Survivable Virtual Infrastructure Mapping in Virtualized Data Centers

Jielong Xu, Jian Tang, Kevin Kwiat, Weiyi Zhang and Guoliang Xue

***Abstract*—In a virtualized data center, survivability can be enhanced by creating redundant Virtual Machines (VMs) as backup for VMs such that after VM or server failures, affected services can be quickly switched over to backup VMs. To enable flexible and efficient resource management, we propose to use a service-aware approach in which multiple correlated VMs and their backups are grouped together to form a Survivable Virtual Infrastructure (SVI) for a service or a tenant. A fundamental problem in such a system is to determine how to map each SVI to a physical data center network such that operational costs are minimized subject to the constraints that each VM's resource requirements are met and bandwidth demands between VMs can be guaranteed before and after failures. This problem can be naturally divided into two sub-problems: VM Placement (VMP) and Virtual Link Mapping (VLM). We present a general optimization framework for this mapping problem. Then we present an efficient algorithm for the VMP subproblem as well as a polynomial-time algorithm that optimally solves the VLM subproblem, which can be used as subroutines in the framework. We also present an effective heuristic algorithm that jointly solves the two subproblems. It has been shown by extensive simulation results based on the real VM data traces collected from the green data center at Syracuse University that compared with the First Fit Descending (FFD) and single shortest path based baseline algorithm, both our VMP+VLM algorithm and joint algorithm 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

Cloud computing has evolved as an important computing model, which enables information, software, and shared resources to be provisioned over the network as services in an on-demand manner. Data centers that consolidate computing and storage capabilities have emerged as the major infrastructure for supporting cloud computing. Virtualization has and will continue to play a key role in cloud computing. In a virtualized data center, Virtual Machines (VMs) are created and configured to host applications and execute 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. The Amazon's recent cloud crash made the whole world realize how important survivability is to cloud computing. A survivable cloud computing system must be able to 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 and do not consider bandwidth demands between VMs [8], [16], [17]. However, it has been shown by recent research [6] that traffic between VMs in a typical Internet data center accounts for about 80% of its total traffic. To enable more flexible and efficient resource management, we propose a *service-aware approach* in which multiple correlated VMs are grouped together to form a *Survivable Virtual Infrastructure (SVI)* for a service or a tenant. In each SVI, backup VMs are also created and synchronized with active VMs such that after failures, affected services can be quickly switched over to backup VMs. Note that our approach aims at those critical applications or services that cannot tolerate any disruption or performance degradation. It is certainly not necessary to create a backup for each VM in a data center. Commercial virtualization software, VMware vSphere 5 [19], for example, supports such a feature. Each data center can host a number of SVIs for various services or multiple tenants.

An SVI can be modeled as a virtual graph, where each vertex corresponds to a VM (primary or backup) and 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*.

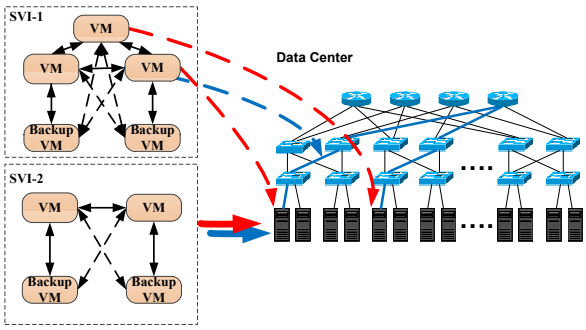


Fig. 1. SVI-physical mapping

A fundamental problem in such a system is to find a mapping from each SVI to the physical data center network such that 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: 1) Computing resource requirements and bandwidth demands are satisfied on servers and links respectively. 2) Bandwidth is reserved on links such that after any single server failure, each failed VM can be replaced by its backup and

Jielong Xu and Jian Tang are with the Department of Electrical Engineering and Computer Science at Syracuse University. Email: {jxu21, jtang02}@syr.edu. Kevin Kwiat is with the Air Force Research Lab (AFRL). Weiyi Zhang is with the AT&T Labs Research. Guoliang Xue is with the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. This research was supported in part by the AFOSR Summer Faculty Fellowship Program and NSF grant 1115129. It was approved for Public Release; Distribution Unlimited: 88ABW-2011-4082 dated July 25, 2011. The information reported here does not reflect the position or the policy of the federal government.

978-0-7695-4755-8/12 $26.00 © 2012 IEEE
DOI 10.1109/CLOUD.2012.100

it can have sufficient link bandwidth to communicate with other VMs. Note that we take into account both bandwidth demands between primary VMs and between primary and backup VMs (for synchronization). This problem shares some similarities with the Network Virtualization (NV) problems, which have been studied by a few recent papers [9], [13], [23], [24]. The key idea of NV is to build a diversified Internet to support 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. However, reserved resources need to be shared since servers in a data center normally will not simultaneously fail. It is very difficult to find the most efficient way to share reserved resources while still guaranteeing zero-disruption and zero-degradation recovery. Furthermore, one-to-one node mapping (i.e., every virtual node must be mapped to a distinct physical node) has been considered for NV while in our problem, multiple VMs in an SVI are allowed to be placed on (mapped to) a common server, which makes our mapping problem much harder. Therefore, none of the existing NV algorithms can be directly applied here.

