J Netw Syst Manage (2012) 20:97–115
DOI 10.1007/s10922-011-9209-x

# A Cost Efficient Design of Virtual Infrastructures with Joint Node and Link Mapping

**Hongfang Yu · Vishal Anand · Chunming Qiao · Hao Di · Xuetao Wei**

Received: 23 September 2010 / Accepted: 30 August 2011 / Published online: 11 September 2011

**Abstract** The virtualization of both servers and substrate networks is a promising technique that will enable the future Internet architecture to support a variety of cloud computing services and architectures. The problem of efficiently mapping a virtual infrastructure (VI) over a wide-area optical network is a key problem in provisioning Infrastructure (and Network) as a Service over multiple datacenters. In this paper, we study the problem of cost efficient VI mapping and propose a novel approach called the virtual infrastructure mapping algorithm (VIMA) that jointly considers node mapping and link mapping. Such a close coordination between the link and node mapping stages increases the efficiency of the VIMA algorithm. We compare the performance of our VIMA algorithm with other VI mapping algorithms

H. Yu · H. Di
School of Communication and Information Engineering, University of Electronic Science and Technology of China 2006, Xi Yuan Ave., Westn High-Tech Zone, Chengdu 611731, People's Republic of China
e-mail: yuhf.uestc@139.com

H. Di
e-mail: dihao@uestc.edu.cn

V. Anand (✉)
Department of Computer Science, The College at Brockport, 350 New Campus Dr., Brockport, NY 14420, USA
e-mail: vanand@brockport.edu

C. Qiao
Department of Computer Science and Engineering, University at Buffalo, State University of New York, Buffalo, NY 14260, USA
e-mail: qiao@buffalo.edu

X. Wei
Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA
e-mail: xwei@cs.ucr.edu

such as NSVIM, vnmFlib and R-ViNE under various performance metrics using simulation. Our simulation results and analysis show that the VIMA algorithm outperforms the other algorithms in terms of VI mapping costs and path length of VI links.

**Keywords** Virtualization · Virtual infrastructure · Node mapping · Link mapping

## 1 Introduction

Virtualization refers to a broad range of approaches for the abstraction of physical resources at different levels. With network virtualization [1], multiple overlay networks with diverse topologies, as well as bandwidth and resilience properties can be created using the same physical substrate network. Thus allowing multiple customizable (e.g., customized for a set of users, application, etc.) network architectures over a common physical infrastructure.

Since each one of these virtual network topologies uses the resources of the same underlying substrate network, it is essential to use these resources efficiently. Making an efficient use of the substrate resources requires effective techniques that map the virtual network on to the substrate network. Although any substrate network with sufficient resources can act as the substrate network, optical networks [2] are a natural choice for the substrate network because of their high speed, enormous bandwidth and protocol transparency.

Cloud computing is a new paradigm built upon (server) virtualization within each datacenter [3, 4]. Cloud computing is a pay-per-use model for enabling convenient, on-demand access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. It refers to both the applications delivered as a service and the hardware and systems software in the datacenters that provide these services. The cloud infrastructure typically includes the network, server, storage, software stacks and applications. With cloud computing the network becomes the most critical asset, thereby increasing the impact on network operators, who now have to build highly resilient, high-performance delivery architectures that can scale on-demand. Dynamic optical transport networks using Wavelength Division Multiplexing (WDM) with on-demand capacity provisioning become key in supporting the elastic nature of cloud computing applications. Furthermore, optical networks can also provide the scalability, availability, reliability, flexibility and agility required by cloud computing services cost-effectively. The Generalized Multiprotocol Label Switching (GMPLS) that extends the MPLS framework, can be used to provide the management and control of the optical network.

With the maturity of cloud computing and optical networking technology, in the near future we will be able to deploy a Federated Computing and Networking System (FCNS) [5], which interconnects a large number of computing *facility* nodes (e.g., clusters and datacenters) with optical networks, to support a variety of distributed applications and services across multiple datacenters. Note that although in this paper we will focus on resource provisioning within such a FCNS, many similar issues exist with a single datacenter and hence, some of the ideas and algorithms in the following discussions may also apply when it comes to resource provisioning with a datacenter.

A virtual infrastructure (VI) request consists of a set of VI nodes, with each node requiring some computing resources (e.g., CPU resource) at a *separate* computing facility node (i.e., no two VI nodes can use the computing resources at the same facility node, even if it has sufficient resources for both VI nodes). A VI node also needs to communicate with another VI node to send intermediate results, file data, or some other information. As a result, a VI request imposes strict connectivity requirements among the VI nodes in terms of topology, bandwidth, and delay guarantees to meet the service level agreements (SLA). Thus, given a VI request, a FCNS has to find a *one-to-one* mapping of the VI request onto the FCNS substrate. That is, assign each VI node to a computing facility node, hereafter simply called a facility node, and establish communication in the substrate network between these facility nodes.

Designing efficient VI mapping algorithms is challenging due to the highly coupled node mapping and link mapping operations. In recent years, several heuristics [6–11] have been proposed for different variants of the VI mapping problem. However, in most of the existing works, node mapping and link mapping are done in separate stages. Even when they are done in the same stage, they are loosely coordinated.

In this paper, we use a joint node and link mapping approach for the VI mapping problem and develop a virtual infrastructure mapping algorithm (VIMA). In our VIMA, we perform the node mapping and link mapping in the same stage, i.e., after determining the mapping of a VI node the mapping of the associated VI links is performed. The advantage of this single stage approach is that link mapping constraints and costs are taken into account at each step of the node mapping. Moreover, when selecting the node mapping we consider its effect on the future link mapping cost thus avoiding inefficient mappings. Our experimental evaluations show that our VIMA algorithm results in better performance.

The remainder of this paper is organized as follows. After giving an introduction to the problem in this section, we describe some of the relevant related work in Sect. 2. In Sect. 3 we describe our network model and the virtual infrastructure mapping problem. Section 4 describes our proposed heuristic algorithm and its analysis. In Sect. 5 we present the simulation model and the simulation parameters employed, and a discussion of the results of the performance evaluation of our heuristic algorithm. In Sect. 6 we conclude the paper and also give directions for future work.

