# Path Protection with Explicit Availability Constraints for Virtual Network Embedding

Sandra Herker, Xueli An, Wolfgang Kiess
DOCOMO Communications Laboratories Europe GmbH
Munich, Germany
{herker, an_de_luca, kiess}@docomolab-euro.com

Andreas Kirstädter
Institute of Communication Networks and Computer Engineering (IKR), Universität Stuttgart, Germany
andreas.kirstaedter@ikr.uni-stuttgart.de

*Abstract*—**Network reliability and survivability are important features for hosting high-demanding services in virtual network environments. Since multiple virtual networks can share the physical resources of the underlying substrate, even a single failure in the substrate can affect a large number of virtual networks and the services they offer. To overcome the physical link failure impact on virtual network operations, backup path mechanisms are investigated in many related works for path protection. However, this one to one path protection cannot handle explicit path availability requirements. In this work, we present a virtual network embedding algorithm which provides path protection with explicit virtual link availability constraints. Evaluation results show that our algorithm performs close to theoretical thresholds and consumes less link resources when delivering a virtual network embedding solution in polynomial time compared to a conventional path backup algorithm.**

## I. Introduction

Network virtualization is receiving more and more attention lately. The sharing of physical resources by subdividing a physical node or link into many virtual nodes or virtual links allows service specific networks to be embedded onto a physical network in a dynamic way. Using end-to-end (E2E) virtualization, it will be possible to create various service specific networks within one operator's network. The network can be tailored to the specific needs of a service with respect to topology, routing or QoS. Multiple configurations of Virtual Networks (VNs) may be created over the same physical setup. Some configurations may be more efficient than others in terms of different requirements such as, optimal use of physical resources, maximizing the revenue and/or minimizing the power consumption. The calculation of the effective allocation of the physical resources (e.g. bandwidth for the connections or capacity for the servers) among the Virtual Network Requests (VNRs) is known as the Virtual Network Embedding (VNE) problem which generally is NP hard [1][2]. Since multiple VNs can share the physical resources of the underlying substrate, even a single failure in the substrate can affect a large number of VNs and the services they offer. Thus, the problem of efficiently mapping a VNR to a substrate while guaranteeing the VNs survivability in the event of failures in the substrate becomes important.

To guarantee survivability of the VNs, the reliability of the physical network, their components like servers and links can be explicitly considered. The idea is to embed the VNR with the requested link availability onto the physical network. Physical links and network elements are always subject to failures - mainly fiber cuts and router failures. High available links provide high operational performance like higher uptime and shorter repairing time.

In this work, we formulate the VNE problem with explicit availability requirements on links to achieve an overall network availability. To the best of our knowledge, this is the first work to consider explicit availability values for embedding in the network virtualization environment. We propose a greedy algorithm which accomplishes VNE that ensures high network availability for links and runs in polynomial time. Extensive mathematical analysis and simulation show that our algorithm performs close to a Mixed Integer Programming (MIP) based ideal solution, and consumes less network resources comparing to conventional VNE algorithms that consider link protection.

The rest of the paper is organized as follows. Section II provides an overview of related work. In Section III the network model and the VNE problem with availability is formulated. In Section IV the heuristic algorithm is presented and Section V provides the evaluation results. Section VI concludes the paper with a summary and future work.

## II. Related work

As there exist many research works on VNE and some on survivable VNE, only the relevant and recent works concerning link embedding/link backup are presented here. The objective of the VNE is to find an effective and efficient mapping for the VNR. Embedding has been proven to belong to the NP-hard category of problems in [1][2].

