IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2019, accepted June 9, 2019, date of publication June 14, 2019, date of current version July 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923096

Toward SLAs Guaranteed Scalable VDC Provisioning in Cloud Data Centers

GANG SUN[1], ZHU XU[1], HONGFANG YU[1], VICTOR CHANG[2], XIAOJIANG DU[3], AND MOHSEN GUIZANI[4], (Fellow, IEEE)

[1]Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu 610051, China
[2]International Business School Suzhou, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
[3]Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[4]Department of Computer Science and Engineering, Qatar University, Doha, Qatar

Corresponding authors: Gang Sun (gangsun@uestc.edu.cn) and Hongfang Yu (hfyu@uestc.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61571098, and in part by the 111 Project under Grant B14039.

ABSTRACT Cloud computing has been a cost-efficient paradigm for deploying various applications in datacenters in recent years. Therefore, efficient provisioning for virtual data center (VDC) requests from different service providers (SPs) over physical data centers plays a vital role in improving the quality of service (QoS) and reducing the operational cost of SPs. Therefore, a significant attention has been paid for the VDC provisioning problem. However, few approaches have been proposed for the problem of reliable VDC embedding across multiple data centers, as most of them only consider the problem of VDC mapping within a single data center. In this paper, we study the problem of QoS-aware VDC provisioning 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 formulate this problem as a mathematical optimization problem by using integer linear programming (ILP) and propose an efficient heuristic algorithm called reliable VDC embedding (RVDCE) algorithm to solve this NP-hard problem. The simulation results show that the proposed algorithm performs better in terms of blocking ratio, CPU resource consumption, and bandwidth consumption of backbone network than the existing solution. In addition, this paper has also incorporated integrated security to minimize security vulnerabilities seen in other similar approaches. Apart from demonstrating how to resolve security challenges in our VDC proposal, cost calculations have been implemented to demonstrate the robustness, resiliency, validity, and effectiveness of the VDC provisioning solution for cloud computing.

INDEX TERMS Provisioning, reliability, virtual data center, service level agreements, cloud computing.

I. INTRODUCTION

Cloud computing has become a promising paradigm that enables users to share the various distributed resources [1], [2]. In the cloud computing environment, an infrastructure provider (InP) owns the physical infrastructure and virtualizes the physical resources (i.e., physical data centers) into virtual resources; and offers the virtualized resources to Service Providers (SPs). Such virtualization of physical resources brings flexible and efficient management of physical resources in data center and improves their utilizations [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Liehuang Zhu.

Large number of users sharing the resource of data centers. A resource request from a user can be abstracted as a request in virtual data center (VDC) [4], which is a collection of demands on not only virtual machines (VMs) with computing, memory, and storage resources but also virtual links with bandwidth resources. VDCs are able to provide better isolation and utilization of network resources, thereby improving the performance of service applications. The main challenge associated with VDC management in cloud data centers is efficient VDC provisioning (i.e., mapping or embedding), which aims at finding a mapping of VMs and virtual links to physical components (i.e., servers, switches and physical links) [5] while complying with the service level agreements (SLAs) that have been agreed upon with the customers or users.

Due to the use of a large number of resources at various locations in cloud data centers, providing QoS-aware (e.g., reliability, security and resource demands) cloud services is an important issue that needs attention. For example, a service disruption may lead to SLA violations and results in customer dissatisfaction and loss revenue. Moreover, restoring failed services is costly. Thus, many cloud services have been deployed in distributed data centers to improve the QoS and meet the SLAs [6], [7]. In addition, some services may be required to be within the proximity of end-users (e.g., Web servers) whereas others may not have such location constraint and can be deployed in any data center (e.g., MapReduce jobs) [5]. Therefore, QoS-aware VDC provision across distributed infrastructures is particularly appealing for SPs as well as InPs.

However, the existing researches ignore the difference of bandwidth costs between intra-data center and inter-data center, as they do not consider the location constraints of VMs nor data exchanging requirements between VMs. Moreover, there are differences between VDC embedding and VN embedding in some aspects. A VN is a combination of active and passive network elements (network nodes and network links) on top of a Substrate Network (SN) [4]. In addition, a VN node cannot be embedded on a physical node (e.g., server) 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 QoS-aware VDC provisioning in multiple data centers. Therefore, it is essential for us to propose new algorithm to solve the problem in this research area.

Therefore, in this work, we study the problem of reliable VDC embedding (RVDCE) across multiple data centers, such that the total bandwidth consumption of backbone network is minimized, while satisfying the QoS requirements of VDC request, and propose an efficient resource scheduling algorithm for solving the studied problem.

The main contributions of this work are summarized as follows:

- We design an efficient scheduling algorithm to provide reliable VDC provisioning against the failures of physical components (e.g., servers or links) or the individual failures (e.g., the wrong configuration or malicious attacks).
- We efficiently allocate the bandwidth resources to the VDC request for improving the bandwidth resource utilization and avoiding longer network latency.
- For large-scale cloud applications, our proposed method can deploy VDC requests across multiple cloud data centers for improving the scalability.
- We rationally use the physical server resources (e.g., CPU, storage or memory) for improving the acceptance ratio of online cloud service scenarios.

The remainder of this paper is organized as follows: Section II reviews the related work. Section III gives the problem descriptions. Section IV explains the formulation of the studied problem by using integer linear programming. The details of proposed algorithm for solving this problem are described in Section V. The simulations for evaluating the performance of proposed algorithm are given in Section VI. Section VII presents the security evaluations; and finally, Section VIII concludes this paper.

II. RELATED WORK

Recently, the problem of reliable virtual data center embedding within single data center is researched with extra dimensions of consideration. Guo et al. [8] proposed a novel data center network virtualization architecture, SecondNet, in which the VDC as the granularity of resource allocation for multiple tenants in the cloud. Zhani et al. [9] designed a migration-aware dynamic virtual data center embedding framework for efficient VDC planning. An efficient online VM placement algorithm, Virtual Knotter, had been proposed in [10] to reduce congestion with controllable VM migration traffic as well as to lower time complexity. The authors in [11] developed a novel analytical model to evaluate the performance of heterogeneous VMs on the same physical machine by applying the Continuous Time Markov Chain (CTMC).

Few existing research efforts have yet carefully addressed the problem of reliable VDC embedding across multiple data centers, where they only considered the case where all the VDC components are allocated within the same data center. For example, Zhang et al. [12] proposed a VDC mapping algorithm, with the goal of maximizing the total income of Cloud Provider (CP), while minimizing total cost of recovering hardware failure and the unreliable service. However, this research mentioned above has not addressed the difference of bandwidth costs between intra-data center and inter-data center; it does not consider the location constraints of VMs nor data exchanging requirements between VMs. There are also some studies on virtual data center or virtual network embedding across multiple domains [5], [13], [14]. For example, the authors in [5] studied the problem of virtual data center embedding across distributed infrastructures. They had proposed a management framework, Greenhead, for the problem of VDC mapping across multiple infrastructures. The goals of Greenhead include minimizing energy cost and maximizing total revenue for InP, while ensuring the environment to be as friendly as possible. Sun et al. [13] proposed an algorithm for implementing the resource efficient virtual infrastructure mapping across multi-domain networks, while improving the response delay and acceptance ratio.