## 2 Related Work

In recent years a number of heuristics have been proposed for solving different variants of the virtual network mapping problem. The work in [6] focuses on the off-line version where the virtual infrastructure traffic requests are known before hand, whereas [7–11] focus on the on-line version where the requests arrive according to a specific distribution. To reduce the problem complexity the work in [6–8] restricts the search space by separating the node and link mapping stages, e.g., handling the node mapping in the first stage and doing the link mapping in the second stage based on the shortest path, k-shortest paths and multi-commodity flow algorithms. The work in [6–8] also uses preselected node mappings without considering its relation to the link mapping stage. Although such approaches simplify the mapping problem and make it manageable, they result in poor performance.

The work in [9] developed an MILP formulation that coordinates the node and link mapping phases and presented two algorithms D-ViNE and R-ViNE based on LP relaxation with deterministic rounding and randomized rounding strategies, respectively. In the node mapping stage, the work in [9] relaxed the integer program to obtain a linear programming formulation that can be solved in polynomial time, and then used the deterministic (D-ViNE) and randomized (R-ViNE) virtual network embedding algorithms to approximate the values of the binary variables in the original mixed integer program (MIP). Once all the virtual nodes have been mapped, multi-commodity flow algorithms are used to get the link mapping solution.

Based on the work in [9, 10] proposed a mechanism to differentiate among resources based on their importance in the substrate topology and re-optimization. However, the integer linear program (ILP) relaxation weakens the coordination between the node and link mapping stages. The authors of [11] have recently proposed to map nodes and links during the same stage, and presented a backtracking algorithm based on a subgraph isomorphism. In this work the VI links are mapped onto paths shorter than a predefined distance (or hop) value. The authors propose two algorithms to determine the proper distance value, vnmFlib-simple with a fixed distance value and vnmFlib-advanced with an incremental distance value. The vnmFlib-advanced outperforms the vnmFlib-simple algorithm. However, both these approaches do not take the link mapping cost during the mapping of the nodes into account. Moreover, this work focuses more on finding an isomorphic subgraph to map the virtual infrastructure request but not on finding a cost-efficient solution. More recently the work in [12] studies the problem of survivability in virtual infrastructure design.

In this paper, we propose the VIMA heuristic that jointly does the VI node and link mappings such that the VI request is mapped on to the FCNS network while minimizing the cost.

## 3 Network Model and Problem statement

In this section we describe our network model and the problem statement. We use a similar framework as presented in our earlier work to model the VI mapping problem.

### 3.1 FCNS Substrate

We model the FCNS as an undirected graph $G_S = (V_S, E_S) = (V_F \cup V_X, E_S)$, where $V_S$ is the set of substrate nodes, and $E_S$ corresponds to the set of bidirectional fiber

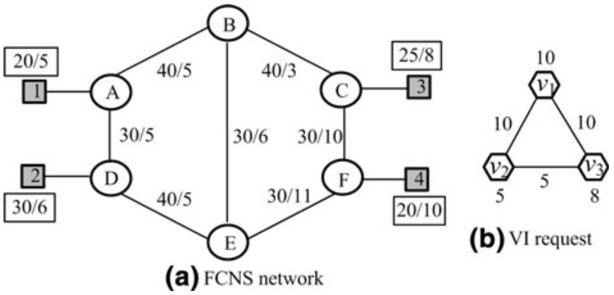

**Fig. 1** FCNS network and VI Request

links. Note that $V_S$ is composed of $V_F$ and $V_X$, where $V_F$ is the set of facility nodes and $V_X$ is the set of optical switches.

Each facility node $v \in V_F$ that can provide computing resources, is associated with a weight $c(v)$ that represents the available capacity of computing resources at facility node $v$. The cost of a computing resource unit on a facility node $v$ is $cf(v)$. For each link $e \in E_S$, the available bandwidth capacity is $w(e)$ and the cost of a unit of bandwidth is $cl(e)$. In this paper, we assume $c(v) = 1$ for all facility nodes, and $cl(e) = g$ for all fiber links $l$, where $g$ is the ratio of the unit cost of computing resources to that of the bandwidth resources.

Figure 1a shows a substrate network with 6 switching nodes A to F (unshaded circles) and 4 facility nodes 1–4 (shaded squares). The facility node 1 is connected to switching nodes A, 2 to D, 3 to C and 4 to F, respectively. It is the facility nodes that perform the actual computations, storage etc. The switching nodes are connected to each other by high bandwidth communication links, and the facility nodes are connected to the switch nodes through low(er) bandwidth access links. The numbers over the links represent the available bandwidth and the cost of a bandwidth unit, and the numbers in the rectangles next to the facility nodes represent the available computing resources and the cost of the computing resource at the facility nodes.

## 3.2 VI Request

A VI request with quality of service (QoS) requirements is modeled as an undirected graph $G_L = (V_L, E_L)$, where $V_L$ corresponds to the set of VI nodes, and $E_L$ denotes the set of bidirectional communication demands among the VI nodes.

Each VI node $v \in V_L$ requires a certain amount of computing resources for its execution, denoted by $\varepsilon(v)$. Each communication demand $e \in E_L$ has a bandwidth requirement, denoted by $b(e)$. Figure 1b shows one VI request with three VI nodes and three VI links, and associated computing and communication resource requirements. In addition we also consider a location constraint for the VI nodes that limits the set of substrate facility nodes that a particular VI node may be mapped on to. This is more practical since in reality there might be restrictions (e.g., delay, distance, security and administration constraints) on the mapping of VI nodes. Thus for each VI node $v \in V_L$, we use $\Omega(v)$ to denote the set of possible substrate nodes that it may be mapped on to. If $\Omega(v) = V_F$, $\forall v \in V_L$, we call it unrestricted node mapping; otherwise, we call it restricted node mapping.

