This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures

Hongfang Yu[1], Vishal Anand[2], Chunming Qiao[3], Gang Sun[1]
[1]School of Communication and Information Engineering, University of Electronic Science and Technology of China, China
[2]Department of Computer Science, The College at Brockport, State University of New York, USA
[3]Department of Computer Science and Engineering, State University of New York at Buffalo, USA

***Abstract*–As network virtualization becomes popular, the problem of efficiently mapping a virtual infrastructure (VI) over a substrate network while guaranteeing its survivability in the event of failures becomes increasingly important. In this paper, we study the survivable VI mapping problem to recover from facility node failures. We develop two solutions namely the *1*-redundant scheme and the *K*-redundant scheme for surviving facility node failures while minimizing network resource costs. We also model the two schemes as a MILP problem and propose efficient heuristics based on the MILP formulations. We compare the efficiency of our solutions using simulation under various performance metrics.**

I. INTRODUCTION

Network *virtualization* [1] allows multiple virtual networks (VNs) to use the resources of the same physical substrate network. Since each one of these VNs use the resources of the underlying substrate network, it is essential to use these resources efficiently by using intelligent techniques that map the VN on to the substrate network.

A virtual infrastructure (VI) request consists of a set of VI nodes, with each node requiring some computing resources (e.g., CPU, memory and storage resources) at a *separate* facility node. A VI node also needs to communicate with another VI node to send intermediate results, file data, or some other information thus imposing strict connectivity requirements among the VI nodes in terms of topology, bandwidth, and delay guarantees to meet the service level agreements (SLA). Thus, given a VI request we need to find a *one-to-one mapping* of the VI request onto the substrate *facility nodes*. That is, assign each VI node to a computing facility node, and establish communication between these facility nodes.

Due to the shared nature of virtualization, survivability, i.e., ability to recover from failures is of prime importance. To guarantee survivability, spare (or redundant) computing resources and communication bandwidth need to be allocated for each VI.

In this work we focus on the failure recovery of the facility nodes to jointly minimize the total amount of computing and bandwidth resources (or cost). A facility node can fail if the computer(s), processors, disks, memory etc. fail, even if there is no substrate network failure (i.e., even when none of the substrate nodes and/or links fail). When a facility node fails, we would like to (fully) restore the VI node (or task) mapped to it by e.g., migrating to another backup facility node. In addition, the associated virtual connections also need to be migrated.

We adopt a two-step paradigm to fully restore a VI from any single facility node failure by *first* enhancing a VI with backup virtual nodes and links requiring spare computing and communications resources, and *then* mapping the enhanced VI to the substrate network. More specifically, we proposed two new approaches whereby an N-node VI is first enhanced to a *1*-redundant and *K*-redundant VI with N+1 and N+K nodes, respectively, in addition to an appropriate number of redundant virtual links. In the subsequent mapping of the enhanced VI to the substrate network, maximal amount of sharing of the computing and communication bandwidth among the nodes and links in the enhanced VI is exploited. Accordingly, it is possible that some N+k ($1 \le k \le K$) substrate nodes are chosen by the algorithm when mapping the enhanced VI with N+K nodes. How to enhance the VI to get the enhanced VI and then maximize sharing in the subsequent mapping of the enhanced VI are both open problems, and the latter is in fact more challenging than a usual VI mapping problem due to the need to exploit sharing.

Our work differs from the previous works in [2-4] which studied the VI mapping problem, but not survivability. More recently the works in [5-9] have considered VI survivability but do not use the two-step paradigm and accordingly were not concerned with the problems such as how to enhance a VI and how to map the enhanced VI with sharing among the virtual nodes and links. In [5,6], a regional failure containing one or more facility nodes is assumed and an N-node VI is mapped to N+m facility nodes but which N+m facility nodes to use depends on which facility nodes are affected by the regional failure. The work in [7] only considered the failure of substrate links, and studied how to re-route the virtual connections in order to utilize the pre-allocated spare link capacity in the substrate networks. In [8], a set of N+K facility nodes are assumed to be given, and the objective is to find a set of lightpaths such that after a facility node fails, the N-node VI can still be mapped to the surviving substrate network. In [9], each VI node is duplicated 2 or more times and then mapped to 2 or more facility nodes in order to tolerate concurrent facility node and substrate link failure. Here, we will only deal with one facility node failure by adding 1 or more VI nodes using different and more cost-efficient solutions.

The remainder of this paper is organized as follows. We describe our network model and the general two-step survivable VI problem in Section II. Section III describes the two proposed *1* and *K*-redundant approaches and in particular, how to construct the enhanced VI graphs and how to exploit resource sharing during the mapping step. In Section IV we give the MILP formulation for mapping the enhanced graphs. In section V we present our heuristic solution for survivable mapping of the VI. In Section VI we present the simulation results of the various strategies. In Section VII we conclude this paper.