Most of the existing researches related to data center can at best be described as attempts to fix existing problems, rather than conscious and focused push to build a complete data center environment. Since multiple aspects of virtual data center provision/deployment need to be explored, the existing research work requires modification and improvement. The key problems of data center virtualization can be summarized as follows:

- Efficient bandwidth consumptions in cloud data centers. Nowadays, the emergence of data-intensive applications has brought us into the "big data" era. Big data applications can generate huge volumes of data that the conventional systems can hardly capture, manage, store and analyze. Therefore, the bandwidth in a data center is a scarce and costly resource making the data center network valuable to users. An efficient bandwidth resource management is of great importance in cloud data centers [15]–[17].
- Large-scale concurrent job/activity management. More and more data center applications are large-scale systems with high performance requirements. In order to meet the increasing demands for services and better performance, the physical infrastructure should scale gracefully to accommodate concurrent jobs enabling incremental expansion without affecting the existing services. Correspondingly, the scheduling strategy should be scalable and can be easily adapted to the new expanded cloud services [18], [19].
- Lowering the network latency of cloud data centers. In cloud data centers, lower network latency should be offered as a basic feature which enabling the data center to provide faster services to users. Primarily the network latency consists of the queuing delay at each hop, transmission delay and propagation delay, of which the buffer queuing at each hop is the major contributor to latency [20]–[22].
- Reliable/survivable virtual data center provisioning. The reliability/survivability aspect of the VDC deployments, in terms of (*i*) hardware failure characteristics on which the service is hosted, and (*ii*) the impact of individual failures on service availability, should be considered while provisioning VDC requests. In particular, many cloud services have high availability requirements, because service outage can potentially incur high penalty in terms of revenue and customer satisfaction [23]–[25].
- Energy-efficient scheduling for cloud data centers. Recent years, research in the areas of "green" and low power consumption cloud infrastructures are of great importance for both infrastructure service providers and equipment manufacturers. Since cloud data center operators expect to minimize the long-term energy cost with uncertainties in electricity price, workload, renewable energy generation, and power outage state [26], [27].
- Data and service security for the cloud services hosted in data centers. The rapid data and service growth poses challenges for the integrated security for the cloud services hosted in the data center, hence offering real-time security for petabytes of data is important for cloud computing. For example, programmability of network elements can increase vulnerability if secure programming models and interfaces are unavailable [28], [29]. Security concerns on VDCs can be real issues for providing real-time services; and functions to minimize security vulnerabilities should be demonstrated.

III. PROBLEM STATEMENT

A. 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 five VMs, interconnected by six virtual links. A virtual machine has node resources demand, and a virtual link has bandwidth requirement. A VDC requires physical resources on servers and links to provide services for users.

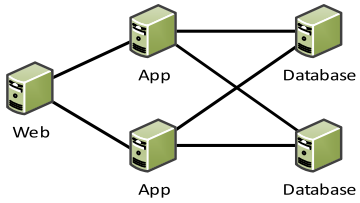


FIGURE 1. Example of a VDC request.

The i-th VDC request can be represented as an undirected weighted graph $G^i = (V^i, E^i)$, where V^i denotes the set of virtual machines, E^i is the set of virtual links. Some of the VMs have location constraints. That is, the VMs with location constraints can only be embedded onto their specified data centers, whereas the VMs without location constraints can be mapped to any data center in the substrate infrastructure. We use *Res* to denote the types of resources (e.g., CPU or memory) offered by each physical server. We use c_v^{ir} to denote the requirement on resource r of virtual machine v, b_e to present the amount of bandwidth required by virtual link e. We define s_{ve} and d_{ve} as binary parameters that indicate whether virtual node (i.e., VM) v is the source or destination of link e. We use rr to denote the reliability requirement of VDC request.

B. DISTRIBUTED INFRASTRUCTURE

We consider a distributed infrastructure that is composed by backbone network and data centers managed by an InP. Thus, the InP knows the information of all distributed data centers. Usually, the cost of per unit bandwidth in backbone network is more expensive than the cost of per unit bandwidth within data center [30]. We thus only consider the cost of bandwidth consumption of backbone network, and ignore the cost of bandwidth consumption within a data center.

We model the physical infrastructure as an undirected graph $G = (\overline{V} \cup \overline{BV}, \overline{E} \cup \overline{BE})$, where $\overline{V}$ denotes the set of physical servers in data centers, $\overline{BV}$ represents the set of nodes (i.e., switches) in backbone network, $\overline{E}$ denotes the set of physical links within data centers, and $\overline{BE}$ indicates the set of physical links of the backbone network. Let $G^k = (\overline{V}^k, \overline{E}^k)$ represents the physical data center k, where $\overline{V}^k$ denotes the set of physical servers and $\overline{E}^k$ denotes the set of physical links in the data center. We use $c_{\bar{v}}^r$ to indicate the capacity of resource r on the physical server $\bar{v}$, and $b_{\bar{e}}$ to indicate the

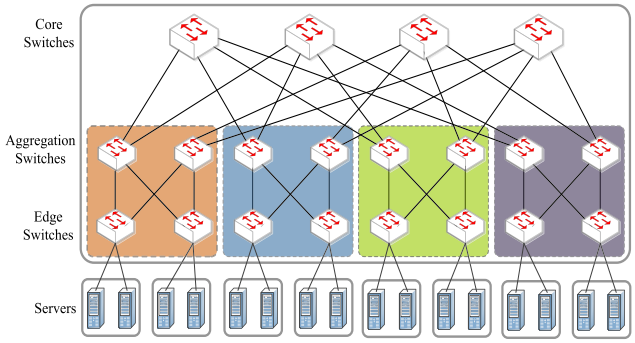


FIGURE 2. Topology of fat-tree.

bandwidth capacity of physical link $\bar{e}$. Let $s_{\bar{v}\bar{e}}$, $d_{\bar{v}\bar{e}}$ be indicators that denote whether $\bar{v}$ is the source or destination of physical link $\bar{e}$. Each server is characterized by a certain failure probability, per unit resource cost and resource capacity. Each link is characterized by its bandwidth capacity. There may be many types of failure in data center, including servers, Top-of-Rack switches, aggregation switches and power supply equipment, etc. In this work, we assume that each data center has a Fat-Tree [31] topology.

C. RELIABLE VDC EMBEDDING

We define the reliability of a VDC as the probability that the service is still available while multiple physical servers failed. Recent analyses [32], [33] on data center hardware reliability has shown that physical data center components have non-uniform failure rates. Thus, we assume that the failure probabilities of different physical server in a data center are also different. Similar to [12], we use replication groups to guarantee the reliability requirements of VDC. 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 at least one of the VMs in the group is available. The VMs in different replication groups implement 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. For example, a VDC with a three-layer structure of web service, including web servers, application servers and database servers. These three replication groups form the complete VDC service function. When any group fails, the entire service would be unavailable, since the rest groups cannot form a complete service.

The reliability rl of an embedded VDC can be calculated as follows:

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

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

Reliable VDC provisioning/embedding problem aims at mapping resource (e.g., virtual machines, switches and communication bandwidth) requests onto the physical infrastructure (e.g., physical servers and links), while guaranteeing the reliability the user or the tenant required. Some services need to be deployed close to end-users (e.g., Web servers) whereas others may not have such location constraints and can be placed in any data center (e.g., Mapreduce jobs) [5]. However, achieving reliable VDC embedding across distributed data center introduces a nontrivial challenge for cloud providers, which aims at mapping the VDC onto the distributed data centers while satisfying the reliability requirements of VDC request. 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. Figure 3 shows an example of mapping a VDC across multiple domains.

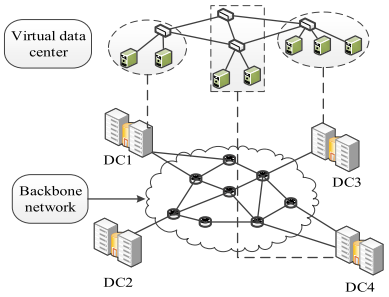


FIGURE 3. Example of mapping a VDC across multiple domains.

IV. PROBLEM MODELING

In this section, we model the problem of reliable VDC embedding across distributed infrastructures by using integer linear programming (ILP).