## 3.3 Virtual Infrastructure Mapping

A basic VI mapping solution includes (1) a one-to-one (but not onto) mapping from $V_L$ to $V_F$; for each VI node $n \in V_L$, its corresponding facility node is $\eta(n) \in V_F$, and (2) mapping of each VI link $(u, v)$ in $G_L$ to a path $P(\eta(u), \eta(v))$ in $G_S$. The number of hops in the path and associated costs are defined as $h_{\eta(u), \eta(v)}$ and $cp_{\eta(u), \eta(v)}$, respectively. Thus, $cp_{\eta(u), \eta(v)}$ is given by (1), which is the product of the path cost (i.e., sum of all unit cost links on the path) and the bandwidth requirement $b(u,v)$ of VI link $(u,v)$.

$$cp_{\eta(u),\eta(v)} = b((u,v)) \times \sum_{e \in P(\eta(u),\eta(v))} cl(e). \tag{1}$$

Figure 2 shows the result of the mapping of the nodes of a VI request onto the substrate nodes, $v_1 \rightarrow 1$, $v_2 \rightarrow 3$ and $v_3 \rightarrow 2$. Similarly the mapping of the links of the VI request is as follows: $(v_1,v_2) \rightarrow \{1\text{–}A\text{–}B\text{–}C\text{-}3\}$, $(v_1,v_3) \rightarrow \{1\text{–A–D–}2\}$ and $(v_2,v_3) \rightarrow \{3\text{–C–B–A–D–}2\}$.

## 3.4 Problem Statement

*Given*: a FCNS $G_S = (V_F, E_S)$ and a VI request $G_L = (V_L, E_L)$.

*Question*: how to jointly allocate computing and networking resources, such that the total cost of mapping of the VI request, i.e., sum of computation and communication cost is minimized?

The total cost $C$ is defined as in (2) below.

$$C = \sum_{v \in N_L} cn(\eta(v)) \times \varepsilon(v) + \sum_{e \in E_L} cp_e. \tag{2}$$

The amount of physical bandwidth required depends on the physical location of the VI nodes. Due to this tight coupling between VI node mapping and VI link

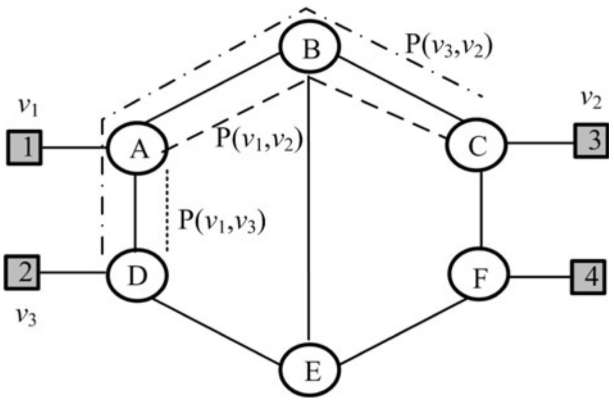

**Fig. 2** Example of a VI mapping

mapping, we need to develop a joint node and link mapping approach for the cost-efficient resource allocation.

## 4 Virtual Infrastructure Mapping Algorithm (VIMA)

It is worth noting that the problem of obtaining a minimum-cost mapping is NP-hard. In this section we present an improved algorithm, namely a VIMA for efficient mapping of VI requests on to the substrate network. In addition we also give the lower bound on the cost of the VI request mapping and the time complexity of our algorithms.

Since the proposed VIMA builds and improves on our earlier heuristic algorithm, called Non-Survivable VI Mapping (NSVIM), we will describe NSVIM in detail first.

### 4.1 Non-Survivable VI Mapping Algorithm (NSVIM)

In the NSVIM algorithm [13, 14], we map each VI node to a facility node and allocate the requested amount of computing resources on the facility node. Then we map each VI link to one or more fiber links in the FCNS, and allocate the requested bandwidth on the fiber links. The objective of NSVIM is to minimize the total cost, including computing costs and communication costs.

The main idea of the NSVIM algorithm is (1) node mapping and link mapping are done in the same stage; (2) the node mapping considers potential (future) link mapping costs. A detailed description of the NSVIM algorithm is given in Figure 3.

NSVIM uses two sets, $\Gamma^0$ for mapped VI nodes and $\Gamma^1$ for unmapped VI nodes, to keep track of mapped and unmapped VI nodes, respectively. In addition, it uses $\mathrm{T}^0$ for allocated facility nodes and $\mathrm{T}^1$ for unallocated facility nodes to keep track of allocated and unallocated facility nodes, respectively. Clearly, there should be a one-to-one and onto mapping between $\Gamma^0$ and $\mathrm{T}^0$ as each VI node is mapped to a unique facility node.

In NSVIM, we first sort the VI nodes in decreasing order of node degree. In Step 3.2, to select a candidate facility node $u \in \mathrm{T}^1$ to which the VI node can be mapped, we note that the amount of available computing resources at $u$ must be no less than the requested amount $\varepsilon(v)$. In addition, let $A$ $(v)$ be the subset of VI nodes that are adjacent to $v$ in $G_L$, and the maximum communication bandwidth requested between $v$ and each VI node in $A(v)$ be $\mathrm{B}(v)$. For the mapping to be feasible, the maximum amount of available link bandwidth on all links adjacent to $u$ should be no less than $\mathrm{B}(v)$. In the proposed NVSIM algorithm, we first select a candidate set $S$ of facilitate nodes that meet the above requirements on the available computing and networking (bandwidth) resources, and then choose a mapping that may result in a minimum cost.

Based on whether the VI node has been mapped, we divide A $(v)$ the set of VI nodes that are adjacent to $v$ in $G_L$, into the subsets $A^0$ $(v)$, the set of mapped VI nodes that are adjacent to $v$, and $A^1$ $(v)$, the set of unmapped VI nodes that are adjacent to $v$, respectively. The VI link between $v$ and VI nodes in $A^0$ $(v)$ is called the

```
NSVIM Algorithm
Input: a FCNS $G_S = (V_F, E_S)$, a VI request $G_L = (V_L, E_L)$
Output: mapping solution
Step 1: Initialization
Initialize the sets $\Gamma^0$, $\Gamma^1$, $T^0$ and $T^1$. Let $\Gamma^0 = T^0 = \Phi$, $\Gamma^1 = V_L$, and $T^1 = V_F$.
Step 2: Sorting the VI nodes
  Sort the VI nodes in decreasing order of node degree and store in list $\Lambda$.
Start with the first node in $\Lambda$, i.e., node $v_L$ with the largest degree.
Step 3: While ($\Gamma^1 != \Phi$) do
    3.1 select an unmapped VI node $v_L$
      Select the VI node from $\Gamma^1$.
    3.2 Determine the candidate facility node subset
      Find a subset of candidate facility nodes $S$ in $T^1 \cap \Omega(v)$ (as discussed
    as above). If $S = \Phi$, return FAILURE.
    3.3Assignment of $v_L$ to a facility node
      Assign $v_L$ to $v_F$ in $S$ with the minimum mapping cost $C(v \rightarrow u)$ (see
    Eq.2 and related discussion). Reserve $\varepsilon(v)$ computing resource on the
    facility node $u$.
    3.4 Assignment of $L^0(v_L)$ to substrate links
      For each determined VI link $e$ in $L^0(v)$ find the minimum cost path $P$
    in FCNS (see discussion below). If no such path can be found, return
    FAILURE. Otherwise, Reserve bandwidth $b(e)$ on every substrate link
    along the path $P$.
    3.5 Update $\Gamma^0$, $\Gamma^1$, $T^0$ and $T^1$
      Remove $v$ from $\Lambda$, move $v$ from $\Gamma^1$ to $\Gamma^0$, and $u$ from $T^1$ to $T^0$.
Step 4: return SUCCESS and output the mapping solution $M$
```

