Efficient Provisioning for Multicast Virtual Network under Single Regional Failure in Cloud-based Datacenters

Dan Liao[1,2], Gang Sun[1,2], Vishal Anand[3] and Hongfang Yu[1]
[1] Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China
Chengdu 611731 - China
[e-mail: gangsun@uestc.edu.cn]
[2] Institute of Electronic and Information Engineering in Dongguan, UESTC, China
[3]Department of Computer Science, the College at Brockport, State University of New York,USA
*Corresponding author: Gang Sun

Received April 14, 2014; revised May 20, 2014; accepted May 21, 2014; published July 29, 2014

Abstract

Network virtualization technology plays a key role in cloud computing, which serves as an effective approach for provisioning a flexible and highly adaptable shared substrate network to satisfy the demands of various applications or services. Recently, the problem of mapping a virtual network (VN) onto a substrate network has been addressed by various algorithms. However, these algorithms are typically efficient for unicast service-oriented virtual networks, and generally not applicable to multicast service-oriented virtual networks (MVNs). Furthermore, the survivable MVN mapping (SMVNM) problem that considers the survivability of MVN has not been studied and is also the focus of this work.

In this research, we discuss SMVNM problem under regional failures in the substrate network and propose an efficient algorithm for solving this problem. We first propose a framework and formulate the SMVNM problem with the objective of minimizing mapping cost by using mixed integer linear programming. Then we design an efficient heuristic to solve this problem and introduce several optimizations to achieve the better mapping solutions. We validate and evaluate our framework and algorithms by conducting extensive simulations on different realistic networks under various scenarios, and by comparing with existing approaches. Our simulation experiments and results show that our approach outperforms existing solutions.

Keywords: Multicast virtual network; regional failures; failure-aware; mapping; cloud-based datacenters

This research was partially supported by the National Grand Fundamental Research 973 Program of China under Grant (No. 2013CB329103), Natural Science Foundation of China grant (No. 61271171), the Fundamental Research Funds for the Central Universities (ZYGX2013J002, ZYGX2012J004, ZYGX2010J002, ZYGX2010J009), Guangdong Science and Technology Project (2012B090500003, 2012B091000163, 2012556031).
The research of Dr. Vishal Anand is supported in part by the Provost Fellowship and Scholarly Incentive Grant at the College at Brockport, SUNY.

http://dx.doi.org/10.3837/tiis.2014.07.007

1. Introduction

The traditional Internet architecture faces many challenges due to the explosive increase and diversification in user demands and Internet services. The ossification of the current Internet architecture prevents it from supporting diverse applications and services. Network virtualization technology can prevent this ossification and diversify the Internet by allowing multiple heterogeneous virtual networks (VNs) share the resources of the same underlying substrate network [1]. Users can access applications or services and infrastructure resources provisioned by cloud-based datacenters using thin clients without having to know the actual location and characteristics of the resource, or how they are delivered. In network virtualization, infrastructure providers (InP) manage the substrate platforms and lease resources to support various operations according to a usage-based pricing model [1]. Service providers (SP) rent resources from one or many infrastructure providers to serve the end users. More specifically, SPs instantiate on demand virtual machines (VMs) for the purpose of running users' tasks or applications. VMs hosted on hundreds of thousands of interconnected servers in multi-data centers allow the isolation of applications from the underlying hardware and other VMs, and also allow the customization of the platform to suit the specific needs of the end-user. Today an increasing number of applications such as web service, large-scale simulation, high-performance computing and virtual labs have been deployed in cloud-based datacenters [2]. Some examples of commercial cloud computing products include Amazon EC2, Microsoft Windows Azure platform, Google App Engine and Yahoo! Datacenters [3-6].

Network virtualization plays an important role in cloud computing and serve as a key enabler for cloud computing [7]. A network virtualization environment (NVE) [8] [9] consists of shared substrate network with computing and bandwidth resources and abstracted applications or services in the form of virtual network (VN) requests. Each VN request consists of a set of VN nodes and VN links. Each VN node requires a certain amount of node resources (i.e., CPU and memory storage resources) for executing the applications or services. Each VN link connecting two VN nodes requires a certain amount of communication bandwidth for the purpose of data and information exchange between the connected VN nodes. Multiple data centers spread across multiple geographical locations interconnected by a network constitute the infrastructure i.e., substrate network. In the substrate network each substrate node typically has a certain amount of node resources and each substrate link possess a certain amount of bandwidth resources subject to transmission delay and transmission delay variation constraints.

Since multiple VN requests are hosted on the same substrate network, and for sharing the resources of the substrate network, an efficient cloud resource provisioning strategy that intelligently uses the resources of the substrate network is important to both users and infrastructure providers (InPs). How to efficiently map or embed a VN request onto the substrate network is a challenging problem in network virtualization.

Most existing research on the problem of VN provisioning/mapping only consider the case of unicast service oriented VN requests [8-13], and accordingly aim at designing efficient provisioning strategies for such VN requests. However, in many real-world situations the applications or services are abstracted as multicast oriented virtual network requests. Multicast virtual networks are expected to support many real-time or interactive applications with diverse performance requirements, such as video-conferencing, distributed database replication, and online games. Multicasting typically scales well to a larger receiver

population and uses network infrastructure efficiently by requiring the source to send the information only once, even if it needs to be delivered to a large number of receivers. A multicast virtual network (MVN) has a tree topology with a single tier of leaf nodes. Multicast applications and traffic require that data received by the destinations are within specified delay bounds and the difference in delay (i.e., delay variation) at multiple destinations should be within specified limits or minimal [14]. Thus, the constraint on transmission delay and delay variation between any two MVN links must also be considered while mapping the MVN links. In addition, in this work we also consider substrate network failures and study the problem of how to optimally map MVN requests while considering survivability, i.e., the survivable MVN mapping (SMVNM) problem. In this paper, we formulate the survivable MVN mapping problem as a mathematical optimization problem by using mixed integer linear programming (MILP). Our optimal problem focuses on minimizing the MVN mapping cost while satisfying all of the resource requirements and constraints. Since the optimal VN embedding problem is NP-hard [8], we also propose heuristic algorithms to solve the mapping problem in a reasonable amount of time as well as achieving good results. We evaluate the performance of our heuristics by conducting extensive simulation experiments under various substrate network and scenarios.

The remainder of this paper is organized as follows. Section 2 discusses the related works. We give the detailed problem description in Section 3. In Section 4 we formulate the problem by using MILP. Section 5 presents our efficient heuristic algorithms for solving the SMVNM problem. Section 6 presents our simulation environment and simulation results that evaluate the performance of our approaches. Section 7 concludes the paper.

2. Related Work

In this section, we describe recent research related to network resource provisioning in cloud computing and the problem of virtual network mapping.

2.1 Resource Managing in Cloud Computing

Network virtualization serves as an important technology in cloud computing. The development and rapid popularization of cloud computing has led to a large amount of research on network virtualization and techniques for virtual network provisioning. In [15] the authors present a peer-to-peer approach for managing services in large scale, dynamic and evolving cloud computing environments. The research in [16-18] focuses on reducing energy consumption in data centers while fulfilling resource provisioning without violating the quality of service (QoS) or service level agreements (SLAs). Research in [19] considers several reliability problems that arise when allocating applications to processing resources in a cloud computing platform. The goal of the approach proposed in [19] is to assess the impact of the reliability constraint on the complexity of resource allocation problems.

2.2 Virtual Network Provisioning

As mentioned network virtualization encourages the development of cloud computing. There has been a significant amount of research on network virtualization and techniques for virtual network mapping/provisioning due to the increasing popularity of cloud computing. Some recent works model the VN mapping problem as mathematical optimization problem with the objective of minimizing the VN provisioning/mapping cost under a number of constraints imposed by resource capacities of substrate network components include as link resource (such as bandwidth) capacity and node resource (such as CPU, memory and storage

resources, etc.) capacities. The research in [8] proposes the online VN mapping algorithms D-ViNE and R-ViNE, with coordination between the VN node mapping and VN link mapping process, in order to increase the acceptance ratio and the revenue while decreasing the VN mapping cost. The authors in [10] addressed the problem of optimally deploying the VN as the substrate network evolves. This work focuses on minimizing the upgrading cost of virtual network while satisfying node resource constraint and path delay constraint. In [11], the authors address the key issue of how to fairly and efficiently share the resources of InPs among multiple SPs by using game theory and modeling the network resources allocation problem as a non-cooperative game model. A virtual network mapping strategy with topology-aware node ranking is proposed in [12], in which the topology-aware node resource rank is computed based on Markov Random Walks before mapping the virtual network components for improving the long-term average revenue and the VN acceptance ratio. Two shared backup VN provision schemes shared on demand and shared pre-allocation for VN embedding are proposed in [20]. By using these sharing schemes the required backup bandwidth can be reduced and more substrate resource can be left for future VN requests. The authors in [21] propose an algorithm for solving the survivable VN mapping such that the mapped virtual network can recover from substrate node failures. However, this work does not consider regional failure(s) i.e., the failure of a set of substrate nodes and links. In [22] the authors study the survivable VN mapping (SVNM) problem and use a failure-dependent protection approach that enables efficient sharing of protection resources among different failure scenarios. The approach in [22] performs well for regional failure(s), but it is not suitable for multicast service-oriented VN (MVN) request. The authors in [23] presented an online VN mapping algorithm that maximizes the number of coexisting VNs leading to good utilization and revenue of the substrate by using resource evaluation while implementing the VN mapping. The authors in [14] conducted research on the basic version of the problem of mapping multicast oriented virtual network onto substrate network and proposed an algorithm for solving this problem. However, this research cannot guarantee the survivability of MVN requests.