II. NETWORK MODEL AND PROBLEM STATEMENT

A. Substrate Network

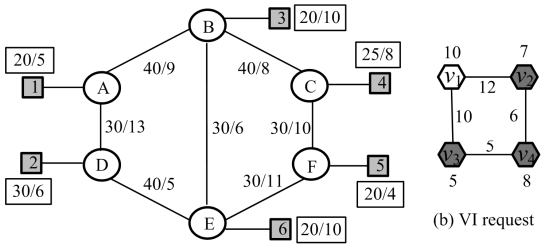
We model the substrate network as an undirected graph $G_S=(N_S, E_S)=(N_F \cup N_X, E_S)$, where N_S is the set of substrate nodes, and E_S corresponds to the set of bidirectional fiber links and access links. Note that N_S consists of N_F and N_X, where N_F is the

978-1-61284-233-2/11/$26.00 ©2011 IEEE

set of facility nodes and N_X is the set of optical switches. Facility nodes access the network through access links connecting them to the optical switches N_X.

Each facility node $n \in N_F$, the available computing capacity is ε_n and the unit computing cost is c_n. For each link $(u,v) \in E_S$, the available bandwidth capacity is w_{uv}, and unit bandwidth cost is c_l. In this paper, we assume c_n=1 for all facility node n, and c_l= g for all fiber link l, where g is the ratio of the unit cost of computing resources to that of bandwidth resources

Fig. 1(a) shows a substrate network with 6 facility nodes (shaded squares) 1 to 6, which are connected to 6 switching nodes (unshaded circles) A to F that actually perform the computations, storage etc. The switching nodes are connected to each other by high bandwidth communication links, and the facility nodes are connected to the switch nodes through low(er) bandwidth access links. The numbers over the links represent the available bandwidth and the cost of a bandwidth unit, and the numbers in the rectangles represent the available computing resources and cost of computing resource at the facility nodes.



(a) Substrate network

Fig. 1: Substrate network and a VI request

B. VI Request

A VI request (with QoS requirements) is modeled as an undirected graph G_V= (N_V, C, E_V), where N_V corresponds to the set of VI nodes, C denotes the set of *critical* VI nodes (i.e., nodes that need to have a backup) and E_V denotes the set of bidirectional communication demands among the VI nodes. Each VI node $u \in N_V$ needs a certain amount of computing resources for its execution, denoted by ε_u. Each communication demand $(u,v) \in E_V$, $u,v \in N_V$ has a bandwidth requirement, denoted by α_{uv}. Fig. 1 (b) shows one VI request with four VI nodes and VI links, and associated computing and communication requirements.

In addition we also assume that different VI nodes can have different reliability requirements. For example, certain VI nodes may not require any fault tolerance, and hence when the corresponding facility node fails no backup is required. This is realistic because there may be situations where either the data or task occurring at the VI node is not important and its loss can be tolerated, or the facility node running the task is very reliable or has some redundancy already built into it. Such critical VI nodes are shown shaded in Fig. 1(b). The solutions to be presented in this paper can be easily extended to the case when the set C includes all the nodes in the VI request.

C. Problem Statement

Given: a substrate network G_S=(N_S, E_S), a VI request G_V=(N_V, C, E_V).

Question: how to design a reliable VI graph (or enhanced VI graph) and find a mapping of the same on to the substrate network by jointly allocating computing and networking resources to recover from the failure of one facility node such that the total cost of resources i.e., sum of the cost of the computation and bandwidth resources is minimized?

To survive the facility node failure, the original VI graph (or request) has to be augmented to form a reliable VI graph with redundant VI nodes. In addition to minimizing the resource costs ,the reliable VI graph should be designed in such a way that when the backup nodes resume execution of the failed critical nodes they have connections (guaranteed bandwidth) to all original (i.e., before failure) neighbors of the critical nodes.

In this paper, we take a two-step approach to the complicated survivable VI mapping problem. In the first step, the *reliable VI graph design* stage, the VI request is augmented with redundant virtual nodes and redundant links that have sufficient computational and bandwidth resources. This augmented VI request is a reliable VI graph that can tolerate facility node failures. In the second step, the *reliable VI graph mapping* stage, the reliable VI graph is mapped to the substrate network to minimize the total cost with guaranteed resource requirement to tolerate any facility node failure.

III. Reliable VI Graph Design and Mapping

A. 1-redundant solution

In this solution, we first design a reliable VI graph with one redundant VI node and redundant connections and then map this reliable VI graph onto the substrate network. While performing the mapping it is important to take advantage of possible resource sharing to minimize the costs.