There are two main survivability methods: protection and restoration. Failure protection works in a proactive way to reserve the backup resources before any failure happens. In contrast, restoration mechanisms are reactive and start the backup restoring mechanism only after the failure occurs. In [3], a reactive backup mechanism to protect against a single substrate link failure for VNE is proposed. The idea is a fast rerouting of the links and to reserve bandwidth for backups on each physical link. This backup mechanism is a restore approach, and after a failure it cannot guarantee 100% recovery. With increase in traffic load, a failure can cause a big amount of data loss and the backup mechanism may not restore the VN. Similar to [3], the authors in [4] solve the problem with a proactive link based backup approach with shared backup. They propose two schemes, *Shared On-Demand* and *Pre-Allocation* approach where bandwidth resources are allocated to the primary flows and to backup flows when a new VNR arrives or pre-allocated during the configuration phase before any VNR arrives. These two approaches are link based methods, where each link of the primary path is backed up by a pre-configured bypass path. The approaches described in this paper are path based, where each E2E primary path is backed up by a disjoint path from the source node to the destination node. In [5], each E2E primary substrate path is protected by a backup path with optimizing the used resources through a node migration technique of the end nodes of the paths. Compared to the traditional backup protection where only one path needs to be migrated, in their approach several links and at least one node need to be migrated. A mechanism, named *QoSMap* considering both Quality of Service (QoS) and resiliency using backup paths in constructing VNs over a substrate network is presented in [6]. In [7] the same problem as of QoSMap is considered and improved by using the substrate topology information in the mapping procedure.

Existing work cannot provide VNE for explicit service availability requirements. Only plain VN link protection exists in related work considering only 1+1 path protection. The number of required paths for a backup solution is not tight with service availability requirements, e.g. several backup path could be needed. Our approach is a proactive path-based VN link protection with explicit high availability guarantee.

## III. Network model and Problem description

In a network virtualization environment, a Virtual Network Operator (VNO) requests for VN on the physical/substrate network from a Virtual Network Provider (VNP). The VNP requests resources which meet the requirements of the VNR from the Physical Infrastructure Provider (PIP), who owns the physical network. The VNR with explicit link availability requirement from a VNO has to be mapped. The availability of a network component (e.g. link or node) is defined as the percentage of time when the component is operational. A simple formula for calculating the system availability, based on the uptime and downtime of the system, is given as the uptime divided through the sum of the uptime and downtime [8]. The graphical presentation of the substrate network and the VNR model is shown in Figure 1.

### A. Substrate Network

The substrate network can be represented as a graph $G_S = (V_S, E_S)$ with nodes and links. $V_S$ and $E_S$ represent the set of nodes and the set of links in the substrate network.

The substrate nodes represent resource nodes. One resource node could be one data center or a cluster of data centers belonging to the same PIP in the same region. The nodes are associated with resources like CPU capacity, storage capacity, etc. The resource nodes have unique geographical locations. The location of a node *i* is loc(*i*). Resource node capacity is

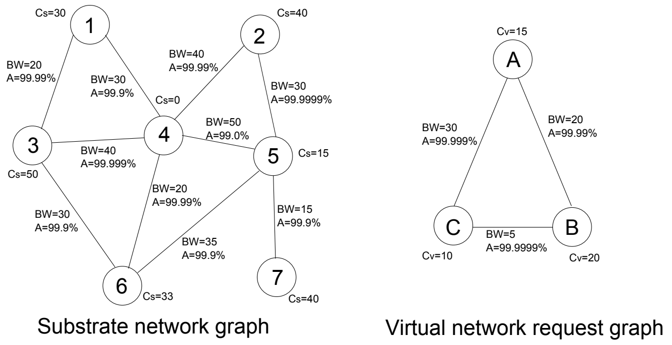

Fig. 1. Network model

specified by $C_S$, e.g. the capacity of a node $i$ is $C_S(i)$. The residual capacity of a substrate node $C_S^R(i)$ is defined as the available capacity of the substrate node $i \in V_S$.

