Reliable Virtual Data Center Embedding Across Multiple Data Centers

Gang Sun[1,2], Sitong Bu[1], Vishal Anand[3], Victor Chang[4] and Dan Liao[1]
[1]*Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), UESTC, Chengdu, China*
[2]*Institute of Electronic and Information Engineering in Dongguan, UESTC, Chengdu, China*
[3]*Department of Computer Science, The College at Brockport, State University of New York, New York, U.S.A.*
[4]*Leeds Beckett University, Leeds, U. K.*

Keywords: Virtual Data Center, Embedding, Reliability, Multi-domain.

Abstract: Cloud computing has become a cost-effective paradigm for deploying online service applications in large data centers in recent years. Virtualization technology enables flexible and efficient management of physical resources in cloud data centers and improves the resource utilization. A request for resources to a data center can be abstracted as a virtual data center (VDC) request. Due to the use of a large number of resources and at various locations, reliability is an important issue that should be addressed in large and multiple data centers. However, most research focuses on the problem of reliable VDC embedding in a single data center. In this paper, we study the problem of reliable VDC embedding across multiple data centers, such that the total bandwidth consumption in the inter-data center backbone network is minimized, while satisfying the reliability requirement of each VDC request. We model the problem by using mixed integer linear programming (MILP) and propose a heuristic algorithm to address this NP-hard problem efficiently. Simulation results show that the proposed algorithm performs better in terms of lowering physical resource consumption and VDC request blocking ratio compared with existing solution.

1 INTRODUCTION

Cloud computing has become a promising paradigm that enables users to take advantage of various distributed resources (Armbrust et al., 2010). In a cloud computing environment, the cloud provider(s) owning the physical resource pool (i.e., physical data centers) offer these resources to multiple users/clients. A user's demand for resources can be abstracted as a virtual data center (VDC), also known as a virtual infrastructure. A typical VDC consists of virtual machines (VMs) connected through virtual switches, routers and links with communication bandwidth. The VDCs are able to provide better isolation and utilization of network resources, thereby improving the performance of services and applications. The main challenge associated with VDC management in cloud data centers is efficient VDC embedding (or mapping), which aims at finding a mapping of VMs and virtual links to physical components (e.g., servers, switches and links).

Due to the use of a large number of resources and at various locations in cloud data centers, providing reliability of cloud services is an important issue that needs attention. A service disruption can lead to customer dissatisfaction and revenue loss. However, restoring failed services is costly. Thus, many cloud services are deployed in distributed data centers to meet their high reliability requirements. Furthermore, some services may require to be within the proximity of end-users (e.g., Web servers) whereas others may not have such location constraints and can be embedded in any data center (e.g., MapReduce jobs) (Amokrane et al., 2013). Therefore, reliable VDC embedding across distributed infrastructures is appealing for Service Providers (SP) as well as Infrastructure Providers (InPs).

However, there is limited research on the problem of reliable VDC embedding across multiple data centers, with most research focusing on only VDC embedding within a single data center. For example, C. Guo et al. (2013) proposed a data center network virtualization architecture, SecondNet, in which VDC is used as the basic unit of resource allocation for multiple tenants in the cloud. However, they do not consider the VDC reliability. M. Zhani et al. (2013) designed a migration- aware dynamic VDC embedding framework for efficient VDC planning, which again

Sun, G., Bu, S., Anand, V., Chang, V. and Liao, D.
Reliable Virtual Data Center Embedding Across Multiple Data Centers.
DOI: 10.5220/0005842101950203
In *Proceedings of the International Conference on Internet of Things and Big Data (IoTBD 2016)*, pages 195-203
ISBN: 978-989-758-183-0

without reliability considerations. Zhang et al. (2014) proposed a framework, Venice, for reliable virtual data center mapping. The authors prove that the reliable VDC embedding problem is NP-hard problem. However, this study is limited to a single data center.

There are also some studies on virtual data center or virtual network (VN) embedding across multiple domains. The work in (Amokrane et al., 2013) studied the problem of VDC embedding across distributed infrastructures. The authors in (Yu et al., 2014) presented a framework of cost efficient VN embedding across multiple domains, called MD-VNM. Di et al. (2014) proposed an algorithm for reliable virtual network embedding across multiple data centers. However, there are differences between VDC embedding and VN embedding in some aspects. For example, a VDC is a collection of VMs, switches and routers that are interconnected through virtual links. Each virtual link is characterized by its bandwidth capacity and its propagation delay. VDCs are able to provide better isolation of network resources, and thereby improve the performance of service applications. A VN is a combination of active and passive network elements (network nodes and network links) on top of a Substrate Network (SN) (Fischer et al., 2013).In addition, a VN node generally cannot be embedded on a physical node that hosts another VN node of the same VN, whereas each physical server can host multiple VMs from the same VDC in the VDC embedding problem. Therefore, most existing VDC embedding approaches cannot be directly applied to solve the problem of reliable VDC embedding across distributed infrastructures.