1. *1*-redundant VI graph design

In this solution we transform the original non-survivable VI graph G_V to the *1*-redundant VI graph G^1_V with redundancy by adding *one additional* VI node *b*. If any of the critical VI nodes in set C fails, the failed VI node is migrated to this redundant node b, and the connections associated with the failed nodes need to be migrated too. Hence the redundant node b should have connections (called redundant VI links) with all neighbors of all critical VI nodes. Formally, the set of redundant links that are added to G_V is:

$$E^1_B = \{(b,v) \mid \exists (r,v) \in E_V, \forall c \in C, v \in N_V\}$$

The set E^1_B connects the backup node to all neighbors of all critical VI nodes. Hence, the redundant node b must have sufficient computation resources, i.e., $\max\{\varepsilon_c \mid \forall c \in C\}$, and bandwidth resources to neighbors of v, i.e., $\max\{\alpha_{cv} \mid \forall c \in C, (c,v) \in E_V\}$. Now G^1_V can be denoted as:

$$G^1_V = (N^1_V, E^1_V), \text{ where } N^1_V = N_V \cup b,\ E^1_V = E_V \cup E^1_B$$

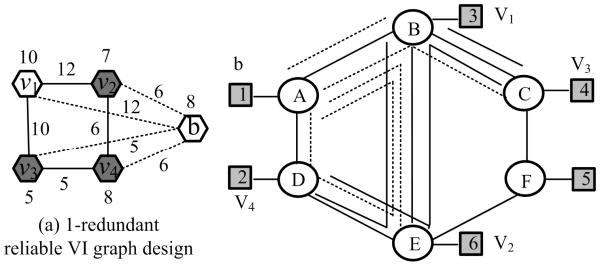
We call the set of redundant VI links that would be used simultaneously upon the failure of a certain VI node v as a *migratory association backup-link group*, denoted by *BG(v)*.

We illustrate the working of the *1*-redundant solution in Fig. 2(a) that shows a VI request graph with three critical VI nodes v_2, v_3 and v_4, and a redundant VI node b, which is added to the VI request. Since b should connect with all neighbors of v_2, v_3 and v_4, four redundant VI links (v_1,b), (v_2,b), (v_3,b) and (v_4,b) are added. Obviously, (v_1,b) and (v_3,b) are the components of $BG(v_2)$, (v_1,b) and (v_4,b) are the components of $BG(v_3)$, and (v_2,b) and (v_3,b) are the components of $BG(v_4)$. The computing resource requirement on redundant node b is 8 units, which is the maximum of the computing resources of all the critical nodes.

Next we calculate the network bandwidth resource requirement as follows. When either of nodes v_2 or v_3 fails, the failed node will be migrated to b and redundant VI link (v_1,b) will be used to transport the communication traffic between v_2 and v_1 or between v_3 and v_1.The bandwidth requirement α_{v1b} on redundant VI link (v_1,b) is $\max(\alpha_{v1v2},\alpha_{v1v3})=\max(12,10)=12$ units. Similarly, α_{v2b} is 6 units, α_{v3b} is 5 units, and α_{v4b} is 6 units.

2. *1*-redundant VI graph Mapping

Now we need to do the actual mapping of the redundant reliable VI graph onto the substrate network. While doing this mapping it should be noted that as only one facility node v may fail at any one time not all redundant VI links belonging to different $BG(u)$ will be used simultaneously, and hence we can share the physical link resources when mapping them onto the substrate network. When we share the resources among different backup paths belonging to different $BG(v)s$, we call such a sharing strategy as *backup share*. One should also note that since we are considering facility node failures, none of the physical network nodes or links fails, and hence when a facility node fails the original working path to the switch connected to the facility node remains intact. These original working resources can be reused by the corresponding backup paths. Thus, now we can also share the bandwidth link resources between the original working path and its associated backup path. We call such a sharing strategy between working and backup paths as *cross share***.** Similar to the $BG(v)$ definition, we define this set of working VI links that would be simultaneously migrated to the backup VI links upon the failure of a certain VI node v as a *migratory association working-link group*, denoted by $WG(v)$. A $BG(v)$ has a corresponding $WG(v)$. The backup VI links in a $BG(v)$ may share bandwidth with the working VI links in the corresponding $WG(v)$ via the cross share strategy.