To the best of our knowledge none of the research on VN requests provisioning and VN design studies the problem of mapping multicast oriented VN requests onto substrate network considering their survivability against regional failures (i.e., failure of a set of nodes and links) in the substrate network.

3. Problem Statement

In this section, we describe the survivable multicast oriented VN provisioning problem that is the focus of this work. We use and build upon the frameworks discussed in [22] and [14].

3.1 MVN Request

In the cloud computing paradigm users use the substrate network resources in a pay-as-you-go manner. A user's service or application request with QoS demands (i.e., link bandwidth, delay, delay variation, node resource etc.) submitted to a cloud-based datacenter can be abstracted as a multicast virtual network request. We model the multicast virtual network request as an undirected weighted graph $G_V = (N_V, E_V, C_N, C_L, C_D, C_{DV})$, where $N_V = \{v_1, v_2 ..., v_t\}$ denotes nodes of MVN, where t is the number of MVN nodes and E_V indicates the set of MVN links. C_N represents the node resources constraint and $C_L = \{x_1, x_2 .....x_{|E_V|}\}$ represents link bandwidth

constraint, respectively. We use x_{e_v} to denote the bandwidth required by virtual link $e_v \in E_V$. We use C_D and C_{DV} to denote the constraints on the maximum delay and delay variation of substrate paths to host the virtual links.

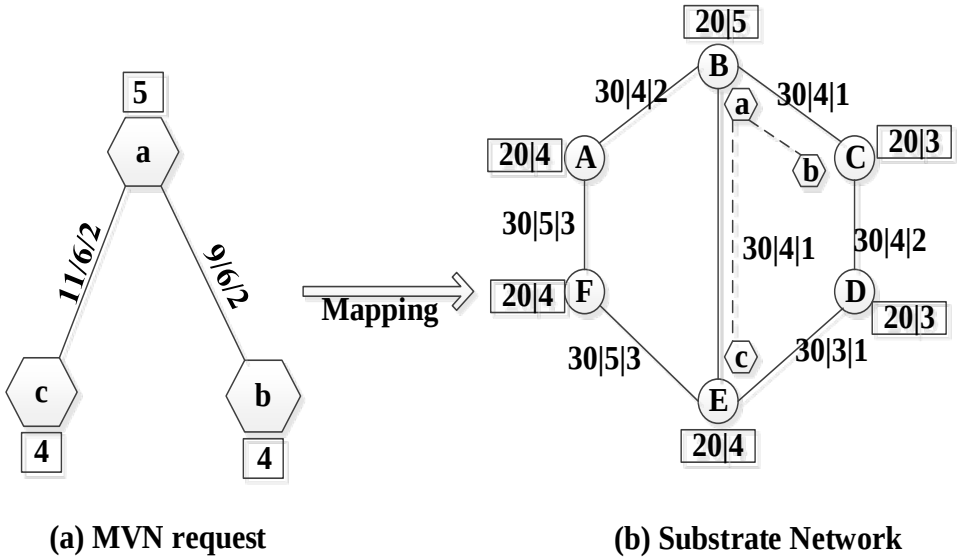


Fig. 1. Example of MVN mapping without regional failure(s)

For each multicast virtual node $n_v \in N_V$, we assume $\varepsilon(n_v)$ is the amount of node resources requested from a specific MVN node. Specifically, a multicast service-oriented VN (MVN) request can be represented by a 2-tier tree. **Fig. 1(a)** shows an example of a MVN request. In this example, a is the root node and b and c are leaf nodes of the MVN request. The numbers in the rectangles next to the MVN nodes represent the resources demands of nodes, and the numbers next to the MVN links represent the requested link resources/delay/delay variations.

3.2 Substrate Network

The substrate network consists of multiple or multi-data centers spread across multiple geographical locations that are interconnected by a network. Similar to the MVN request described above, we model the substrate network as a weighted undirected graph $G^S = (N^S, E^S, C^L, C^N)$. Where N^S and E^S represent the set of substrate nodes and the set of substrate links, respectively. We use C^N and C^L to denote the attributes of the substrate nodes and substrate links. The typical attributes of substrate nodes that are of interest include node resources such as CPU, memory and storage capacities. The typical attributes of substrate links include bandwidth capacity and delay.

For each substrate node $n_s \in N^S$, which is able to provision node resources for a MVN node, the amount of available node resource capacity and the cost of per unit of node resource is denoted as $c(n_s)$ and $p(n_s)$, respectively. For each substrate link $e_s \in E^S$, we define the amount of available link resource capacity, the cost per unit of link resource and transmission delay as $b(e_s)$, $p(e_s)$ and $d(e_s)$, respectively. **Fig 1(b)** presents an example of a substrate network where the numbers in rectangles next to the nodes represent the amount of available

node resources/cost of per unit resource of the nodes, and the numbers next to the links represent the available link resource/cost of per unit resource/transmission delay of links.

We use P to denote the set of end-to-end substrate paths, and each substrate path p, $p \in P$ is a subset of E^S. Accordingly, the end to end delay and available bandwidth resources of substrate path p, denoted as $D(p)$ and $B(p)$, can be computed as follows:

$$D(p) = \sum_{l \in p} d(l), \quad \forall p \in P \tag{1}$$

$$B(p) = \min_{l \in p} \{b(l)\}, \quad \forall p \in P \tag{2}$$

3.3 Regional Failure

In [22] the authors assume a set of possible regional failures R is given. We use this definition of regional failures in this paper. Each regional failure $r \in R$ will simultaneously destroy one or more substrate nodes and links denoted by $G(r) = (N_r, E_r)$, where $N_r \in N^S$ and $E_r \in E^S$. We note that if we allocate the resource from $G(r)$ to the MVN request, we must reserve resource from regions which do not include r for the MVN request to recover from regional failure r. **Fig 2(b)** shows an example with two regional failures $r1$ and $r2$. In the figure $G(r1) = (N_{r1}, E_{r1})$, where N_{r1} = {D, E} and E_{r1} = {C-D, D-E, B-E, E-F}, and $G(r2) = (N_{r2}, E_{r2})$ where N_{r2} = {A, F} and E_{r2} = {A-B, A-F, E-F}.

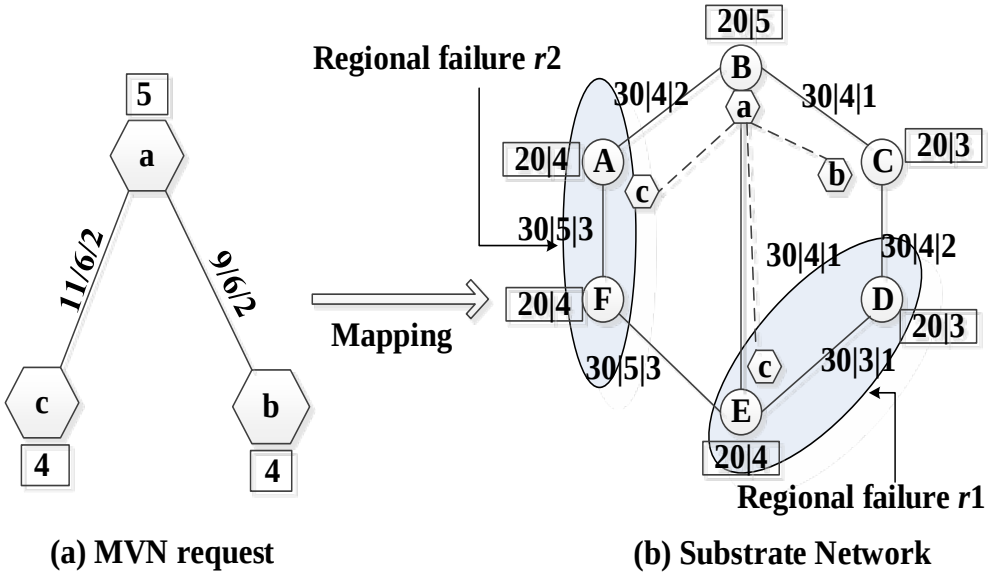


Fig. 2. Example of MVN mapping with regional failure(s)

3.4 MVN Mapping Under Single Regional Failure

The procedure of mapping a MVN request G_V onto a substrate network G^S with survivability against any single regional failure r includes four steps: initial MVN request mapping, backup substrate nodes, backup substrate paths, MVN request migration.