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:

1) We present a general optimization framework for the SVI mapping problem such that different VMP and VLM algorithms can be incorporated into it.

2) We present 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 presented for the VMP problem.

3) Extensive simulation results are presented to justify the effectiveness of the proposed algorithms, using the real VM data traces collected from the green data center at Syracuse University [5].

The rest of this paper is organized as follows. We discuss related works in Section II. We describe the system model and formally define the problem in Section III. The proposed algorithms are presented in Section IV. The simulation results are presented in Section V and the paper is concluded in Section VI.

II. 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 [18]) have been developed to determine VMP according to resources at hosting servers such as CPU, memory, etc. The problems of determining how to place VMs with the objective of minimizing server power consumption have been studied in [8], [16], [17]. In [8], 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 [16], 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 data center. Unlike [16], 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 [20].

In [17], Verma *et al.* presented the design, implementation and evaluation of a power-aware application placement controller, pMapper, in virtual server clusters. They presented multiple ways to formulate the cost-aware VMP problem and present simple and practical algorithms to solve them. In [10], Meng *et al.* , for the first time, considered VMP with the objective of minimizing the communication cost. The problem was showed to be NP-hard. The authors designed a two-tier heuristic algorithm to solve it. In [15], 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 [9], [13], [23], [24]. In [24], the authors developed heuristic algorithms for two versions of the problem: Virtual Network (VN) assignment without reconfiguration and VN assignment with reconfiguration. In [23], 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. In [9], a fast VN mapping algorithm based on subgraph isomorphism detection was presented, which maps nodes and links during the same stage. The authors of [13], 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 very recent paper [7], in which an NV architecture called SecondNet was designed and evaluated for virtualized data centers. SecondNet introduces a centralized resource allocation algorithm for bandwidth guaranteed virtual to physical mapping. Moreover, the proposed approach was implemented using source routing and the Multi-Protocol Label Switching (MPLS) [11].

The differences between our work and these related works are summarized as follows: 1) The commercial software tools [18] and most of the recent works on VM management [8], [16], [17] did not consider bandwidth demands between VMs. 2) as mentioned above, our mapping problem

is different from the NV problems [9], [13], [23], [24] since multiple VMs in an SVI are allowed to be placed on a common server. 3) Survivability has not been addressed in NV works [9], [23], [24] or other closely related works [7], [10], [15], [20]. 4) Unlike heuristic algorithms presented in [13] which cannot provide any performance guarantees, our algorithms can ensure sufficient link bandwidth for failover traffic after any single server failure.

III. Problem Definition

TABLE I
MAJOR NOTATIONS

C_l	The available bandwidth of link $l \in L$
$G/G_X/G_P$	Virtual graph/enhanced virtual graph/physical graph
$L/N/N_R$	The set of physical links/servers/switches in the data center
P	The set of all server-server paths
V/E	The set of vertices/virtual links on G
$V_X/E_X/\hat{E_X}$	The set of vertices/virtual active links /virtual inactive links on G_X

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) [14] 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 [6]. 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 [19] and Xen [22], 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 [4] and it is also supported by commercial virtualization software, such as VMware vSphere 5 [19].

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 E 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 an 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 an SVI leaves, corresponding VMs are removed from servers and reserved bandwidth is released from links. A mapping of an 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 [21], which has been shown to be NP-hard.

IV. The Proposed Mapping Algorithms

In this section, we discuss the proposed mapping algorithms. First, we present a general optimization framework that solves the SVIM problem in two steps. Second, we present an efficient heuristic algorithm for the VMP subproblem as well as an LP-based algorithm that solves the VLM subproblem optimally. Both algorithms can be used as subroutines in the proposed framework. In addition, we present an effective algorithm to jointly solve the two subproblems.

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 III.

B. Virtual Machine Placement (VMP)

The VMP problem (with the objective of minimizing the number of active servers) can considered as the bin packing problem [21] with VMs as items and servers as bins. However, the well-known bin packing algorithms, such as the First Fit Descending (FFD), does not work well for our VMP sub-problem 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.

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
Remove infeasible mapping tuples 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

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 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.

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 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 interacts 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.

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!";

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. We omit it due to space limitation.

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:

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

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

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

V. Simulation Results

Our simulation runs were conducted based on the real VM data traces collected from the green data center at Syracuse University [5]. Specifically, we have measured the CPU utilization, memory utilization and network bandwidth utilization 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 [16], a quantile approach was used to pre-process the raw data and generate inputs for our algorithms. Specifically, we considered a day as an observation period and an hour as an observation interval. In the data trace, 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.

In each simulation run, 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. For physical data center networks, we used two widely used network topologies, fat-tree [12] and VL2 [6]. 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 data center with off-the-shelf Ethernet switches.