**Fig. 3** Pseudo code for the NSVIM algorithm

determined VI link; the VI link between $v$ and VI nodes in $A^0$ ($v$) is called the undetermined VI link. Similarly, we divide $L(v)$ the set of VI links connected with $v$ into the $L^0(v)$ and $L^1(v)$. Where $L^0(v)$ is the set of determined VI links between $v$ and VI nodes in $A^0$ ($v$), and $L^1(v)$ is the set of undetermined VI links between $v$ and VI nodes in $A^1$ ($v$).

The key idea of NSVIM is to select a facility node in $S$ such that the sum of the computing costs and the communication costs associated with the mapping of $v$ to $u$, given in (3) below, is minimized.

$$C(v \rightarrow u) = x(v \rightarrow u) + y^0(v \rightarrow u) + y^1(v \rightarrow u) \tag{3}$$

where x($v \rightarrow u$) is the computing cost which is equal to $cf(u) \times \varepsilon(v)$. $y^0(v \rightarrow u)$ and $y^1(v \rightarrow u)$ are the communication costs of the VI links in $L^0(v)$ and $L^1(v)$, respectively.

More specifically, $y^0(v \rightarrow u)$ is the sum of the path cost between $u$ and a subset of facility nodes in T$^0$ that have been allocated to the VI nodes in set $A^0$ ($v$) [see (4)]. On the other hand, $y^1(v \rightarrow u)$ is the sum of the potential (future) communication costs of undetermined VI links between $u$ and a subset of facility nodes in T$^1$that might be allocated to the unmapped VI nodes in $A^1$ ($v$). Since one end-node of the VI link is unmapped, we use the average value of the communication cost to estimate the cost of an undetermined VI link, as shown in given in (5) and (6).

$$y^0(v \rightarrow u) = \sum_{(v,v')\in L^0(v)} cp_{\eta(v),\eta(v')} = \sum_{v'\in A^0(v)} cp_{u,\eta(v')} \tag{4}$$

$$y^1(v \rightarrow u) = \sum_{(v,v')\in L^1(v)} \underline{cp}_{n(v),\eta(v')} = \sum_{v'\in A^1(v)} \underline{cp}_{u,\eta(v')} \tag{5}$$

$$\underline{cp}_{u,\eta(v')} = \sum_{\eta(v')\in T^1\cap\Omega(v')} cp_{u,\eta(v')}/|T^1 \cap \Omega(v')| \tag{6}$$

where $\underline{cp}_{u,\eta(v')}$ is the average cost for the undetermined VI link ($v$,$v'$). Since the end-node v′ is unmapped, it may be mapped onto any unallocated facility node in $\Omega(v')$. We use (6) to estimate the future cost of the VI link ($v$,$v'$).

The rationales behind including $y^1(v \rightarrow u)$ are (1) since NSVIM is a greedy algorithm that maps one VI node at a time, there is always a risk that a node $u$ that looks good now because for example, it has a lower sum of $x(v \rightarrow u)$ and $y^0(v \rightarrow u)$, may turn out to be a bad choice overall, and (2) to avoid choosing such a node $u$, by looking ahead at potential future communication costs to be incurred when additional facility nodes in $T^1$ are allocated to map the VI nodes in $A^1$ ($v$).

After a VI node mapping is done, its adjacent VI links whose two end-nodes are determined are mapped immediately. This is key, and leads to close coordination between the node mapping and link mapping.

## 4.2 Virtual Infrastructure Mapping Algorithm (VIMA)

We improve upon the NSVIM algorithm and propose a new efficient algorithm, called the VIMA.

Compared to NSVIM, the VIMA algorithm uses an efficient VI node sorting strategy to map those unmapped VI nodes first that require large computing and bandwidth resources. In addition, VIMA also uses a more efficient way of calculating the VI node mapping cost that further increases the coordination between the node mapping and link mapping, and eventually reduces the costs incurred while mapping VI nodes onto the substrate network.

As noted above in VIMA, first we sort the VI nodes in decreasing order of resource requirements to determine the order of the VI nodes to be mapped.

To quantify the amount of resources (AR) required by a VI node $v$ and its connected VI link(s), $L(v)$ ($L(v) \subseteq E_L$), we define AR($v$) as follows.

$$AR(v) = \varepsilon(v) + g \times \sum_{e \in L(v)} b(e). \tag{7}$$

In (7) $g$ is the ratio of the unit communication cost to the unit computing cost. Intuitively, mapping "big" resource-hungry VI nodes first can give these big VI nodes a higher priority to reserve resources with lower cost and in turn, reduce the total overall cost. Accordingly, we select an unmapped VI node with maximum $AR$ first.

In [14] the authors use the parameter $\beta$ to control and improve the precision of the estimated link mapping cost during node mapping. In this paper, we introduce a new neighbors-selection strategy to further improve the performance. This strategy limits the set of candidate facility nodes that an unmapped VI node can be mapped onto. The strategy does not permit all unassigned facility nodes to be used for mapping, but only those unassigned facility nodes that are not "far" from the facility node $u$ to be used for mapping the unmapped end VI node of an undetermined VI link. We use a parameter $h$ to control the effect of the potential communication cost $y^1(v \to u)$ on the total VI mapping cost by modifying (6) to (8) as follows. We define the neighbor candidate facility node set $NC(u)$ of facility node $u$ as follows:

$$NC(u) = \left\{ n | H_{u,n} \le h, n \in N_F,\ n \in T \right\} \tag{8}$$

$$\underline{cp}_{u,\eta(v')} = \sum_{\eta(v') \in NC(u) \cap \Omega(v')} cp_{u,\eta(v')} / |NC(u) \cap \Omega(v')|. \tag{9}$$

Note: if $NC(u) \cap \Omega(v') = NULL$, e.g., there is no candidate facility node that the VI node $v'$ can be mapped onto, we set $cp_{u,\eta(v')}$ to a large value to avoid the VI node $v$ being mapped onto $u$ due to its potentially high VI link $(v,v')$ cost.