(a) 1-redundant reliable VI graph design

(b) 1-redundant VI graph mapping

Fig. 2: A *1*-redundant VI graph design and mapping example

Fig. 2(b) illustrates the mapping and sharing strategies discussed above for the mapping of the *1*-redundant reliable VI graph designed in Fig. 2(a). Fig. 2(b) shows a substrate network where the VI nodes v_1, v_2, v_3 and v_4 are mapped onto facility nodes 3, 6, 4 and 2 respectively. As shown in Fig. 2(b), original working VI links are mapped onto solid working paths and redundant VI links are mapped onto dashed backup paths. Redundant VI link (v_1,b) is mapped onto path A-B, redundant VI link (v_2,b) is mapped onto path A-B-E. Note that redundant link (v_1,b) is only used when critical node v_2 or v_3 fails; while redundant link (v_2,b) is only used when critical node v_4 fails. Hence the corresponding redundant paths A-B and A-B-E can share the backup bandwidth on fiber link A-B. This sharing is an example of backup share. Similarly, redundant VI links (v_1,b) and (v_3,b) in $BG(v_2)$, (v_2,b) and (v_4,b) in $BG(v_3)$, and (v_3,b) and (v_4,b) in $BG(v_4)$ can do backup sharing. Further, assume that redundant VI link (v_3,b) is mapped onto path A-B-C, and original working link (v_3,v_4) is mapped onto path C-B-E-D. When critical node v_3 fails, original VI link (v_3,v_4) in $WG(v_3)$ is migrated to redundant link (v_3,b) in $BG(v_3)$, so the corresponding redundant path A-B-C can reuse the bandwidth released by original working path C-B-E-D on fiber link B-C. This sharing is an example of cross share. Similarly, redundant link (v_4,b) in $BG(v_2)$ can reuse the resource of original working links (v_1,v_2) and (v_2,v_4) in $WG(v_2)$.

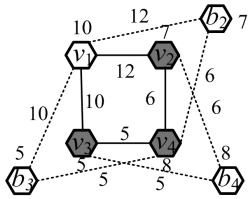
B. K-redundant solution

In the *1*-redundant VI graph design while finding a feasible physical node that a backup VI node b may be mapped onto we need to establish multiple VI links from one backup node b to each neighbor of all the critical nodes. In some cases, especially when the resources are limited it is not cost-efficient or difficult or even impossible to find resources to support such a *1*-redundant mapping. Hence, we propose a more flexible solution, namely the *K*-redundant solution.

In the *K*-redundant solution, we first design a *K*-redundant reliable VI graph, in which we permit each critical node to have a corresponding backup node. Then we map the *K*-redundant VI graph onto the substrate network, such that no more than *K* backup facility nodes are used, where *K* is equal to the number of critical nodes in set *C*. The actual number *k* of backup facility nodes could be anywhere from 1 to *K*. Since at any time only one working VI node will fail (due to the failure of the corresponding facility node), different redundant VI nodes can be mapped onto the same facility node depending on the cost.

1. *K*-redundant VI graph design

When we design a *K*-redundant reliable VI graph, a set of redundant backup virtual nodes B^K are added to the original non-survivable VI graph G_V such that for each VI node c in C, there exists a corresponding redundant virtual node b(c) in B^K. Obviously, $B^K= \{b(c),\forall c\in C\}$.



K-redundant VI graph

Fig. 3: A *K*-redundant solution

Formally, the set of redundant links E_B that are added to G_V is:

$$E^K{}_B= \{(b(c),v) \mid \exists(c,v)\in E_V\ , c\in C,\ v\in N_V\}$$

The set $E^K{}_B$ connects all backup nodes to all neighbors of all VI nodes. The resource requirement on each redundant link equals its corresponding original VI link. Now the *K*-redundant VI graph $G^K{}_V$ can be denoted as:

$$G^K{}_V= (N_V\cup B^K,\ E_V\cup E^K{}_B)$$

In the *K*-redundant solution, all redundant VI links connected to the same backup VI node b(c) belong to $BG(c)$; and a redundant VI link is member of only one BG. Fig. 3 shows a *K*-redundant VI graph with 3 backup redundant VI nodes b_2, b_3 and b_4, which are used upon the failure of the corresponding 3 critical nodes v_2, v_3 and v_4. The redundant VI links (b_3,v_4) and (b_3,v_1) belong to $BG(v_3)$; while the redundant VI links (b_2,v_4) and (b_2,v_1) belong to $BG(v_2)$.

2. *K*-redundant VI graph Mapping

When mapping the *K*-redundant VI graph onto the substrate nodes, we permit these backup VI nodes to be mapped onto multiple facility nodes. The actual number K of backup facility nodes used depends on physical network topology, VI request and optimization objective.

Similar to the *1*-redundant solution, in the *K*-redundant solution we can take advantage of the backup share and cross share. Moreover, we can take advantage of node share, i.e., if the backup VI nodes are mapped onto the same facility node, then the computing resources required on this facility node is only the maximum required resources of all corresponding VI nodes. The *1*-redundant solution is a special case of the *K*-redundant solution, when all backup VI nodes are mapped onto one facility node.