In this paper, we study the problem of reliable VDC embedding (RVDCE) across multiple data centers. In our work, we minimize the total backbone bandwidth consumption, while satisfying the reliability requirement of VDC request. We formulate the RVDCE problem as a mixed integer linear programming (MILP) problem and propose a cost efficient algorithm for solving this problem. Simulation results show that the proposed algorithm performs better than the existing approach.

2 PROBLEM STATEMENT

In this section, we give the descriptions on the problem of reliable VDC embedding across distributed infrastructures.

2.1 VDC Request

A VDC request consists of multiple VMs and virtual links that connect these VMs. Figure 1 shows an example of a VDC request with 5 VMs, interconnected by virtual links. The *i*-th VDC request is represented by $G^i = (V^i, E^i)$, where V^i denotes the set of virtual machines, E^i is the set of virtual links. The VMs with location constraints can only be embedded onto the specified data centers, whereas the VMs without location constraints can be embedded onto any data center in the substrate infrastructure. We use R to denote the types of resources (e.g., CPU or memory) offered by each physical node. We use c_v^{ir} to denote the requirement of resource of virtual node v, b_e to present the amount of bandwidth required by virtual link e. Similarly, we define s_{ve} and d_{ve} as the variables that indicate whether virtual node v is the source or destination of link e.

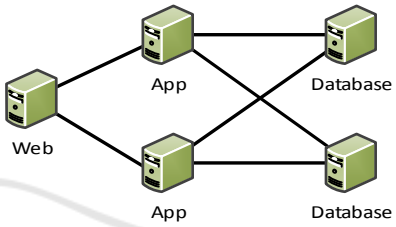

Figure 1: Example of a VDC request.

2.2 Distributed Infrastructure

The distributed infrastructure consists of the backbone network and data centers managed by infrastructure providers (InP).Thus, the InP knows the information of all the distributed data centers. Typically, the cost of per unit bandwidth in the backbone network is more expensive than the cost of per unit bandwidth within data center (Greenberg et al., 2008).

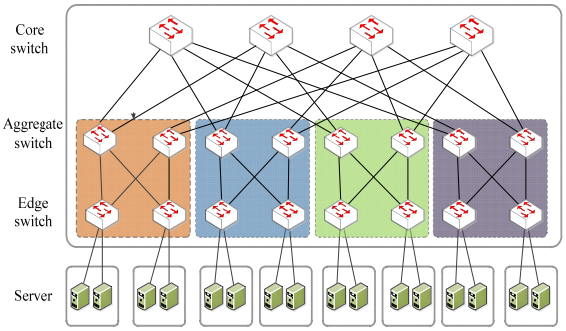

Figure 2: The Fat-Tree topology.

We model the physical infrastructure as a undirected graph $G = (\bar{V} \cup \overline{BV}, \bar{E} \cup \overline{BE})$, where $\bar{V}$ denotes the set of physical nodes in the data centers, $\overline{BV}$ denotes

the set of nodes in backbone network, $\bar{E}$ denotes the set of physical links of data centers, and $\overline{BE}$ denotes the set of physical links of the backbone and physical links that connect the backbone and data centers. Let $G^k = (\bar{V}^k, \bar{E}^k)$ represent the physical data center k, where $\bar{V}^k$ denotes the set of physical nodes and $\bar{E}^k$ denotes the set of physical links in the data center. We use $c_{\bar{v}}^r$ to indicate the capacity of resource on the physical node $\bar{v}$, and $b_{\bar{e}}$ to indicate bandwidth capacity of physical link $\bar{e}$. Let $s_{\bar{v}\bar{e}}, d_{\bar{v}\bar{e}}$ be indicator variables that denote whether $\bar{v}$ is the source or destination of physical link $\bar{e}$ .Without loss the generality, in this work, we assume that each data center has a Fat-Tree (Leiserson, 1985) topology as shown in Figure 2.

2.3 Reliable VDC Embedding

We define the reliability of a VDC as the probability that the service is still available even after the failure of physical servers. Similar to (Zhang et al., 2014), we use replication groups to guarantee the reliability requirements of data centers. The basic idea is that if one VM in the replication group fails, the other VM in the same replication group can run as a backup. In other words, a replication group is reliable as long as one of the VMs in the group is available. The VMs in different replication groups have different functionalities, and the set of all replication groups form the complete service. Therefore, when any group fails, the entire service is unavailable, since the rest of groups cannot form a complete service. In this work, the key objective is to guarantee the reliability of the whole VDC, while satisfying its resource requirements.