First, we map the MVN request without consideration of any possible regional failure. Then we need to allocate a separate substrate node and required computing resources for each MVN

node of the MVN request, as well as find paths and reserve required bandwidth resources to transmit data among the MVN nodes. If we assume that the number of MVN nodes is N_1 then exactly N_1 substrate nodes are allocated in this initial request mapping. The above MVN mapping procedure can be formulated as follows:

(1) MVN Node Mapping, denoted as:

$$M_N : (S_V, C_N) \xrightarrow{M} (N^{S*}, C^{N*})$$

$$M_N : (N_V, C_N) \xrightarrow{M} (N^{S*}, C^{N*})$$

$$M_N(v) \in N^{S*}, \quad \forall v \in N_V$$

$$R_n(M_N(v)) \geq \varepsilon(v), \quad \forall v \in N_V$$

Where $N^{S*} \subset N^S$, C^{N*} denotes the node resources allocated to the MVN request.

(2) MVN Link Mapping, denoted as:

$$M_E : (E_V, C_L) \xrightarrow{M} (P^*, C^{L*})$$

$$M_E(e) = p, \quad \forall e \in E_V, \exists p \in P^*$$

$$B(M_E(e)) \geq x_e, \quad \forall e \in E_V$$

$$D(M_E(e)) \leq C_D, \quad \forall e \in E_V$$

$$| D(M_E(i)) - D(M_E(j)) | \leq C_{DV}, \quad \forall i, j \in E_V \quad and \quad i \neq j$$

Where $P^* \subset P$, C^{L*} denotes the link resources allocated to the MVN request.

In this work, in order to mapping a MVN request onto a substrate network with survivability against any single regional failure, we assume that if a substrate node n_{s1} initially mapped for a MVN node *v* is within the failed region, we need at least another backup substrate node n_{s2} outside the failed region in order to remap this MVN node *v*. Note that the choice of the backup substrate node(s) depends on the failure scenario, and the selected backup substrate node may be within another failure region. To ensure that such a substrate node can be found in the event of a regional failure, we allocate it before a failure actually happens. Since a regional failure may cause multiple substrate nodes allocated for initial MVN request mapping to fail, a sufficient set of backup substrate nodes have to be pre-assigned. Similarly, we also need to pre-assign enough backup bandwidth resource on the substrate links to support the communication between the MVN nodes under any single regional failure. In particular, if a MVN node *v* is to be remapped at a pre-assigned spare substrate node, we must set up new spare path(s) between that spare substrate node and other surviving substrate nodes.

Fig. 2 shows an example of MVN mapping when there is a regional failure. In this example MVN nodes *a*, *b* and *c* are initially (under no failures scenario) mapped onto the substrate nodes *B*, *C* and *A* communicating through substrate link *B*-*C* and *B*-*A*, respectively. Regional failure *r*1 destroys substrate nodes *D* and *E*, as well as destroys substrate links *C*-*D*, *D*-*E*, *B*-*E*, *E*-*F* which do not influence the initial mapping, thus we need no backup nodes and links resource for recovering this failure. Regional failure *r*2 destroys substrate nodes *A* and *F*, as well as destroys substrate links *A*-*B*, *A*-*F*, *E*-*F*. The substrate network remaps the MVN request such that the MVN node *c* is now mapped onto substrate node *E*, as well the MVN nodes a and

b are still mapped onto *B* and *C*. The new paths are *B*-*E* and *B*-*C*. So, we have to migrate the MVN node *c* from *A* to *E* and migrate communications on MVN link *c*-*a* from *B*-*A* to *B*-*E* for recovering from regional failure *r*2.

3.5 Resource Sharing

Since in this work we assume that only one regional failure occurs at any one time, the resources reserved on the substrate nodes and substrate links may be shared among the different failure scenarios.

The MVN mapping solution without consideration of any regional failure can be described as follows: a one-to-one mapping from N_V to N^S and mapping of each $e_v \in E_V$ to a path in G^S. Similarly, for each regional failure $r \in R$, a survivable mapping solution consists of a mapping from N_V to N^S-N_r (one-to-one) and mapping of each $e_v \in E_V$ to a path in G^S-$G(r)$.

Accordingly to ensure survivability under any regional failure, the total node resources that are allocated for the MVN request on substrate node *n*, denoted by rn_n, is the maximum of all node resources under any failure. Similarly, the total bandwidth resources allocated for the MVN request on substrate link *l*, denoted by rl_l, is the maximum of all bandwidth resources under any failure. rn_n and rl_l can be calculated by using the equations below:

$$rn_n = \max\{rn_{n,0}, rn_{n,1}, ..., rn_{n,|R|}\} \tag{3}$$

$$rl_l = \max\{rl_{l,0}, rl_{l,1}, ..., rl_{l,|R|}\} \tag{4}$$

where, $rn_{n,r}$ and $rl_{l,r}$ are the node and link resources allocated on substrate node *n* and substrate link *l* under *r*-th failure in *R*, Note that *r*=0 means no failure scenario.

4. MILP Formulation

In this section, we model the problem of survivable MVN mapping as a mathematical optimization problem by using mixed integer linear programming (MILP).

4.1 Problem Description

Given: A substrate network $G^S = (N^S, E^S, C^L, C^N)$, a MVN request $G_V = (N_V, E_V, C_N, C_L, C_D, C_{DV})$ and a list of all possible regional failures *R*.

Problem: How to jointly allocate node and link resources of substrate network, including the backup resources, such that the total cost of survivable mapping of the MVN request, i.e., sum of the total cost of node and link resource is minimized.

4.2 MILP Formulation for SMVNM Problem

In this paper, in order to formulate the MILP model for SMVNM problem, we apply a graph transformation on substrate network GS similar to [21] to get an augmented graph. We add $|N_V|$ extended nodes into G^S; each extended node added corresponds to a MVN node of the MVN requests. Each extended node is set to have infinite resource capacity. We assume that each extended node v^* added into GS is connected to all available substrate nodes. We call the links connecting extended nodes and substrate nodes as extended links. We assume that the bandwidth resources on each extended link are unlimited and the delay of each extended link

is zero. Each extended node v^* is unaffected by any regional failure in R. However, for each extended link connecting an extended node v^* with a substrate node n_s, if n_s is within a region $r \in R$, then that extended link is also inside the failure region r. Accordingly we achieve the augmented graph as $G^* = (N^*, E^*)$, where $E^* = E^S \cup \left\{e \middle| Sc(e) \in N^S, Dt(e) \in N_V\right\} \cup \left\{e \middle| Sc(e) \in N_V, Dt(e) \in N^S\right\}$ and $N^* = N^S \cup N_V$, where $Sc(e)$ and $Dt(e)$ denote source and sink node of link e, respectively.

Based on the above transformation, the SMVNM problem can be formulated as a mixed integer multi-commodity flow (MCF) problem, in which the bandwidth requirement on each virtual link $e_v \in E_V$ of MVN request G_V is considered as a commodity. In each regional failure r, each extended node must choose only one extended link to connect itself and an available substrate node. This constraint makes sure that each virtual node corresponding to the extended node is mapped onto one substrate node (one-to-one mapping) under each regional failure. We present the detailed MILP formulation for survivable MVN mapping problem below. The key notations used in our model are summarized in **Table 1**.

Table 1. Summary of key notations

Notation	Meaning
E^*	The set of all links of augmented graph G^*.
N^*	The set of all nodes of augmented graph G^*.
R	The set of all possible regional failures in the network. ($r_0 \in R$ means no failure scenario)
$\delta_{e,r}$	Binary variable denoting whether link $e \in E^*$ is assigned under the failure of region r; 1 if there is resource demand on link $e \in E^*$ under the failure of region r, and 0 otherwise.
rl_e	Total amount of link resources provisioned by substrate link e to support either initial mapping or failure tolerance.
rn_m	Total amount of node resources provisioned by substrate node m to support either initial mapping or failure tolerance.
$p(e_s)$	Cost of per unit resource of substrate link e_s.
$p(n_s)$	Cost of per unit resource of substrate node n_s.
C_D	The max delay of MVN links.
C_{DV}	The delay variation of MVN links.
$\varepsilon(n_v)$	The amount of resources required by MVN node n_v.
$R_n(n_s)$	The available resources of substrate node n_s.
X	A big constant.
$Sc(e)$	The source of link e of augmented graph.
$Dt(e)$	The sink of link e of augmented graph.
$\eta(e;r)$	Binary value to indicate whether link e is in r, where $e \in E^*$ and $r \in R$.
$b(e)$	The resource required by MVN link e.
$\lambda(m;r)$	Binary value to indicate whether node m is in r, where $m \in N^S$ and $r \in R$.
$M_E(e)$	The path in substrate network that hosting MVN link e.
$D(p)$	The delay of path p.