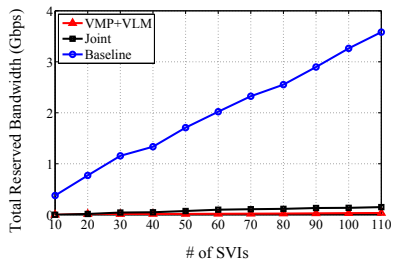
Since we are the first to study such a survivable mapping problem, we used a baseline algorithm for comparison. In this algorithm, the FFD algorithm was used for VMP because it is known to perform well in terms of minimizing the number of active servers (bins) [21]. For link mapping, all traffic between 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 inputs.

We performed simulation runs on two network topologies (fat-tree and VL2) for a non-busy hour (light resource utilizations) and a busy hour (heavy resource utilizations). The simulation results are presented in Figs. 2–5.

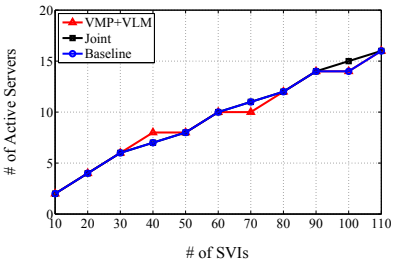
We made the following observations from these results:

1) As expected, on average, 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. This is because our LP-based VLM algorithm (used by both algorithms) finds the best way to share reserved bandwidth for each SVI. From the simulation results, 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 comparable to that of the baseline algorithm, which uses the FFD to handle VMP. The FFD is known to be one of the best algorithms for VMP [21]. 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 all the algorithm only turn on new servers when absolutely necessary.

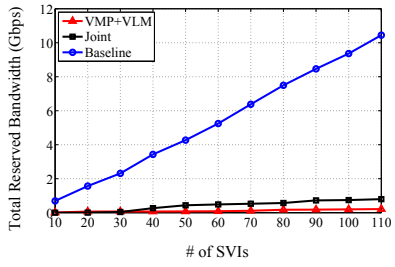


(a) Total reserved bandwidth

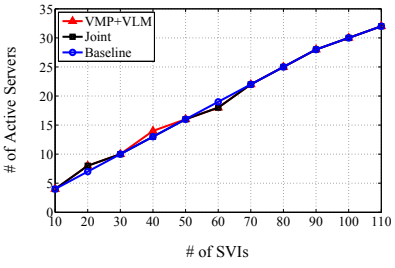


(b) The number of active servers

Fig. 2. Performance on a fat tree network with light workloads



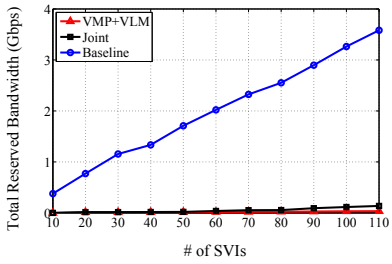
(a) Total reserved bandwidth



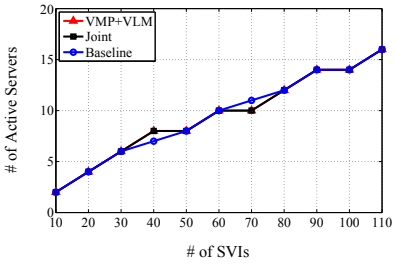
(b) The number of active servers

Fig. 3. Performance on a fat tree network with heavy workloads

3) The two typical data center topologies are similar: 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.

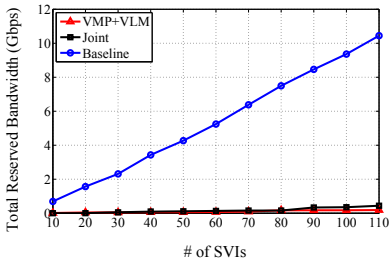


(a) Total reserved bandwidth

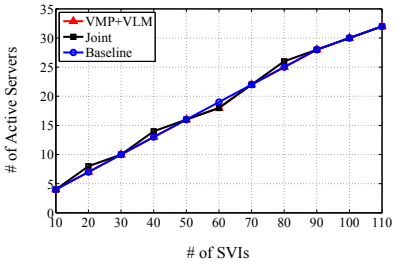


(b) The number of active servers

Fig. 4. Performance on a VL2 network with light workloads



(a) Total reserved bandwidth



(b) The number of active servers

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

VI. Conclusions

In this paper, we proposed to use a service-aware approach for resource management in virtualized data centers and consider a related optimization problem, SVIM. We found that the SVIM problem can be naturally divided into two subproblems: VMP and VLM. We present 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 present an effective algorithm to solve the two subproblems jointly. It has been shown by extensive simulation results based on the real VM data traces collected from the green data center at Syracuse University that compared with 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] B. Cully, *et al.* , Remus: High availability via asynchronous virtual machine replication, *USENIX NSDI'2008*, pp. 161–174.

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

[6] 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

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

[8] 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.

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

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

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

[12] 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.

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

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

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

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

[17] 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.

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

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

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

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

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

[23] 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.

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