The reliability *rl* of a VDC which has been embedded can be calculated as follows:

$$rl = \sum_{i \in RC} \prod_{\bar{v} \in F} fr_{\bar{v}} \prod_{\bar{v} \in NF} (1 - fr_{\bar{v}}) \,, \forall \bar{v} \in PN \,, \qquad (1)$$

where *PN* denotes the set of physical nodes which host the VMs in the VDC; *RC* denotes the set of cases which can guarantee the availability of the VDC; $fr_{\bar{v}}$ denotes the failure probability of the physical node $\bar{v}$; *F* is the set of the failed physical nodes in all data centers; and *NF* is the set of the available physical nodes in data centers.

The problem of VDC embedding across multiple domains (i.e., multiple data centers) means that VMs in a VDC may be embedded in multiple data centers. Since VMs have different location constraints, they may not be embedded in the same data center. Figure 3 shows an example of VDC embedding across multiple domains.

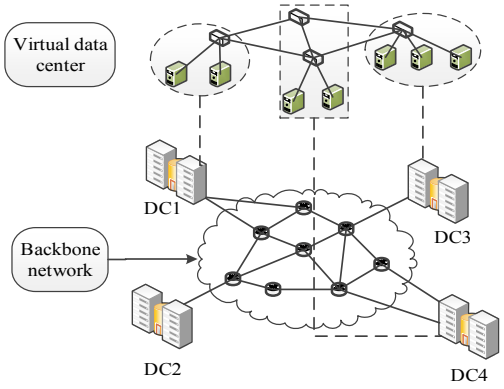

Figure 3: Example of VDC cross-domain embedding.

3 MILP MODEL

In this section we describe the constraints and objective of our researched problem and then construct a MILP model.

3.1 The Constraints

In order to ensure that the VDC embedding does not violate the physical resource capacity limits, the following capacity constraints must be met.

$$\sum_{i \in I} \sum_{v \in V^i} x_{v\bar{v}}^i c_{\bar{v}}^{ir} \le c_{\bar{v}}^r, \ \forall \bar{v} \in \bar{V}, r \in R \,, \qquad (2)$$

$$\sum_{i \in I} \sum_{e \in E^i} b_{e\bar{e}}^i \le b_{\bar{e}}, \ \forall \bar{e} \in \bar{E} \,, \qquad (3)$$

Where $x_{v\bar{v}}^i$ is a variable denoting whether the virtual node *v*(i.e., VM v)of VDC*i* is embedded on the physical node $\bar{v}$; $b_{e\bar{e}}^i$ denotes the amount of bandwidth resources provisioned by physical link $\bar{e}$ for virtual link *e* of VDC *i*. In addition, the flow conservation constraints below must also be satisfied.

$$\sum_{e \in E} s_{ve}^i b_{e\bar{e}}^i - \sum_{e \in E} d_{ve}^i b_{e\bar{e}}^i = \sum_{v \in V^i} x_{v\bar{v}}^i s_{ve}^i b_e - \sum_{v \in V^i} x_{v\bar{v}}^i d_{ve}^i b_e, \forall i \in I, e \in E^i, \bar{v} \in \bar{V} \qquad (4)$$

We define $\tilde{x}_{v\bar{v}}^i$ as a variable that indicates whether the VM *v* can be embedded on physical node $\bar{v}$. Then the virtual node placement constraints can be formulated as follows:

$$x_{v\bar{v}}^i \le \tilde{x}_{v\bar{v}}^i, \forall i \in I, \ v \in V^i, \bar{v} \in \bar{V} \,, \qquad (5)$$

$$\sum_{\bar{v}\in V} x^i_{v\bar{v}} = 1, \quad \forall i \in I, v \in V^i . \tag{6}$$

We define $y_{\bar{v}}$ as a variable to indicate whether the physical node $\bar{v}$ is active. If a physical node hosts at least one VM, then this node must be active, otherwise inactive. It means that the following constraints should be satisfied:

$$y_{\bar{v}} \geq x^i_{v\bar{v}}, \quad \forall i \in I, v \in V^i, \bar{v} \in \overline{V} , \tag{7}$$

$$y_{\bar{v}} \geq \frac{1}{b_e} b^i_{e\bar{e}} s_{\bar{v}e}, \quad \forall i \in I, \bar{v} \in \overline{V}, e \in E^i, \bar{e} \in \overline{E} , \tag{8}$$

$$y_{\bar{v}} \geq \frac{1}{b_e} b^i_{e\bar{e}} d_{\bar{v}e}, \quad \forall i \in I, \bar{v} \in \overline{V}, e \in E^i, \bar{e} \in \overline{E} . \tag{9}$$

Furthermore, the following geographical location constraints must be satisfied:

$$z^i_{kv} = \begin{cases} 1 & \text{if } v \text{ can be assigned to DC } k \\ 0 & \text{otherwise} \end{cases} , \tag{10}$$

$$w^i_{kv} = \begin{cases} 1 & \text{if } v \text{ is assigned to DC } k \\ 0 & \text{otherwise} \end{cases} , \tag{11}$$

$$w^i_{kv} \leq z^i_{kv}, \quad \forall v \in V^i , \tag{12}$$

$$\sum_{v \in V^i} w^i_{kv} = 1, \quad \forall k : \mathrm{G}^k . \tag{13}$$

3.2 The Objective

We use A_i to represent the reliability of *i*-th VDC. If the InP fails to meet the reliability requirement A_i, there will be a penalty:

$$P^{unavail} = \sum_{i \in I} (1 - A_i)\pi_i , \tag{14}$$

where π_i denotes the penalty from failing to meet the reliability of the *i*-th VDC.