$R_e(e)$	The available resources of link e, where $e \in E^*$.
$fl_{e_s,r}^{e_v}$	Variable denoting the total required resource on link e_s for MVN link e_v under the failure of region r.

Objective function:

$$Minimize\left\{\sum_{e\in E^S}(rl_e \times p(e)) + \sum_{m\in N^S}(rn_m \times p(m))\right\} \tag{5}$$

The objective function in (5) tries to minimize the total provisioning cost (i.e., the cost of substrate node and link resources) for a MVN request considering survivability.

Constraints:

$$\sum_{Dt(e)\in N^S}\delta_{e,r} = 1, \forall Sc(e) \in N_V, \forall r \in R \tag{6}$$

$$\sum_{Sc(e)\in N_V}\delta_{e,r} \le 1-\lambda(Dt(e);r), \forall Dt(e) \in N^S, \forall r \in R \tag{7}$$

$$\delta_{e1,r} = \delta_{e2,r}, \quad \forall e1, e2 \in E^*, Sc(e1) = Dt(e2), Dt(e1) = Sc(e2) \tag{8}$$

Constraint (6) and (7) are related to the mapping between the virtual and substrate facility nodes. Constraint (6) ensures that only one substrate node is selected for mapping the virtual node of MVN request under any failure r, and constraint (7) ensures that only one virtual node is mapped onto a substrate node under any failure r. Constraint (8) makes sure that the binary variables for the two links, in the opposite directions between any node pair of the augmented graph G^* have the same value.

$$\begin{aligned}&\sum_{e_v\in E_V}(fl_{e1,r}^{e_v} + fl_{e2,r}^{e_v}) \le 2\times R_e(e1)\times\delta_{e1,r}, \quad \forall r \in R\\ &\forall e1, e2 \in E^*, Sc(e1) = Dt(e2), Sc(e2) = Dt(e1)\end{aligned} \tag{9}$$

$$\begin{aligned}&\delta_{e,r}\times\varepsilon(Sc(e)) \le R_n(Dt(e)), \quad \forall r \in R\\ &\forall e \in E^*, Sc(e) \in N_V, Dt(e) \in N^S\end{aligned} \tag{10}$$

Constraints (9) and (10) are link and node capacity constraints that ensure that the total required bandwidth and node resources must remain within the residual link bandwidth and node resource capacities.

$$\begin{aligned}&\sum_{\substack{e\in E^*,\\ Sc(e)=m}} fl_{e,r}^{e_v} - \sum_{\substack{e\in E^*,\\ Dt(e)=m}} fl_{e,r}^{e_v} = 0, \quad \forall r \in R\\ &\forall e_v \in E_V, \forall m \in N^* \setminus \{Sc(e_v), Dt(e_v)\}\end{aligned} \tag{11}$$

$$\sum_{\substack{e\in E^*,\\ Sc(e)=Sc(e_v)}} fl_{e,r}^{e_v} = b(e_v), \quad \forall e_v \in E_V, \forall r \in R \tag{12}$$

$$\sum_{\substack{e\in E^*,\\ Dt(e)=Dt(e_v)}} fl_{e,r}^{e_v} = b(e_v), \quad \forall e_v \in E_V, \forall r \in R \tag{13}$$

$$\sum_{\substack{e\in E^*,\\ Dt(e)=Sc(e_v)}} fl_{e,r}^{e_v} = 0, \quad \forall e_v \in E_V, \forall r \in R \tag{14}$$

$$\sum_{\substack{e\in E^*,\\ Sc(e)=Dt(e_v)}} fl_{e,r}^{e_v} = 0, \quad \forall e_v \in E_V, \forall r \in R \tag{15}$$

Constraints (11)-(15) are flow conservation constraints that constrain the net flow of a node $n^* \in N^* \setminus \{Sc(e_v), Dt(e_v)\}$ to zero and the net flow of a node $n^* \in \{Sc(e_v), Dt(e_v)\}$ to $b(e_v)$, where $e_v \in E_V$.

$$\delta_{e,r} \le 1-\eta(e;r), \quad \forall r \in R, \forall e \in E^* \tag{16}$$

$$\sum_{Dt(e)\in N^*} \delta_{e,r} \le X*(1-\lambda(Sc(e);r)), \forall Sc(e) \in N^S, \forall r \in R \tag{17}$$

Constraint (16) ensures that only when link e is available under regional failure *r*, it is permitted to allocate resources. Similarly, constraint (17) ensures that only when substrate node $Sc(e)$ is available under regional failure *r*, its adjacent links are permitted to allocate resources.

$$rl_e \ge \sum_{e_v\in E_V} fl_{e,r}^{e_v}, \quad \forall e \in E^S, \forall r \in R \tag{18}$$

$$rn_m \ge \sum_{\substack{Sc(e)\in N_V\\ Dt(e)=m}} \delta_{e,r} * \varepsilon(Sc(e)), \quad \forall m \in N^S, \forall r \in R \tag{19}$$

Constraint (18) makes sure that the total required bandwidth resource on each substrate link is no less than the required bandwidth resource of the MVN link under any failure. Constraint (19) guarantees that the total required node resource on each substrate node is no less than the required resource of the MVN node under any failure.

$$D(M_E(e)) \le C_D, \forall e \in E_V \tag{20}$$

$$\left|D(M_E(e)) - D(M_E(f))\right| \le C_{DV}, \forall e, f \in E_V \tag{21}$$

Constraints (20) and (21) are transmission delay constraints. Where constraint (20) checks that for any MVN edge its transmission delay does not exceed the maximum allowed delay. The transmission delay variation among paths hosting the MVN links is controlled within an acceptable range by equation (21).

5. Algorithms Design

Since the survivable MVN provisioning is an NP-hard problem, using the MILP to find an

optimal solution for this problem is computationally intractable. For achieving survivable MVN mapping in a reasonable time, we propose efficient heuristics to solve this NP-hard problem. Our main idea is to start with an "original" mapping without consideration of any regional failure, and then find the backup resource for each regional failure. More specifically, our algorithm consists of three parts below:

a) Achieve "original" mapping with non-survivable MVN mapping algorithm.

b) Achieve survivable MVN mapping with non-survivable MVN mapping algorithm for each regional failure by fixing the substrate network.

c) Eliminate redundant resource and mappings for all mappings above with the strategy of resource sharing in section 3 and min-cost set cover algorithm.

5.1 Non-Survivable MVN Mapping Algorithm

The problem of non-survivable MVN mapping is different from the non-survivable VN mapping problem as now delay and delay variation constraints must be considered while mapping. In this paper, we design the non-survivable MVN mapping algorithm (NSMVNM) while considering the topology and delay constraints of the MVN request.

Algorithm 1: NSMVNM algorithm

Input: 1. Substrate network $G^S = (N^S, E^S, C^L, C^N)$;
2. A MVN request $G_V = (N_V, E_V, C_N, C_L, C_D, C_{DV})$.

Output: Mapping solution M

1: Initialize the lists $UMN^S = \varnothing$, and $UMN_V = N_V$.
2: Sort the substrate nodes in descending order according to Equation (22) and store in UMN^S.
3: Find the root node of G_V denoted as v.
4: **for** each $n_s \in UMN^S$, **do**
5: calculate and record $Cost(v \rightarrow n_s)$ according Equation (23).
6: **end for**
7: Mapped v onto n_s with minimum $Cost(v \rightarrow n_s)$, update M, $UMN_V = UMN_V - v$ and $UMN^S = UMN^S - n_s$
8: **for** each $v_L \in UMN_V$, **do**
9: **for** each $n_k \in UMN^S$, **do**
10: calculate and record $Cost(v_L \rightarrow n_k)$ according Equation (24).
11: **end for**
12: Mapped v_L onto n_k with minimum $Cost(v_L \rightarrow n_k)$, update M $UMN_V = UMN_V - v_L$ and $UMN^S = UMN^S - n_k$, find a candidate path p connecting n_k and n_s, then stored in M_p.
13: **end for**
14: Adjust the paths in M_p ensure that the delays of paths in M_p are all falling into [$D_{\max}$-C_{DV}, $D_{\max}$].
15: Update M according to M_p.
16: **return** M

Fig. 3. Pseudo code of the NSMVNM algorithm

Since the MVN request can be represented by a 2-tier tree, the root node has maximum connectivity among all MVN nodes. We first map the root node onto the substrate network and then map the leaf nodes based on the mapping of the root node. While mapping each leaf node we also map each MVN link to a path (a set of substrate links) in substrate network. The objective is to minimize the total cost including computing cost and bandwidth cost. Note that the path found while mapping the leaf node is not necessarily the working path for the MVN link because the delay variation may not be satisfied. We call the paths found while mapping

leaf nodes as candidate paths, which have to meet the delay variation constraint.

A detailed description of the NSMVNM algorithm is shown in **Fig. 3**, where the notation $D_{\max}$ denotes the maximum transmission delay of the substrate paths for all of MVN links.