IV. MILP FORMULATIONS

In this section we give the MILP formulations for the mapping of the *1* and *K*-redundant VI graphs.

A. Reliable VI graph Construction

The reliable VI graph G_R, $G_R=(N_W,N_B,E_W,E_B)$ with N_W working VI nodes and N_B backup VI nodes, E_W working VI links and E_B backup VI links serves as input to our MILP.

Given the original VI graph $G_V=(N_V,C,E_V)$, the reliable VI graph G_R for the *1*-redundant solution is as follows:

$$N_W = N_V,\ N_B=B^1,\ E_W=E_V,\ E_B=E^1{}_B$$

Similarly, the reliable VI graph G_R for *K*-redundant solution is as follows:

$$N_W = N_V,\ N_B = B^K,\ E_W = E_V,\ E_B = E^K{}_B$$

B. Augmented Substrate Graph Construction

To address the tight coupling between node mapping and link mappings of a redundant VI graph onto a physical substrate network, we use a joint node and link mapping approach for the resource allocation problem. We apply the following graph transformation similar to [4] to the substrate network G_S. We add $|N_W|$ "virtual working nodes" and $|N_B|$ "virtual backup nodes" into G_S, , and each virtual backup node is connected to all the facility nodes. We call the link connecting a "virtual working node" and facility node as a "virtual working link", and the link connecting a "virtual backup node" and facility node as a "virtual backup link". We assume that the bandwidth resources on each virtual working link and backup link are unlimited. This transformed augmented graph is denoted as: $G_S^*=(N_S^*,E_S^*)$, where $N_S^*=N_F\cup N_x\cup N_W\cup N_B$ and $E_S^*=E_S\cup E_W\cup E_{B,}$. The virtual links can be seen as flows between virtual nodes.

The difference from [4] is that in this paper we should consider not only the original node and link mapping, but also the backup nodes and links mapping. Moreover, we also need to consider the link share between backup VI links and, between the backup VI links and the original VI links. Furthermore, we also need to consider node sharing.

C. MILP formulation for 1 and K-redundant solution

The MILP formulation to map the reliable VI graph onto the substrate network for the *1* and *K*-redundant solutions is essentially the same; the only difference is in their input, i.e., the augmented substrate graph or reliable VI graph. In the *1*-redundant solution there is only one backup node, while in the *K*-redundant solution the maximum number of backup VI nodes is the same as the number of critical nodes.

Variables:

λ: big constant

$f^0_{uv}[ij]$: the amount of working flows between two virtual nodes $u,v\in N_W$ which pass through link (i,j)

$f^l_{bv}[ij]$: the amount of backup flows between a virtual node $b\in N_B$ and a virtual node $v\in N_W$ which pass through link(i,j).

x^0_{uv}: Binary variable denoting whether virtual working link is assigned; 1 if there is flow on vitual working link (u,v), and 0 otherwise.

x^l_{uv}: Binary variable denoting whether virtual backup link (u,v) is assigned; 1 if there is flow on link (u,v), and 0 otherwise.

$\delta[ij]$: the reserved backup resources on a link (i,j) to recover from any critical node failure.

θ^l_u: Binary variable denoting whether physical node u is mapped for redundant VI node.

ρ^l_i: Variable denoting the total required backup computing resource on the backup facility node i.

Objective:

$$\min \sum_{v\in N_F}\Big(\sum_{u\in N_W} x^0_{uv}\times\varepsilon_u+\rho^1_v\Big)+g\times\sum_{(i,j)\in E_S}\sum_{(u,v)\in E_W}\big(f^0_{uv}[ij]+(\delta[ij]\big) \quad (1)$$

The objective function minimizes the sum of all computing and communication costs. The first part of Eq. (1) is the sum of all working and backup computing node costs, and the second part is the sum of all working and backup link bandwidth costs.

Node capacity constraints:

$$\sum_{u\in N_V} x^0_{uv}\times\varepsilon_u+\rho^1_v\le\tau_v,\quad \forall v\in N_F \quad (2)$$

Node mapping constraints:

$$\sum_{v\in N_F} x^0_{uv}=1,\quad \forall u\in N_W\cup N_B \quad (3)$$

$$\theta^1_v\le\sum_{u\in N_B} x^1_{uv}\le\lambda\times\theta^1_v,\quad \forall v\in N_F \quad (4)$$

$$\sum_{u\in N_W} x^0_{uv}+\theta^1_v\le 1,\quad \forall v\in N_F \quad (5)$$

Eqs. (2)-(5) ensure that there are enough working and backup computing resources at the facility nodes, and only one facility node is selected for each virtual working node and each virtual backup node.

Flow conservation constraints:

Due to space limitation we omit the details of the flow conservation equations (6)-(11) that ensure that both the working and backup flows follow the flow conservation constraints.