Recovery costs come from restarting VMs and reconfiguring the network equipment. We thus define the failure recovery costs of node $\bar{v}$ and link $\bar{e}$ as shown in Equation (15) and (16), respectively. Let $\rho_{\bar{v}}$ and $\rho_{\bar{e}}$ represent the energy cost of an active node $\bar{v}$ and link $\bar{e}$, respectively.

$$P^{restore}_{\bar{v}} = \rho_{\bar{v}} + \sum_{i \in I} \sum_{v \in V^i} x^i_{v\bar{v}} \lambda_v + \sum_{e \in E^i} b^i_{e\bar{e}} \mu_{\bar{v}e} \lambda_e, \quad \forall \bar{v} \in \overline{V} , \tag{15}$$

$$P^{restore}_{\bar{e}} = \rho_{\bar{e}} + \sum_{i \in I} \sum_{e \in E^i} b^i_{e\bar{e}} \lambda_e, \quad \forall \bar{e} \in \overline{E} \setminus \overline{BE} , \tag{16}$$

where λ_v and λ_e are the recovery costs of virtual node v and virtual link e, and $u_{\bar{v}\bar{e}} = max\{s_{\bar{v}\bar{e}}, d_{\bar{v}\bar{e}}\}$ denotes that whether node $\bar{v}$ is the source or destination of link $\bar{e}$. The cost for embedding e onto backbone network is:

$$P^{backbone}_{\bar{e}} = \sum_{e \in E^i} b^i_{e\bar{e}} \lambda_{eb}, \quad \forall \bar{e} \in \overline{BE} , \tag{17}$$

where λ_{eb} denotes the provisioning cost of virtual link e in backbone network.

We use $F_{\bar{v}}$ and $F_{\bar{e}}$ to denote the failure probability of node $\bar{v}$ and link $\bar{e}$, respectively. Then the total cost of unreliable VDC is as follows:

$$P_A = P^{unavail} + \sum_{\bar{v} \in \overline{V}} F_{\bar{v}} P^{restore}_{\bar{v}} + \sum_{\bar{e} \in \overline{E}} F_{\bar{e}} P^{restore}_{\bar{e}} + P^{backbone}_{\bar{e}} . \tag{18}$$

Therefore, the objective function to minimize the total cost defined as follows:

$$Minimize \quad P_A . \tag{19}$$

4 ALGORITHM DESIGN

In this section, we propose an efficient algorithm for solving the problem of reliable VDC embedding (RVDCE) across multiple domains. The RVDCE problem consists of two key issues: *i*) how to ensure the reliability of the VDC request; *ii*) how to reduce the backbone bandwidth consumption.

Multiple VMs may be embedded on the same physical node, and the reliability of embedding all VMs of a VDC on the same physical node is higher than that of embedding the VMs on different nodes. The reason is as follows: according to the definition of the reliability requirement of a VDC, reliable service implies that all replication groups are reliable. All replication groups form the complete service, and each plays a unique role as explained in the third part of Section 2.Therefore, when any group fails, the entire service is unavailable. Hence, if we embed all VMs belonging to different replication groups on the same physical node, the service reliability is equal to the reliability of that physical node. On the other hand, if we embed all VMs on different physical nodes, the service reliability is equal to the product of the reliability of those physical nodes. In addition, in order to improve the service reliability, we need to embed VMs in the same replication group on to different physical node so that these VMs can backup

each other. However, physical nodes with limited resources and VMs with location constraints may not allow all VMs in a VDC to be embedded onto the same physical node, or even the same data center.

Moreover, reducing the backbone bandwidth consumption is the other issue we have to address in this work. For addressing this issue, we partition a VDC into several partitions before mapping it. The reasons are as follows: *i*) if we place VMs one by one, it will cause a large amount of backbone bandwidth consumption, because the backbone bandwidth consumption is not considered in the VM mapping process; *ii*) the network resource in inter-data centers is more expensive than that of intra-data centers networks. We put the VMs that have a large amount of communication bandwidth requirement between each other into the same partition, and the VMs in the same partition will be mapped into the same data center. Thus, the backbone bandwidth consumption can be reduced. On the other hand, the way of mapping VMs one by one mentioned above will cause much higher reliability than required. It is unnecessary that the data centers provide much higher reliability than that required by VDC request, so it is necessary to just meet the required reliability for reducing the resource consumption.