The substrate links represent physical connections between the nodes. A link between node *i* and *j* is given by $e_s(i,j)$ and $\forall e_s(i,j) \in E_S$. Each substrate link has a bandwidth resource $BW$ and a link availability $A$. The link bandwidth of a link $e_s \in E_S$ is $BW(e_s)$. The residual bandwidth of a substrate link $BW^R(e_s)$ is defined as the total amount of bandwidth available on the substrate link $e_s \in E_S$. The link availability of a link $e_s$ is defined as $A(e_s)$. A path from node *s* to node *t* is denoted as $p(s,t)$, which is the collection of the links along the path. The length of the path (the number of hops) is given by $|p(s,t)|$, which is defined by the number of links along the path. $P_S$ is the set of all feasible substrate paths in $G_S$, and $p \in P_S$. The bandwidth of a substrate path $p \in P_S$ is the minimum bandwidth of all substrate links $e_s$ on the path $p$: $BW^R(p) = \min BW^R(e_s), \forall e_s \in p$.

Physical links are not reliable, therefore the availability of links is less than 100%. In this work, we assume, the availability of single links to be independent from each other. We concentrate in this work on the availability of links and complete paths, because resource nodes are commonly implemented with high internal redundancy. Therefore we assume node availability of 100%.

### B. Virtual Network Request

A VNR consists of virtual nodes and virtual links represented by a graph $G_V = (V_V, E_V)$ with node and link requirements. The virtual link requirements are availability $A$ and bandwidth $BW$. A virtual link is given by a connection between two virtual nodes *i* and *j*, $e_v(i,j) \in E_S$. The bandwidth of $e_v(i,j)$ is given by $BW(e_v)$ and the link availability by $A(e_v)$. The virtual nodes have some requirements like CPU, storage, geographical location, etc. The node capacity requirement is specified by $C_V(i)$ for a virtual node $i \in V_V$.

### C. Virtual Network Embedding

The VNE can be considered as a process with two stages: virtual node mapping and virtual link mapping. In the first stage, virtual nodes are mapped to resource nodes in the substrate network as $\mathcal{M}_n : V_V \mapsto V_S$. In the link mapping, the feasible paths between all substrate nodes mapped from the virtual nodes are established by using function $\mathcal{M}_l : E_V \mapsto P_S$ where $\mathcal{M}_l(e_v(i,j)) = p(\mathcal{M}_n(i), \mathcal{M}_n(j)), \forall i,j \in V_V$. Note that we assume virtual nodes to be in different geographical locations and are not mapped to the same substrate node, i.e. each virtual node is mapped to a separate substrate node.

## *D. VNE Problem Formulation with Link Availability Constraints*

Our VNE goal is to embed the VNR in the substrate network with least bandwidth that fits the requirements of the VNR (i.e. sufficient availability and minimized bandwidth). Therefore we design a VNE algorithm to provide high available paths for the VNR. In the mathematical formulation only one path with the required link availability is considered for each virtual link. The VNR embedding can then be formulated mathematically as following:

**Objective:**

$$\text{minimize} \sum_{e_v \in E_V} \sum_{e_s \in E_S} BW(e_v) x_{e_v e_s} \tag{1}$$

**Bandwidth Constraints:**

$$\sum_{e_v \in E_V} BW(e_v) x_{e_v e_s} \leq BW^R(e_s), \forall e_s \in E_S \tag{2}$$

$$x_{e_v e_s} \in \{0,1\}, \forall e_v \in E_V, \forall e_s \in E_S \tag{3}$$

**Path Availability Constraints:**

$$A(e_v) \leq \prod_{e_s \in E_S} A(e_s) x_{e_v e_s}, \forall e_v \in E_V \tag{4}$$

**Link Constraints:**

$$\sum_{w | e_s(u,w) \in E_S} x_{e_v e_s} - \sum_{w | e_s(w,u) \in E_S} x_{e_v e_s} = \begin{cases} 1, & u = s \\ -1, & u = t \\ 0, & u \in V_S \setminus \{s,t\} \end{cases} \quad \forall e_v \in E_V \tag{5}$$

