Enhancing Survivability in Virtualized Data Centers: A Service-Aware Approach

Jielong Xu, Jian Tang *Member, IEEE,* Kevin Kwiat, Weiyi Zhang, *Member, IEEE,* and Guoliang Xue, *Fellow, IEEE*

***Abstract*—In this paper, we propose a service-aware approach to enhance survivability in virtualized data centers. The idea is to create and maintain a Survivable Virtual Infrastructure (SVI) for each service or tenant, which includes Virtual Machines (VMs) hosting the corresponding application and their backup VMs. A fundamental problem is to determine how to map each SVI to a data center network with minimum operational costs while satisfying each VM's resource requirements and bandwidth demands between VMs before and after failures. This problem can be naturally divided into two subproblems: VM Placement (VMP) and Virtual Link Mapping (VLM). We first present a general optimization framework. Then we propose an efficient algorithm for VMP, and a polynomial-time optimal algorithm for VLM, which can be used as subroutines in the framework. We also present an effective heuristic algorithm that jointly solves two subproblems. It has been shown by extensive simulation results based on the real VM workload traces collected from Syracuse University's green data center that compared to the First Fit Decreasing (FFD) and shortest path routing based baseline algorithm, the proposed algorithms significantly reduce the reserved bandwidth, and yield comparable results in terms of the number of active servers.**

***Index Terms*—Cloud Computing, Data Center, Service-aware, Survivability, Virtual Machine Management.**

I. Introduction

IN A VIRTUALIZED data center, Virtual Machines (VMs) are created and configured to host applications and perform computing tasks for cloud service users. Each (physical) server can host multiple VMs as long as it has sufficient resources (i.e. CPU, memory, bandwidth, etc).

Servers are prone to failures caused by cyber attacks (to the hypervisor) or hardware/software faults. For example, in April 2011, Amazon's cloud EC2 crashed and the outage lasted for more than 24 hours. Therefore, it is very important to develop a survivable cloud computing system that can quickly recover from failures without any service disruption. However, scant research attention has been paid to survivability issues in cloud computing. Furthermore, most existing VM management methods treat each VM individually without considering bandwidth demands between related VMs [9], [17], [18], [22]. However, it has been shown that traffic between VMs accounts for about 80% of the total traffic in a typical Internet data center [7].

In this paper, we propose a *service-aware approach* to enhance survivability in virtualized data centers. The idea is to create and maintain a *Survivable Virtual Infrastructure (SVI)* for a service or a tenant. Each SVI includes VMs that host the corresponding application. Moreover, in each SVI, backup VMs are created and synchronized with their active peers such that affected services can be quickly switched over to backup VMs after failures. The owner of an SVI can specify the number of backup VMs for each primary VM according to its importance, which could be 0, 1 or even more. Most virtualization software packages, such as VMware vSphere 5 [20], support such a fault-tolerance feature. Our approach aims at those critical applications or services that cannot tolerate any disruption or performance degradation. A data center can host a large number of SVIs for various services or multiple tenants. Note that whether or not to create backup VMs should be determined according to specifical service requirements. Sometimes it may not be necessary to have a backup for every VM in an SVI.

An SVI can be modeled as a virtual graph, where each vertex corresponds to a VM (primary or backup). There is an edge connecting a pair of vertices if the corresponding VMs need to communicate with each other. We call such edges *virtual links*. A fundamental problem in such a system is to find a mapping from each SVI to the physical data center network. Specifically, each VM is mapped to a server and each virtual link in the virtual graph is mapped to one or multiple paths between corresponding servers in the physical network as shown in Fig. 1, such that

- computing resource requirements and bandwidth demands are satisfied on servers and links respectively;
- bandwidth is reserved on links such that after any single server failure, each failed VM can be replaced by its backup and the backup can have sufficient link bandwidth to communicate with other VMs.

Note that we take into account both bandwidth demands between primary VMs and demands between primary and backup VMs (for synchronization).

This problem shares some similarities with the Network Virtualization (NV) problems [4], [10], [14], [26], [28]. The key idea of NV is to build a diversified Internet to support

Manuscript received December 15, 2012; revised July 31, 2013. A shorter version of this paper was presented in the IEEE Cloud'2012. This work was supported in part by the AFOSR Summer Faculty Fellowship Program, ARO grant W911NF-09-1-0467 and NSF grant 1217611. It was approved for Public Release; Distribution Unlimited: 88ABW-2013-2837 dated 13 JUN 2013. The information reported here does not reflect the position or the policy of the federal government.

J.Xu and J. Tang are with the Department of Electrical Engineering and Computer Science at Syracuse University (e-mail: {jxu21, jtang02}@syr.edu).

K. Kwiat is with the Air Force Research Lab (AFRL).

W. Zhang is with the AT&T Labs Research.

G. Xue is with the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University.

Digital Object Identifier 10.1109/JSAC.2013.131203.

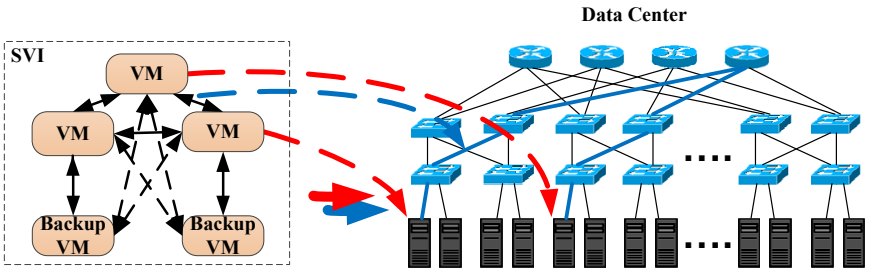


Fig. 1. SVI-physical mapping

a variety of network services and architectures through a shared substrate (physical) network by assigning physical resources (such as node, link, bandwidth, etc) to multiple virtual networks in an on-demand manner.

However, survivability has not been well studied in the context of NV. To support survivability, resources (such as link bandwidth) need to be reserved to guarantee successful failover operations. Reserved resources should be shared since servers in a data center are not likely to fail simultaneously. It is difficult to find the most efficient way to share reserved resources with zero-disruption and zero-degradation recovery. Furthermore, NV often deals with one-to-one node mapping (i.e., every virtual node must be mapped to a distinct physical node), while VM mapping allows multiple VMs to be placed on (mapped to) a common server. Therefore, our problem is harder and none of the existing NV algorithms can be directly applied to solve it.

In this paper, we consider an SVI mapping problem with the objective of minimizing operational costs. We find that the problem can naturally be divided into subproblems: VM Placement (VMP) (determine how to place VMs on servers) and Virtual Link Mapping (VLM) (determine how to map each virtual link in the virtual graph to routing paths in the physical network and how to allocate bandwidth along those paths). To the best of our knowledge, we are the first to address this survivable mapping problem in the context of virtualized data centers. Our contributions are summarized as follows:

- We establish a general optimization framework for the SVI mapping problem, into which different VMP and VLM algorithms can be incorporated.
- We propose a polynomial-time optimal algorithm to solve the VLM subproblem, which can guarantee sufficient link bandwidth for failover traffic after any single server failure with minimum reserved bandwidth. An efficient heuristic algorithm is developed for the VMP subproblem.
- We present extensive simulation results obtained based on the real VM workload traces collected from the green data center at Syracuse University [6] to justify the effectiveness of the proposed algorithms.