A. THE CONSTRAINTS

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

$$\sum_{i \in I} \sum_{v \in V^i} x^i_{v\bar{v}} c^{ir}_v \le c^r_{\bar{v}}, \quad \forall \bar{v} \in \overline{V}, \quad r \in R \tag{2}$$

$$\sum_{i \in I} \sum_{e \in E^i} b^i_{e\bar{e}} \le b_{\bar{e}}, \quad \forall \bar{e} \in \overline{E} \tag{3}$$

Constraints (2) and (3) are physical server and link capacity constraints, respectively; they can be used to guarantee that the total amount of required resources must neither exceed server resource capacity nor link bandwidth capacity.

In addition, the flow conservation constraints must be satisfied on each physical server. If a physical server does not host the source or destination node of a virtual link, then for

TABLE 1. Main notations used in our formulation.

Notation	Description
$\overline{V}$, $\overline{E}$	The set of physical servers and links
$\overline{E}$	The set of physical links
$\overline{BE}$	The set of links in backbone network
R	The set of resource types
I	The set of VDC requests
E^i, V^i	The set of virtual links and nodes of VDC i
G^k	The VDC k
$x^i_{v\bar{v}}$	A 0-1 variable to denote whether the virtual node v (i.e., VM v) of VDC i is embedded on the physical server $\bar{v}$
c^{ir}_v	The demand of resource r of VM v
$c^r_{\bar{v}}$	The capacity of resource r on the physical server $\bar{v}$
b_e	The amount of bandwidth required by virtual link e
$b^i_{e\bar{e}}$	A variable to denote the amount of bandwidth resources provided by physical link $\bar{e}$ for virtual link e of VDC i
$b_{\bar{e}}$	The available bandwidth capacity of physical link $\bar{e}$
$s_{\bar{v}\bar{e}}$, $d_{\bar{v}\bar{e}}$	0-1 parameters to denote whether $\bar{v}$ is the source or destination of physical link $\bar{e}$
s^i_{ve} , d^i_{ve}	0-1 parameters to denote whether v is the source or destination of virtual link e
$\widetilde{x}^i_{v\bar{v}}$	A 0-1 parameter indicates whether the VM v of VDC i can be embedded to server $\bar{v}$
$x^i_{v\bar{v}}$	A 0-1 variable to indicate whether the VM v of VDC i is embedded to server $\bar{v}$
$y_{\bar{v}}$	A 0-1 variable to indicate whether the physical server $\bar{v}$ is active
rr	The reliability requirement of VDC request
Ω	The set of VDCs whose reliability requirements have not been satisfied
π_i	The penalty of failing to satisfy the reliability requirement of i-th VDC
$\rho_{\bar{v}}$, $\rho_{\bar{e}}$	The energy costs of an active physical server $\bar{v}$ and link $\bar{e}$
λ_v , λ_e	The recovery costs (the costs on resource consumptions) of virtual node v and link e

that physical server, the incoming flow and outgoing flow of that virtual link should be equal. This can be formulated as in Constraint (4).

$$\sum_{\bar{e}\in E} s_{\bar{v}\bar{e}} b^i_{e\bar{e}} - \sum_{\bar{e}\in E} d_{\bar{v}\bar{e}} b^i_{e\bar{e}} = \sum_{v\in V^i} x^i_{v\bar{v}} s^i_{ve} b_e - \sum_{v\in V^i} x^i_{v\bar{v}} d^i_{ve} b_e, \quad \forall i \in I,\ e \in E^i,\ \bar{v} \in \overline{V}. \tag{4}$$

There are also some constraints need to be satisfied while performing the VM embedding. These constraints are used to guide embedding the VMs to appropriate physical servers. The VM placement constraints can be formulated as follows.

$$x^i_{v\bar{v}} \le x^i_{v\bar{v}}, \quad \forall i \in I,\ v \in V^i,\ \bar{v} \in \overline{V} \tag{5}$$

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

Constraint (5) guarantees that a virtual machine only can be embedded on the physical server that it can be placed on. Constraint (6) makes sure that one VM only can be embedded on one physical server.

If a physical server hosts at least one VM, then this server must be active, otherwise inactive. This means that the following constraints must be met.

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

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

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

Constraint (7) ensures that only when a server is active, then it is valid to assign a VM onto this server. Constraints (8) and (9) denote that if a physical link provide resources to a virtual link, then the source and destination node (i.e., server) of this physical link must be active.

Furthermore, the following location constraints must be satisfied:

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

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

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

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

Equation (12) denotes that the VMs only be embedded in the data center that the virtual machines can be embedded in. And Equation (13) guarantees that one VM only can be embedded in one data center.

Moreover, the following VDC reliability requirement must be satisfied.

$$rl \ \ge \ rr \tag{14}$$

where rl defined in Equation (1), which denotes the reliability of an embedded VDC. Constraint (14) ensures that the reliability of an embedded VDC must be no less than its reliability requirement.

B. THE OBJECTIVE

We use rr to represent the reliability requirement of i-th VDC. If the InP fails to meet the reliability requirement rr, there will be a penalty. The penalty for an InP can be calculated as in Equation (15).

$$P^{unreliable} = \sum_{i\in\Omega} \pi_i, \tag{15}$$

where $\pi_i$denotes the penalty of failing to satisfy the reliability requirement of i-th VDC.

Recovery costs come from restarting VMs and reconfiguring the network equipment. We thus define the failure

recovery costs of physical server $\bar{v}$ and physical link $\bar{e}$ as shown in Equations (16) and (17), respectively.

$$P_{\bar{v}}^{recovery} = \rho_{\bar{v}} + \sum_{i\in I}\sum_{v\in V^i} x_{v\bar{v}}^i \lambda_v + \sum_{e\in E^i} b_{e\bar{e}}^i \mu_{\bar{v}\bar{e}} \lambda_e, \quad \forall \bar{v} \in \bar{v} \tag{16}$$

$$P_{\bar{e}}^{recovery} = \rho_{\bar{e}} + \sum_{i\in I}\sum_{e\in E^i} b_{e\bar{e}}^i \lambda_e, \quad \forall \bar{e} \in \overline{E}\backslash\overline{BE} \tag{17}$$

where λ_v and λ_e are the recovery costs of virtual node v and virtual link e; and $\mu_{\bar{v}\bar{e}} = max\{s_{\bar{v}\bar{e}}, d_{\bar{v}\bar{e}}\}$. The cost for embedding e onto backbone network can be defined as in Formula (18).

$$P_{\bar{e}}^{backbone} = \sum_{e\in E^i} b_{e\bar{e}}^i \zeta_{eb}, \forall \bar{e} \in \overline{BE}, \tag{18}$$

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

Then the total cost caused by unreliability is as follows:

$$P_R = P^{unreliable} + \sum_{\bar{v}\in\overline{V}} P_{\bar{v}}^{recovery} + \sum_{\bar{e}\in\overline{E}} P_{\bar{e}}^{recovery} + P_{\bar{e}}^{backbone}. \tag{19}$$

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

$$Minimize\ P_R \tag{20}$$

V. ALGORITHM DESIGN

Since the problem of optimal VDC provisioning described in Section 3 is NP-hard, in this section, we propose an efficient heuristic 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; and *ii*) how to reduce the bandwidth consumption of backbone network.

Multiple VMs may be embedded on the same physical server, hence the service reliability of embedding the VMs from different replication groups of a VDC on the same physical server is higher than that of embedding the VMs on different servers. The reason is as follows: according to the definition of reliability of a VDC, reliable service implies that all replication groups are reliable. All replication groups form the complete service, and each group plays 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 server, the service reliability is equal to the reliability of that physical server. If we embed VMs on different physical nodes, the service reliability is equal to the product of the reliabilities of those physical servers. Furthermore, in order to improve the reliability of service, we have to embed VMs in the same replication group on to different physical servers so that these VMs can backup for each other. However, physical servers with limited resources and VMs with location constraints may not allow all VMs in a VDC to be embedded onto the same physical server, nor even the same data center.