**Node Constraints:**

$$C_V(m) z_{mj} \leq C_S^R(j), \forall m \in V_V, \forall j \in V_S \tag{6}$$

$$dist(loc(m), loc(j)) \leq D_{loc}, \quad \forall m \in V_V, \forall j \in V_S \text{ and } j = \mathcal{M}_n(m) \tag{7}$$

$$\sum_{j \in V_S} z_{mj} = 1, \forall m \in V_V \tag{8}$$

$$\sum_{m \in V_V} z_{mj} \leq 1, \forall j \in V_S \tag{9}$$

The objective function (1) aims to minimize the required bandwidth for the embedding of the VNR. $x_{e_v e_s}$ is a binary variable as indicated by (3) denoting, whether substrate link $e_s$ is part of the mapping of virtual link $e_v$: 1 if true, otherwise 0. Equation (2) represents the bandwidth constraint that the total bandwidth of all the virtual links on the substrate link $e_s$ is limited by its bandwidth constraint $BW^R(e_s)$. Equation (4) represents the path availability constraint. This constraint is a nonlinear constraint. It calculates the path availability and ensures that the path for the virtual link $e_v$ has equal or higher availability than the requested link availability $A(e_v)$ of $e_v$. Equation (5) can be viewed as path continuity constraints, where $s$ and $t$ are source and destination nodes of the path. For all the intermediate nodes on the path, the number of incoming links is equal to the number of outgoing links. Equation (6) represents the capacity constraint for a VNR. $z_{mj}$ is a binary variable, which is 1 if virtual node $m$ is mapped to substrate node $j$, otherwise, it is 0. Equation (7) is the location constraint on the VNR. The distance between the requested location of $m$ and node $j$ it is mapped to should be equal or lower than a defined distance threshold $D_{loc}$. Equations (8) and (9) enforce, that one substrate node can only accommodate one virtual node for one VNR.

## IV. Heuristic for solving the problem

As an exact solution of the mathematical formulation above cannot be found with polynomial effort we developed a heuristic for the node and link mapping.

### *A. Algorithm Design Motivation*

Substrate network links cannot always provide the requested availability, thus several independent parallel links or paths are combined to achieve the availability. Our VNE with path protection is a deterministic algorithm for finding candidate E2E path pairs (primary and backup paths) that meet the availability and bandwidth requirements of the request. However, always calculating a backup path is not bandwidth efficient. If one single path can be found and it fulfills the requested availability constraint, no backup path is required. If no path within the substrate network that can fulfill the requested availability constraint is available, multiple paths are used together to provide the requested availability. One path is used as primary path, the others are backups.

### *B. Heuristic Idea*

The mapping of virtual links to substrate paths is first determined by a graph search algorithm. Here the nonlinearity of the availability constraints (as opposed to e.g. bandwidth constraints) has to be considered: for a path consisting of $x$ substrate links the availability is calculated as $A_{path} = \prod_{i=1}^{x} A_i$. Therefore, if each substrate link on the path fulfills the availability requirement $A_l$ as $A_i > A_l$, $\forall i \in [1,x]$, the overall availability of this path is not necessary satisfying the required VN link availability, i.e. $A_{path} < A_l$. Hence, we calculate the paths using Constrained-based Shortest Path (CSPF) algorithm on bandwidth and afterwards check for the link availability constraint. For each virtual link we compute $k$ candidate paths from which the most cost efficient constraint satisfying path is chosen.

- Step 1: Calculate and select a substrate node candidate that fulfills the virtual node requirements (capacity and location constraint, ...) for each virtual node.
- Step 2: Find out $k$ primary paths using CSPF algorithm on bandwidth that satisfies bandwidth requirements for each virtual link and check for these found paths if the virtual link's availability requirement is fulfilled.
- Step 3: If the found primary paths do not fulfill the availability constraints, find out a disjoint backup path

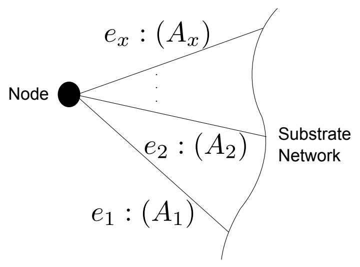

Fig. 2. Incident link failure rate of a substrate node

that satisfies bandwidth requirements for each found primary path.