NSMVNM algorithm includes three steps: a) the mapping of root node of MVN request; b) leaf nodes mapping; c) the mapping of MVN links. The two sets UMN_V and UMN^S are used to ensure the one-to-one mapping between the MVN nodes and the substrate nodes. Before the mapping procedure we introduce some pre-optimizations (i.e., sorting the substrate nodes) on substrate network. In order to elaborate on this strategy of sorting nodes, we define $Con(n_s)$ as the connectivity of substrate node n_s which is equal to the number of adjacent nodes of n_s. $Adj(n_s)$ is the set of adjacent nodes of n_s. Here we can calculate $Con(n_s)$ by using following equation.

$$Con(n_s) = | Adj(n_s) | \tag{22}$$

While mapping the root node of the MVN request we must calculate the possible mapping cost for each available substrate node, such that we can find the substrate node with minimum cost to be the mapped node of root node. In this work, we use the following equation to calculate the possible mapping cost $Cost(v_L \rightarrow n_k)$.

$$Cost(v \rightarrow n_s) = (p(n_s) + (MC - Con(n_s))) * \varepsilon(v) \tag{23}$$

where MC is $Con(n_i)$ with n_i having the maximum connectivity among all substrate nodes. Equation (23) guides the mapping of the root node onto the substrate node with large connectivity.

Similar to the mapping of root node, we also need to calculate the possible mapping cost for each available substrate node while mapping the leaf nodes, such that we can find the substrate node with minimum cost to be the mapped node of leaf node. The cost for a leaf node mapping onto a substrate node can be calculated by the following equation.

$$Cost(v_L \rightarrow n_k) = CNn_k + CPn_k \tag{24}$$

where CNn_k is the node resource cost and CPn_k is the bandwidth cost computed as follows.

$$CNn_k = p(n_k) * \varepsilon(v_L) \tag{25}$$

$$CPn_k = \sum_{e \in P} p(e) * x_{e_v} \tag{26}$$

where P is the substrate path connecting n_s (the substrate node hosting the root node) and n_k, e_v is the MVN link connecting v_L and v (the root node of MVN request).

If there exists a path whose delay does not fall into [Dmax-CDV, Dmax] window, we use the k-shortestpaths [24] to find k paths and choose a shortest one whose delay is in [$D_{\max}$-C_{DV}, $D_{\max}$] to substitute for the original path.

5.2 Survivable MVN Mapping Algorithm

For achieving survivable MVN mapping with NSMVNM algorithm, we use the framework

similar to [22]. In this section, we will reuse part of SOUM* algorithm to design our survivable MVN mapping algorithm.

Similar to SOUM*, we decompose the SMVNM problem into $|R|$ separate NSMVNM problems. For each regional failure ri, we fix the substrate network $G^S = G^S\text{-}G(r\text{i})$ and then achieve the mapping by using algorithm NSMVNM. Note that $r0$ means no failure ($G(r0) = \varnothing$). The algorithm based on SOUM* proposed here for survivable MVN mapping is named SOUM*-M. Since SOUM*-M deal with the $|R|$ problems independently, some mapping solutions or resources may be unnecessary, which we can remove from the final mapping solution(s) by the greedy min-cost set cover algorithm. To describe the algorithm SOUM*-M clearly, we use the following definitions. $M_S = \{M_0, M_1, M_2, ..., M_{|R|}\}$ is the mapping set corresponding to failure set $R = \{r_0, r_1, r_2, ..., r_{|R|}\}$. $c(M_i)$ is used to denote the cost of mapping $M_i \in M_S$.

A detailed description of the SOUM*-M algorithm is shown in **Fig. 4**.

Algorithm 2: SOUM*-M algorithm

Input: 1. Substrate network $G^S = (N^S, E^S, C^L, C^N)$;
2. A MVN request $G_V = (N_V, E_V, C_N, C_L, C_D, C_{DV})$;
3. A list of regional failures R.

Output: The min-cost mapping set M^* to recover from any specified regional failures.

1: Initialize the cost $c(M_i)$ of all mappings in M_S to zero.
2: **for** each $ri \in R$, **do**
3: Update $G^S = G^S\text{-}G(r\text{i})$.
4: Call **NSMVNM** algorithm to get the mapping solution M_i.
5: Calculate $c(M_i)$ according to M_i.
6: **end for**
7: Call Greedy min-cost set cover algorithm and get min-cost mapping set M^*.

Fig. 4. Pseudo code of the SOUM*-M algorithm

Algorithm 3: Greedy min-cost set cover algorithm

Input: 1. All mapping solutions M_S;
2. The list of regional failures R.

Output: The min-cost mapping set M^* to recover from any specified regional failures.

1: Initialize the set $M^* = \varnothing$.
2: **while** $R \neq \varnothing$, **do**
3: Pick up mapping M selected by equation (27).
4: Add M to M^*, and remove all failures that M recovers from R.
5: **end while**
6: **Return** min-cost mapping set M^*.

Fig. 5. Pseudo code of the Greedy min-cost set cover algorithm

For the Greedy min-cost set cover algorithm we have:

$$Min\{c(M)/|U(M)|\}, M \in M_S \tag{27}$$

where $U(M)$ is the subset of regional failures that can be recovered by the mapping M.

Fig. 6 explains the relation between M_S and R, and gives an example of Greedy min-cost set

cover algorithm. **Fig. 6** shows a scenario with two regional failures r_1 and r_2 and assumes that M_0 is the no-failure mapping solution (i.e., resource allocations), and M_1 and M_2 are the mappings for regional failures r_1 and r_2, respectively. Since r_2 does not affect M_0, M_0 can still be used to recover from regional failure r_2. Similarly, since r_1 does not affect M_1, hence M_1 can be used to recover from r_1. By the same argument we can deduce that M_2 can recover from both r_1 and r_2. Assuming that M_2 has a lower cost than $\boldsymbol{M_0 \cup M_1}$, then we should choose mapping M_2 as our solution to recover from $r_1$and r_2, and M_0 and M_1 are redundant and not required any more. Then M_2 should be added to M^* and $r_1$and r_2 should be removed from R.

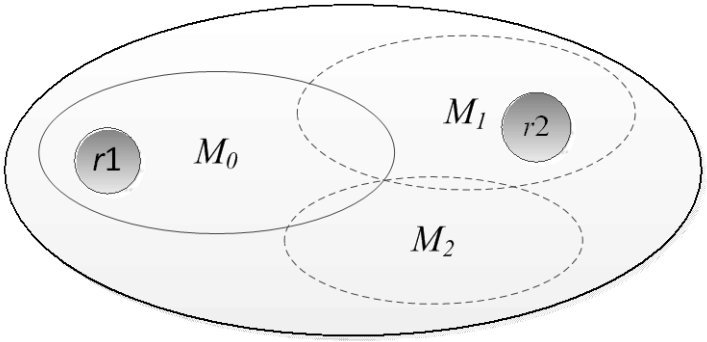


Fig. 6. Example of the Greedy min-cost set cover algorithm

5.3 Improved Non-Survivable MVN Mapping Algorithm

Since the number of leaf nodes of a MVN request is typically greater than 1, the order of the leaf nodes in UMN_V in the NSMVNM algorithm may affect the final mapping result. Since the regional failure R is given before mapping, and if we can make the mapping aware of the failures, we can achieve better mapping solutions.

Based on this we introduce two strategies to improve the NSMVNM algorithm proposed in **Fig. 3**.

(1) Sort on all leaf nodes

For achieving a better mapping result, we sort the leaf nodes of MVN request in decreasing order of resource requirements ($DR(v_L)$). This resource requirement is decided by the leaf node, and also dependent on the MVN link (e_v) connecting it to the root node. We define $DR(v_L)$ as follows.

$$DR(v_L) = \varepsilon(v_L) + \lambda * x_{e_v} \tag{28}$$

where λ is the ratio of unit bandwidth cost to the unit node resource cost. Intuitively, mapping "larger" resource-hungry MVN nodes first can give these MVN nodes higher priorities to reserve resources with lower cost and in turn, reduce the total overall cost.

(2) Failure-Awareness in NSMVNM

In order to make NSMVNM aware of the failures while mapping the MVN request, we introduce the *virtual* unit cost of substrate resource. We define $p^*(n_s)$ and $p^*(e_s)$ to present virtual cost of per unit resource of substrate node n_s and virtual cost of per unit resource of substrate link e_s, respectively. Then $p^*(n_s)$ and $p^*(e_s)$ can be calculated as follows.

$$p^*(n_s) = p(n_s) + \alpha * AF(n_s) \tag{29}$$

$$p^*(e_s) = p(e_s) + \alpha * AF(e_s) \tag{30}$$

Where α is a weight factor that is used to control the mapping procedure; when α=0 the mapping proceeds oblivious to the failure of any region, and if α>0 the mapping procedure avoids the failures of the regions. $AF(n_s)$ is the number of regional failures that affect substrate node n_s. The main idea is to select the substrate nodes and links that not only have low cost, but also are less likely to be affected by the various given regional failures.