Resource constraint on shared links:

The actual amount of bandwidth reserved on a physical link (i,j):

$$\sum_{(b(c),v)\in BG(c)}\big(f^1_{b(c),v}[ij]+f^1_{v,b(c)}[ij]\big)-\sum_{(c,v)\in WG(c)}\big(f^0_{cv}[ij]+f^0_{vc}[ij]\big)\le\delta[ij]$$
$$\forall(i,j)\in E_S,\forall c\in C \quad (12)$$

The first part of the Eq. (12) ensures that backup paths to adjacent neighboring nodes will use bandwidth resources at the same time and hence no resource sharing is possible. That means backup paths in different $BG(v)$s have a chance to share bandwidth. The second part of the equation captures the fact that the resources on the working paths that are migrated to the corresponding backup paths can be reused by these backup paths. Thus the backup paths in a $BG(v)$ may share bandwidth with the working paths in the corresponding $WG(v)$.

Resource constraints on shared nodes:

The actual amount of computing resources reserved on a facility node v after considering overlaps of different backup VI nodes can be captured by the following constraint:

$$\varepsilon_u \times x^1_{uv} \le \rho^1_v, \quad \forall u \in N_B, \forall v \in N_F \qquad (13)$$

Eq. (13) gives the maximum computing resources required at a facility node upon the failure of a critical node.

Link capacity constraints:

$$\sum_{(u,v)\in E_W} (f^0_{uv}[ij]+f^0_{uv}[ji]) + (\delta_{ij}+\delta_{ji}) \le 2\times w_{ij} \quad \forall (i,j) \in E_s \qquad (14)$$

$$\sum_{(u,v)\in E_W} (f^0_{uv}[ij]+f^0_{uv}[ji]) + \sum_{(b,v)\in E_B} (f^1_{bv}[ij]+f^1_{bv}[ji]) \le \lambda \times x^0_{ij} \quad \forall i \in N_W, j \in N_F \qquad (15)$$

$$\sum_{(u,v)\in E_W} (f^0_{uv}[ij]+f^0_{uv}[ji]) + \sum_{(b,v)\in E_B} (f^1_{bv}[ij]+f^1_{bv}[ji]) \le \lambda \times x^1_{ij} \quad \forall i \in N_B, j \in N_F \qquad (16)$$

$$x^0_{ij} = x^0_{ji}, \forall i \in N_W, j \in N_F \qquad (17)$$

$$x^1_{ij} = x^1_{ji}, \forall i \in N_B, j \in N_F \qquad (18)$$

V. Heuristic Algorithm Design

It should be noted that even the problem of finding a minimum-cost non-survivable VI mapping is a NP-Hard problem. The problem is even more complicated when we need to consider VI request survivability. Hence we rely on heuristics to solve the survivable VI mapping problem.

Heuristic Algorithm

Input: VI request $G_V=(N_V,C,E_V)$, substrate network $G_S=(N_F \cup N_X, E_S)$

Output: Working and backup mappings

Step 1: Obtain the working mapping solution

1.1: Establish an augmented substrate network G^* based on the original VI request G_V.

1.2: Use D-ViNE to obtain the working mapping solution $\{\{x_{ij}\},\{f_{uv}[ij]\}\}$, where $\{x_{ij}\}$ denotes whether VI node i is mapped onto facility node j ,$f_{uv}[ij]$ denotes the amount of flow on VI link (u,v) that passes through physical link (i,j).

Step 2: Obtain the backup mapping solution

2.1: Establish a reliable VI graph $G_R=(N_W,N_B,E_W,E_B)$

According to *1*-redundant VI graph design strategy, we establish a reliable VI graph with *1*-redundant node for the VI request.

2.2 Establish an augmented substrate network G^*_S based on G_R

2.3: Fix the variables x^0_{ij} and $f^0_{uv}[ij]$ according to original working node mapping and link mapping solution as follows:

$x^0_{ij}=x_{ij}, i\in N_V, j\in N_F$

$f^0_{uv}[ij]=f_{uv}[ij], (u,v)\in E_V, (i,j)\in E_S$

2.4: Solve MILP under G^*_{Sub} with fixed x^0_{ij} and $f^0_{uv}[ij]$, obtain the backup mapping solution $\{\{x^1_{ij}\},\{f^1_{uv}[ij]\}\}$

Fig. 4: Heuristic algorithm for the *1*-redundant solution

The basic idea of our algorithm is to first use the D-ViNE algorithm [4] to find the working mapping for the original VI request. We then fix this working mapping and solve the simplified MILP to obtain only the backup solution using CPLEX. Since we fix the value of variables x^0_{uv} and $f^0_{uv}[ij]$, constraints related to x^0_{uv} and $f^0_{uv}[ij]$ can be ignored or simplified too. This reduces the time complexity of the MILP. The details of the algorithm are shown in Fig. 4.