- Step 4: Check if primary path and backup path in parallel fulfill the virtual link availability requirement.
- Step 5: If the virtual link availability requirement is not fulfilled for the combination of one primary plus one backup path, calculate another disjoint backup path (repeat Step 4 and 5) until a combination of primary and backup path or paths is found that fulfill availability requirements.
- Step 6: Combine candidate primary and backup paths.
- Step 7: Using a linear program to select the best bandwidth efficient combination of path pairs out of these candidate pairs to minimize the overall bandwidth consumption for the VNE.

The following section describes our algorithm in detail.

## C. Candidate Node Selection

For each virtual node a substrate node needs to be found that fulfills the requirements. First the virtual nodes are sorted according to their capacity requirement in descending order. The virtual node with the highest capacity constraint, node *i*, is selected first for mapping. From all substrate nodes, the nodes that fulfill the capacity and location requirements of *i* are selected as possible candidates. From the candidate nodes the overall incident link failure rate is calculated. The overall incident link failure rate of a node is the multiplication of all failure rates of the incident links. The failure rate of a link is 1 minus availability value. For example for a node with $x$ incident links $e_1$ to $e_x$, the overall incident link failure rate is $\prod_{i=1}^{x}(1 - A_i)$, see Figure 2. The node with the lowest incident link failure rate is selected. If two or several nodes have the same overall incident link failure rate, the second criteria is the incident link bandwidth, so the node with the highest bandwidth of the incident links is selected. Otherwise, node mapping cannot be found and the algorithm stops.

After all virtual nodes are successfully mapped to substrate nodes, candidate paths and path pairs need to be found for the connection between the nodes that fulfill the requirements of the virtual links.

## D. Link Mapping with Availability Constraint

*1) Calculation of candidates for link embedding:* The next step is the calculation of candidate paths and path pairs for the virtual links. The virtual links are sorted according to the link availability requirements in decreasing order. If two or more virtual links have the same availability value, they are sorted according to the bandwidth requirements in decreasing order.

For each virtual link $e_v$ (start with the one with the highest link availability requirement) calculate *k* CSPF as primary paths between the mapped nodes with the constraint on the bandwidth of the path. The *k* is an operational parameter which is a number greater or equal than 2 and can be chosen by the VNO to calculate several candidates. The found paths which fulfill the bandwidth requirements for virtual link $e_v$ are considered as the candidates for the primary path. For each possible primary path the path availability is calculated and compared to the requested link availability of $e_v$. The E2E path availability of a path $A_{path}$ with $n$ links is the link availability of each link under the assumption that the single link failures are independent, $A_{path} = \prod_{i=1}^{n} A_i$. If the availability requirement is fulfilled for the primary path, this primary path is added to the set of candidate primary paths for link $e_v$ and no further backup path need to be calculated.

If the path availability is lower than the requested link availability of $e_v$, an additional path (backup path) is needed for fulfilling the availability requirement. A link-disjoint path (=backup path) to the existing primary path from the source to destination node fulfilling the requested bandwidth requirements is calculated (using CSPF algorithm with constraints on bandwidth). We selected link-disjoint paths and not completely disjoint paths as the unavailability of nodes is neglected here due to the fact that the nodes are commonly implemented with high internal redundancy. In case a link-disjoint path can be found, the availability constraint must be checked for this path. Even though the backup path might not be the shortest path, it can have a higher path availability than the primary path as the availability constraint is not a linear constraint. Therefore if the availability requirement is fulfilled for the backup path, this path becomes the primary path and is added to the set of candidate primary paths for link $e_v$.