Moreover, reducing the bandwidth consumption of backbone network is the other issue to be addressed in this paper. For addressing this issue, we partition a VDC into several partitions before embedding it. The reasons are as follows: *i*) if we embed VMs one by one, it will consume large amount of bandwidth of backbone network, because the bandwidth consumption of backbone network is not considered in the VM embedding process; *ii*) per unit bandwidth resource in inter-data center is more expensive than that of intra-data center network. We put the VMs that have large amount of communication bandwidth requirement between each other into the same partition, and the VMs in the same partition will be embedded into the same data center. Thus, bandwidth consumption of backbone network can be reduced. On the other hand, the way of embedding VMs one by one mentioned above will result in a much higher reliability than required. It is unnecessary that the data centers provide much higher reliability than that required by VDC request, so it just needs to satisfy the reliability requirements of virtual data centers for reducing the resource consumptions.

In addition, in order to use physical server resource with different reliabilities rationally, we group the servers in data center according to their reliabilities.

Therefore, the RVDCE algorithm consists of three main steps: *i*) *group the physical servers; ii*) *partition the VDC*; and

Algorithm 1 RVDCE Algorithm

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

iii) *embed the VDC partitions*. *Algorithm* 1 shows the pseud code of RVDCE algorithm.

A. GROUP PHYSICAL SERVERS

Due to resources are limited in physical data center, in order to improve acceptance ratio of VDC requests, we should embed the VMs on the servers with low reliability. If we map VMs with low reliability requirements on the servers with high reliability, the physical server resources could be waste resulting in lower VDC acceptance ratio. Therefore, in order to use physical server resource with different reliabilities rationally, we sort the servers in descending order of their reliabilities, and then group the servers into groups based on their reliability levels that servers with higher level have higher reliabilities. In the embedding process, for example, 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 the requirement after VDC embedding, it is necessary to increase the reliability level l and re-embed the VDC. If the reliability is lower than the required reliability for any available physical servers, we have to change the partition size and re-embed the VDC.

B. PARTITION A VDC REQUEST

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 smaller than $K$and can be adjusted (i.e., there are at most K VMs in a partition). 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 $K = N$, the bandwidth consumption of backbone network is minimized. Otherwise, reduce Kuntil the VDC is embedded successfully.

In the VDC partition process, we first make each virtual node (i.e., VM) as a partition, and calculate the total amount of bandwidth demands between the partitions. Then the algorithm traverses each VM $v \in V^i$, and finds partition P that allows us to move v from its original partition to P and satisfy the follow conditions: *i*) reduce the amount of bandwidth consumption in backbone network; *ii*) the VMs in P have same location constraint; *iii*) the number of VMs in P does not exceed K. If partition P meets the above conditions, we can move VM v to P. As long as there are VMs moving between different partitions, algorithm keeps traversing the VMs until no VM needs 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.

Figure 4 shows an example for partitioning a VDC request. We assume that there are two data centers 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

Procedure 2 VDC Partition

```
Let flag ← true;
while (flag)
    Denote each node (i.e., VM) of G^i as a partition;
    Record the total bandwidth demands between
     partitions;
    whileVMs need to be moved between partitionsdo
     for each v ∈ V^i, do
          Find a partition P and move v to P, such that:
          a) the number of VMs in P does not exceed
K;
          b) total bandwidth consumption is minimized;
          c) all of the VMs inP have same location
             constraint.
     end for
    end while
if current bandwidth demands < initial bandwidth
 demands
          Change G^i to be the graph of partitions;
    else
          flag ← false
    end if
end while
```

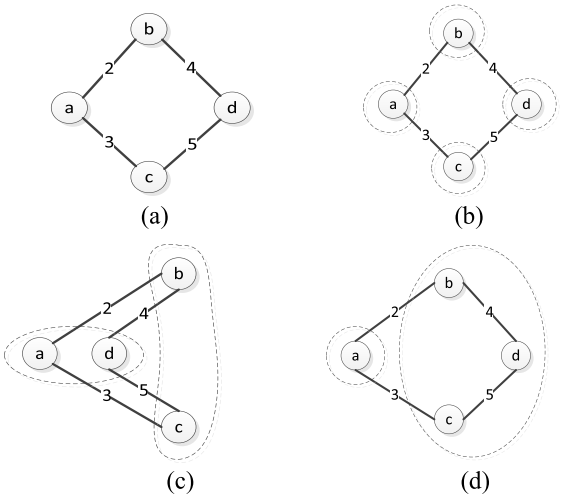


FIGURE 4. Example of partitioning a VDC. (a) Original VDC request *m* (b) The partitioning for *K*=1, 2 or 3 (c) The partitioning for *K*= 2 or 3 (d) The partitioning for *K*= 3.

VMs are as follows: *i*) VM a need to be embedded in $DC1$; *ii*) VM b and VM c need to 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), Figure 4(c) and Figure 4(d).

C. EMBED THE VDC 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 VM in a replication group as the working VM, the other VMs in that replication group can be used for backup. We take different approaches for embedding the working VM and backup VMs.

1) EMBED THE WORKING VMs

Since the VMs in partition have location constraints, each partition has a corresponding location constraint. Partitions can only be embedded on to the data centers that meet the location constraints of VMs. Assuming the size of partition is K, we map VMs on the level l servers. If VDC reliability cannot meet the reliability requirement under the partition size K, the algorithm will increase the level of the selected servers.

We randomly choose a partition 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 available resources of the physical server are not enough, we need to embed the VM on a ''new'' available server with the highest reliability.

After embedding one partition, we choose the next partition that has the maximum bandwidth with the embedded partitions in the set of remaining partitions need to be embedded, until all partitions are embedded.

2) EMBED THE BACKUP VMs

We calculate reliability of VDC according to Equation (1), after embedding the VM. If the reliability does not meet the requirement, we adjust the physical servers with lower or higher reliability until it meets the requirement.

Note that all of the VMs belonging to 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 hosted the VMs belonging to the same partition. In addition, it is important to note that the VMs belonging to same replication group cannot be embedded on the same server. After embedding the backup VMs, we calculate the reliability and check whether the reliability meets the reliability requirement of VDC.

D. COMPLEXITY ANALYSIS

The proposed RVDCE algorithm consists of Procedure 1 and Procedure 2. We analysis the complexity of our proposed RVDCE algorithm as follows:

(1) The complexity of Procedure 1 is $O(|V^i| \times |V^i|)$, where $|V^i|$ is the number of VMs in *i-th* VDC.

(2) The complexity of Procedure 2 is also $O(|V^i| \times |V^i|)$.

Procedure 3 Partition Embedding

```
Randomly choose a partition Q from partition set Partitions;
Choose a data center dc has the highest reliability and enough resources for the VDC partition from data center set DCs;
PartitionToDC = PartitionToDC ∪ < Q, dc >;
M: the set of VMs that have been embedded, let M = φ;
while Partitions is not empty do
    for all v in Q do
        if none of VM in the replication group that v belongs to has been embedded
            if v can be embedded on the server s ∈ S hosting the VMs in M
                Embed v on server s;
            else
                Choose the server with highest reliability that belongs to dc and s for hosting v;
                M ← M ∪ {v};
            end if
        end if
    end for
    Partitions ← Partitions\{Q};
    Q ← the partition inPartitions which has the largest amount of bandwidth demand to communicate with the embedded partitions;
    Choose dc that can provide highest reliability and enough resources for Q from DCs;
    PartitionToDC = PartitionToDC ∪ < Q, dc >;
end while
Compute the current reliability rl of VDC request according to Equation (1);
for all < Q, dc > in PartitionToDC do
    if rl > rr //rr is the reliability requirement of VDC
        for each VM (denoted as v) in Q do
            Embed v on the server with enough resources and the lowest reliability in dc;
        end for
    else
        for remaining VMs in Q do
            if v can be embedded on the server that VM u in Q has been embedded on and the replication groups that v and u belong to are different
                Embed v on s;
            else
                Embed v on the server with highest reliability;
            end if
        end for
    end if
end for
```

Therefore, the complexity of our RVDCE algorithm is $O(K \times (|V^i| \times |V^i| + L \times |V^i| \times |V^i|)) \approx O(K \times L \times |V^i|^2)$, where K is the size of partitions and$L$ is the number of levels of servers.