II. Problem Definition

We consider a data center with servers connected by high capacity switch/routers and links via Local Area Networks (LANs). We model the (physical) data center network using a directed graph $G_P(N \bigcup N_R, L)$, where each vertex in N corresponds to a server, each vertex in N_R corresponds to a switch/router and each edge in L corresponds to a physical link. We also label servers in G_P from 1 to $|N|$ and switches/routers from $|N|+1$ to $|N|+|N_R|$, respectively. Furthermore, we consider a realistic routing model where a set P_{ij} of candidate routing paths are given for each server pair (i, j) in advance, which can be found by a standard multipath routing protocol, such as Open Shortest Path First-Equal-Cost MultiPath (OSPF-ECMP) [15] which can find equal cost paths for each source-destination pair in terms of fixed measures such as line speed or hop count and has been widely used in data center networks [7]. Since a data center usually has a special tree-like or hypercube-like topology, paths or shortest paths (in terms of hop count) between any pair of servers can be easily enumerated and the total number is very limited.

In a data center, virtualization software, such as VMware [20] and Xen [24], is used to create and manage VMs. Usually, each VM hosts a guest operating system and application program(s). Multiple VMs can be placed on a common server as long as for each kind of resource (such as CPU, memory, etc), its total utilization does not exceed the capacity of that server. One or multiple redundant VMs can be created to serve as backup for an active VM. They need to be perfectly synchronized with the primary VMs. VM synchronization can be achieved using existing methods [5] and it is also supported by commercial virtualization software, such as VMware vSphere 5 [20].

As mentioned above, multiple correlated VMs (primary or backup) are grouped together to form an *SVI* for a service or a tenant. We model each SVI using a directed virtual graph $G(V, E)$, where each vertex in V corresponds to a VM (primary or backup) and there is an edge connecting a pair of such vertices if corresponding VMs have interactions. Essentially, there are edges connecting vertices corresponding to primary VMs that need to communicate with each other and there are also edges connecting backup VMs with their primary VMs since information needs to be exchanged frequently for synchronization. We call these edges in a virtual graph *virtual links*, each of which is assigned a weight to indicate its bandwidth demand. In this graph, there is also a special vertex corresponding to the external network (i.e., the Internet in most cases) and edges connecting this special vertex with all other vertices if their corresponding VMs send/receive packets to/from the external network.

Every time when a SVI mapping request arrives, a mapping algorithm will be used to determine: how to place its VMs; how to allocate bandwidth on links to route traffic between VMs; and how to reserve bandwidth on physical links for

failover traffic. Every time when a SVI leaves, corresponding VMs are removed from servers and reserved bandwidth is released from links. A mapping of a SVI is said to be feasible if the following conditions are satisfied: 1) For each kind of resource (CPU, memory, etc), the available capacity of each server is no smaller than the total demands of VMs placed on that server. Moreover, conflicting VMs are placed on different servers (e.g., a primary VM and its backup must be placed on two different servers). 2) Each link has sufficient available bandwidth to carry traffic for regular communications between primary VMs, and between primary VMs and their backups. 3) Bandwidth is reserved on links such that after any single server failure, each failed VM can be replaced by its backup and there is sufficient link bandwidth to communicate with other VMs. Now we are ready to define our mapping problem.

Definition 1 (SVIM): Given a SVI $G(V, E)$ and a data center network $G_P(N \bigcup N_R, L)$, available resources on each server and available bandwidth on each link, the **Survivable Virtual Infrastructure Mapping (SVIM)** problem seeks a *feasible* mapping $\mathcal{M} : G \mapsto (N', \mathbf{f}, \hat{\mathbf{f}})$ (where $N' \subseteq N, \mathbf{f} = [\cdots, f_p^k, \cdots], p \in P_k, k \in \{1, \cdots, |E|\}, \hat{\mathbf{f}} = [\cdots, \hat{f}_l, \cdots], l \in L$) with minimum operational costs.

Note that the SVIM problem can be naturally divided into two subproblems: *Virtual Machine Placement (VMP)* (find $\mathcal{M}(v) \in N$ for each $v \in V$) and *Virtual Link Mapping (VLM)* (determine the amount of bandwidth f_p^k that needs to be allocated on path $p \in P_k$ to route traffic for each virtual link $e_k \in E$ and the amount of bandwidth that needs to be reserved on each link $l \in L$ to carry failover traffic, where P_k is the set of paths between servers to which two ending vertices (VMs) of virtual link e_k are mapped). Note that since the candidate path set P_k is assumed to be given for any pair of servers, the VLM is essentially a bandwidth allocation problem.

Our primary goal is to minimize operational costs. The costs of operating a data center mainly come from energy consumption. IT devices, especially servers, are major energy consumers. One of the most efficient methods for power savings is to consolidate servers by packing VMs on a minimum number of servers such that idle servers, which usually consume more than 70% of peak power [2], can be shut off or put into a low power sleeping model. For simplicity, we aim at minimizing the number of active servers in this work. However, the algorithms presented here can be easily extended to other cost functions. In addition, bandwidth needs to be reserved and shared on links to support survivability. Hence, our secondary goal is to minimize the total reserved bandwidth thereby allow the data center to accommodate more SVIs.

The SVIM problem is obviously NP-hard since without even considering bandwidth demands or survivability, the VMP problem (with the objective of minimizing the number of active servers) is basically the well-known bin-packing problem [23], which has been shown to be NP-hard.

III. The Proposed Mapping Algorithms

A. A General Optimization Framework

Here, we present a general optimization framework for the SVIM problem, into which different VMP and VLM algorithms can be incorporated.

Algorithm 1 A general framework: Framework(G, G_P)

```
Step 1  Apply an algorithm to find multiple VMP solutions;
        flag := FALSE;
Step 2  for each VMP solution
          Apply an algorithm to determine the corresponding
          VLM and examine its feasibility;
          if feasible
            Record the corresponding VMP and
            link mapping solution as well as its total
            reserved bandwidth and active servers;
            flag :=TRUE;
          endif
        endfor
Step 3  if (flag=TRUE)
          output the best VMP and VLM solution;
        else  output "There is no feasible solution!";
        endif
```

If the VMP algorithm can enumerate all possible placement solutions in the first step and in the second step, the link mapping algorithm can optimally test the feasibility of each solution, then we will have an optimal algorithm for the SVIM problem. However, the number of possible VMP solutions is exponentially large. Therefore, it is acceptable to find a subset of "good" placement solutions that hopefully lead to low-cost and feasible solutions for the SVIM problem, which will be discussed in the next section. In Step 2, feasibility will be tested according to the constraints described in Section II.