In case the disjoint path has lower availability, the availability of the combination of primary and backup path needs to be checked. The availability of these two paths (primary and backup path) is calculated as following: the availability of two parallel paths is $A_{parallelPaths} = 1 - (1 - A_{primary}) \times (1 - A_{backup})$, where $A_{primary}$ is the availability value of the primary path and $A_{backup}$ is the availability value of backup path. If the availability of primary and backup path is equal or greater than the requested link availability of $e_v$, then generate a pair (primary path, backup path) and add this pair the set of candidate path pairs. In case the link availability can still not be satisfied with the primary and backup path another link-disjoint backup path can be calculated to increase the availability and meet the requirement. The second backup path is calculated in the same way as the first backup path. The primary and backup paths can be combined in a way that two backup paths satisfy the virtual link availability requirement without the primary path. Then the backup path with the highest availability value is the primary path. After successful completion of this process for all virtual links, every virtual link has a set of candidate paths and/or candidate path pairs.

*2) Path Selection:* After finding candidate path pairs for each virtual link in the VNR, a bandwidth efficient combination of these candidates for the complete VNE will be calculated using Integer Linear Programming (ILP) that selects the suitable candidate for each connection:

**Objective:**

$$\text{minimize} \sum_{e_v \in E_V} \sum_{p \in P_{e_v}} |p| BW(e_v) x_{e_v p} \tag{10}$$

**Bandwidth Constraints:**

$$\sum_{e_v \in E_V} \sum_{p \in P_{e_v}} BW(e_v) x_{e_v p} 1_{e_s \in p} \leq BW^R(e_s), \forall e_s \in E_S \tag{11}$$

$$x_{e_v p} \in \{0, 1\}, \forall e_v \in E_V, \forall p \in P_{e_v} \tag{12}$$

**Link Constraints:**

$$\sum_{p \in P_{e_v}} x_{e_v p} = 1, \forall e_v \in E_V \tag{13}$$

$$\sum_{e_v \in E_V} x_{e_v p} \leq 1, \forall p \in P_{e_v} \tag{14}$$

The objective function (10) minimizes the bandwidth for VNR embedding while selecting the paths which require the least bandwidth. $P_{e_v}$ is the set of pre-selected constrained disjoint candidate path pairs (primary + backup path) for virtual link $e_v$. The pre-selected constrained disjoint candidate path pairs step is done in the previous subsection. The length of a path/pair $|p|$ is the sum of the links along the paths, which is pre-calculated for each path pair. $x_{e_v p}$ is a binary variable as indicated by (12) denoting, whether virtual link $e_v$ is assigned to the disjoint path pair $p$ of primary and backup paths: 1 if true, otherwise 0. Equation (11) ensures that the bandwidth of substrate link $e_s$ is not overused by all virtual links that are mapped to a substrate path $e_s$. $1_{e_s \in p}$ is a indicator which is 1 if $e_s$ is part of the path $p$, and 0 otherwise. Equation (13) and (14) imply that only one substrate path pair is selected out of the candidate pairs for virtual link $e_v$. After the ILP finds the solution, the selected paths or path pairs are mapped to the corresponding virtual links.

## V. Evaluation

In this section, the performance of our algorithm is evaluated. To this end, we simulate the arrival of VNR as discrete events and compare our heuristic algorithm (with only up to one backup path) to the optimal MIP algorithm and a simple algorithm with shortest link-disjoint paths, which we call link-disjoint paths heuristic. The simple shortest disjoint paths algorithm approach uses Suurballe's algorithm [9] to find the shortest link-disjoint paths for each virtual link. These two link-disjoint paths are used to check if the requested virtual link availability is fulfilled. For node mapping, our algorithm is used and the linear programs are solved with glpk [10].

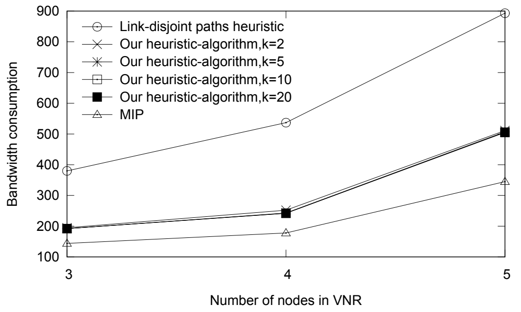

Fig. 3. Comparison of bandwidth consumption of heuristics and optimal solution embedding algorithms with average nodal degree of 4