The algorithm for the *K*-redundant solution is almost same; the only difference with the *1*-redundant solution is the reliable VI graph design. Thus, we change step 2.1, and establish a reliable VI graph according to the *K*-redundant graph design strategy.

VI. Simulation Results

A. Simulation environment and Comparison Methodology

We compare the working of our algorithms for a large substrate network with 27 node and 41 links. The computing capacity at facility nodes and bandwidth capacity on the links follow a uniform distribution between 100 and 300 units.

The VI requests are generated randomly based on (i) the number of VI nodes in the VI request *N*, (ii) the average probability of connectivity between any two nodes in the VI request, which we set to 0.5, (iii) the number of critical nodes, which we set to N/2, (iv) the average computing requirement of a VI node and (v) the average bandwidth requirement of a VI link. The computing and bandwidth requirements follow a uniform distribution from 10 to 30 and 10 to 50, respectively.

We compare the performance of using (i) both cross and backup share (labeled as "*share*"), (ii) only backup share ("*bshare*") and (iii) no share ("*noshare*") using the *redundancy ratio* performance metric, which is the ratio of the total backup resource cost to the total working resource cost.

For the *bshare* case, we revise constraint (12) to (12.a) as follows so that now it only considers backup sharing:

$$\sum_{(b(c),v)\in BG(c)} (f^1_{b(c),v}[ij] + f^1_{v,b(c)}[ij]) \le \delta[ij] \quad \forall (i,j) \in E_S, \forall c \in C \qquad (12.a)$$

For the *noshare* case, we revise constraint (12) to (12.b) so that there is no backup or cross sharing on any link (i,j):

$$\sum_{c\in C} \sum_{(b(c),v)\in BG(c)} (f^1_{b(c),v}[ij] + f^1_{v,b(c)}[ij]) \le \delta[ij] \quad \forall (i,j) \in E_S \qquad (12.b)$$

In addition, we also compare the performances of the *1*-redundant solution and *K*-redundant solution. We consider the following performance parameters: 1) *node cost ratio*, 2) *link cost ratio* and 2) *total cost ratio*, which are the ratios of the node redundancy cost, link redundancy cost and the total redundancy cost incurred by the *K*-redundant solution to that incurred by the *1*-redundant solution.

B. Simulation analysis

Fig. 5 shows the redundancy ratio of the various sharing schemes of the *1*-redundant solution with the increasing size of VI request. The Fig. shows that the redundancy ratio of *noshare* is considerably high. We also note that *bshare* with resource sharing among backup paths significantly reduces the redundancy ratio by requiring fewer redundant resources to tolerate any facility node failure. Furthermore, *share* further decreases the redundancy ratio and cost by permitting the sharing of resources between backup and working paths. We explain these results as follows. Since only one critical node can fail at any one time, only a limited number of backup VI links are used simultaneously; these VI links can share the resources with the other unused redundant VI links, thus reducing the resources in *bshare*. In addition the *share* scheme also promotes the reuse/sharing of resources between the backup and original working paths, further reducing the resource redundancy ratio. For example, when the number of VI nodes in VI request is 6, the redundant resources (or cost) used by the share scheme is very low.

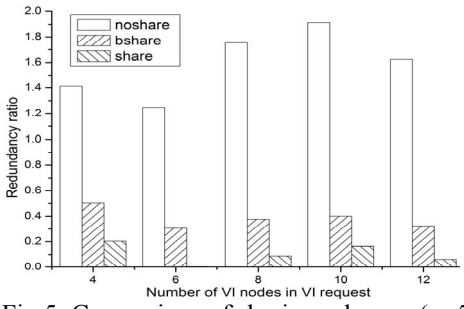


Fig.5: Comparison of sharing schemes (g=5)

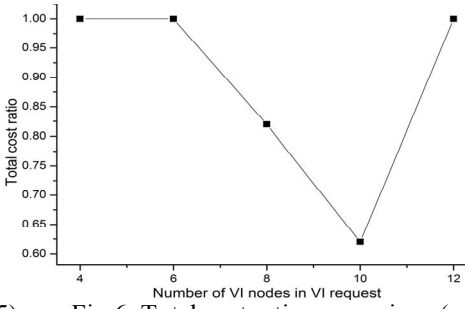


Fig.6: Total cost ratio comparison (g=5)

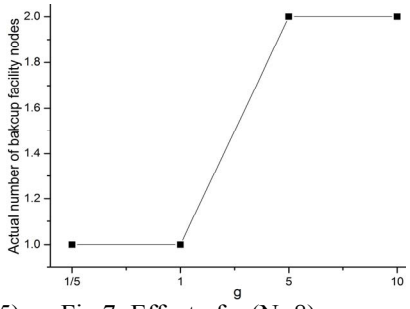


Fig.7: Effect of g (N=8)