B. Virtual Machine Placement (VMP)

The VMP problem (with the objective of minimizing the number of active servers) can be considered as the bin packing problem [23] with VMs as items and servers as bins. However, the well-known bin packing algorithms, such as the First Fit Decreasing (FFD), does not work well for our VMP subproblem because of its bandwidth constraints. Our simulation results verified this mismatch. Our algorithm to solve our VMP subproblem is a back-tracking algorithm that performs a Depth First Search (DFS)-like search to enumerate a subset of possible VMP solutions. The algorithm is formally presented as follows.

In the algorithm, m is a global variable, which is defined outside this function. Even though the total number of VMP solutions may be exponentially large, the algorithm generates up to M VMP solutions to avoid exponentially long running time. After quite a few trials, we found out that setting $M = 30$ leads to good performance and reasonable running time. So we used this setting in our simulation. It is possible to enumerate all possible VMP solutions for small scale networks by removing the if statements corresponding to M in the algorithm. Q_M is used to store a VMP solution and it is built progressively during the procedure of the algorithm. In the algorithm, $V^{sub} \subseteq V$ consists of vertices (VMs) in the virtual graph that have been mapped, however, $\overline{V^{sub}}$ contains vertices in $V - V^{sub}$ that are associated with either an incoming edge from a vertex $u \in V^{sub}$ or an outgoing edge to a vertex $u \in V^{sub}$. When generating node mapping tuples, a vertex (server) in N should not be removed after it is used

Algorithm 2 The VMP algorithm: VMP($V^{sub}, \overline{V^{sub}}, G, G_P$)

Step 1 **if** $(m = M)$ **return**; **endif**
Step 2 $V^{sub} := \emptyset$; $\overline{V^{sub}} := \emptyset$; $Q_M := \emptyset$;
if $(\overline{V^{sub}} = \emptyset)$
$Q := V \times N$
else $Q := \overline{V^{sub}} \times N$;
endif
Perform a preliminary tetsting on each node mapping tuple to remove infeasible ones from Q;
Step 3 **for** $((v, v') \in Q)$
if $((v, v')$ is feasible)
$Q_M := Q_M + \{v, v'\}$;
$V^{sub} := V^{sub} + \{v\}$;
Update $\overline{V^{sub}}$ with v;
if $(V^{sub} = V)$
$m := m + 1$; **output** Q_M;
else VMP($V^{sub}, \overline{V^{sub}}, G, G_P$);
endif
Restore $Q_M, V^{sub}, \overline{V^{sub}}$;
endif
if $(m = M)$ **return**; **endif**
endfor

since multiple VMs are allowed to be placed on a common server (as long as it has sufficient resources). In Step 2, a preliminary testing is conducted for each node mapping tuple $(v, v'), v \in V, v' \in N$ to make sure that the server (corresponding to v') has sufficient available resources (CPU, memory, etc) to host the VM corresponding to v. Infeasible node mapping tuples will be removed from the list Q to further reduce running time.

In Step 3, node mapping pairs are examined one by one and their feasibilities are further checked by verifying if server v' still has enough resources to host VM v after placing VMs in the current Q_M. In addition, another preliminary testing is conducted for link bandwidth to making sure links directly connecting servers with the rest of the network has sufficient bandwidth to support the bandwidth demand between the newly placed VM and the other VMs in the current Q_M. Even though feasibility is checked here, a VMP solution generated by this algorithm may still be infeasible for the given SVIM problem instance. The actual feasibility can be determined by the VLM algorithm presented later. However, if the mapping solution fails testing here, then it must be infeasible. Therefore, feasibility testing can reduce time complexity. If a node mapping pair passes the testing, it will be added into the node mapping (VMP) solution Q_M. Recursive calls are also made in Step 3 to back-track all possible mappings. The time complexity of this algorithm is $O(M|V|)$ because every time, the recursive algorithm performs a linear search to find a feasible VMP solution.

C. Virtual Link Mapping (VLM)

After the VMP is determined, we create a directed enhanced virtual graph $G_X(V_X, E_X \bigcup \hat{E_X})$, where each vertex in V_X corresponds to a *server* on which one or multiple VMs are placed (according to the obtained VMP solution) and there is an edge connecting a pair of such servers if the corresponding VMs have interactions. We call the set of such edges *active virtual links*, which are denoted as E_X and are labeled from 1 to $|E_X|$. There is also an edge connecting a server where the backup of a primary VM v_i is placed, with a server where another primary VM (that v_i has interactions with) is placed, or a server where the backup of another primary VM v_j is placed (if $\mathcal{M}(v_i) = \mathcal{M}(v_j)$ and $\mathcal{M}(backup(v_i)) \neq \mathcal{M}(backup(v_j))$. We call the set of such edges *inactive virtual links*, which are denoted as $\hat{E_X}$ and are labeled from $|E_X|+1$ to $|E_X|+|\hat{E_X}|$. These virtual links may become active after a server failure.

Next, we show that once the VMP is given, the VLM subproblem can be optimally solved by solving an LP problem which is formally presented as follows. In this formulation, P_k is the set of candidate path between two servers to which the two ending vertices (VMs) of virtual link $e_k \in E_X$ are mapped. C_l is the available bandwidth on link $l \in L$. B_k is the aggregated bandwidth demands corresponding to virtual link e_k which is the summation of bandwidth demands of all VM pairs placed on two servers corresponding to e_k. $E_i \subseteq E_X$ is the set of active virtual links whose corresponding VMs are placed on server i. These links will not exist after the failure of server i. Similarly, $\hat{E}_i \subseteq \hat{E_X}$ is the set of inactive links whose corresponding backup VMs will be used to replace failed primary VMs after the failure of server i. These links will become active after the failure.

LP-VLM:

Unknown decision variables:

1) $f_p^k \geq 0$: The amount of bandwidth allocated along path $p \in P_k$ for active virtual link e_k in E_X ($k \in \{1, \ldots, |E_X|\}$).
2) $\hat{f}_p^k \geq 0$: The amount of bandwidth reserved along path $p \in P_k$ for inactive virtual link e_k in $\hat{E_X}$ ($k \in \{|E_X|+1, \ldots, |E_X|+|\hat{E_X}|\}$).
3) $f_l \geq 0$: The total amount of bandwidth allocated to physical link $l \in L$ for active virtual links in E_X.
4) $\hat{f}_l \geq 0$: The total amount of bandwidth reserved on link $l \in L$ for failover traffic after a single server failure.

$$\min \sum_{l \in L} \hat{f}_l \tag{1}$$

Subject to:

$$\sum_{p \in P_k} f_p^k = B_k, \quad k \in \{1, \ldots, |E_X|\}; \tag{2}$$

$$\sum_{p \in P_k} \hat{f}_p^k = B_k, \quad k \in \{|E_X|+1, \ldots, |E_X|+|\hat{E_X}|\}; \tag{3}$$

$$f_l = \sum_{k=1}^{|E_X|} \Big(\sum_{p \in P_k : l \in p} f_p^k \Big), \quad \forall l \in L; \tag{4}$$

$$\hat{f}_l \geq \sum_{k=|E_X|+1}^{|E_X|+|\hat{E_X}|} \Big(\sum_{\substack{e_k \in \hat{E}_i, \\ p \in P_k : l \in p}} \hat{f}_p^k \Big) - \sum_{k=1}^{|E_X|} \Big(\sum_{e_k \in E_i, p \in P_k : l \in p} f_p^k \Big), \quad \forall i \in \{1, \cdots, |N|\}, \forall l \in L; \tag{5}$$

$$f_l + \hat{f}_l \leq C_l, \quad \forall l \in L. \tag{6}$$

Proposition 1: If the VMP solution is given, the LP-VLM can be used to test whether this placement solution can yield a *feasible* SVI mapping. If feasible, then solving it can give a corresponding VLM with minimum reserved bandwidth in polynomial time.