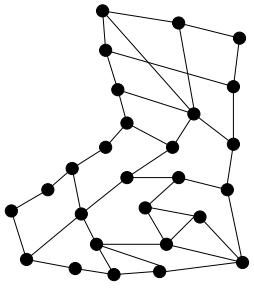
While mapping with the NSMVNM algorithm, we sort all leaf nodes using equation (28) and use the virtual cost calculated by equations (29) and (30) instead of $p(n_s)$ and $p(e_s)$. We call the algorithm SOUM*-M with these two strategies as SOUM*-MF.

6. Simulation and Results

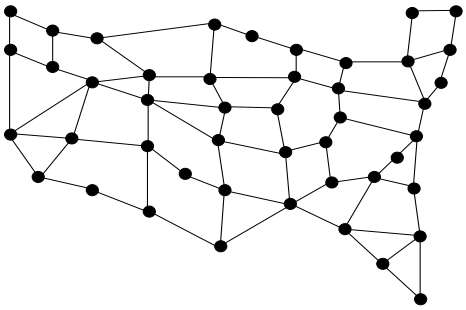
In order to evaluate and validate the effectiveness of our proposed framework and algorithms we have conducted extensive simulations. In this section, we first describe the simulation environment, and present several performance parameters used in our simulations that show the advantages of our algorithms under different substrate networks and different scenarios. Then, we present our main simulation results. In our simulation experiments we compare the MVN mapping cost, average number of migrations and unrecoverable ratio for all failure regions of our algorithms with existing approach.

6.1 Simulation Environment

We use two real networks namely the England Network and USANET shown in **Fig. 7** as the substrate network in our simulations. These two networks vary in terms of the number of nodes, links and connectivity and thus provide a good basis for evaluation of our approaches. The computing capacity at substrate nodes and bandwidth capacity on the substrate links are all equal to 1000 units under the unconstrained capacity scenario and follow a uniform distribution from 50 to 150 units when the capacity is constrained. We also assume that the per unit node resource cost and the per unit link bandwidth cost are all equal to 1 unit. The transmission delay of each substrate link is equal to 1 unit.



(a) Net-1: England Network Topology with 27 nodes and 41 links



(b) Net-2: USANET with 46 nodes and 76 links

Fig. 7. Substrate networks used in our simulation

In our simulations, the node numbers of MVN requests vary from 4 to 12 in step of 2 to simulate the various scenarios of MVN requests. The resource requirement of each MVN node is randomly generated from 10 to 30 units. The resource requested by each MVN link follows a uniform distribution from 10 to 50 units. We assume that the constraint on transmission delay of each MVN link is less than 6 units, and the delay variation of each two MVN links must not exceed 2 units in our simulation.

In our simulation experiments, we have not only compared the performances of our algorithms SOUM*-M and SOUM*-MF with the algorithm SOUM* proposed in [22], but also compared the performance of SOUM*-MF for different values of α.

For each regional failure we randomly choose various numbers of adjacent nodes to fail (e.g., three nodes). Since the scale of Net-2 is larger than Net-1, in our simulation experiments, we define 5 failure regions for Net-1 and 8 failure regions for Net-2.

We have implemented the algorithms that are compared in our simulations by using Microsoft Visual Studio 2005 and the C++ programming language. All algorithms are run on a computer with a memory of 4 GB and 3.2-GHz CPU.

6.2 Performance Metrics

The MVN mapping cost, average number of migrations and unrecoverable ratio are three key parameters in the algorithm for MVN request mapping under any single regional failure. Accordingly, we use the following three metrics to evaluate the performance of the algorithms compared in our work. The first two are applicable when the amount of computing and bandwidth resources available in substrate network is sufficient to recover from any failure, while the last is applicable when the resources are constrained or limited.

(1) **Cost**: It is the total cost of reserving substrate network resources for mapping the MVN request to tolerate any regional failure in the substrate network. More specifically, this is the sum of computing cost on all substrate nodes and bandwidth cost on all substrate links for provisioning a MVN request while considering its survivability.

(2) **Average Number of Migrations**: It is the average number of migrations, which is defined as follow:

$$AM = \sum_r am_r / |R|,$$

where am_r is the sum of the number of MVN nodes and MVN links that need to be migrated to new substrate node and substrate links under failure r under unconstrained resource. Migrations produce additional cost.

(3) **Unrecoverable Ratio**: This is the ratio of the number of unrecoverable failure scenarios to the total number of failure scenarios when the substrate network has limited constrained

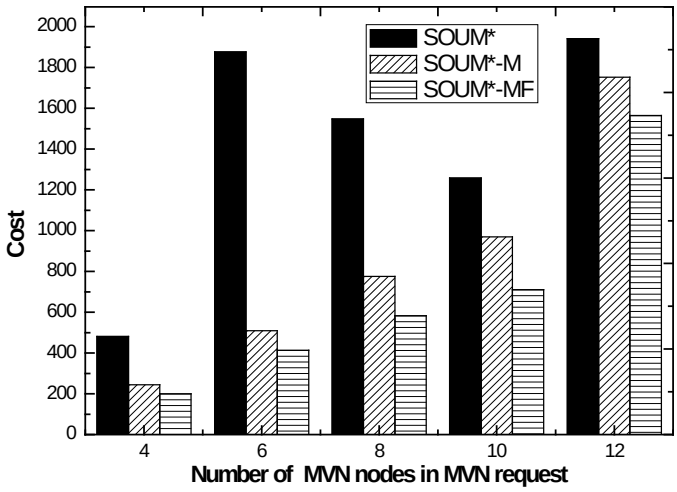
capacity.

Table 2. Algorithms compared

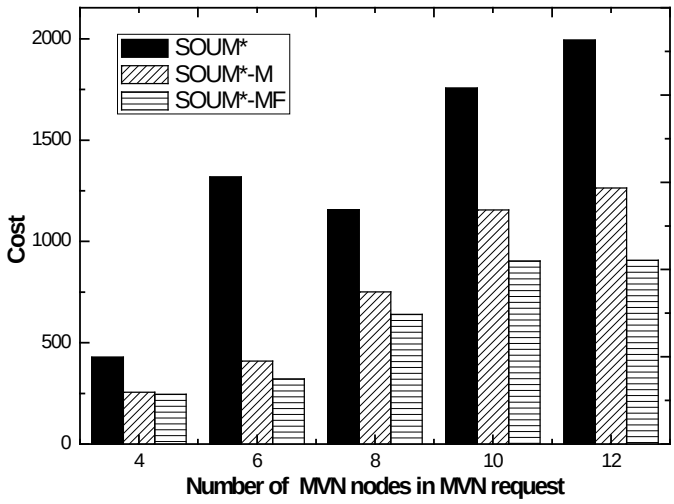
Notations	Brief description
SOUM*	Separate optimization with unconstrained mapping & redundancy elimination algorithm proposed in [21].
SOUM*-M	Algorithm proposed in this work.
SOUM*-MF	The improved SOUM*-M algorithm proposed in this work.

6.3 Simulation Experiment Results

We have compared the performance of three algorithms in **Table 2** in terms of mapping cost, average number of migrations and unrecoverable ratio for MVN mapping under single regional failure in different substrate networks.



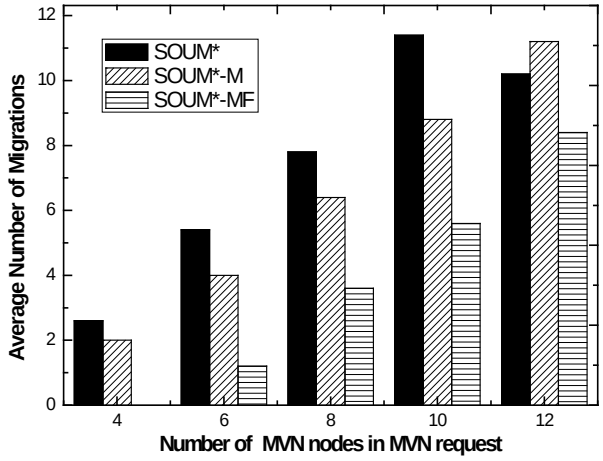
(a) Simulation Results on Net-1



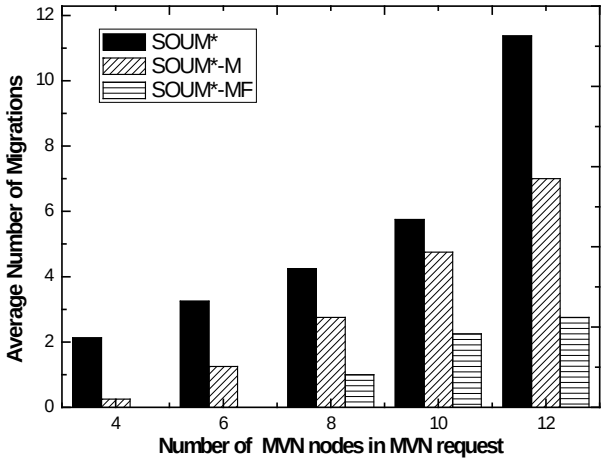
(b) Simulation Results on Net-2

Fig. 8. Total mapping cost achieved in different substrate networks with various size of MVN request

Fig. 8 shows the comparison of total mapping cost obtained by the two approaches proposed in this work and in [21], where the MVN node number varies from 4 to 12 in increments of 2. From the figure we see that the total mapping cost of SOUM*-M and SOUM*-MF are much lower than SOUM* in two different substrate network. This is because the SOUM*-M and SOUM*-MF algorithms introduce pre-optimizations such as sorting of substrate nodes before mapping which leads to better mapping solutions. **Fig. 8(a)** and **8(b)** show that SOUM*-MF performs better than SOUM*-M in both Net-1 and Net-2. This is because that algorithm SOUM*-MF introduce more pre-optimizations (i.e., sorting on MVN nodes and failure-awareness) on MVN request. The fixed parameter α defined in equations (29) and (30) helps SOUM*-MF achieve a better solution by avoiding more failures which reduces the required backup resources compared to SOUM* and SOUM*-M.