By setting $h = 0$ in (8), e.g., for any unassigned facility node $u$, the corresponding $NC(u)$ is $NULL$ and we disregard the effect of $y^1(v \to u)$. At the same time by setting $h$ to a very large value, we relax the limitation of $h$ and reduce (9) to (6).

## 4.3 Lower Bound on the Cost of VI Mapping

As defined in (2), the mapping cost $C$ is the sum of the VI node mapping cost and the VI link mapping cost. To simplify the calculation of the lower bound, we assume that the cost of a unit of computing resource on all facility nodes is the same and is equal to 1. In addition we also assume that the unit bandwidth costs on all fiber links are the same, and the ratio of the computing unit cost and bandwidth costs is $g$. So the cost $C$ is simplified as follows:

$$C = \sum_{v \in N_L} cn(\eta(v)) \times \varepsilon(v) + \sum_{e \in E_L} cp_e = 1 \times \sum_{v \in N_L} \varepsilon(v) + g \times \sum_{e \in E_L} h(cp_e) \times b(e) \tag{10}$$

where $h(cp_e)$ is the number of hops along path $cp_e$ that the VI link $e$ is mapped onto.

Obviously, the most cost-efficient solution will use only one-hop (i.e., one link) path, e.g., $h(cp_e) = 1$, for mapping each VI link. So the lower bound $B$ on cost of the mapping solution is as follows:

$$B = \sum_{v \in N_L} \varepsilon(v) + g \times \sum_{e \in E_L} b(e). \tag{11}$$

That is, the lower bound is determined by the weighted sum of total computing resources, and the total communication resources where the relative weight associated with a unit computing cost is 1 and that with a unit bandwidth cost is $g$.

From the above analysis, we find that the gap between the mapping cost of the algorithm and the lower bound on the mapping cost is largely dependent on the number of hops of each path for mapping a VI link, and a cost-efficient solution should shorten the average hop number of a path for each VI link.

## 4.4 Time Complexity of the NSVIM and VIMA Algorithms

To calculate the time complexity of the VIMA algorithm, we note that steps 3.2 and 3.3 are keys to the working of the VIMA algorithm. We assume that there are $N$ substrate nodes in the substrate network and $M$ VI links in a VI request. For each VI link, we need to estimate the potential communication costs once when it is an undetermined VI link, i.e., calculating $y^1(v_L \rightarrow v_F)$ in Step 3.3, and calculating its real path cost twice, one time to get $y^0(v_L \rightarrow v_F)$ in Step 3.3, and again to decide the link mapping in Step 3.4. Every time we calculate the potential communication cost of a VI link, we need to run the Dijkstra algorithms at most $|\Omega(v)|$ times. Since the complexity of the Dijkstra's algorithm is $O(N^2)$, the complexity of the NSVIM is $O(M \times (|\Omega(v)| + 2) \times N^2)$. Due to the hop limit in the VIMA algorithm the number of times the Dijkstra's algorithm is run to obtain the estimated cost of an undetermined VI link will decrease depending on the size of set $NC(u) \cap T^1 \cap \Omega(v')$. Let $w = |NC(u) \cap T^1 \cap \Omega(v')|$, accordingly the complexity of the VIMA algorithm is $O(M \times (w + 2) \times N^2)$.

## 5 Simulation Results

In this section, we describe the simulation environment including the simulation parameters and present our simulation results of the proposed algorithms.

### 5.1 Simulation Environment

We compare the working of our algorithms for a small 10-node, 15-link substrate network and a large 27-node, 41-link substrate network. The computing capacity at facility nodes and bandwidth capacity on the links follow a uniform distribution from 50 to 150. We assume that the ratio of the unit bandwidth cost and the unit computing cost g is 3, e.g., the unit computing cost (e.g., using 1,000 CPU hours) on every facility node is the same and equal to 1, the unit bandwidth cost on every fiber link (e.g., using 1 Mbps) is 3 [14].

The VI requests are generated randomly based on four main parameters: (1) the number of VI nodes in the VI request $|N|$, (2) the average probability of connectivity between any two nodes in the VI request, which we set to 0.5, (3) the average computing requirement of a VI node and (4) the average bandwidth requirement of a VI link. The computing and bandwidth requirements follow a uniform distribution from 10 to 30 and 10 to 50, respectively. We compare the effect of the mapping location constraint on performance by comparing two cases (1) unrestricted node mapping and (2) restricted node mapping. For the unrestricted node mapping case, there is no placement constraints, i.e., a VI node can be mapped onto any facility node and hence for each VI node $v$, $\Omega(v) = V_F$. For the restricted node mapping case, a VI node can only be mapped onto certain facility nodes, i.e., those facility nodes that meet the corresponding placement constraint. In our simulations, we limit how far a virtual node can be placed from the location specified. For each VI node $v$, we specify the location area and choose the substrate nodes within the location area as $\Omega(v)$.

We use two metrics to evaluate the performance of our mapping algorithms: (1) the cost of VI mapping that is the sum of the computing cost on the facility nodes and the bandwidth cost on the fiber links. This metric is our problem's optimization objective, (2) the average number of substrate path hops that are used to map the VI links. A shorter path results in a lower delay for the VI link and has a better QoS performance.

## 5.2 Comparison of VI Mapping Algorithms

In our simulations we compare four algorithms including NSVIM, vnmFlib-advanced, R-ViNE and our proposed VIMA algorithm that use different node mapping and link mapping strategies in various simulation environments. Note that since the vnmFlib-advanced outperforms the vnmFlib-simple algorithm and R-ViNE algorithm outperforms the D-ViNE algorithm we use the vnmFlib-advanced and the R-ViNE algorithms in our performance comparison.