Proof: In this formulation, Constraints (2) make sure that bandwidth demands of primary VM pairs and primary-backup pairs are satisfied. Constraints (3) ensure that after the failure of a primary VM, sufficient bandwidth is reserved for communications between its backup VM and other VMs. Variable f_l calculates the aggregated bandwidth demands on link l for active communications (before a failure). Sufficient bandwidth needs to be reserved on each link l to carry failover traffic after any single server failure, which is guaranteed by Constraints (5). Due to the assumption that only one server (and its VMs) will fail at a particular time, reserved bandwidth needs to be shared to improve resource (reserved bandwidth) utilization, i.e., the reserved bandwidth on each link ($\hat{f}_l$) should be set to the maximum (instead of the summation of) bandwidth needed to carry failover traffic after a single server failure. Moreover, after a failure, link bandwidth that was previously used to carry traffic related to the failed server can be re-used to carry failover traffic, which is also ensured by Constraints (5). In this way, resource utilization can be further improved. The objective in expression (1) is to minimize the total reserved bandwidth, which results in a VLM with minimum reserved bandwidth. This LP problem includes $((|E_X| + |\hat{E_X}|) * |P| + 2|L|)$ variables and only $(|E_X| + |\hat{E_X}| + 2|L| + |N||L|)$ constraints, where P is the set of all server-server paths. Therefore, it can be efficiently solved by existing algorithms [1] in polynomial time. This completes the proof. ■

D. A Joint Mapping Algorithm

Unlike the proposed framework, the algorithm presented here jointly solves the two subproblems. This algorithm deals with the mapping of links one by one and determines a feasible VMP (node mapping) concurrently. We create a $|N| \times |N|$ matrix C^E and an $|L|$ vector C to keep track of the end-to-end available bandwidth between a pair of servers and the available bandwidth on each link respectively. Note that $C^E_{ii} = \infty$ because if two VMs are placed on the same server then we assume they have infinite bandwidth between them.

The joint mapping algorithm is formally presented as Algorithm 3. The basic idea of this algorithm is to view the SVIM problem as a virtual link packing problem, i.e, the problem of packing virtual links in the virtual graph $G(V, E)$ into the physical network $G_P(N + N_R, L)$. Sorting in Step 1 ensures that difficult cases will be dealt with first, which usually leads to good performance. Since the primary goal is to minimize the number of active servers, the selection of the server pair $(i_{\max}, j_{\max})$ for packing each active virtual link should, if at all possible, avoid turning on new servers. Note that in a server pair (i, j) considered here, i, j may be the same. Moreover, multiple tests need to be performed to check the feasibility of using this server pair: 1) We need to check whether the two servers have sufficient capacities to host the corresponding VMs. 2) If two virtual links share a common vertex in the virtual graph G, the shared vertex (VM) must be mapped to a common server. 3) The selected server pair $(i_{\max}, j_{\max})$ must have sufficient bandwidth to accommodate virtual link e_k, i.e, $C^E_{i_{\max}, j_{\max}} \geq B_k$, where B_k is the bandwidth demand of virtual link e_k. After determining the server pair for packing virtual link e_k, we can figure out how to allocate bandwidth along paths between the two servers by solving the LP-BA. The LP aims at finding a feasible bandwidth allocation with balanced link loads by minimizing maximum link load. In addition, after determining both VMP and VLM, reserved bandwidth can be easily computed by solving the LP-VLM presented above using the flow values of active virtual links (f^k_p) obtained before. The available end-to-end bandwidth matrix C^E needs to be updated following the packing of a virtual link. This can be done by solving a maximum-flow-like LP similar to the LP-BA except that its objective function is to maximize end-to-end flow and there is no bandwidth demand constraint for each pair of servers.

Algorithm 3 The joint mapping algorithm: Joint (G,G_P)

Step 1 Sort active virtual links in E in the descending order of their bandwidth demands and store them in the list E_S;
$flag$:= TRUE;

Step 2 **for** ($e_k \in E_S$)
Find a pair of server $(i_{\max}, j_{\max})$ such that $(i_{\max}, j_{\max}) = B_{ij}$ among all server pairs (i, j) that are feasible to host VMs corresponding to e_k and need to turn on minimum number of servers.
if $(i_{\max}, j_{\max})$ does not exist)
$flag$:= FALSE; **break**;
endif
Solve the LP-BA to determine bandwidth allocation $(f^k_p, p \in P_{i_{\max}, j_{\max}})$;
Update matrix C^E and vector C;
Mark new servers that need to be turned on;
if (no feasible solution)
$flag$:= FALSE; **break**;
endif
endfor

Step 3 **if** ($flag$=TRUE)
Solve the LP-VLM to determine reserved bandwidth on each link $(\hat{f}_l, l \in L)$ by using the values of f^k_p obtained above;
if (there exists a feasible solution)
output the VMP and VLM solution;
return;
endif
endif
output "There is no feasible solution!";

LP-BA:
Unknown decision variables:

1) $f^k_p \geq 0$: The amount of bandwidth allocated along path $p \in P_k$ for virtual link e_k in E ($k \in \{1, \ldots, |E|\}$).
2) $f_l \geq 0$: The total amount of bandwidth allocated to link $l \in L$.

$$\min \beta \tag{7}$$

Subject to:

$$\sum_{p \in P_k} f_p^k = B_k, \quad k \in \{1, \ldots, |E|\}; \tag{8}$$

$$f_l = \sum_{k=1}^{|E|} \Big(\sum_{p \in P_k : l \in p} f_p^k \Big) \leq \beta, \quad \forall l \in L; \tag{9}$$

$$f_l \leq C_l, \quad \forall l \in L. \tag{10}$$

IV. Implementation Issues

In this section, we discuss some practical issues related to implementation. First, a virtualization software needs to be used for creating, deleting, migrating VMs (primary and backup) and for synchronizing primary VMs with their backups. VMware vSphere 5 [20] is a good candidate. The VMware fault tolerance component [20] supports VM synchronization and allows instantaneous failover between two VMs in the event of server failures.