(a) Simulation Results on Net-1

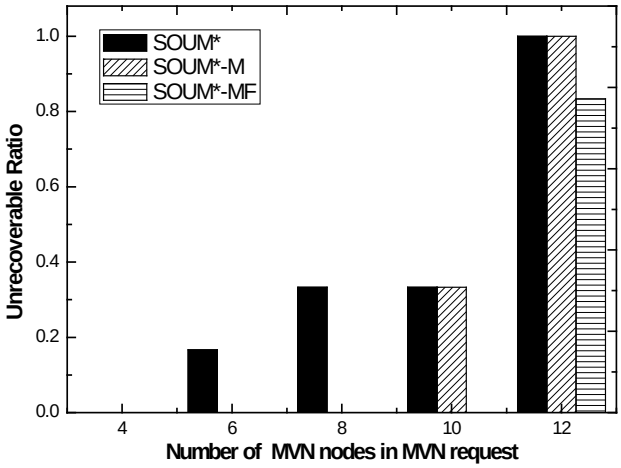


(b) Simulation Results on Net-2

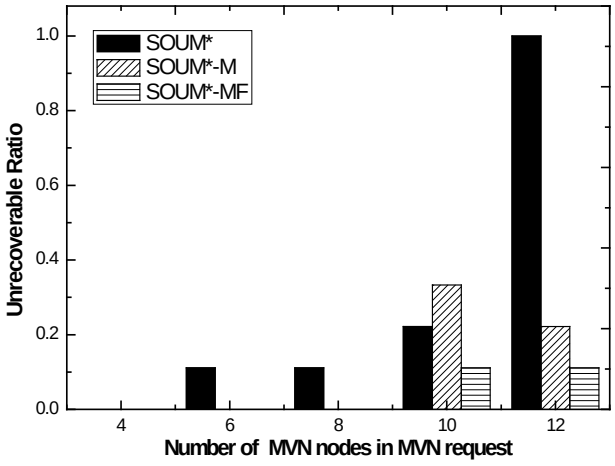
Fig. 9. Average number of migrations achieved in different substrate networks with various size of MVN request

Fig. 9 presents the average number of migrations of the three algorithms in **Table 2**, where the MVN node number varies from 4 to 12 in increments of 2. From **Fig. 9** we can see that our algorithms SOUM*-M and SOUM*-MF incur fewer average number of migrations than

SOUM* in both Net-1 and Net-2. Thus implying that SOUM*-M and SOUM*-MF can recover from any single regional failure with smaller migration cost than SOUM*. This is because the fixed parameter α in SOUM*-MF can guide the mapping away from the failure regions. All algorithms perform better in **Fig. 9(b)** compared to **Fig. 9(a)**, this is because Net-2 is bigger than Net-1, and can provide more available substrate nodes and links which offer more optimized mapping solutions. From **Fig. 9(a)** we can see that in the case of large MVN requests with 12 nodes, the average number of migrations for SOUM*-M is higher than SOUM*. However, the cost of SOUM*-M is lower than SOUM* (see **Fig. 8(a)**). This is because when the size of the MVN request is large fewer migrations can lead to the use of longer more resource expensive substrate paths while mapping the MVN links.



(a) Simulation Results on Net-1



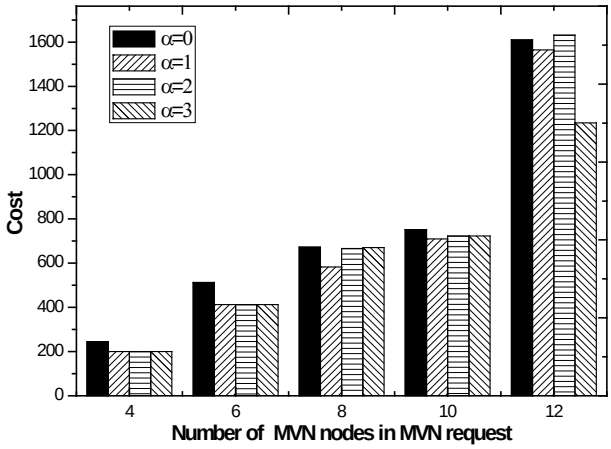
(b) Simulation Results on Net-2

Fig. 10. Unrecoverable ratio on different substrate networks with various size of MVN request

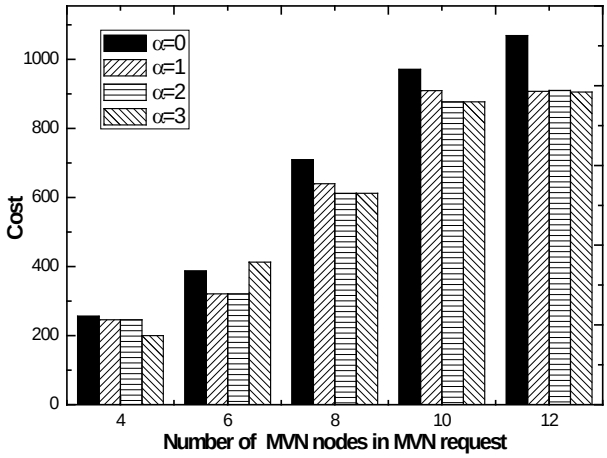
Fig. 10 compares the unrecoverable ratio of three compared algorithms when the resource capacity of the substrate network is constrained. We can see that our algorithms SOUM*-M and SOUM*-MF lead to better unrecoverable ratio than SOUM*. This is due to that the optimized strategy introduced into SOUM*-M and SOUM*-MF lead to not only lower mapping cost but also less migrations, as shown in **Fig. 8** and **Fig. 9**. The lower mapping cost means higher probability of more successful mapping when the substrate network has limited

resource capacity. On the other hand, with fewer migrations more failure regions can be covered using the original mapping. Thus with a fixed amount of resources SOUM*-MF can recover from more failure regions compared to SOUM*, and accordingly the unrecoverable ratio of SOUM*-MF is lower than SOUM*. From **Fig. 10** we also see that the performance in **(a)** is better than **(b)**, this is because larger substrate network provides more optimized mapping solutions that leads to lower unrecoverable ratio.

Fig. 11 shows the effect of parameter α on mapping cost of algorithm SOUM*-MF under various networks. As shown in **Fig. 11**, in this set of simulations the resources capacities of the substrate networks are sufficient to recover from any failure. We can see that there is a significant difference between the cost obtained by the algorithm SOUM*-MF with different α. In the case with no α (i.e., $\alpha = 0$), the cost of SOUM*-MF algorithm is higher than when α is not 0. The fixed value of α for SOUM*-MF depends on the unit cost of the resource of substrate network and the number of failure regions. However, fixed α makes SOUM*-MF achieve better performance in term of mapping cost. This is because parameter α leads to mapping that avoids failures, which lowers the backup resources necessary for recovering from any failure.