VI. SIMULATION AND ANALYSIS

A. SIMULATION ENVIRONMENT

In our simulations, we use the NSFNET [34] as the backbone network, there are six data centers attached to the NSFNET as shown in Figure 5. The bandwidth capacity of a link in backbone network is 100 units. Each data center has a Fat-Tree [31] topology which is shown in Figure 2. We assume that each physical server has 32 CPUs. We refer to the existing work [35] to set the bandwidth capacity of the intra-data center links. In each data center, the bandwidth capacity of each physical link which directly connecting server is 10 units and the bandwidth capacity of each switch-to-switch physical link is 40 units.

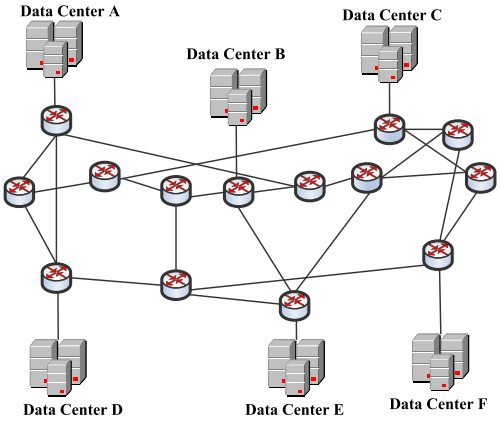


FIGURE 5. The physical infrastructure.

We generate the VDC requests by using GT-ITM [36] and the parameter settings are similar with [5]. We consider two cases in our simulations, i.e., *Case* 1 and *Case*2. The number of VMs in *Case*1 is randomly set from 3 to 30. The number of VMs in *Case* 2 is randomly set from 3 to 15. The CPU resource demand of each VM is generated randomly from 8 to 16 units. The bandwidth requirement of each virtual link is randomly set from 1 to 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 2.

B. SIMULATION EXPERIMENT RESULTS

Figure 6 shows the performance of bandwidth consumption of backbone network for provisioning VDC requests. The K in Figure 6(a) and Figure 6(b) represents the size of the partition. And *rr* indicates the reliability requirement of VDC request. It can be seen that the RVDCE algorithm results in a lower bandwidth consumption in backbone network compared to that of RVNE and RVDCE_R. This is because RVDCE divides a VDC requests as multiple partitions and consumes the bandwidth of backbone network as few as possible while embedding these partitions.

For example, in Figure 6(c) and Figure 6(d), the RVDCE algorithm always leads to a significant lower bandwidth

TABLE 2. 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 [13] for reliable virtual network embedding across multiple domains.

consumption of backbone network compared to that of RVNE and RVDCE_R, whatever the reliability requirement *rr* is. Similarly, it can be seen in Figure 6(a) and Figure 6(b), our RVDCE algorithm always leads to a lower backbone bandwidth consumption compared to that of RVNE and RVDCE_R, whatever the partition size K is. Furthermore, bandwidth consumption of backbone network of RVDCE decreased with the growth of the value of K. Since larger partition size may lead to lower bandwidth consumption between partitions.

Figure 7 shows the performance of blocking ratios of VDC requests under various partition sizes or reliability requirements. It is clear that when the number of VDCs is small, the blocking ratios of these three algorithms are very low. For example, in Figure 7(a), when the number of the VDC is less than 20, the blocking ratios of these algorithm are zero. The blocking ratio increased with the growth of the number of VDC request. The blocking ratio of RVDCE is lower than that of the other two algorithms under different reliability requirements. It is due to the fact that RVDCE algorithm embeds VMs on servers with as lower reliability as possible while satisfying the reliability requirements of VDCs, for avoiding over provisioning and thus can admit more VDC requests. Specifically, in Figure 7(a) and Figure 7(b), the RVDCE algorithm always leads to a lower blocking ratio compared to that of RVNE and RVDCE_R, whatever the value of K is. Similarly, as shown in Figure 7(c) and Figure 7(d), the RVDCE algorithm always leads to a lower blocking ratio compared to that of RVNE and RVDCE_R, whatever the value of *rr* is.

Furthermore, it can be seen in Figure 7(c) and Figure 7(d), blocking ratio of RVDCE is increased with the growth of the value of *rr*. This is because that embedding a VDC with higher reliability requirement need to consume more resources.

Figure 8 presents the result of total CPU resource consumptions under various number of VDC requests. The cumulative CPU resource consumption increased with the growth of the number of VDC requests. Furthermore, it can be seen from Figure 8 that the CPU resource consumption of RVDCE is lower than that of RVNE. This is because that our

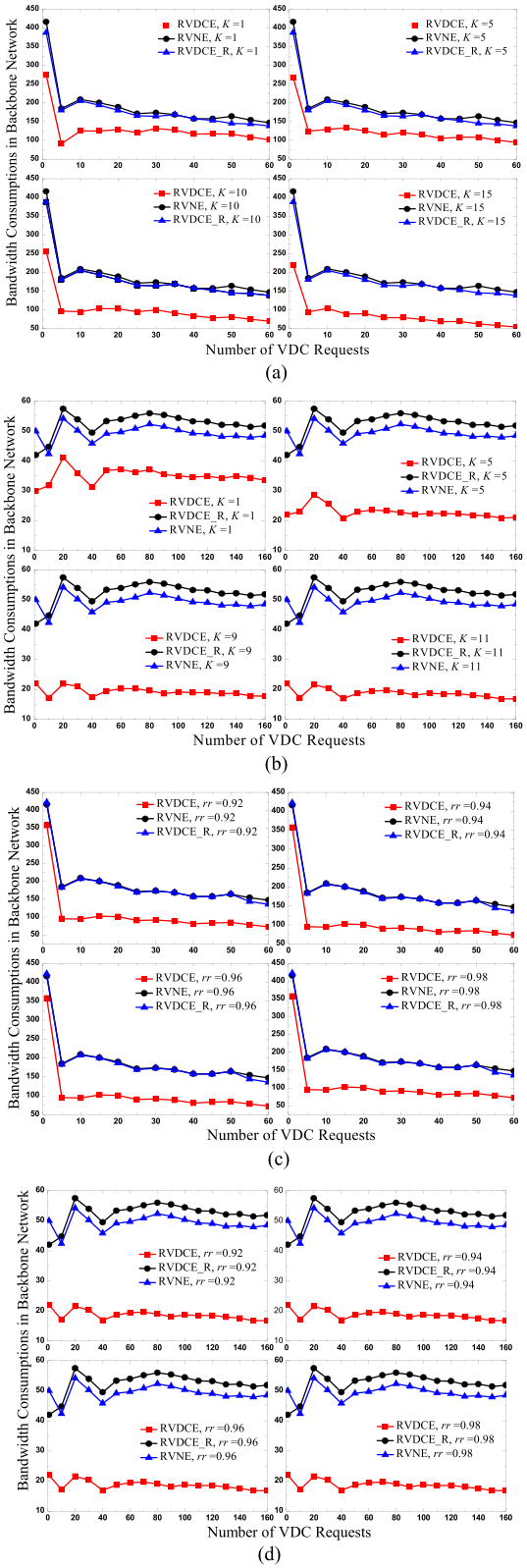


FIGURE 6. Bandwidth consumption of backbone network. (a) Simulation results for *Case* 1 (b) Simulation results for *Case* 2 (c) Simulation results for *Case* 1 (d) Simulation results for *Case* 2.