Source routing can be used to enable scalable routing and implement our idea of link mapping (bandwidth allocation and reservation). With source routing, switches become stateless and do not need to maintain any SVI mapping and bandwidth reservation states. They simply perform packet forwarding based on the source routing information carried in packet headers. Commodity servers and switches are currently used in most data centers to provide the best performance-price tradeoff. However, source routing may not be supported by some commodity switches. The port-switching based source routing (PSSR) introduced in [8] can be applied to enable source routing in data centers with commodity switches.

Basically, the proposed approach can be implemented as a global mapping manager using the APIs of a virtualization software. For example, the VMware vSphere's SDK [21] can be used for implementation. The manager interacts with the VMware vCenter (a software tool that provides centralized control and visibility at every level of a virtualized data center) to control VMs. Once a SVI mapping request is given, the manager can apply a proposed algorithm to find a SVI mapping, place the corresponding VMs based on the VMP solution given by our algorithm, and disseminate mapping information to hypervisors installed on servers. Similar as in [8], hypervisors can be used to store and maintain mapping, routing paths and bandwidth reservation states. Moreover, they encode source routing paths into packet headers, and perform bandwidth allocation and reservation according to the VLM solution given by our mapping algorithm. In addition, a data center network usually has a special tree-like topology such as fat-tree [13] and VL2 [7], and therefore it is easy to enumerate all possible paths or shortest paths (in terms of hop count) between a pair of servers.

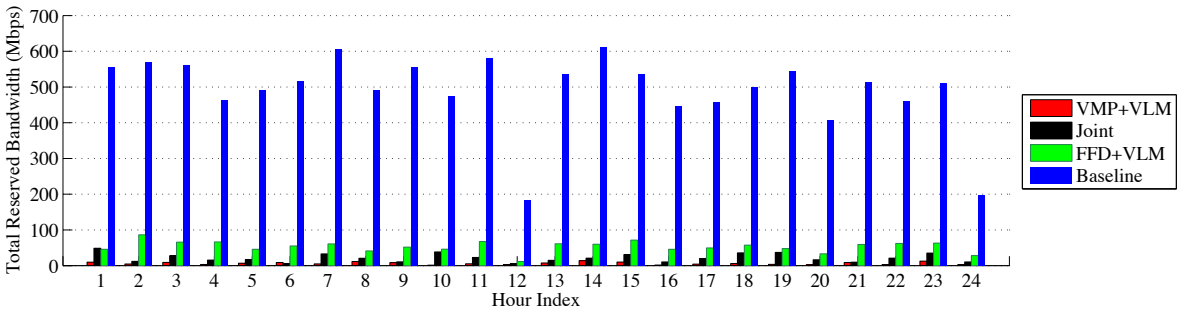
In order to always achieve the best resource utilization, the mappings of SVIs may need to be adjusted over time due to changes in VM resource utilization, changes in bandwidth demands between VMs, and/or the arrival and removal of SVIs. To handle this issue, a monitor can be implemented to track current VM resource utilization, make a prediction based on historical data and instantaneous numbers using methods such as that in [3], and feeds the information to the mapping manager. By re-running our mapping algorithm, the mapping manager can adjust the VMP (via live migrations) and VLM (via traffic shaping by hypervisors). However, such a re-mapping procedure should be not performed frequently due to large overheads introduced by VM migrations. The criteria of triggering re-mapping and the tradeoff between the overhead and the overall performance will be studied in our future work.

V. Simulation Results

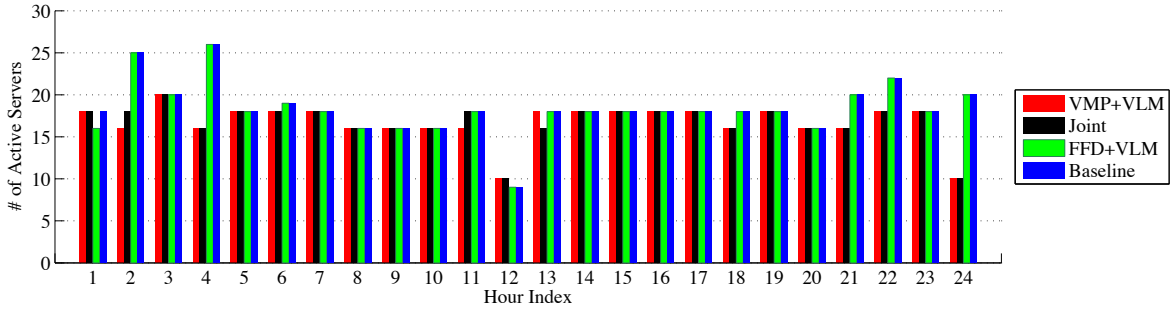
Our simulation runs were conducted based on the real VM workload traces collected from the green data center at Syracuse University [6]. We measured the CPU, memory and network bandwidth utilizations of 100 typical VMs hosting various services in our data center every 5 minutes for over one month (3/7/2011-4/11/2011). Similar to [17], a simple estimation approach was used to pre-process the raw data and generate inputs for our algorithms. Specifically, we considered one day as an observation period and an hour as an observation interval. In the workload traces, given a VM and a resource type, there are 12 consecutive sample utilization values corresponding to a particular hour, and there are $12 \times 36 = 432$ such values corresponding to that hour during that month which are aggregated together to form a set. Therefore, for a particular type of resource on each VM, there are a total of 24 such data sets. In each data set, the 90th percentile is chosen to represent the utilization of that kind of resource on that VM. In addition, in the data center, most servers are IBM 3850X5 servers, each of which has 40 CPU cores (each at 2.4GHz) and 1TB memory. We used two typical network topologies, fat-tree [13] and VL2 [7], for simulation Both topologies have three layers. The capacities of links on the bottom layer and on the top two layers were set to 1Gbps and 10Gbps respectively, which are typical in production data centers.

Since we are the first to study such a survivable mapping problem, we used a baseline algorithm for comparison. In this algorithm, the First Fit Decreasing (FFD) algorithm was used for VMP because it is known to perform well in terms of minimizing the number of active servers (bins) [23]. For link mapping, all traffic between a pair of VMs is routed via a shortest path, and moreover, bandwidth is reserved according to the bandwidth needed by all inactive links in the enhanced virtual graph. In the simulation, we compared our 2-step mapping algorithm (denoted as VMP+VLM) and the joint mapping algorithm (denoted as Joint) against the baseline algorithm (denoted as Baseline) in terms of the number of active servers and the total reserved bandwidth. In both of our algorithms, all shortest (minimum hop-count) paths between any pair of servers were enumerated to serve as input.