For each size of VI request, i.e., number of VI nodes in the VI request, we randomly generate 10 VI requests and calculate the average result for each algorithm.

### *5.2.1 Effect of h on the VIMA Algorithm*

Figures 4 and 5 show the cost incurred by the VIMA algorithm for different values of $h$. In the figures we compare the working of VIMA for the unrestricted node mapping case and the restricted node mapping case, while varying $h$ from 0 to 5. From the figures we note that the cost incurred by VIMA increases as the size of the VI request increases (i.e., number of VI nodes in the VI request increases). There are two extremes, (1) by setting h to 0 and 1, we do not take (or take less) into account the effect of potential (future) communication costs and (2) by setting h to 5, we take into full/maximum account the effect of future communication costs. We note that both these cases have a bad performance in the unrestricted and restricted node mapping scenarios. The best performance is achieved when h is set around 2 and 3. This implies that taking the future communication cost into account is reasonable; while overestimating it will lead to a worse performance and selecting the proper value of $h$ to restrict the estimation range is important. An appropriate $h$ value can reduce the gap between the estimated communication cost and real communication

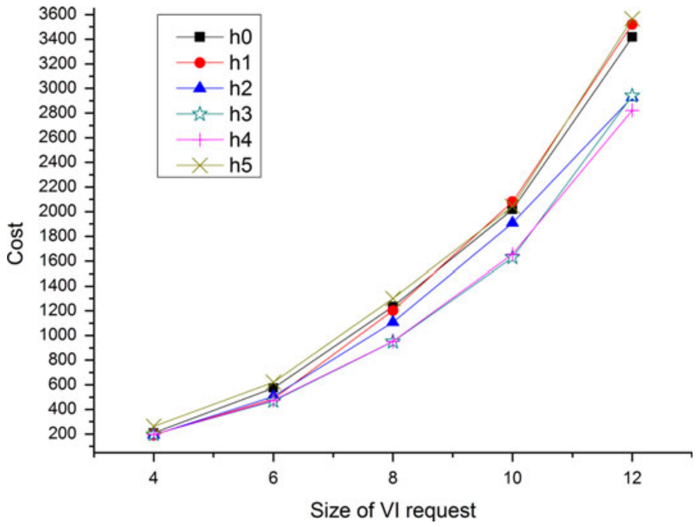

**Fig. 4** Effect of h under unrestricted node mapping scenario in VIMA

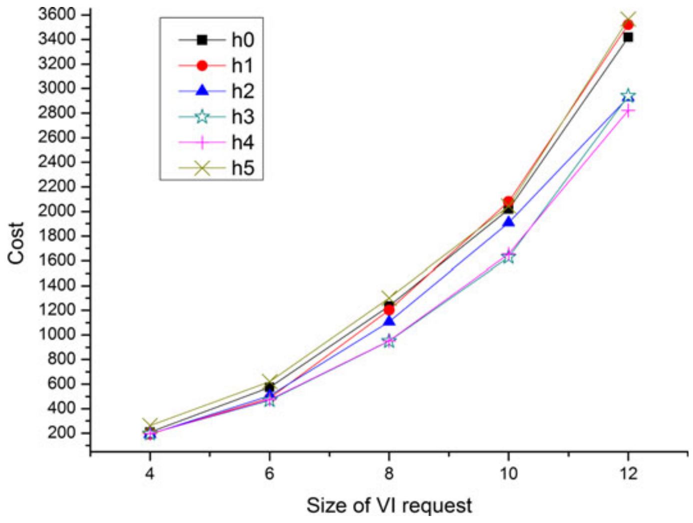

**Fig. 5** Effect of h under restricted node mapping scenario in VIMA

cost i.e., it reduces the risk of choosing a node that looks good now but may turn out to be bad later.

In addition, when the size of the VI request is small, the smaller value of *h* will be better, especially in the unrestricted node mapping scenario. The reason is that when the VI request size is small, most of VI links can be mapped onto one (or a few) hop substrate paths, and small values of *h* reduce the overestimating of future

**Table 1** Comparison of VIMA and MILP (small network)

| Cost | Size of VI request | | |
|---|---|---|---|
| | 2 | 3 | 4 |
| VIMA (h = 3) | 165 | 190 | 356 |
| MILP | 165 | 190 | 356 |
| Lower bound | 165 | 190 | 356 |

communication costs and improves the performance. In the rest of paper, we set *h* to 3 when considering the VIMA algorithm.

#### 5.2.2 *Comparison of VIMA and the MILP with the Low Bound*

In this section, we compare the performance of VIMA and MILP with the lower bound using the small 10-node network. We solve the MILP using CPLEX 8.0.

Table 1 shows that VIMA has a near-optimal (i.e., the same cost as the MILP) performance for a small-size VI request. The reason is that VI links can be easily mapped onto one-hop physical paths when the size of the VI request is small.

### 5.3 Comparison of the Four Mapping Algorithms

In this section, we compare the performances of VIMA, NSVIM, vnmFlib and R-ViNE in terms of the cost and average hops of the paths.

Figures 6 and 7 show the cost incurred by the various algorithms under the unrestricted and restricted node mapping scenarios. From Figs. 6 and 7 we find that in both scenarios, the performance gap between four algorithms and the lower bound (denoted as "LB" in the Figures) increase as the size of the VI request