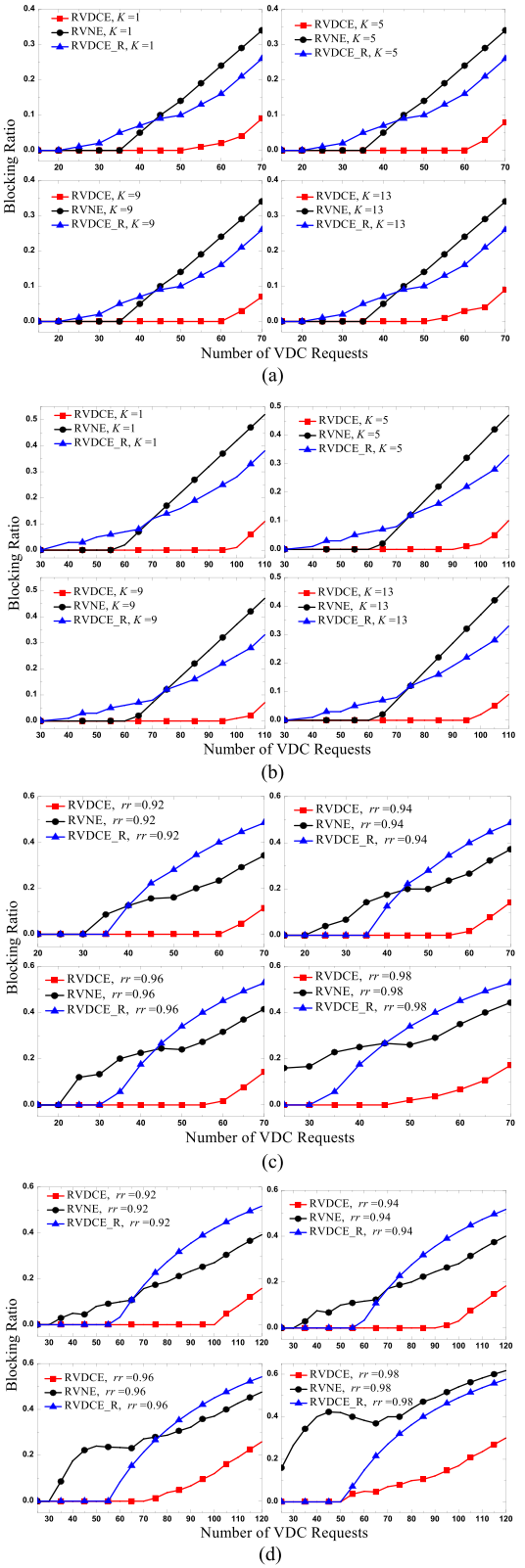


FIGURE 7. Blocking ratios under different reliability requirements. (a) Simulation results for *Case* 1 (b) Simulation results for *Case* 2 (c) Simulation results for *Case* 1 (d) Simulation results for *Case* 2.

RVDCE algorithm does not use redundant resources as the backups for satisfying the reliability requirements of VDCs. Furthermore, our proposed algorithm selects the servers to host the VMs considering the dependences between VMs, thus avoiding over provisioning and results in lower resource consumption.

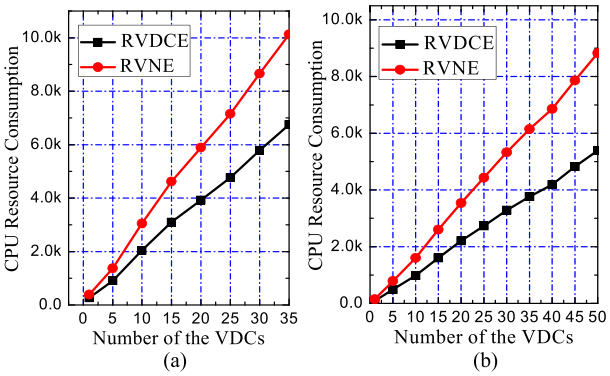


FIGURE 8. Total CPU resource consumptions. (a) Simulation results for *Case* 1 (b) Simulation results for *Case* 2.

VII. SECURITY AND PERFORMANCE

In order to achieve the required key parameters of QoS-aware framework with security, resiliency, reliability, latency and bandwidth, we need to incorporate QoS-aware strategies with our heuristic reliable VDC embedding algorithm (RVDCE). Herewith we presented the experimental results for the development of VDC as follows.

A. EXPERIMENTS WITH REGARD TO INTEGRATED SECURITY PROCESS AND MODEL

Integrated security model with relevant process to implement the security techniques is paramount in achieving QoS-aware VDCs [37] on the rationale of building an integrated security to minimize security vulnerabilities and hacking in VDCs. The proposal is based on integrated security model and associated implementation process. Therefore, in this simulation experiment, we **use** integrated security techniques such as integrated security layer composed by firewall, identity management and access control monitoring framework, and performed **a** large scale penetration testing on VDCs to test the validity, robustness and resiliency of the security solution [29]–[37]. Since VDCs can be used and be independent of the locations, VDCs have been implemented in London and Southampton to perform the test. Each VM contains 100 GB disk space and each VDC can contain up to 10 TB. Three layers consist of (*i*) firewall; (*ii*) identity management and (*iii*) encryption. We ensure that all VDC, including all the VMs, have three layers of protection. The 2014 known-vulnerabilities, including 10000 common viruses have been used for large scale penetration testing to test the robustness of the VDC security solution. Figure 9 (a) shows the number of viruses/Trojans blocked by the integrated security model. 5278 viruses have been detected and blocked by the firewall. Another 3744 viruses/trojans have been detected and blocked by identity management and intrusion prevention systems. 838 viruses/trojans are then have been detected and blocked by the encryption. All detected viruses/trojans have been killed.

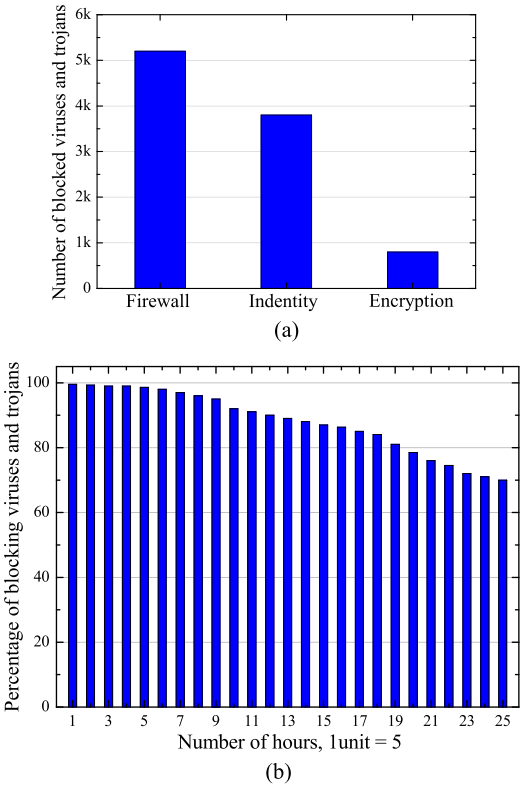


FIGURE 9. Integrated security penetration tests with RVDCE. (a) Number of viruses blocked by each layer (b) Percentage of blocking viruses and Trojans.

With regard to penetration tests, firstly, we define the Penetration Test Efficiency (PTe) as in Formula (21).

$$PTe = \left(\sum_1^{TN} VS + \sum_1^{TN} TR\right) \Big/ TN \times 100\% \qquad (21)$$

where *VS* denotes the number of virus have been detected and blocked, TR denotes the number of Trojans have been detected and blocked, and *TN* is the total number of virus and Trojans.

Secondly, we define the Security Test Efficiency (STe) as in Formula (22).

$$STe = \left(\sum_1^{TN} SAs \Big/ \sum_1^{TN} SIs\right) \times 100\% \qquad (22)$$

where *SAs* represents the number of surface attacks have been detected, blocked and killed, *SIs* represents the total number of system surface interfaces.

Thirdly, we can calculate the Business Process Efficiency (BPe) according to Formula (23).

$$BPe = \left(PTe \times BPN \Big/ Hr\right) \times 100\% \qquad (23)$$

where *BPN* denotes the total number of business process, and *Hr* denotes the total number of penetration test hours.