In the first scenario, we conducted simulation runs on a fat-tree network for a total of 24 hours. Note that each VM's resource utilizations change over time. For each hour, we randomly selected 10 VMs from the set of 100 VMs and added 10 backup VMs (with the same sizes as their primary VMs) to form an SVI. The virtual graph was assumed to be a complete graph, i.e., every VM needs to communicate with every other. In each simulation run, we kept adding SVIs to



(a) Total reserved bandwidth



(b) The number of active servers

Fig. 2. Performance on a fat-tree network with 60 SVIs

the data center and tracking the number of active servers and total reserved bandwidth until the number of accommodated SVIs reached 60. In addition, we considered an algorithm, "FFD+VLM", to ensure more fair comparison, which uses the FFD for VMP and the proposed algorithm for VLM. The corresponding simulation results are presented in Fig. 2.

We make the following observations from this figure:

1) As expected, the proposed algorithms reserve much less bandwidth than the baseline algorithm. Specifically, on average, the total reserved bandwidth given by our VMP+VLM and the joint algorithms is only 1.31% and 4.44% of that given by the baseline algorithm, respectively. This is mainly because our LP-based VLM algorithm (used by both algorithms) finds the best way to share reserved bandwidth for each SVI. However, the baseline algorithm does not carefully consider the inter-VM traffic demands for routing or bandwidth reservation for survivability.

2) The numbers of active servers given by our algorithms are comparable to those given by the baseline algorithm, which uses the FFD algorithm to handle VMP. Specifically, in 14 out of 24 hours three algorithms used exactly the same number of active servers. In 9 out of the rest 10 hours, the VMP+VLM and joint algorithms even outperform the baseline algorithm. This shows that our algorithms can achieve much lower reserved bandwidth without increasing the number of active servers.

3) The number of active servers and reserved bandwidth change over time because the resource (CPU, memory) utilizations of each VM change over time. Note that our data center carries heavy workloads even in early morning hours because some scientific computing applications were scheduled to run from late night to early morning (in the next day).

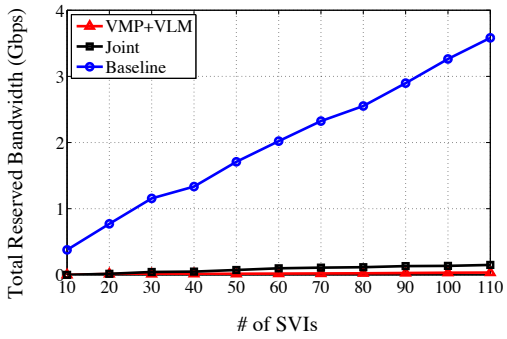
4) The proposed VMP+VLM algorithm and the joint algorithm outperform the FFD+VLM algorithm by 87.3% and 57.2% on average in terms of reserved bandwidth while offering almost the same performance in terms of the number of active servers. This shows the proposed VMP algorithm (Algorithm 2) outperforms the well-known FFD algorithm since the same algorithm is used for VLM.

In the second scenario, we performed simulation runs on both network topologies (fat-tree and VL2) for a non-busy hour (light CPU utilizations) and a busy hour (heavy CPU utilizations). We used almost the same settings as the first scenario except that we kept adding SVIs to the data center until over the number of accommodated SVIs reached a certain value (as shown in the figure) in each run. The corresponding results are presented in Figs. 3–6.

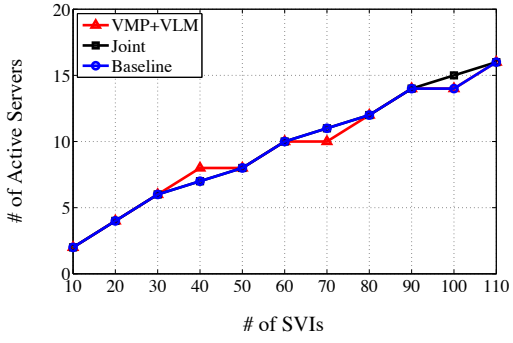
We make the following observations from these results:

1) Similarly, we can see that the total reserved bandwidth given by our VMP+VLM and the joint algorithms is only 1.48% and 3.57% of that given by the baseline algorithm, respectively. We can also observe that the total reserved bandwidth increases very slowly with the number of SVIs because sometimes after a server failure, link bandwidth that is released for failed primary VMs may be re-used for backup VMs such that no additional link bandwidth needs to be reserved for failover traffic.

2) In terms of the number of active servers, the performance of our algorithms are also comparable to that of the baseline algorithm. Specifically, the average differences between our algorithms and the baseline algorithm are only 1.44% and 1.15% respectively. In addition, no matter which algorithm is used, the number of active servers increases slowly with the number of SVIs because the primary objectives of all the algorithms are to minimize the number of active servers and they only turn on new servers when absolutely necessary.

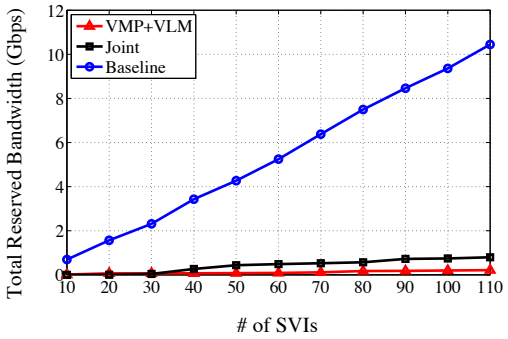


(a) Total reserved bandwidth

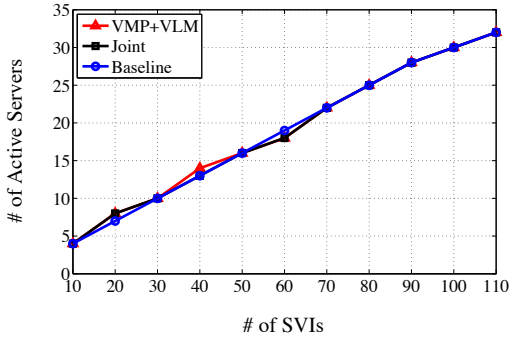


(b) The number of active servers

Fig. 3. Performance on a fat-tree network with light workloads



(a) Total reserved bandwidth



(b) The number of active servers

Fig. 4. Performance on a fat-tree network with heavy workloads

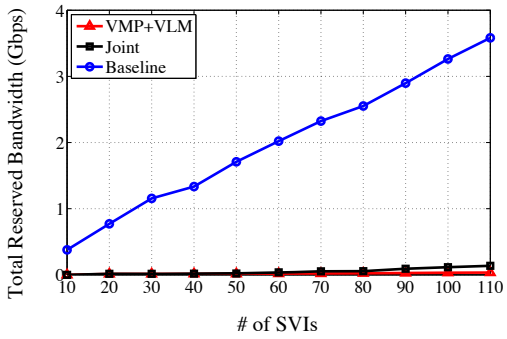
3) The two typical data center topologies are similar since they both have a tree-like topology with 3 layers. Therefore, every algorithm yields similar performance on these two network topologies with regards to both metrics.