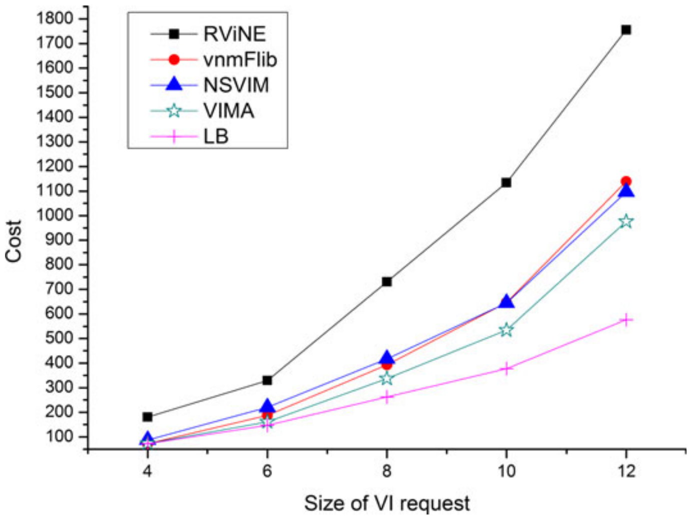

**Fig. 6** Comparison in terms of cost under unrestricted node mapping scenario

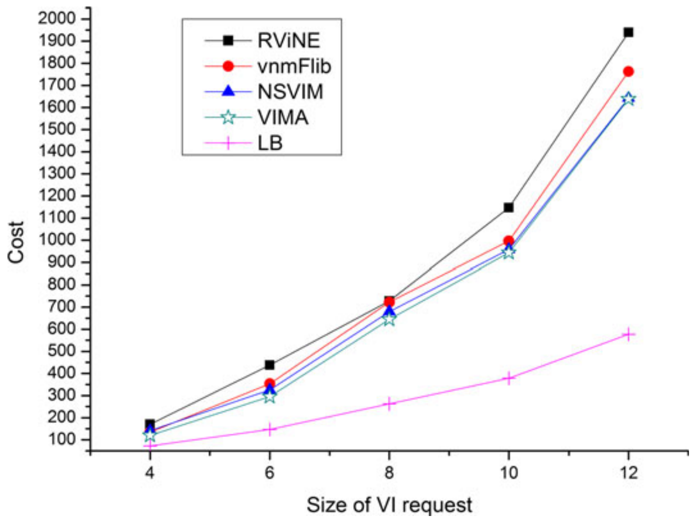

**Fig. 7** Comparison in terms of cost under restricted node mapping scenario

increases. Generally speaking, our VIMA algorithm has the best performance (lowest cost) and R-ViNE has the worst performance (highest cost). The vnmFlib algorithm performs better than R-ViNE but worse than the VIMA and NSVIM algorithms. The reason is that in R-ViNE, its relaxation-based approach weakens the coordination between node mapping and link mapping, which results in poor performance. Meanwhile, in vnmFlib, its node mapping and link mapping are in the same stage, but its node mapping does not consider the link mapping cost, thus, resulting in poor performance. NSVIM does the node mapping and link mapping in the same stage, and the node mapping takes into account the link mapping cost, but it permits all unassigned facility nodes including “far” nodes to have a chance to be mapped onto, and leads to the over-estimate of future link costs resulting in a worse performance than VIMA.

Figures 8 and 9 show the average hop length of mapped VI links as a result of using the four algorithms under the unrestricted and restricted node mapping scenarios. From the figures we note that the VIMA algorithm results in the shortest average hop length (except when size of VI request is 4), while R-ViNE results in the largest average hop length. In case of the unrestricted node mapping, when the size of the VI request is 4, vnmFlib has a lower average hop length compared to VIMA, because when the size of the VI request is small, vnmFlib can map the VI request with 1-hop subgraph isomorphism, while VIMA maps some of the VI links onto longer paths taking into account future link costs.

## 6 Conclusion and Future Work

Network virtualization can help diversify the Internet by supporting multiple virtual network architectures on a shared substrate network. Since multiple virtual networks

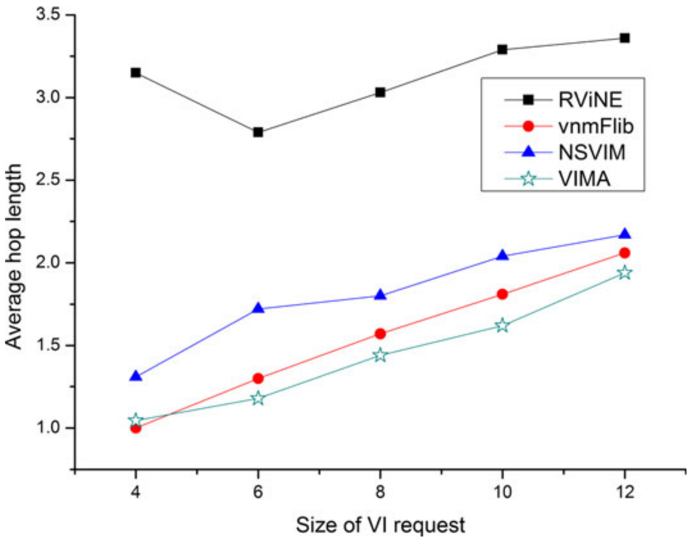

**Fig. 8** Comparison in terms of average hop length under unrestricted node mapping scenario

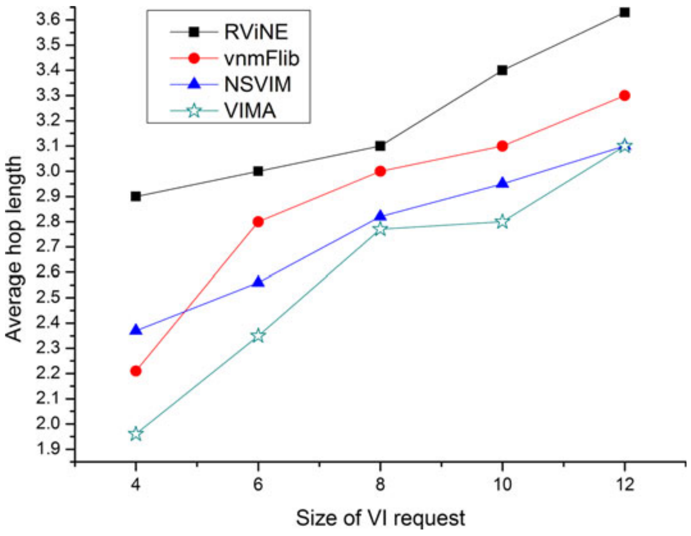

**Fig. 9** Comparison in terms of average hop length under restricted node mapping scenario

are now supported over a common substrate network, it is important to use the resources of the substrate intelligently. Making efficient use of the underlying substrate network resources requires smart algorithms for mapping the virtual networks on to the substrate network. Providing an efficient VI with a low cost is critical for emerging applications using paradigms such as FCNS and Cloud Computing. In this paper, we developed a new algorithm called VIMA for the VI mapping problem. The VIMA algorithm realizes close coordination between node and link mapping stages while mapping the virtual network. We compared the performance of our proposed VIMA algorithm with the NSVIM, vnmFlib and R-ViNE algorithms for virtual network mapping under various performance metrics using simulation. Our simulation results show that the VIMA algorithm outperforms the aforementioned algorithms in terms of mapping costs and hop length of the VI links.