### A. Simulation Settings

To evaluate our algorithm, we have implemented a Java based simulator. For the substrate network (SN), a random graph with an average nodal degree between two and four is created. Each node has a capacity resource between [0, 300], each link gets a bandwidth resource from the interval [10, 200] and a link availability between 99% and 99.999%. For the virtual network request (VNR), the number of nodes is distributed between [2, 5]. The probability of connectivity between every two virtual network nodes is 0.5. The VNR is a connected graph. Each node has a capacity demand between [2, 10]. Each link has bandwidth demand between [1, 100] and a desired link availability (between 99.9% and 99.9999%). The resource and demand values are randomly created.

### B. Evaluation Results

*1) Overall comparison:* For simulations, we compare the used bandwidth consumption of embedding a VNR for the heuristics with the optimal results derived from exhaustive search obtained from the MIP. The MIP model is built based on the mathematical formulation at III-D which requires only primary path and no backup path. The non-linear availability constraint is solved using the logarithm to eliminate the non-linearity and reformulate it as a linear constraint. Figure 3 shows the averaged results after 50 iterations of the bandwidth consumption of VNE obtained from different approaches using a SN with ten nodes and an average nodal degree of four. For the heuristics we use the same end node mapping mechanism to eliminate the influence from the node embedding, to compare the bandwidth consumption efficiency from different VNE solutions. The optimal solutions are obtained by searching the complete problem space, hence they indicate the results that are not only optimized for link mapping but also for node mapping. The optimal results are better than the heuristic algorithms, but the required computing resources and computing time cannot be neglected. For our proposal we use different values of k, the parameter which indicates how many primary candidates are offered to the optimization. With increasing k, the bandwidth consumption is getting closer to the optimal value due to selecting the best paths out of more possible candidates. Similar results are achieved using a SN with average nodal degree of two. In this case for the VNR the different values of k have nearly no influence on the bandwidth

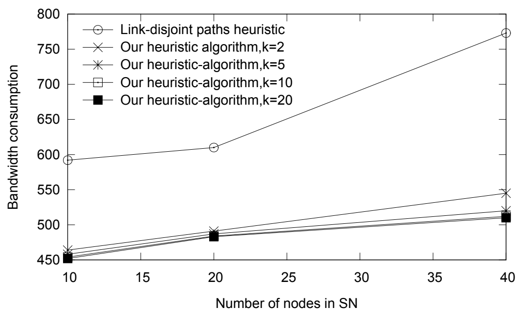

Fig. 4. Comparison of bandwidth consumption of heuristics, number of nodes in VNR=4, SN link availability 99% to 99.99%

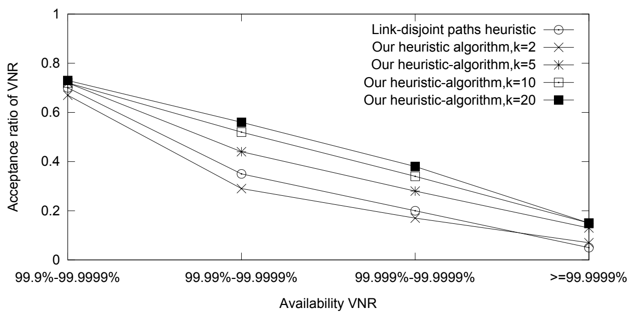

Fig. 5. Acceptance rate of heuristics, number of nodes in SN=10

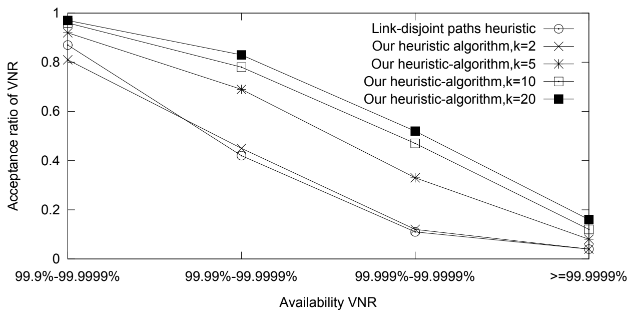

Fig. 6. Acceptance rate of heuristics, number of nodes in SN=40

usage, i.e. with k=2 a near optimal solution for our heuristic can be found.