VI. Related Work

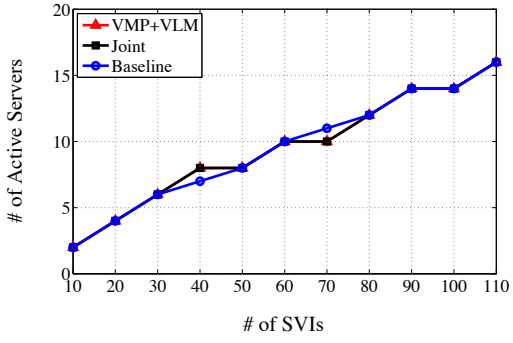
VM management has attracted research attention from both industry and academia due to its potential for reducing operation costs of data centers. A few commercial software tools (such as VMware Capacity Planner [19]) have been developed to determine VMP according to resources at hosting servers such as CPU, memory, etc. In [9], Li *et al.* proposed a power efficient approach named EnaCloud, which uses application scheduling and VM live migration to minimize the number of running servers. In [17], mathematical programming formulations were presented for various VMP problems and heuristic algorithms were presented to solve them. Extensive simulations were conducted based on a large set of real server load data from a production data center. Unlike [17], the VMP problem with bandwidth demand constraints was formulated into a stochastic (instead of deterministic) bin packing problem and an online approximation algorithm was presented in [22]. In [18], Verma *et al.* presented the design, implementation and evaluation of a power-aware application placement controller, pMapper, in virtual server clusters. In [11], Meng *et al.* , for the first time, considered bandwidth demands between VMs and aimed to minimize the communication cost. The problem was showed to be NP-hard. The authors designed a two-tier heuristic algorithm to solve it. In [16], the authors introduced a computationally efficient scheme, AppAware, to incorporate inter-VM dependencies and the underlying network topology into VM migration decisions. Using simulations, they showed that it decreases network traffic by up to 81% compared to a well-known method that is not application-aware.

The mapping problem studied in this paper shares some similarities with the NV problems, which have been studied in recent works. In [28], the authors developed heuristic algorithms for two versions of the problem: Virtual Network (VN) assignment without reconfiguration and VN assignment with reconfiguration. In [26], Yu *et al.* presented heuristic algorithms to solve an NV problem which allows the substrate (physical) network to split a virtual link over multiple substrate paths and employ path migration to periodically re-optimize the utilization of the physical network. Joint node mapping and link mapping algorithms based on Linear Programming (LP) rounding were presented in a later work [4]. In [10], a fast VN mapping algorithm based on subgraph isomorphism detection was presented, which maps nodes and links during the same stage. The authors of [14], for the first time, presented heuristic algorithms to solve a survivable NV problem, in which a certain percentage of bandwidth of each link is reserved to support survivability.

The closest work is a recent paper [8], in which an NV architecture called SecondNet was designed and evaluated for virtualized data centers. In the SecondNet, a centralized resource allocation algorithm is used for bandwidth guaranteed virtual to physical mapping. The SecondNet was implemented using source routing and MPLS [12]. In addition, the authors

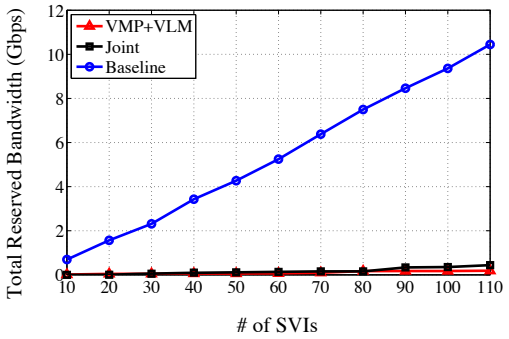


(a) Total reserved bandwidth

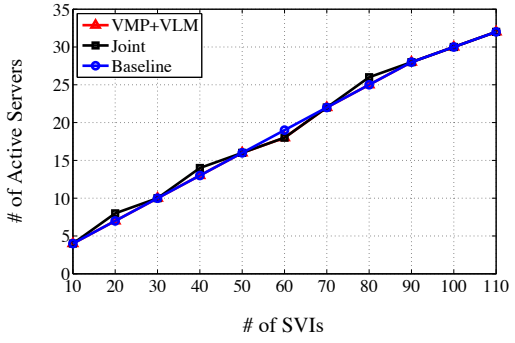


(b) The number of active servers

Fig. 5. Performance on a VL2 network with light workloads



(a) Total reserved bandwidth



(b) The number of active servers

Fig. 6. Performance on a VL2 network with heavy workloads

of [25] considered the problem of efficiently allocating resources in a virtualized physical infrastructure for VIs with reliability constraints and presented a mixed integer programming formulation. In [27], Yu *et al.* presented a mixed integer linear programming formulation and two heuristic algorithms for a VI mapping problem in a federated computing and networking system with multiple data centers under single regional failures.

The differences between our work and these related works are summarized as follows: 1) The commercial software tools [19] and most of the recent works on VM management [9], [17], [18], [22] did not consider bandwidth demands between VMs. 2) As described in the first section, our mapping problem is different from the NV problems studied in [4], [10], [14], [26], [28] since multiple VMs are allowed to be placed on a common server. 3) Survivability has not been addressed in the works on NV [4], [10], [26], [28] or other closely related works [8], [11]. 4) Unlike heuristic algorithms presented in [8], [11], [14], [27] which cannot provide any performance guarantees, our algorithms can ensure sufficient link bandwidth for failover traffic after any single server failure. 5) Inter-data-center resource allocation and regional failures were considered in [27], while our work deals with intra-data-center issues and considers a different failure model. 6) Unlike [25] (which aimed to find the relationship between reliability and redundant resources), we aim at minimizing backup resources while guaranteeing zero-disruption zero-degradation failover.

VII. Conclusions

In this paper, we proposed to use a service-aware approach to enhance survivability in virtualized data centers and considered a related optimization problem, SVIM. We found that the SVIM problem can be naturally divided into two subproblems: VMP and VLM. We presented a general optimization framework based on such an observation. We also presented a polynomial-time optimal algorithm for the VLM subproblem and an efficient heuristic algorithm for the VMP subproblem, which can be used as subroutines in the framework to solve the SVIM problem. In addition, we presented an effective algorithm to solve the two subproblems jointly. It has been shown by extensive simulation results based on the real VM workload traces collected from the green data center at Syracuse University that compared to the FFD and single shortest path based baseline algorithm, both our VMP+VLM and joint algorithms significantly reduce the reserved bandwidth, and yield comparable results in terms of the number of active servers.

References

[1] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows (3rd Edition)*, John Wiley & Sons, 2005.

[2] A. Beloglazov and R. Buyya, Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers, *ACM MGC'2010*.

[3] N. Bobroff, A. Kochut and K. Beaty, Dynamic placement of virtual machines for managing SLA violations, *IEEE IM'2007*, pp. 119–128.

[4] N. Chowdhury, M. Rahman, R. Boutaba, Virtual network embedding with coordinated node and link mapping, *Proc. IEEE Infocom'2009*, pp. 783–791.