Many directions for future research work in this area are possible. Applications such as e-health, virtual reality, etc., are rapidly becoming popular; providing flexible virtual infrastructure for such emerging applications that require different bandwidth connections (e.g., wavelength to sub-wavelength) is important. In our future work, we will extend the previous approaches to accommodate such sub-wavelength bandwidth requirements simultaneously. Accordingly, new challenges and design choices arise in efficiently allocating substrate networks. For instance, when a new VI request arrives, is it preferable to execute the job on the current existing virtual networks (with the option of reconfigurations) or establish a new virtual network for the job? Another area of research is how to develop new virtual infrastructure mapping algorithms for different traffic such as when the VI requests arrive one after the other or in batches or are known before hand. Algorithms and techniques that do efficient traffic grooming can make efficient use of resources, but may increase mapping times and communication paths. The study of the potential tradeoff of traffic grooming will be an interesting direction to pursue.

Since with virtualization many virtual networks share the same substrate resources, even small failures in the substrate network can cripple many applications. Thus, the topic of VI survivability becomes important. Another topic of research is how to develop efficient techniques to guarantee VI survivability to recover from different kinds of network failures. In our future work, we will also consider how the characteristics of the underlying optical network affect the design of virtual infrastructure. For example what is the effect of having wavelength conversion capabilities in the underlying optical network, or how can the various optical network survivability schemes be leveraged to provide VI survivability?

**Acknowledgments** This research was partially supported by Natural Science Foundation of China grant no. 60872032 and 60972030, the National Grand Fundamental Research 973 Program of China under Grant no. 2007CB307104, National Higher-education Institution General Research and Development project no. E022050205. The research of Dr. Qiao is supported in part by HDTRA1-09-1-0032. The information reported here does not reflect the position or policy of the federal government.

## References

1. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization computer **38**(4), 34–41 (2005)
2. Mukherjee, B.: Optical WDM Networks. Springer (2006)
3. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceedings of Grid Computing Environments Workshop, pp. 1–10 (2008)
4. Baranovski, A., et al.: Enabling distributed petascale science. J. Phys. Conf. Ser. **78**, 012020 (2007)

5. Liu, X., Qiao, C., Yu, D., Jiang, T.: Application-specific resource provisioning for wide-area distributed computing. IEEE Netw. (2010) (Special issue on future internet: new applications and emerging services", accepted for publication)
6. Luand, J., Turner, J.: Efficient mapping of virtual networks onto a shared substrate. Washington University, Technical report, WUCSE-2006-35, (2006)
7. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of IEEE INFOCOM, pp. 1–12 (2006)
8. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comput. Commun. Rev. **38**(2), 17–29 (2008)
9. Mosharaf, N.M.M.K., Rahman, M.R., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. IEEE INFOCOM (2009)
10. Butt, N.F., Chowdhury, M., Boutaba, R.: Topology-awareness and reoptimization mechanism for virtual network embedding. Lect. Notes Comput. Sci. **6091**(4), 27–39 (2010)
11. Lischka, J., Karl, H.: A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, Spain pp. 81–88 (2009)
12. Yu, H., Qiao, C., Anand, V., Lu, X., Di, H., Sun, G.: Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. IEEE Globecom (2010)
13. Yu, H., Anand, V., Qiao, C., Di, H., Wang, J.: On the survivable virtual infrastructure mapping problem. IEEE ICCCN (2010)
14. Michael, A., Armando, F., Rean, G., Anthony, D.J., Randy, K., Andy, K., Gunho, L., David, P., Ariel, R., Ion, S., Matei, Z.: Above the clouds: a berkeley view of cloud computing. University of Berkeley, Technial report, UCB/EECS-2009-28, (2009)

## Author Biographies

**Hongfang Yu** received the M.S. degree and Ph.D. degree in Communication and Information Engineering in 1999 and 2006 from University of Electronic Science and Technology of China (UESTC), respectively. From 2009 to 2010, she was a visiting scholar in Department of Computer Science and Engineering, University at Buffalo. She is currently an associate professor at UESTC. Her current research interest includes cloud computing, network survivability, and next generation Internet.

**Vishal Anand** received his Ph.D. degree in Computer Science and Engineering from the University at Buffalo (UB), SUNY in 2003. Currently he is an associate professor at The College at Brockport, SUNY. He has worked as a research scientist at Bell Labs, Lucent technologies and Telcordia Technologies (ex-Bellcore. He is the co-inventor of a patent that cost-effectively improves the switching speed of traffic in the Internet backbone, and is the recipient of the "Rising Star" and the "Promising Inventor Award" award from the Research Foundation of The State University of New York (SUNY), and the recipient of the "Visionary Innovator" award from UB. His research interests are in the area of wired and wireless computer communication networks and protocols including optical networking, cloud and grid computing.

**Chunming Qiao** directs the Lab for Advanced Network Design, Analysis, and Research (LANDER), which conducts cutting-edge research with current foci on optical and wireless networking and survivability issues in cloud computing, cyber transportation systems, low-cost and low-power sensors and mobile sensor networking. He has published more than 95 and 150 papers in leading technical journals and conference proceedings, respectively. His pioneering research on Optical Internet in mid 1990, in particular, the optical burst switching (OBS) paradigm has produced some of the highest cited works. In addition, his work on integrated cellular and ad hoc relaying systems (iCAR), started in 1999, is recognized as the harbinger for today's push towards the convergence between heterogeneous wireless technologies, and has been featured in BusinessWeek and Wireless Europe, as well as at the websites of New Scientists and CBC. He was elected to IEEE Fellow for his contributions to optical and wireless network architectures and protocols.

**Hao Di** received the M.S. degree in communication and information systems from the School of Communication and Information Engineering, University of Electronic Science and Technology of China

(UESTC), Chengdu, China, in 2006. He is currently a PhD candidate at UESTC. His research interests include network virtualization and next generation network.

**Xuetao Wei** received the M.S. degree in communication and information systems from the School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2008. He is currently a PhD candidate with the department of Computer Science and Engineering, University of California, Riverside, USA. His research interests include performance and security analysis in networked systems, cloud computing, Android applications, social and information networks.