In addition, if we map virtual components with low reliability requirement on the servers with high reliability, the physical server resources will be wasted and may result in increasing in the blocking ratio of the VDC requests in the long term. Therefore, in order to use physical server resource with different reliability rationally, we grade the servers in data center according to their reliability.

The RVDCE algorithm consists of three main steps: i) group the physical servers; ii) partition the VDC; iii) embed the VDC partitions. The detailed RVDCE algorithm as shown in Algorithm 1.

Algorithm 1: Reliable VDC Embedding.

Input: 1. Size of partitions: N;
2. The VDC request: $G^i = (\mathrm{V}^i, \mathrm{E}^i)$;
3. Substrate data center: $G = (\overline{V} \cup \overline{BV}, \overline{E} \cup \overline{BE})$.
Output: The VDC embedding result.
1: Sort servers in data centers in ascending order of their reliability;
2: Divide the servers in data centers into L levels, where a lower level has a lower reliability;
3: The initial partition size denoted as K, **let** $K = N$;
4: **let** *isSuccessful* ← false;
5: **while** $K > 0$ **do**
6: Call **Procedure1** to partition the VDC request;
7: **for** all $l \in L$ **do**
8: $S \leftarrow$ the set of servers in all data centers whose levels are lower than or equal to l;
9: Call **Procedure 2** to embed partitions;
10: **if** all partitions are embedded successfully
11: *isSuccessful* ← true;
12: **return** VDC embedding result;
13: **end if**
14: **end for**
15: $K = K - 1$
16: **end while**
17: **if** (*isSuccessful* == false)
18: **return** VDC embedding is failed
19: **end if**

4.1 Group the Physical Servers

We sort the servers in descending order of their reliability, and then group the servers into groups with different reliability levels. That is, higher level means higher reliability. In the embedding process, we chose level *l* servers to host a VDC, meaning that we can use the servers whose levels are less than or equal to *l* for hosting the VDC. If the reliability does not meet VDC requirement after completing the VDC embedding, it is necessary to increase the reliability level *l* and re-embed the VDC. If the reliability is lower than the required for any available physical servers, we have to change the partition size and re-embed the VDC.

4.2 Partition the VDC

Before embedding a VDC, we first partition the VDC into several sub-VDCs, with the aim of minimizing the bandwidth demands between partitions, thereby reducing the bandwidth consumption in the backbone network.

Assume the number of VMs in a VDC is *N*, the partition size is *K*(i.e., the number of VMs in each partition cannot exceed *K*), where the partition size *K* is adjustable. The initial value of *K* is equal to *N*, then the *K* is gradually reduced in the process of adjusting the size. If a VDC is successfully embedded while *K*= *N*, we do not need to adjust the size *K*, since the backbone bandwidth consumption is minimized.

In the VDC partition process, we first make each node as a partition, and calculate the total amount of bandwidth demands between the original partitions. Then the algorithm traversals all nodes $v \in V^i$, and

finds partition P that allows us move v from its original partition to P and satisfy the follow conditions: *i*) reduce the amount of bandwidth in backbone network; *ii*) the VMs in P have same location constraints; *iii*) the number of VMs in P is smaller than k. If a partition P that meets the above conditions is found, we will move VM v to P. As long as there are nodes moving between different partitions, algorithm keeps traversing the VMs until there is no node need to be moved. If the current bandwidth demands in inter-data center is less than the bandwidth demands between original partitions, algorithm will regenerate a new graph where a partition of G^i is as a node in this new graph.

Procedure 1: VDC Partition.

```
1: Let flag← true;
2: while(flag)
3:   Denote each node of G^i as apartition;
4:     Record the initial bandwidths
       between partitions;
5:  while nodes need to be moved between
      partitions do
6:        for each v∈V^i , do
7:           Find a partition P and move
             v to P, such that: a) the
             number of VMs in P does not
             exceed k ; b) bandwidths
             consumptionis minimized; c)
             all of the nodes in P with
             same location constraint.
8:        end for
9:      end while
10:     if current bandwidth demands <
        initial bandwidth demands
11:          Change G^i as the graph of
partitions;
12:     else
13:        flag ←false
14:     end if
15:end while
```

Figure 4 shows an example of partitioning a VDC request. We assume that two data centers included in the substrate network, denoted as *DC1* and *DC2*. The VDC request *m* is shown in Figure 4(a). The location constraints of the four VMs are as follows: *i*)VM *a* must be embedded in *DC1*;*ii*) VM *b* and VM *c* must be embedded in *DC2*; *iii*)VM *d* can be embedded in *DC1* or*DC2*. Therefore, VM a cannot be in the same partition with VM *b* and VM *c*. Figure 4(b) shows the partitioning of VDC m when the partition size is 1. When the partition size is 2, the feasible partitions of *m* are shown in Figure 4(b) and Figure 4(c). When the partition size is 3, the feasible partitions of *m* are shown in Figure 4(b), Figure4(c) and Figure 4(d).

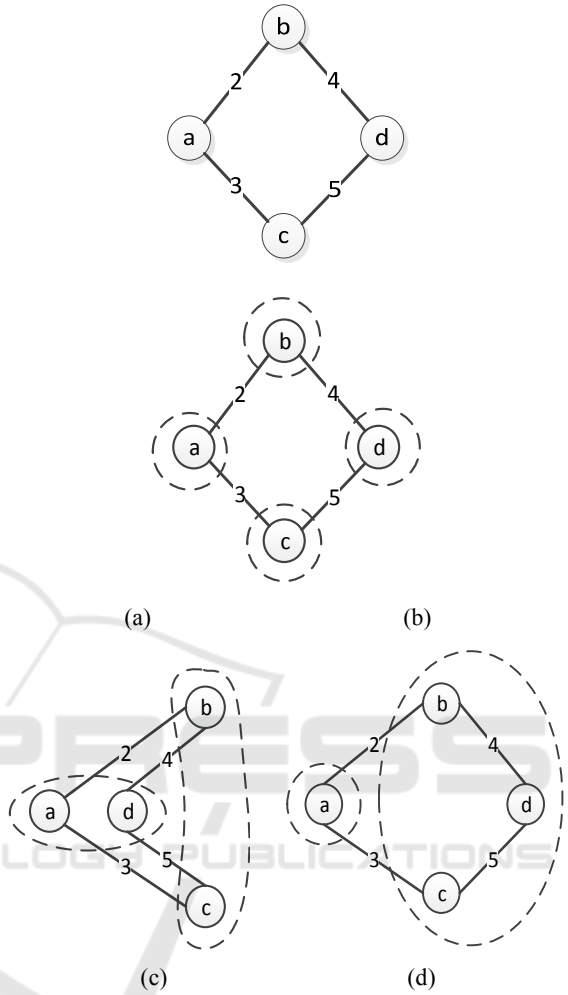

Figure 4: Example of partitioning a VDC.

4.3 Embed the Partitions

We guarantee the reliability of VDC by using replication groups. If one VM in the replication group fails, the other VM in the same replication group can run as a backup. Therefore, a replication group is reliable as long as at least one of the VMs in the group is reliable. We choose one node in a replication group as the working node, the other nodes in that replication group can be used for backup. We take different approaches for embedding the working node and backup nodes.

4.3.1 Embedding of Working Nodes

Since the VMs in partition have location constraints, each partition has a corresponding location constraint.

Partitions can only be embedded on to data centers that satisfy the location constraints.

We choose the first partition randomly and find a data center that can provide the highest reliability for the chosen partition. Then we embed the VMs in the partition according to the following two steps: *1*) embedding the VMs without backups;*2*) embedding the VMs that with backups. If any VM in the replication group has been embedded, the VMs belong to this replication group in the partition should be skipped. In other words, only one VM of each replication group needs to be embedded. We preferentially embed it on the servers that have hosted other VMs belong to the same VDC. If the resources of the physical server are not enough, we need to embed the VM on a new available server which with the highest reliability.

4.3.2 Embedding of Backup Nodes

We calculate the reliability of the VDC by using the reliability computing algorithm (Zhang et al., 2014), after embedding the working node. If the reliability is higher than required, we embed the backups into the physical servers with lower reliability. If the reliability is lower than required, we embed the backups into the physical servers with higher reliability.

Procedure 2: Partition embedding.

```
1: Randomly chose a partition g from partition set Partitions;
2: Choose a data center dc with the highest reliability and enough resources from data center set DCs;
3: PartitionToDC = PartitionToDC ∪ <g, dc>.
4: M: the set of nodes in VDC that have been embedded, let M = φ;
5: while Partitions is not empty do
6:    for all v in g do
7:       if none of the virtual node in the replication group that v belongs to has been embedded
8:          if v can be embedded on the server s ∈ S hosting the nodes in M
9:             Embed v on server s;
10:            M ← M ∪ {v};
11:         else
12:            Chose the server has highest reliability that belongs to dc and s to host v;
13:         end if
14:      end if
15:   end for
16:   Partitions ← Partitions \ {g};
17:   g ← the partition in Partitions which has the largest amount of bandwidth demand to communicate with the embedded partitions;
18: Choose a dc that can provide highest reliability and enough resources to g from DCs;
19:   PartitionToDC = PartitionToDC ∪ <g, dc>;
20: end while
21: Compute the current reliability r of VDC request by using the reliability computing algorithm proposed in (Zhang et al., 2014);
22: for all <g, dc> in PartitionToDC do
23:    if r > ra
24:       for each node v in g do
25:          Embed v on the server with enough resources and the lowest reliability in dc;
26:       end for
27:    else
28:       for remaining nodes v in g do
29:          if v can be embedded on the server that node μ in g has been embedded on and the replication groups of v and μ are different
30:             Embed v on s;
31:          else
32:             Embed v on the server has highest reliability;
33:          end if
34:       end for
35:    end if
36: end for
```

We note that all VMs belonging to the same partition must be embedded in the same data center. Accordingly, when embedding the backup VMs, we need to choose the data center that has the embedded VMs belonging to the same partition. In addition, it is important to note that the VMs belonging to the same replication group cannot be embedded on to the same server. After embedding the backup nodes, we calculate the reliability and check whether the reliability meets the VDC requirement.

5 SIMULATIONS AND ANALYSIS

5.1 Simulation Environment

In our simulation, each data center has a Fat-Tree (Leiserson, 1985) topology composed of 128 physical servers connected through 80 switches. Each physical server has 1000units CPU resource. The capacity of physical link in data center is 100 units. We generate the VDC requests by using GT-ITM tool (GT-ITM) and the parameter settings are similar to (Zhang et al., 2014). The number of VMs in each VDC request is randomly set between 3 and 15.The CPU resource demand of each VM is generated randomly between 8 to 16 units. The bandwidth requirement of each virtual link is set randomly between 1 and 3 units.

For evaluating the effectiveness and correctness of our proposed algorithm, we have implemented three algorithms for comparison purposes. The algorithms compared in our simulation experiments are shown in Table 1.

Table 1: Algorithms compared in our simulations.

Algorithms	Descriptions
RVDCE	The algorithm proposed in this work for provisioning reliable VDC across multiple data centers.
RVDCE_R	The algorithm proposed in this work for provisioning reliable VDC across multiple data centers in which the VMs are embedded on the available servers with the highest reliability.
RVNE	Algorithm proposed in (Yu et al., 2014) for reliable virtual network embedding across multiple domains.

5.2 Simulation Experiment Results

Backbone bandwidth consumption: Figure 5 shows the performance of average backbone bandwidth consumption for provisioning VDC requests against the number of VDC requests. The *K* in Fig. 5 represents the size of the partition. It can be seen that the RVDCE algorithm leads to a lower average backbone bandwidth consumption compared to that of RVNE and RVDCE_R. This is because RVDCE divides a VDC request into multiple partitions and consumes the bandwidth of backbone network as few as possible while mapping these partitions. Furthermore, average backbone bandwidth consumption of RVDCE decreases with the growth of the value of *K*. Since larger partition size leads to a lower bandwidth consumption between partitions.

Blocking ratio of VDC request: Figure 6 shows the performance of blocking ratio under various reliability requirements. In this set of simulations, the reliability requirement *rr* varies from 0.92 to 0.98 with a step of 0.02.It is clear that when the number of VDCs is small, the blocking ratios of the three compared algorithms are very low. For example, in Figure 6(a), when the number of VDC requests is smaller than 20, the blocking ratios of these algorithm is zero. The blocking ratio increases with the growth of the number of VDC requests. The blocking ratio of RVDCE is lower than that of the other two algorithms under different reliability requirements. It is due to fact that RVDCE algorithm embeds VMs onto servers with as lower reliability as possible while satisfying the reliability requirements of VDCs, for avoiding over-provisioning and thus can admit more VDC request.

Cumulative CPU resource consumption: Figure 7 presents the results of total CPU resource consumption for various number of VDC requests. The cumulative CPU resource consumption increases with the growth of the number of VDC requests. It also can be seen from Figure 7 that the CPU resource consumption of RVDCE is lower than that of RVNE. This is because our RVDCE algorithm does not use redundant resources as the backups for satisfying the VDC reliability. Furthermore, our proposed algorithm selects the servers to host the VMs according to the dependencies between VMs, thus avoiding over-provisioning and results in a lower resource consumption.

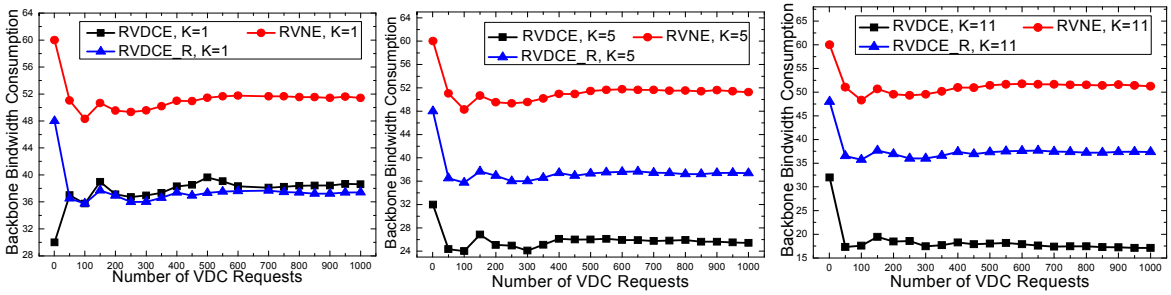

Figure 5: Average backbone bandwidth consumption.

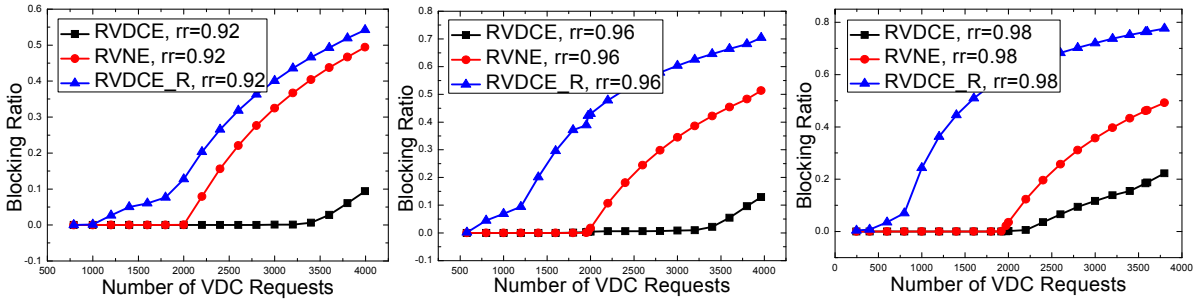

Figure 6: Blocking ratios under different reliability requirements.

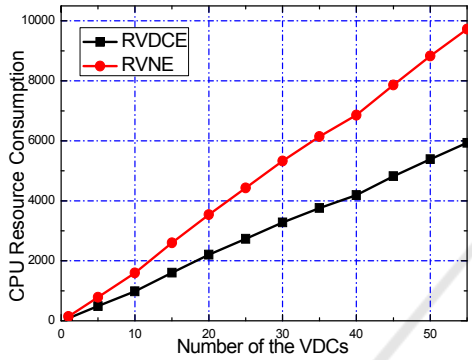

Figure 7: Total CPU resource consumption.

6 CONCLUSION

In this paper, we have studied the problem of reliable VDC embedding across multiple infrastructures and proposed a heuristic algorithm for solving this problem. The aim of our research is to minimize the total bandwidth consumption in backbone network for provisioning a VDC request, while satisfying its reliability requirement. Our algorithm makes a trade-off between backbone bandwidth consumption and reliability. Simulation results show that the proposed algorithm significantly reduced the resource consumption and blocking ratio than the existing approach.

ACKNOWLEDGEMENTS

This research was partially supported by the National Grand Fundamental Research 973 Program of China under Grant (2013CB329103), Natural Science Foundation of China grant (61271171, 61571098), China Postdoctoral Science Foundation (2015M570778), Fundamental Research Funds for the Central Universities (ZYGX2013J002), Guangdong Science and Technology Project (2012B090400031, 2012B090500003, 2012B091000163), and National Development and Reform Commission Project.

REFERENCES

Armbrust, M., Fox, A., Griffith, R., et al, 2010. A view of cloud computing. *Communications of the ACM.*

Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M., 2013. Data center network virtualization: A survey. *IEEE Communications Surveys and Tutorials.*

Amokrane, A., Zhani, M., Langar, R., Boutaba, R., Pujolle, G., 2013. Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures. *IEEE Transactions on Cloud Computing.*

Zhang, Q., Zhani, M., Jabri, M., Boutaba, R., 2014. *Venice: Reliable Virtual Data Center Embedding in Clouds*. IEEE INFOCOM.

Guo, C., Lu, G., Wang, H., Yang, S., Kong, C., Sun, P., Wu, W., and Zhang, Y., 2010. *SecondNet: a data center network virtualization architecture with bandwidth guarantees*. ACM CoNEXT.

Zhani, M. F., Zhang, Q., Simon, G., and Boutaba, R., 2013. *VDC planner: Dynamic migration-aware virtual data center embedding for clouds*. IFIP/IEEE IM.

Yu, H., Wen, T., Di, H., Anand, V., Li, L., 2014. Cost efficient virtual network embedding across multiple domains with joint intra-domain and inter-domain embedding. *Optical Switching and Networking.*

Di, H., Anand, V., Yu, H., Li, L., Lianand D., Sun, G., 2014. Design of Reliable Virtual Infrastructure Using Local Protection. *IEEE International Conference on Computing, Networking and Communications.*

Fischer, A., Botero, J., Beck, M., Meer, H., Hesselbach, X., 2013. Virtual Network Embedding: A Survey. IEEE Communications Surveys &Tutorials.

Greenberg, A., Hamilton, J., Maltz, D., and Patel, P., 2008. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review.*

Leiserson, C., 1985. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. IEEE Transactions on Computers.

GT-ITM. http://www.cc.gatech.edu/projects/gtitm/