[5] B. Cully, *et al.* , Remus: High availability via asynchronous virtual machine replication, *USENIX NSDI'2008*, pp. 161–174.

[6] Green Data Center at Syracuse University, *http://www.syr.edu/greendatacenter/*

[7] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel and S. Sengupta, VL2: A scalable and flexible data center network, *ACM SIGCOMM'2009*, pp. 51–62

[8] C. Guo, *et al.* , SecondNet: a data center network virtualization architecture with bandwidth guarantee, *ACM CoNEXT'2010*.

[9] B. Li, J. Li, J. Huai, T. Wo, Q. Li and L. Zhong, EnaCloud: An energy-saving application live placement approach for cloud computing environments, *IEEE CLOUD'2009*, pp. 17–24.

[10] J. Lischka and H. Karl, A virtual network mapping algorithm based on subgraph isomorphism detection, *ACM VISA'2009*, pp. 81–88.

[11] X. Meng, V. Pappas and L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, *IEEE Infocom'2010*.

[12] Multi-Protocol Label Switching, *http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching*

[13] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, PortLand: A scalable fault-tolerant layer 2 data center network fabric, *ACM SIGCOMM'2009*, pp. 39–50.

[14] M. Rahman, I. Aib and R. Boutaba, Survivable virtual network embedding, *Lecture Notes in Computer Science*, Vol. 6091/2010, pp. 40–52.

[15] G. M. Schneider and T. Nemeth, A simulation study of the OSPF-OMP routing algorithm, *Computer Networks J.*, Vol. 39, No. 4, 2002, pp. 457–468.

[16] V. Shrivastava, *et al.* , Application-aware virtual machine migration in data centers, *Proc. IEEE Infocom'2011*, pp. 66–70.

[17] B. Speitkamp, M. Bichler, A mathematical programming approach for server consolidation problems in virtualized data centers, *IEEE Trans. Services Computing*, Vol. 3, No. 4, 2010, pp. 266–278.

[18] A. Verma, P. Ahuja and A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, *ACM/IFIP/USENIX International Middleware Conference*, 2008.

[19] VMware Inc., VMware Capacity Planner, *http://www.vmware.com/products/capacity-planner/*

[20] VMware Inc., VMware vSphere, *http://www.vmware.com/products/vsphere/*

[21] VMware Inc., VMware vSphere SDK, *http://www.vmware.com/support/developer/vc-sdk/*

[22] M. Wang, X. Meng and L. Zhang, Consolidating virtual machines with dynamic bandwidth demand in data centers, *Proc. IEEE Infocom'2011*, pp. 71–75.

[23] M. A. Weiss, *Data Structures and Algorithm Analysis in Java (2nd Edition)*, Addison Wesley, 2006.

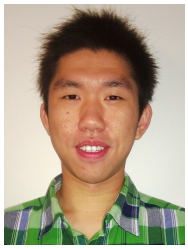
[24] Citrix Inc., Xen Server, *http://www.ctrix.com/*

[25] W-L. Yeow, C. Westphal and U. C. Kozat, Designing and embedding reliable virtual infrastructures, *ACM VISA'2010*, pp. 33–40.

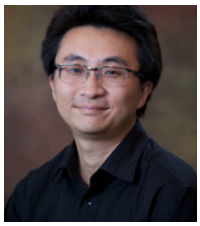
[26] M. Yu, Y. Yi, J. Rexford, and M. Chiang, Rethinking virtual network embedding: substrate support for path splitting and migration, *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, 2008, pp. 17–29.

[27] H. Yu *et al.* , Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures, *IEEE Globecom'2010*.

[28] Y. Zhu and M. Ammar, Algorithms for assigning substrate network resources to virtual network components, *IEEE Infocom'2006*.



Jielong Xu is a PhD candidate in the Department of Electrical Engineering and Computer Science at the Syracuse University. He received a Master degree in Computer Engineering from Syracuse University, and a Bachelor degree in Computer Engineering from Sun Yat-Sen University, China. His research interests include Cloud Computing, Big Data and Social Networking.



Jian Tang (Member 2006) is an associate professor in the Department of Electrical Engineering and Computer Science at Syracuse University. He received his Ph.D. degree in Computer Science from Arizona State University in 2006. His research interests lie in the areas of Cloud Computing, Big Data, Wireless Networking and Green Networking. He received an NSF CAREER Award in 2009. He has been an editor of IEEE Transactions on Vehicular Technology since 2010. He served as a co-chair for the IEEE Globecom'2011 Wireless Networking Symposium. He has also served on the TPC of many international conferences such as IEEE Infocom, ICC, Globecom, etc.

Kevin Kwiat has been with the U.S. Air Force Research Laboratory (AFRL) in Rome, New York for over 30 years. Currently is assigned to the Cyber Assurance Branch. He received the BS in Computer Science and the BA in Mathematics from Utica College of Syracuse University, and the MS in Computer Engineering and the Ph.D. in Computer Engineering from Syracuse University. He holds 4 patents. In addition to his duties with the Air Force, he is an adjunct professor of Computer Science at the State University of New York at Utica/Rome, an adjunct instructor of Computer Engineering at Syracuse University, and a Research Associate Professor with the University at Buffalo. He has been by recognized by the AFRL Information Directorate with awards for best paper, excellence in technology teaming, and for outstanding individual basic research. His main research interest is dependable computer design.

Weiyi (Max) Zhang (Member 2007) received the Ph.D. degree in computer science from Arizona State University, Tempe, in 2007. He is a Senior Member of Technical Staff in the Network Evolution Research Department at AT&T Labs Research, Middletown, NJ. Before joining AT&T Laboratories, he was an Assistant Professor with the Computer Science Department at North Dakota State University, Fargo, ND. His research interests include Small Cell Strategy, Software Defined Network, Network Traffic Modeling, Routing, Scheduling and Cross-Layer Design in Wireless Networks.

Guoliang Xue (Member 1996, Senior Member 1999, Fellow 2011) is a Professor of Computer Science and Engineering at Arizona State University. He received the BS degree (1981) in mathematics and the MS degree (1984) in operations research from Qufu Normal University, Qufu, China, and the PhD degree (1991) in computer science from the University of Minnesota, Minneapolis, USA. His research interests include survivability, security, and resource allocation issues in networks. He has published over 200 papers in these areas, including over 100 journal papers. He received Best Paper Awards at IEEE Globecom'2007, IEEE ICC'2011, IEEE MASS'2011, IEEE Globecom'2011, and IEEE ICC'2012, and a Best Paper Award Runner-up at IEEE ICNP'2010. He is an Associate Editor of IEEE/ACM Transactions on Networking and IEEE Network. He was a TPC Co-Chair for IEEE INFOCOM'2010, a Panels Co-Chair for ACM MOBIHOC'2008. He was a Keynote Speaker at IEEE LCN'2011. He was a Distinguished Lecturer of the IEEE Communications Society in 2010 and 2011. He is a Fellow of the IEEE.