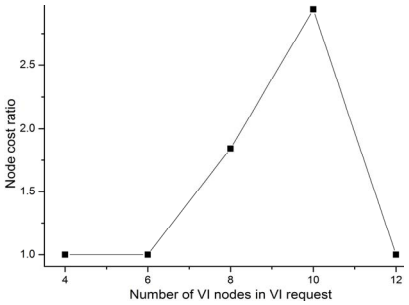


Fig. 8: Node cost ratio comparison (g=5)

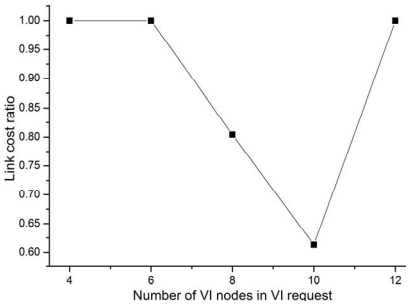


Fig.9: Link cost ratio comparison (g=5)

Fig.6 shows the total cost ratio, i.e., the redundancy cost incurred by the *K*-redundant solution to that incurred by the *1*-redundant solution with the increasing size of VI request. From the Fig. we note that the total cost ratio is no *more* than 1.0, i.e., the *K*-redundant solution is at least as good if not better than the *1*-redundant solution. This is because the *K*-redundant solution has more choices compared to *1*-redundant solution, i.e., to use one or more backup facility nodes for failure recovery. For the *K*-redundant solution, the worst case is same as the *1*-redundant solution, i.e., *K* backup VI nodes are mapped onto only one backup facility node. When the total ratio is less than 1.0, i.e., *K*-redundant solution is better than the *1*-redundant solution, the number of backup VI nodes used by the *K*-redundant solution is larger than 1. This conclusion can be further confirmed by Fig.7, which shows the effect of different *g* on the actual number of backup facility nodes for a given VI request with 8 VI nodes. For example, when g is 5, the number of backup facility nodes used is 2.

Fig.7 also shows that when the value of *g* is smaller, i.e., node computing cost is more expensive than the link communication cost, the *K*-redundant solution uses only one backup facility node. On the other hand when g increases, the *K*-redundant solution uses more than one backup facility nodes to reduce the total cost of resources.

Figs. 8 and 9 show the node cost ratio and link cost ratio of the *K*-redundant solution to the *1*-redundant solution. From the Figs. we note that the node cost ratio is no less than 1.0, while the link cost ratio is no more than 1.0. This is because the backup facility nodes (and node resources) used by the *K*-redundant solution for failure recovery would be no less than the *1*-redundant solution. While the corresponding link cost ratio decreases due to increase in bandwidth sharing with the increase in the number of used backup facility nodes.

VII. CONCLUSION

In this paper, we have studied the problem of survivable virtual infrastructure design to recover from facility node failures. We have designed a *1*-redundant scheme and *K*-redundant scheme to tolerate any facility node failure. We then modeled the *1* and *K*-redundant schemes as a MILP problem, and compared the two schemes via simulation. Simulation results show that the proposed backup and cross share strategies has a significant impact in conserving backup resources and improving resource utilization. Furthermore, we found that under majority of the circumstances the *K*-redundant solution with multiple backup facility nodes is more efficient than the *1*-redundant solution especially when communication costs are higher than the node computing costs.

REFERENCES

[1] T. Anderson et al., "Overcoming the Internet impasse through virtualization," *Computer*. vol. 38. No. 4., 2005.

[2] J.Luand, J.Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University, Tech. Report, WUCSE-2006-35, 2006.

[3] Y.Zhu, M.Ammar, "Algorithms for assigning substrate network resources to virtual network components," IEEE INFOCOM, 2006.

[4] N. M. M. K. Chowdhury et al., "Virtual Network Embedding with Coordinated Node and Link Mapping", IEEE INFOCOM, 2009.

[5] H. Yu et al. "Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional failures", accepted by IEEE Globecom 2010.

[6] H. Yu et al. "On the Survivable Virtual Infrastructure Mapping Problem", IEEE ICCCN, Aug. 2010.

[7] M. R. Rahman et al. "Survivable Virtual Network Embedding", Lecture Notes in Computer Science, Apr. 2010.

[8] X. Yu et al. "Survivable Logical Topology Design for Distributed Computing in WDM Networks", OFC, 2009.

[9] X. Liu et al., "Robust Application Specific and Agile Private (ASAP) Networks Withstanding Multi-layer Failures," OFC/NFOEC, 2009.