In our tests, for example, the total number of viruses and Trojans is 10000 and the total number of detected and blocked viruses and Trojans is 9860; total number of detected, blocked

and killed surface attacks is 9868 and the total number of system surface interfaces is 10000, hence we have,

$$PTe = (9860/10000) \times 100\% = 98.60\%,$$
$$STe = (9868/10000) \times 100\% = 98.68\%.$$

The percentage of maintaining good and protected data is important. Figure 9(b) shows the continuous ethical hacking for 125 hours to test how resilient and robust the VDC security solution is. At the end of a 125-hour attack, the percentage of blocking has dropped to 69.00%, hence the business process efficiency is (69.00% *125 / 125) = 69.00%.

B. RELATONSHIP BETWEEN PERFORMANCE, ENERGY AND COST

The performance is based on the blocking ratio and bandwidth consumption of backbone network tested in different scenarios in Section 5. Bandwidth consumption of backbone network should be the lower the better, since the large amount of network resource consumption can prolong network speed and job delivery. Similarly, the blocking ratio should be the lower the better to minimize the impact caused by bandwidth consumption of backbone network. Figure 8 shows that the CPU resource consumption is directly proportional to the number of VDCs and similarly, the energy consumption (Ec) is equivalent to the multiplication of CPU resource consumption (Rc) and time (T) in Formula (24).

$$Ec = Rc \times T \tag{24}$$

Running VDCs means experiments can be done via private clouds or service providers. In this case, it is the use of private clouds. The energy cost (Cst) of running VDCs is equivalent to energy consumption (Ec) multiplying the price (Pr) of per unit energy [38]. Thus, the cost can be calculated according to Formula (25).

$$Cst = Ec \times Pr \tag{25}$$

In order to measure energy consumption, the execution time to launch and run VDC has been recorded five times and taken the average values while the number of VDC has been increased from 10 to 100. The experiment settings are identical to infrastructures described in [38], whereby physical data centers in London and Southampton can host 100 VDCs at each site. To ensure all results can be synchronized without the impacts to the QoS, all experiments were conducted at the same time.

Figure 10 shows the mean execution time for running VDCs, which are consistent with results conducted in London and Southampton. All the time taken is increased in a linear regression method, starting from 45 seconds for 10 VDCs to 532 seconds for 100 VDCs. The execution time for running 35 VDCs and 50 VDCs is 163.5 and 238 seconds (i.e., 0.0454 and 0.0661 hours), respectively. The electricity price is £0.115 per Kilo Watt, or US$0.1495 (taking £1 = US $1.300 on August 18, 2016).

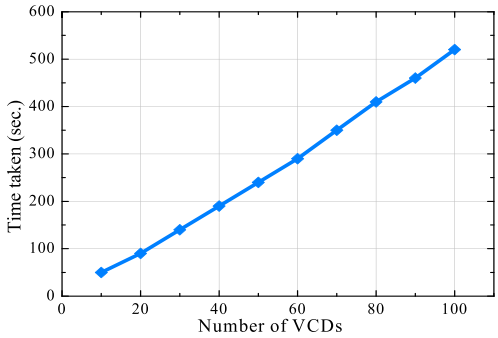


FIGURE 10. The execution time for running VDCs.

Referring back to Figure 8(a), the CPU resource consumption will take 10000 Watts (10 kw) and 6900 Watts (6.9 kw) by RVNE algorithm and RVDCE algorithm, respectively. Therefore, the energy consumption for RVNE is: 10 × 0.0454 = 0.454 kwh, and the energy consumption for RVDCE is: 6.9 × 0.0454 = 0.3132 kwh. Per hour running energy costs of RVNE and RVDCE are 0.4540 × 0.1495 = US $0.0679 and 0.3132 × 0.1495 = US $0.0468, respectively.

Referring to Figure 8(b), the CPU resource consumption will take 8700 Watts (8.7 kw) and 5700 Watts (5.7 kw) for 0.0661 hours of execution time, with the unit price of US$0.1495. In other words, the energy consumptions for RVNE and RVDCE are 8.7 × 0.0661 = 0.5751 kwh and 5.7 × 0.0661 = 0.3768 kwh, respectively. Per running energy costs of of RVNE and RVDCE are 0.5751 × 0.1495 = US $0.0860 and 0.3768 × 0.1495 = US $0.0563, respectively.

The difference between [38] and this paper is that running on [38] is on both physical and virtual systems and also each time the energy consumption can be obtained at the end of each service. In this paper, we can demonstrate that energy consumption can be calculated by multiplying CPU resource consumption and execution time. Prices can be calculated by multiplying the energy consumption and the price of per unit energy. The costs are very low with a low execution time, which can make the private running and management of VDCs economical and effective. However, costs do not include buying of the actual servers, resources maintenance of data centers, which are not within the remit of our work. In comparison to [39], authors develop two algorithms to reduce energy costs while running VMs. They have tested up to 100 virtual machines in their data center. Such measurements should be taken while VMs are utilizing large amount of energy consumptions in situations such as protecting VDCs under security attacks in real time or running services at full scales. Our energy consumption test was performed when all VDCs and VMs were in full utilization of resources for at least 125 hours with low costs achieved.

VIII. CONCLUSION

A large number of business, services and applications have been deployed on the cloud. Cloud providers take the advantage of the worldwide market to deploy their geographically

distributed infrastructures and enlarge their coverage. Therefore, the QoS-aware virtual data center provisioning in distributed infrastructures is particularly appealing for SPs as well as InPs.

In this paper, we study the problem of QoS-aware VDC provisioning and heuristically embedding algorithm for solving this problem in cloud computing. This research has also proposed an integrated security model to simulate the performance and resiliency of the proposed RVDCE algorithm to minimize security vulnerabilities seen in other proposals. This research aims at minimizing the total bandwidth consumption in backbone network for provisioning a VDC request, while satisfying the SLA requirements (such as reliability, access location constraints and resource requirements). These key SLAs are the main aspect of achieving QoS-aware efficient cloud resource provisioning requirement as part of the cloud users' perspective. Our algorithm can make a trade-off between bandwidth consumption in backbone network and reliability. Simulation results show that the proposed algorithm significantly reduced the resource consumption and blocking ratio than the existing approach does. Integrated security solutions have been demonstrated and our research contributions have been supported.

REFERENCES

[1] G. Sun, D. Liao, V. Anand, D. Zhao, and H. Yu, "A new technique for efficient live migration of multiple virtual machines," *Future Gener. Comput. Syst.*, vol. 55, pp. 74–86, Feb. 2016.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] Y. Xia, X. Sun, S. Dzinamarira, D. Wu, X. S. Huang, and T. S. E. Ng, "A tale of two topologies: Exploring convertible data center network architectures with flat-tree," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 295–308.

[4] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.

[5] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual data center embedding across distributed infrastructures," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 36–49, Jan./Jun. 2013.

[6] G. Sun, G. Zhu, D. Liao, H. Yu, X. Du, and M. Guizani, "Cost-efficient service function chain orchestration for low-latency applications in NFV networks," *IEEE Syst. J.*, to be published. doi: 10.1109/JSYST.2018.2879883.

[7] Y. Xiao, X. Du, J. Zhang, and S. Guizani, "Internet protocol television (IPTV): The killer application for the next generation Internet," *IEEE Commun. Mag.*, vol. 45, no. 11, pp. 126–134, Nov. 2007.

[8] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. ACM Co-NEXT*, 2010, Art. no. 15.

[9] M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC planner: Dynamic migration-aware virtual data center embedding for clouds," in *Proc. IFIP/IEEE IM*, May 2013, pp. 18–25.

[10] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "VirtualKnotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," *Comput. Netw.*, vol. 67, pp. 141–153, Jul. 2014.