(a) Simulation Results on Net-1

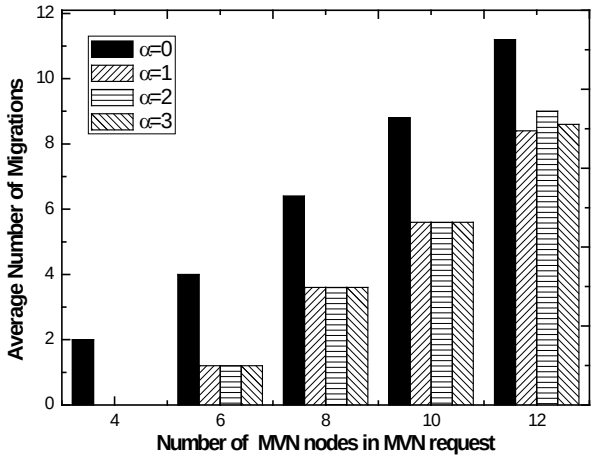


(b) Simulation Results on Net-2

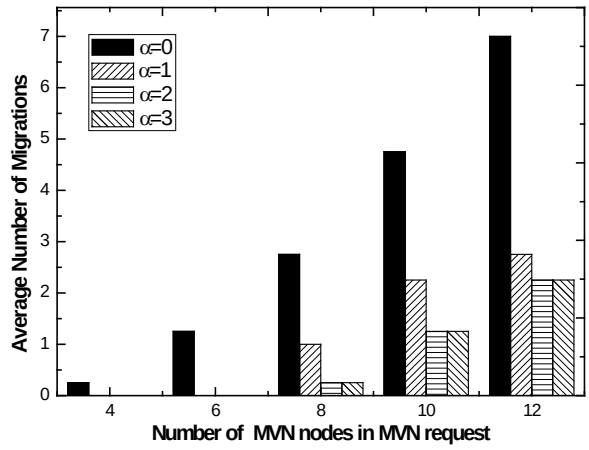
Fig. 11. Total mapping cost using SOUM*-MF on different substrate networks for various values of parameter α

Fig. 12 shows the effect of parameter α on average number of migrations of algorithm SOUM*-MF under various networks. As shown in **Fig. 12**, in this set of simulations the resources capacities of the substrate networks are sufficient to recover from any failure. **Fig. 12** depicts that the algorithm SOUM*-MF with different α obtained different average number of migrations under various substrate networks. The average number of migrations obtained by SOUM*-MF without α (i.e., α=0) is higher than when α is nonzero. This is because α guides the mapping avoiding failure regions, which means original mapping covers more failure regions thus requiring fewer migrations to recover from the regional failure. In addition, the results in **Fig. 12(b)** is better than in f **Fig. 12(a)** due to the fact that larger substrate network offers better mapping solutions.

The above simulation results show that our approaches perform well and lead to lower mapping cost, average number of migrations and unrecoverable ratio for all failures than SOUM*. This is due to fact that our approaches proposed in this work can adjust the mapping solution to balance the mapping cost and the number of migrations, saving more network resource to ensure better performance under different substrate networks and different scenarios (e.g. constrained and unconstrained resource capacity scenarios.).



(a) Simulation Results on Net-1



(b) Simulation Results on Net-2

Fig. 12. Average number of migrations on different substrate networks with various values of parameter α for SOUM*-MF

7. Conclusion

Network virtualization serves as an effective method for resource provisioning in cloud-based datacenters. Multicast service is an important and popular service or application class in cloud computing. In this research, we study the problem optimal provisioning for multicast oriented virtual network (MVN) request in cloud-based data centers considering survivability for recovering from any single regional failure of substrate network. We formulate the problem of survivable MVN provisioning as a mathematical optimization problem with the objective of minimizing the mapping cost by using MILP. Furthermore, we first design an efficient algorithm, NSMVNM, to solve the non-survivable MVN mapping problem. We then extend the NSMVNM to SOUM*-M for achieving the survivable MVN mapping. We also introduce the strategy of sorting on MVN nodes and the factor α into NSMVNM to optimize our mappings which makes our algorithm SOUM*-MF failure-aware. Simulation results show that our algorithm performs better on mapping cost, average number of migrations and unrecoverable ratio than existing approach SOUM*.

References

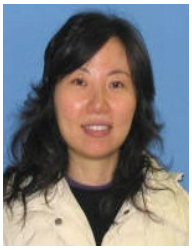
[1] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no.5, pp. 862-876, Apr. 2010. Article (CrossRef Link)
[2] D. Villegas, S.M. Sadjadi, "DEVA: distributed ensembles of virtual appliances in the cloud," *LNCS, Euro-Par Parallel Processing* vol.6852, pp. 467-478, Sept. 2011. Article (CrossRef Link)
[3] Amazon elastic computing cloud, http://aws.amazon.com/ec2/.
[4] Windows azure, http://www.microsoft.com/windowsazure/.
[5] Google app engine, http://code.google.com/appengine.
[6] Y. Chen, S. Jain, V.K. Adhikari, Z.L. Zhang, K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! Datasets," *IEEE INFOCOM*, pp.1620-1628, 2011.
[7] F. Hao, T.V. Lakshman, S. Mukherjee, H. Song, "Enhancing dynamic cloudbased services using network virtualization," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 67-74, Jan. 2010. Article (CrossRef Link)
[8] N.M. Mosharaf, M.R. Rahman, R. Boutaba, "Virtual network embedding with coordinated node and link embedding," *IEEE INFOCOM*, pp.783-791, 2009.
[9] M. Rahman, I. Aib, R. Boutaba, "Survivable Virtual Network Embedding," *NETWORKING*, vol. 6091 LNCS, pp. 40-52, May. 2010. Article (CrossRef Link)
[10] Z. Cai, F. Liu, N. Xiao, Q. Liu, Z. Wang, "Virtual network embedding for evolving networks," *IEEE GLOBECOM*, 2010.
[11] Y. Zhou, Y. Li, G. Sun, D. Jin, L. Su, L. Zeng, "Game theory-based bandwidth allocation scheme for network virtualization," *IEEE GLOBECOM*, 2010.
[12] X. Cheng, S. Su, Z. Zhang, Y. Luo, et al., "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38-47, Apr. 2011. Article (CrossRef Link)
[13] M. Yu, Y. Yi, J. Rexford, M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACMSIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17-29, Apr. 2008. Article (CrossRef Link)
[14] M. Zhang, C. Wu, M. Jiang, Q. Yang, "Mapping multicast service-oriented virtual networks with delay and delay variation constraints," *IEEE GLOBECOM*, 2010.
[15] R. Rajiv, and L. Zhao, "Peer-to-peer service provisioning in cloud computing environments," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 154-184, Jul. 2013.
[16] X. León, L. Navarro, "A Stackelberg game to derive the limits of energy savings for the allocation of data center resources," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 74-83, Jan. 2013. Article (CrossRef Link)

[17] I. Goiri, J.Ll. Berral, J.O. Fitó, F. Julià, et al. "Energy-efficient and multifaceted resource management for profit-driven virtualized data centers," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 718-731, May 2012. Article (CrossRef Link)
[18] A. Beloglazova, J. Abawajyb, R. Buyyaa, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, May 2012. Article (CrossRef Link)
[19] O. Beaumont, L. Eyraud-Dubois, H. Larchevêque, "Reliable Service Allocation in Clouds," *IEEE International Parallel & Distributed Processing Symposium*, 2013.
[20] T. Guo, N. Wang, K. Moessner, R. Tafazolli, "Shared backup network provision for virtual network embedding," *IEE ICC*, 2011.
[21] H.Yu, V. Anand, C. Qiao, G. Sun, D. "Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures," *IEEE ICC*, 2011.
[22] H.Yu, V. Anand, C. Qiao, et al. "On the survivable virtual infrastructure mapping problem," *IEEE ICCCN*, Aug, 2010.
[23] H. Di, H. Yu, V. Anand, L. Li, et al. "Efficient Online Virtual Network Mapping Using Resource Evaluation," *Journal of Network and Systems Management*, vol. 20, no. 4, pp. 468-488, Dec. 2012. Article (CrossRef Link)
[24] J. Hershberger, M. Maxel, S. Suri, "Finding the k shortest simple paths: A new algorithm and its implementation," *ACM Transactions on Algorithms* (*TALG*), vol. 3, no. 4, pp. 26-36, Nov. 2007. Article (CrossRef Link)

Dan Liao is an associate professor at University of Electronic Science and Technology of China (UESTC). He received his B.S. degree in Electrical Engineering in 2001 from UESTC, and his Ph.D. degree in Communication and Information Engineering in 2007 from University of Electronic Science and Technology of China, respectively. His research interests are in the area of wired and wireless computer communication networks and protocols, next generation network.

Gang Sun received his M.S. degree in Signal and Information Processing from Chengdu University of Technology, China in 2009, and the Ph.D. degree in Communication and Information Engineering in 2012 from University of Electronic Science and Technology of China. His research interests are in the area of network virtualization, cloud computing and next generation Internet.

Vishal Anand is an associate professor at The College at Brockport, SUNY. He received his B.S. degree in Computer Science and Engineering from the University of Madras, Madras (Chennai), India in 1996, and the M.S. and Ph.D. degrees in Computer Science and Engineering from the University at Buffalo, SUNY in 1999 and 2003. His research interests are in the area of wired and wireless computer communication networks and protocols, cloud and grid computing.



Hongfang Yu received her B.S. degree in Electrical Engineering in 1996 from Xidian University, her M.S. degree and Ph.D. degree in Communication and Information Engineering in 1999 and 2006 from University of Electronic Science and Technology of China, respectively. From 2009 to 2010, she was a Visiting Scholar at the Department of Computer Science and Engineering, University at Buffalo (SUNY). Her research interests include network survivability and next generation Internet, cloud computing etc.