For high available link requirements at VNR, the optimal MIP algorithm which tries to only find a primary path with the requested availability value cannot be used when the SN links have low availability since short high available paths are hardly found then. When the available link requirements at VNR are higher than the physical link availability value it is obvious that the MIP cannot find a path since it only calculates one primary path.

*2) Bandwidth Consumption:* Bandwidth usage is one of the main metrics to evaluate the VNE algorithm efficiency. Therefore, we compare the bandwidth consumption for each VNR by using the two heuristics, our proposal (with different values for k) and the link-disjoint paths heuristic. The results in Figure 4 show the bandwidth consumption for embedding a VNR and indicate that our proposed algorithm consumes less bandwidth to embed a VNR compared to the link-disjoint paths heuristic. In several cases a VN link can be embedded with our heuristic using a single working path with high availability alone whereas in the link-disjoint paths heuristic all VN links require an backup path in the SN.

*3) Substrate Network Influence and Availability Value Influence:* The topology of the SN and the requested link availability will also influence the VNE performance to a certain extent. To review the influence, we compare with different sizes of nodes in substrate network. Different requirements for VNR availability between 99.9% and 99.9999% were specified. The average nodal degree for the substrate network is 4. The results shown in the Figures 5 and 6 are the mean values over 100 iterations. To obtain the VN acceptance ratio, we divide the number of accepted requests by the total number of inputted requests. As shown in the figures, our proposed algorithm has equal or higher VNR acceptance ratio for k $\geq$ 5 compared to the link-disjoint paths approach for VNE. When the network size is increased, the acceptance ratios increase in both heuristics. When the availability constraint is tightened, the acceptance ratios drop in all cases. For the link-disjoint paths heuristic this is more obvious than for our proposal. In all cases the acceptance ratios drop if the availability constraints are tightened as the latter cannot be satisfied at all by the paths between the selected nodes. When the average nodal degree in the SN gets lower, in all cases the acceptance ratios dropped.

## VI. Conclusions

In this paper, we have formulated the VNE problem with explicit availability requirements for links. For solving this problem a heuristic algorithm was proposed and evaluated. Our heuristic has achieved good bandwidth consumption compared to an optimal solution. This heuristic can be used to provide link protection for VNE using several low cost/low available links and achieving virtual networks for high availability requirement services. In the future we will consider the node availability also, to consider, e.g., the selection between multiple micro data centers for the embedding of a service.

## References

[1] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, March 2008.

[2] D. G. Andersen, "Theoretical Approaches to Node Assignment," Dec. 2002, unpublished manuscript.

[3] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable Virtual Network Embedding," in *IFIP/TC6 NETWORKING*, 2010, pp. 40–52.

[4] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared Backup Network Provision for Virtual Network Embedding," in *IEEE International Conference on Communications (ICC)*, June 2011.

[5] H. Yu, V. Anand, C. Qiao, and H. Di, "Migration based protection for virtual infrastructure survivability for link failure," in *OFC/NFOEC*, March 2011, pp. 1 –3.

[6] J. Shamsi and M. Brockmeyer, "QoSMap: Achieving Quality and Resilience through Overlay Construction," in *4th International Conference on Internet and Web Applications and Services (ICIW)*, May 2009.

[7] X. Zhang, C. Phillips, and X. Chen, "An overlay mapping model for achieving enhanced QoS and resilience performance," in *3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Oct. 2011, pp. 1 –7.

[8] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance,Analysis,and Design*. John Wiley & Sons, Inc., 2002.

[9] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974.

[10] A. Makhorin, "GLPK (GNU Linear Programming Kit)," Available at http://www.gnu.org/software/glpk/glpk.html.