[11] B. Wang, X. Chang, and J. Liu, "Modeling heterogeneous virtual machines on IaaS data centers," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 537–540, Apr. 2015.

[12] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 289–297.

[13] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.

[14] G. Sun, D. Liao, S. Bu, H. Yu, Z. Sun, and V. Chang, "The efficient framework and algorithm for provisioning evolving VDC in federated data centers," *Future Gener. Comput. Syst.*, vol. 73, pp. 79–89, Aug. 2017.

[15] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, no. 5, pp. 36–42, Sep./Oct. 2015.

[16] R. Ranjan, L. Wang, A. Y. Zomaya, D. Georgakopoulos, X.-H. Sun, and G. Wang, "Recent advances in autonomic provisioning of big data applications on clouds," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 101–104, Jun. 2015.

[17] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, "Live migration for multiple correlated virtual machines in cloud-based data centers," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 279–291, Apr. 2018.

[18] D. Guo, J. Xie, X. Zhou, X. Zhu, W. Wei, and X. Luo, "Exploiting efficient and scalable shuffle transfers in future data center networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 997–1009, Apr. 2015.

[19] Y. Peng, K. Chen, G. Wang, W. Bai, Z. Ma, and L. Gu, "HadoopWatch: A first step towards comprehensive traffic forecasting in cloud computing," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 19–27.

[20] J. Luo, L. Rao, and X. Liu, "Temporal load balancing with ser-vice delay guarantees for data center energy cost optimization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 775–784, Mar. 2014.

[21] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance in a large-scale wireless sensor network: Measurement, analysis, and implications," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 186–197, Feb. 2015.

[22] G. Sun, Y. Li, Y. Li, D. Liao, and V. Chang, "Low-latency orchestration for workflow-oriented service function chain in edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 116–128, Aug. 2018.

[23] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. Wang, and G. Chen, "Reliable multicast in data center networks," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 2011–2024, Aug. 2014.

[24] R. Garg, M. Mittal, and L. H. Son, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Comput.*, pp. 1–15, 2019.

[25] A. Kobusińska and C.-H. Hsu, "Towards increasing reliability of clouds environments with RESTful Web services," *Future Gener. Comput. Syst.*, vol. 87, pp. 502–513, Oct. 2018.

[26] L. Yu, T. Jiang, and Y. Cao, "Energy cost minimization for dis-tributed Internet data centers in smart microgrids considering power outages," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 120–130, Jan. 2015.

[27] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Gener. Comput. Syst.*, vol. 91, pp. 347–360, Feb. 2019.

[28] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[29] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Trans. Services Comput.*, vol. 9, no. 1, pp. 138–151, Jan. 2016.

[30] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[31] G. S. Malik and N. Kapre, "Enhancing butterfly fat tree NoCs for FPGAs with lightweight flow control," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2019, p. 308.

[32] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM*, 2011, pp. 350–361.

[33] X. Li, H. Wang, S. Yi, S. Liu, L. Zhai, and C. Jiang, "Disaster-and-evacuation-aware backup datacenter placement based on multi-objective optimization," *IEEE Access*, vol. 7, pp. 48196–48208, 2019.

[34] (2013). *The National Science Foundation Network (NSFNET)*. [Online]. Available: http://www.nsf.gov

[35] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, 2013.

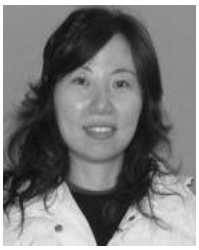
[36] GT-ITM. Accessed: Jul. 26, 2000. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[37] M. Ramachandran, "Integrated security process improvement framework for systems and services," *Int. J. Syst. Service-Oriented Eng.*, vol. 4, no. 1, pp. 53–67, 2014.

[38] V. Chang and G. Wills, "A model to compare cloud and non-cloud storage of Big Data," *Future Gener. Comput. Syst.*, vol. 57, pp. 56–76, Apr. 2016.
[39] X. Dai, J. M. Wang, and B. Bensaou, "Energy-efficient virtual machines scheduling in multi-tenant data centers," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 210–221, Jun. 2016.

GANG SUN is currently an Associate Professor of computer science with the University of Electronic Science and Technology of China. His research interests include network virtualization, cloud computing, high-performance computing, parallel and distributed systems, ubiquitous/pervasive computing and intelligence, and cybersecurity. He has coauthored 80 technical publications, including a paper in refereed journals and conferences, invited papers and presentations, and book chapters. He has also edited special issues at top journals, such as the *Future Generation Computer Systems and Multimedia Tool and Applications*. He has served as a Reviewer for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE COMMUNICATIONS LETTERS, the *Information Fusion*, the *Future Generation Computer Systems*, the *Journal of Network and Computer Applications*, the *Journal of Supercomputing*, the *Journal of Parallel and Distributed Computing*, the *KSII Transactions on Internet and Information Systems*, the *Computers and Electrical Engineering*, and the *Chinese Journal of Electronics*.

ZHU XU is currently pursuing the master's degree in communication and information systems with the University of Electronic Science and Technology of China. His research interests include network virtualization and algorithms.



HONGFANG YU received the B.S. degree in electrical engineering from Xidian University, in 1996, and the M.S. and Ph.D. degrees in communication and information engineering from the University of Electronic Science and Technology of China, in 1999 and 2006, respectively. From 2009 to 2010, she was a Visiting Scholar with the Department of Computer Science and Engineering, University at Buffalo (SUNY). Her research interests include network survivability, network security, and the next-generation Internet.

VICTOR CHANG is currently an Associate Professor with the Xi'an Jiaotong-Liverpool University. He is a leading expert on big data/cloud/security, a Visiting Scholar/Ph.D. Examiner at several universities, an Editor-in-Chief of the IJOCI and OJBD journals, an Editor of FGCS, the Founding Chair of two international workshops and founding Conference Chair of the IoTBD 2016 (www.iotbd.org) and COMPLEXIS 2016 (www.complexis.org). He has given 14 keynotes in international conferences and has received numerous awards, since 2011, including a 2016 European award: Best Project in Research, the Outstanding Young Scientist 2017, the INSTICC Service award, and the ICGI 2017 Special Award.

XIAOJIANG DU received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland College Park, in 2002 and 2003, respectively. He is currently a Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His research interests are security, wireless networks, and systems. He has authored over 250 journal and conference papers in these areas, as well as a book published by Springer. He is a Life Member of the ACM. He has been awarded over five million US dollars research grants from the US National Science Foundation (NSF), Army Research Office, Air Force, NASA, the State of Pennsylvania, and Amazon. He has received the Best Paper Award at the IEEE GLOBECOM 2014 and the Best Poster Runner-Up Award at the ACM MobiHoc 2014. He has served as the Lead Chair of the Communication and Information Security Symposium of the IEEE International Communication Conference (ICC) 2015 and a Co-Chair of the Mobile and Wireless Networks Track of the IEEE Wireless Communications and Networking Conference (WCNC) 2015.

MOHSEN GUIZANI (S'85–M'89–SM'99–F'09) received the B.S. (Hons.) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor with the Computer Science and Engineering Department, Qatar University, Qatar. Previously, he has served in different academic and administrative positions with the University of Idaho, Western Michigan University, University of West Florida, University of Missouri-Kansas City, University of Colorado-Boulder, and Syracuse University. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is a Senior Member of the ACM. He is currently the Editor-in-Chief of the *IEEE Network Magazine*. He serves on the editorial boards of several international technical journals, and the Founder and Editor-in-Chief of the *Wireless Communications and Mobile Computing journal* (Wiley). He is the author of nine books and over 500 publications in refereed journals and conferences. He has guest-edited a number of special issues in the IEEE journals and magazines. He also served as a member, Chair, and General Chair for a number of international conferences. Throughout his career, he has received three teaching awards and four research awards. He has also received the 2017 IEEE Communications Society WTC Recognition Award as well as the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor Networks. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He has served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer.