A Virtual Network Migration Approach and Analysis for Enhanced Online Virtual Network Embedding

Mahboobeh Zangiabady, Christian Aguilar-Fuster and Javier Rubio-Loyola
Centre for Research and Advanced Studies of the National Polytechnic Institute
Ciudad Victoria, Tamaulipas, Mexico
{mzangiabady, caguilar, jrubio}@ tamps.cinvestav.mx

***Abstract*— The Virtual Network Embedding (VNE) problem has atracted a lot of attention in the last decade. Moreover, recent analysis demonstrates that the performance of VNE solutions decreases drastically with large scale substrate networks. This paper proposes a VN migration approach intended to enhance the performance of online VNE in terms of VN acceptance ratio, whose enhancements are more accentuated in large networks. We propose an approach to re-allocate virtual elements considering both, additive and non-additive QoS requirements. We advance the state of the art by proposing a VN migration approach that can be programmed by Infrastructure Providers (InP) to drive migrations systematically. The effectiveness of our approach is demonstrated through extensive simulations and analysis.**

Keywords—Virtual Network Embedding; Virtual network migration; Quality of Service; Virtual Network acceptance ratio improvement; Virtual Link Migration Problem

I. Introduction

Network Virtualization (NV) is a promising technology for overcoming Internet ossification by enabling multiple Virtual Networks (VNs) to coexist on a shared substrate network. Its main objective is to build heterogeneous networks to support a variety of network services and architectures through a shared substrate infrastructure, where each VN can provide services for customers and applications [5]. One of the most critical aspects in NV environments is the capability of infrastructure providers (InPs) to allocate resources in the substrate network to support the VNs in an optimal manner. This is known as the Virtual Network Embedding (VNE) problem that has received considerable attention by the research community [5], [9]. Solutions to the VNE problem are classified as online and offline [5].

Online embedding is the process of mapping VNs once each VN request is received, evaluating the current state of substrate resources at the time of the VN request and selecting the resources that would support its resource requirements. The arrival time, resource requirements, and time of life of the VNs are unknown and as such online VNE is a rather difficult problem. Offline embedding relies on prior knowledge of all VN arrivals, and therefore finding more optimal solutions for a given batch of VN arrivals is easier. In any case, the aim of the VNE problem is to optimize the use of substrate network resources while allocating resources for VNs, aligned to optimization targets that may include, among others, Quality of Service (QoS), energy, efficiency, resilience, VNE cost, and profit.

VNE is a combinatorial optimization problem for which the networking research community has used several methods and algorithms to find optimal VNE solutions [9]. However, recent analysis has demonstrated that the performance of VNE solutions in key metrics like VN acceptance ratio decreases drastically [2] [16] with practical instances, i.e. instances of large size substrate and virtual networks. It is therefore necessary to design mechanisms to overcome these limitations.

This paper proposes an approach to enhance the performance of metaheuristic-based online VN embeddings for large substrate networks. The proposed approach is based on systematic reallocation of substrate resources at VNE runtime. In this paper, the terms reconfiguration and migration of VNs are used interchangeably to refer to runtime reallocation of substrate resources for VNs. The pivotal element of our approach is the Virtual Link Migration Problem (VLMP), which is a concept proposed to drive VN migrations systematically. VLMP is meant to find optimal reconfigurations of Virtual Links (VLs) on the substrate network in runtime. The VLMP is solved with a shortest-path-based multi-constraint optimal path selection algorithm customized for NV. Our approach considers both, additive (i.e. delay and hop count) and non-additive (i.e. CPU and BW) QoS constraints and it has been designed to give InPs the flexibility to reconfigure resources according to different QoS criteria while enhancing the VNE acceptance ratio, especially for practical instances. The paper analyses key configuration aspects that drive the performance of our migration approach and we demonstrate its effectiveness, highlighting its benefits for online VNE.

The rest of this paper is organized as follows. Section II presents the related work related. Section III presents our VN migration approach. Section IV presents results and analysis of our simulations. Section V concludes the paper.

II. Related Work

The majority of work on network migration has been centred on defining mechanisms to reconfigure virtual elements within the substrate network. Authors in [22] proposed VROOM (Virtual Routers on the Move), which is an approach intended to avoid unnecessary changes to the logical topology by allowing (virtual) routers to freely move from one physical node to another. Authors in [1] developed a scheduling approach that minimizes both, the total migration time and overhead. The algorithms determine an optimal single-node-at-a-time sequence of moves to minimize migration overhead and a scheduling process that allows for multiple node moves in parallel. Authors in [6] proposed a method to drive VN migration with a pre-copy approach in which the total downtime as well as service disruption time can be reduced to very low levels. Authors in [3] proposed a method to transfer the memory of virtual machines (VM), VM's file system and ongoing network connections between VMs to reduce service disruption. The above works consider that the target physical resources where the virtual elements will be moved are in fact the most ideal ones. Nevertheless, choosing the most suitable resources to host the virtual elements to be migrated is a critical aspect that has received relatively very little attention. Exceptional works in this direction are described in the remainder of this section.

Authors in [17] proposed an approach based on VL path splitting over multiple substrate paths, which is biased with a link-mapping algorithm to optimize the utilization of the substrate network. The approach relies on path migrations into splitted paths over the substrate network. Authors in [20] propose a scheme that prioritizes the migration of the most critical VNs driven by an algorithm that finds the most stressful physical nodes and links, and henceforth it reconfigures the VN accordingly. A key limitation of the above works is the lack of considerations of VNs' constraints during the migration-oriented decision process. More recently authors in [12] proposed an embedding algorithm with a migration option in which physical link BW utilization is driven to trigger virtual link migration. A key limitation of their work is that it does not consider node-remappings. Authors in [21] present an approach for VN reconfiguration that hypes VNE acceptance ratio and load balancing enhancements. The limitation of their work is that the trigger for the migration process is centred to prioritize highly stressed nodes and links and leave aside the constraints of the VNs.

Unlike the work in this paper, the validations/simulations in all the above works were restricted to small or medium sized substrate network instances in each case not exceeding 100 nodes. Authors in [15] proposed an integer linear programming VNE approach that assesses node migration and link re-mapping. A key limitation of their work is that it leaves aside both, additive and non-additive QoS constraints of the VNs. In addition, the experimental design of this work [15] was limited to substrate networks with 200 nodes. All the above works have addressed some aspects of VN migration. Nevertheless, none of these approaches has defined a systematic approach for virtual network migration, with programmable features that provide InPs flexibility to manage network resources under migration scenes. All in all, none of the above works has been validated their approaches in different network size and link density scenarios, and their feasibility to address real-life VN scenarios is questioned.

A. VN Migration Approach

This sub-section provides a general description of our migration approach, which is designed to work in synchrony with online VNE and it is described with the following steps.

1) The InP configures the migration process

1.1) Migration triggering events are configured, potential ones are: i) a given number of time units; ii) a given number of Virtual Network Requests (VNRs) arrivals during online VNE; iii) thresholds crossings of resource usage on nodes and links; iv) thresholds crossings on the online VNE acceptance ratio.

1.2) Migration policies (explained later) are configured. They define the type of substrate resources that would support the to-be-migrated VLs.

2) Migration approach execution

2.1) Evaluate the triggering condition event. When the triggering condition holds, execute the following steps:

2.2) Define a set VN' that includes the VNs that are embedded and active in the substrate infrastructure.

2.3) Define the set VN'' that includes VNs from VN', whose remaining time is larger than threshold $\mathrm{T_{thresh}}$ at the time of execution of this step 2. $\mathrm{T_{thresh}}$ is also a parameter that can be configured to give InPs the flexibility (if desired) to set limits over the number of potential VNs subject of reconfiguration. The elements in VN'' are ordered from the VN with the longest remaining time to the VN with the lowest one. This approach is adopted from [17] and it is intended increase the chances for VNs with longer remaining times to be reconfigured.

2.4) Define the set VL_{target} that includes VLs from VNs in VN'' that are served by (i.e. mapped in) physical resources (CPU and links) that are highly-utilized at the time of execution of this step 2. The InP defines the percentage of utilization of a physical resource (CPU and links) to be considered highly utilized. The VLs in VL_{target} are ordered in accordance with the order of their VN in VN''.

2.5) For each VL in VL_{target}:

2.5.1) Execute the VLMP algorithm (explained in detail in the next subsection) to find the available, feasible and optimal physical resources that can host the VL under evaluation.

2.5.2) If the three conditions (availability, feasibility and optimality) are not met, continue with next VL in VL_{target} until the VLMP algorithm finds the physical resources that meet the three conditions for the VL under evaluation.

2.5.3) When VLMP finds the resources that meet the three conditions (availability, feasibility and optimality); the VL under evaluation is selected for migration. The latter will be named from now on VL_{on_mig}.

2.5.4) Pause online VNE, start queuing the VNRs until migration is over (step 2.5.6).

2.5.5) Reconfigure the VL_{on_mig} onto the physical resources selected in step 2.4. Reactivate the reconfigured network.

2.5.6) Release the resources allocated for VL_{on_mig} before the migration. Update the state of resources in the substrate network.

2.5.7) Restart online VNE, dequeue the VNRs that arrived during the migration steps (2.5.5 and 2.5.6).

2.5.8) Evaluate the migration trigger condition, if it is met repeat the process from step 2.2.

The following sections give details of key aspects of the VLMP algorithm which in turn is pivotal for our migration approach.

B. VLMP model

We model a physical network as a weighted undirected graph where each node u_i and link (i,j) have associated parameters. Let $G_{phys} = (V_{phys}, E_{phys})$ denote a physical network with physical node set V_{phys} and physical edge set E_{phys}. Let $n_{phys} = |V_{phys}|$ be the number of physical nodes in G_{phys} and $m_{phys} = |E_{phys}|$ be the number of physical links in G_{phys}. Let $p = (u_1, u_2, \ldots, u_i) \in V_{phys}$, denote a simple physical path in G_{phys}. Let $d_{phys} \in [0,1]$ denote the density of the physical network G_{phys}, i.e. the ratio of the number of links in the physical network to the maximum possible number of links. Each physical node $u_i \in V_{phys}$ is associated with an available CPU percentage $cpu_{phys}^{avail}(u_i) \in [0,100]$. Likewise, each physical link $(i,j) \in E_{phys}$ is associated with an available BW percentage $bw_{phys}^{avail}(i,j) \in [0,100]$ and a delay $dly_{phys}(i,j)$. A physical link is also associated to a cost $cost_{phys}(i,j)$. Additionally, a physical path p is associated with a hop count $hop_{phys}(p) = |p| - 1$.

We model a virtual network as a weighted undirected graph, where each virtual node v_i and virtual link (k,l) have associated requirements, namely requested CPU and requested BW respectively. Let $G_{virt} = (V_{virt}, E_{virt})$ denote a VNR with virtual node set V_{virt} and virtual edge set E_{virt}. Let $d_{virt} \in [0,1]$ denote the density of the virtual network G_{virt}. Each virtual node $v_i \in V_{virt}$ has a requested CPU percentage $cpu_{virt}^{req}(v_i) \in [0,100]$. Each virtual link $(k,l) \in E_{virt}$ is associated with a requested BW percentage $bw_{virt}^{req}(k,l) \in [0,100]$. The VLMP concept is proposed to find the substrate resources where a given VL can be re-mapped. We opted for a VL migration approach because network link failures occur about 10 times more than node failures [14]. Let $VL_{on\_mig}(k_{mig}, l_{mig}) \in E_{virt}$ be a VL candidate to be migrated (see steps 2.5.1 to 2.5.3 in sub-section III.A), let expressions $cpu_{constr}(VL_{on\_mig})$, $bw_{constr}(VL_{on\_mig})$, $dly_{constr}(VL_{on\_mig})$, and $hop_{constr}(VL_{on\_mig})$ be the CPU, BW, delay and path hop constraints of VL_{on_mig} respectively, and let $C_{VL_{on\_mig}} = (cpu_{constr}, bw_{constr}, dly_{constr}, hop_{constr})$ be the set of VL_{on_mig} constraints. VLMP is aimed at finding a physical path $p_{vlmp} = (u_s, \ldots, u_t) \in V_{phys}$ from a source physical node u_s, onto which virtual node k_{mig} is mapped, to a destination (target) physical node u_t, onto which virtual node l_{mig} is mapped, such that the following three conditions are met:

VLMP availability condition

This condition guarantees that all substrate links along $p_{vlmp} = (u_s, \ldots, u_t) \in V_{phys}$ will have enough BW and enough CPU required by VL_{on_mig}. Let $cpu_{phys}^{avail}(p_{vlmp}) = min_{\,u \in p_{vlmp}}\{cpu_{phys}^{avail}(u)\}$ be the minimum available CPU all the way along $p_{vlmp} = (u_s, \ldots, u_t) \in V_{phys}$, let $cpu_{constr}(VL_{on\_mig}) = cpu_{VL_{on\_mig}}^{req}(p_{vlmp})$ be the CPU needed to achieve the desired forwarding rate $VL_{on\_mig}\ \{k_{mig}, l_{mig}\} \in E_{virt}$, let $bw_{phys}^{avail}(p_{vlmp}) = min_{(i,j) \in p_{vlmp}}\{bw_{phys}^{avail}(i,j)\}$ be the minimum available BW all the way along p_{vlmp}, and let $bw_{constr}(VL_{on\_mig}) = bw_{VL_{on\_mig}}^{req}(k_{mig}, l_{mig})$ be the requested BW by the $VL_{on\_mig}\ \{k_{mig}, l_{mig}\} \in E_{virt}$. The VLMP availability condition is formally defined as:

$$cpu_{phys}^{avail}(p_{vlmp}) \geq cpu_{constr}(VL_{on\_mig}) \ \ldots.. (1.1)$$

$$bw_{phys}^{avail}(p_{vlmp}) \geq bw_{constr}(VL_{on\_mig}) \ \ldots... (1.2)$$

In order to meet the availability condition the physical network G_{phys} is pruned to: i) discard any physical node whose available CPU is lower than $cpu_{constr}(VL_{on\_mig})$; and ii) to discard any physical link whose available BW is lower than $bw_{constr}(VL_{on\_mig})$. If conditions 1.1 and 1.2 are met, the result is a pruned physical network $G'_{phys} = (V'_{phys}, E'_{phys})$ whose paths between u_s and u_t meet the VLMP availability condition, where $V'_{phys} \subseteq V_{phys}$ and $E'_{phys} \subseteq E_{phys}$.

VLMP feasibility condition

Let $dly_{phys}(p_{vlmp}) = \sum_{(i,j) \in p_{vlmp}} dly_{phys}(i,j)$ be the sum of the delays along p_{vlmp}, and let $hop_{phys}(p_{vlmp})$ be the hop count of path $p_{vlmp} = (u_s, \ldots, u_t) \in V_{phys}$. Expressions 2.1 and 2.2 are constraints that are considered while evaluating the VLMP feasibility condition of p_{vlmp}.

$$dly_{phys}(p_{vlmp}) \leq dly_{constr}(VL_{on\_mig}) \ \ldots.. (2.1)$$

$$hop_{phys}(p_{vlmp}) \leq hop_{constr}(VL_{on\_mig}) \ \ldots... (2.2)$$

Expression 2.1 evaluates whether the sum of the physical links' delays of all substrate nodes along $p_{vlmp} = (u_s, \ldots, u_t) \in V_{phys}$ is not higher than $dly_{constr}(VL_{on\_mig})$, whereas expression 2.2. evaluates if the hop count of path p_{vlmp} is not higher than $hop_{constr}(VL_{on\_mig})$. These two expressions are used define two feasibility cost functions (FCF) which were originally proposed in [23] and that we have customized for our VN migration approach as follows:

$$g^{dly}(p_{vlmp}) = \frac{dly_{phys}(p_{vlmp})}{dly_{constr}(VL_{on\_mig})} \qquad (2.3)$$

$$g^{dual}(p_{vlmp}) = \frac{dly_{phys}(p_{vlmp})}{dly_{constr}(VL_{on\_mig})} + \frac{hop_{phys}(p_{vlmp})}{hop_{constr}(VL_{on\_mig})} \qquad (2.4)$$

The FCF of expression 2.3 enables InPs to define the feasibility condition as a function of the fulfillment of the delay constraint, whereas the FCF of expression 2.4 defines feasibility as a function of both, delay and hop count constraints. The feasibility condition is met by running a Reverse Dijkstra algorithm [23], which is a variation of Dijkstra that explores the graph G'_{phys} from the destination (target) substrate node u_t to the source substrate node u_s and it eventually returns a path that minimizes the cost function (the one selected by the InP from expressions 2.3 or 2.4). For example, let us consider that the InP opts to define feasibility as a function of both, delay and hop-count constraints. When evaluating feasibility, a weight $w_{ij} = \frac{dly_{phys}(i,j)}{dly_{constr}} + \frac{hop_{phys}(i,j)}{hop_{constr}}$ is assigned to every physical link $\{i,j\} \in E'_{phys}$ and eventually, Reverse Dijkstra will find a path that minimizes expression 2.4, if such a feasible path exists. VLMP exploits this approach to find a set of feasible paths fp_{vlpm} that could host VL_{on_mig} on the pruned physical network G'_{phys}, in which each feasible path p_{vlmp_fea} in fp_{vlmp} meets either: i) the delay constraint minimizing FCF 2.3; or ii) both, delay and hop count constraints minimizing FCF 2.4. This programmable approach is intended to give InPs flexibility to manage QoS levels of the VNs deployed on the substrate, and if needed, to make counter-offers when the required levels of QoS cannot be maintained.

VLMP optimality condition

This condition is evaluated to find, from the set of feasible paths P_{vlmp_fea}, the one that uses the lowest amount of resources. This condition is formally defined as follows:

$$p_{vlmp\_opt} = \mathrm{argmin}_{p_{vlmp\_fea} \in P_{vlmp\_fea}} \left\{ \sum_{(i,j) \in p_{vlmp_{fea}}} cost_{phys}(i,j) \right\},$$

where P_{vlmp_fea} is the set of all feasible paths, and $cost_{phys}(i,j)$ defines the cost of p_{vlmp_fea} in terms of the resources' utilization along the path. The first step towards optimality is to evaluate the utilization of the substrate resources associated to the set P_{vlmp_fea}. Let G''_{phys} be the pruned physical network of the substrate resources associated to set P_{vlmp_fea}. Let $cpu_{phys}^{use}(u_i) = 100 - cpu_{phys}^{avail}(u_i)$ be the CPU currently used by the substrate node $u_i \in p_{vlmp\_fea}$ and let $bw_{phys}^{use}(i,j) = 100 - bw_{phys}^{avail}(i,j)$ be the BW currently in use by link $(i,j) \in p_{vlmp\_fea}$. In order to define optimality we opted for two cost functions originally proposed in [23], customized for our VN migration approach as follows:

$$h^{cpu}(p_{vlmp\_fea}) = \sum_{u_i \in p_{vlmp\_fea}} \left(\frac{cpu_{phys}^{use}(u_i)}{cpu_{thresh}^{use}} \right)^{\beta} \quad (3.1)$$

$$h^{dual}(p_{vlmp\_fea}) = \sum_{u_i,(i,j) \in p_{vlmp\_fea}} max \left\{ \frac{cpu_{phys}^{use}(u_i)}{cpu_{thresh}^{use}}, \frac{bw_{phys}^{use}(i,j)}{bw_{thresh}^{use}} \right\}^{\beta} \quad (3.2)$$

Expression 3.1 is an optimality cost function (OCF) that models the cost of a feasible path p_{vlmp_fea} as proportional to the sum of the CPU in use of the nodes along the path. OCF in expression 3.2 models the cost of a feasible path as a function of both, CPU and BW in use of the nodes and links along p_{vlmp_fea}. The resources in use are normalized by percentage threshold values for cpu_{thresh}^{use} and bw_{thresh}^{use} as to express when the resources are highly-utilized. The parameter β helps the Look-ahead Dijkstra algorithm to find paths that contain bottlenecks, i.e. paths having nodes and/or links whose resources in use are higher than the thresholds, this way penalizing them during the search process. In our approach a value β=5 gives the best VLMP performance results.

The VLMP optimality condition is met by running a Look-ahead Dijkstra algorithm [23] to explore G''_{phys} for each $(i,j) \in p_{vlmp\_fea} \subset P_{vlmp\_fea}$. Look-ahead Dijkstra eventually returns a path that minimizes an OCF (the one selected by the InP from expressions 3.1 or 3.2) from the set P_{vlmp_fea}. For example, let us consider that the InP opts to define optimality as a function of both, CPU and BW in use of the nodes and links along feasible path p_{vlmp_fea}. When evaluating optimality a weight $w'_{ij} = \left\{ \frac{cpu_{phys}^{use}(u_i)}{cpu_{thresh}^{use}}, \frac{bw_{phys}^{use}(i,j)}{bw_{thresh}^{use}} \right\}^{\beta}$ is assigned to every link $\{i,j\} \in G''_{\mathrm{phys}}$ in the physical network and eventually Look-ahead Dijkstra finds a path p_{vlmp_opt} that minimizes expression 3.2 if such optimal path exist. Making use of the Look-ahead Dijsktra algorithm [23] VLMP finds an optimal path p_{vlmp_opt} that could host VL_{on_mig} on the pruned physical network G''_{phys} that meets either: i) the cpu constraint by minimizing OCF 3.1; or ii) both, CPU and BW constraints by minimizing OCF 3.2. This programmable approach is intended to give InPs flexibility to manage QoS levels of the VNs deployed on the substrate, and if needed, to make counter-offers when the required levels of QoS cannot be maintained.

III. VLMP SIMULATION RESULTS AND ANALYSIS

A. Configuration Setup

Migration is triggered by triggering events (TEs) configured by the InP. The four TEs and their corresponding values considered in our evaluation are: TE1) a number of time units (25 time units as in [20]); TE2) a number of VN embeddings (four VN embeddings as in [15]); TE3) when the 25% of physical nodes or links are highly utilized (utilization above 60%); TE4) when the VNE acceptance ratio is below 80%. Feasibility and optimality options are configured making use of the policies shown in Table I, which have been defined to link the migration process with QoS-aware decisions. The table indicates the conditions and cost functions subject of minimization by VLMP. The name of each policy indicates the type of physical paths that would host the to-be-migrated VLs.

TABLE I. QoS-ORIENTED MIGRATION POICIES

Migration Policy	**Function to be minimized**
Paths with high avail. CPU	Cond. 1.1 and 1.2 OCF 3.1
Paths with high avail. CPU and BW	Cond. 1.1 and 1.2 OCF 3.2
Paths with high available CPU and BW and low delay	Cond. 1.1 and 1.2 FCF 2.3, OCF 3.2
Paths w/ high available CPU and BW, low delay and hop count	Cond. 1.1 and 1.2 FCF 2.4, OCF 3.2

An exhaustive analysis of the optimal values of the parameters of our FCFs and OCFs is out of the scope of this paper as they are mostly dependant on specific scenarios, most probably with values linked to high-level objectives. However, for validation purposes, we tried four ranges of delay and hop count VLMP constraints and four ranges of BW and CPU VLMP thresholds, whose impact to online VNE was evaluated. The configuration values that produced the best results in terms of migration success ratio are shown in Table II and these values have been used to validate the efectiveness of our approach.

TABLE II. VLMP CONSTRAINTS AND THRESHOLDS USED IN VALIDATION

$\boldsymbol{dly_{constr}}$	$\boldsymbol{hop_{constr}}$	$\boldsymbol{cpu_{thresh}^{use}}$	$\boldsymbol{bw_{thresh}^{use}}$
300 msec	3 hops	60 %	50 %

B. Simulation setup

Our migration approach has been implemented over an online VNE simulator. Our simulation platform includes an instance generator that produces the substrate networks and VNRs. The topologies (substrate and virtual) are produced with the GT-ITM tool [18], which is one of the most widespread tools to generate connected graphs that reproduce effectively the hierarchical topologies in Internet. We added attributes of the graph elements, specifically, CPU processing capacity of substrate and virtual nodes, BW and delay of substrate and virtual links, and time of life of the VNs. These attributes are generated for each VNR taking into account probability density functions. CPU and BW capacities and delay have been generated using a minimum and maximum range following a uniform distribution. The time of life of VNs assigned follows an exponential distribution with a desired mean. VNRs are generated following a Poisson distribution that determines the order and arrival time for each VNR. In our evaluation we have used a simulation setup similar to the ones reported in the literature for online VNE [19], [7], [13], [17]. Tables III and IV summarize the physical and virtual network configurations used for our simulations. Finally, our simulation setup incudes implementations of Genetic Algorithm (GA) [4], Particle swarm optimization (PSO) [19], Ant Colony (ACO) [4][8], and Harmony Search (HS) [10], this with the intention to analyse the robustness of our migration approach with four metaheuristics widely used in online VNE.

TABLE III: PHYSICAL NETWORK CONFIGURATION

Property	Symbol	Min	Max
Physical nodes	n_{phys}	100, 150, 200 and 250 nodes	
Density	d_{phys}	0.06, 0.12, 0.25 and 0.6	
Available CPU	cpu_{phys}^{avail}	50%	100%
Available BW	bw_{phys}^{avail}	50%	100%
Link delay	dly_{phys}	0 msec.	150 msec.

TABLE IV. VIRTUAL NETWORK CONFIGURATION

Property	Symbol	Min	Max
Density	d_{virt}	0.5	
Lifetime	l_{virt}	500 time units	
Virtual nodes	n_{virt}	2	10
Requested CPU	cpu_{virt}^{req}	1%	20%
Requested BW	bw_{virt}^{req}	1%	50%
Number of VNRs	1000		
Simulation time	25000 time units		

C. Experiment selection and results

An exhaustive evaluation of a full factorial combination of migration triggers (4), migration policies (4), VNE metaheuristics (4), physical network sizes (4), and densities (4), executing experiments that would lay to statistically valid results is impractical as it may take several months of simulations even with HPC machines. To reduce the number of experiments to a subset of a full factorial design while preserving the most important variable interactions, we used a Covering Array (CA) [11]. A CA is a mathematical object that can be used for software testing purposes where testing all possible configurations of a software program can be redundant and it might be possible to achieve an adequate simulation of all of the program's behavior with a much smaller number of carefully chosen configurations. The size of the smallest CA $\mathrm{CA}(t,k,v)$ is parameterized by t, v, and k, where t is strength of the CA and measures how "fully" we cover the possible variable interactions in the array, v is the number of possible values that each entry in the array can take, and k is the number of columns in the array. For our experiments we chose the CA(3,5,4) to consider $k = 5$ variables (metaheuristic, policy, trigger, physical network size and density), and $v = 4$ different values for each variable. A value of $t = 3$ means that we experimented with *all* the value combinations of *any* subset of three variables (metaheuristic, policy and trigger). Our intention for the above is to get non-exhaustive but representative evaluation of the behaviour of our migration approach. This way, we carried out experiments with 94 combinations of our five variables (metaheuristic, policy, trigger, physical network size and density), where a subset of combinations of metaheuristic, policy and triggers generated by the CA have been fully tested with all network sizes and all densities. For each of the 94 combinations, aligned to the rule of thumb of the Central Limit Theorem to obtain statistically valid comparisons, we executed 31 simulations for each combination. We also executed simulations of online VNE without migration to compare the performance of our migration approach. Fig. 1 to 4 show the average relative VNE acceptance ratio enhancement for each of the four Triggering Events (TE) proposed, for the four densities and the four network sizes considered in our experimentation. Each column in each figure is representative of the behaviour of the migration approach for the metaheuristics and policies considered in our evaluation, in the corresponding substrate network density and size. A first observation of our results is that TE3 is not an appropriate trigger to control the timing of our migration approach. The simple detection of highly utilized physical resources should not be considered as a triggering condition to drive the migration process proposed in this paper should an InP wants to enhance the VNE acceptance ratio.

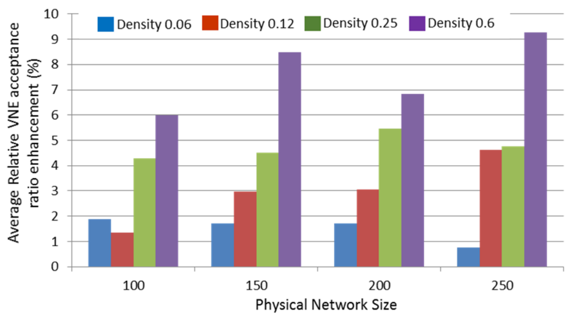

Figure 1. Average Relative VNE enhancement with TE1

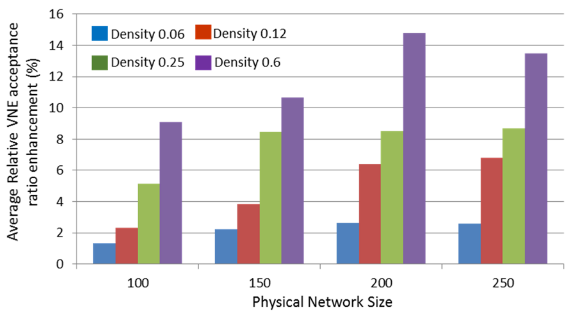

Figure 2. Average Relative VNE enhancement with TE2

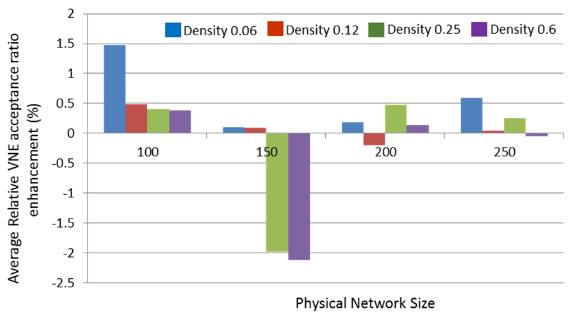

Figure 3. Average Relative VNE enhancement with TE3

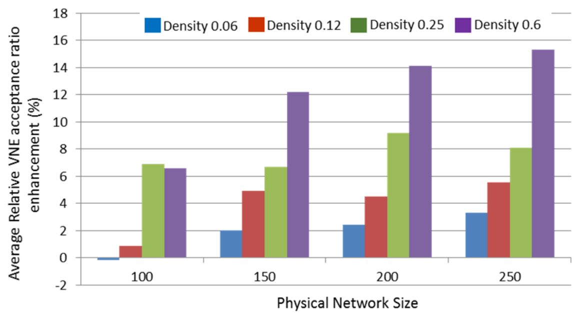

Figure 4. Average Relative VNE enhancement with TE4

Moreover, sensible enhancements were achieved with TE1, TE2, and TE4, specially for larger networks and higher densities. For density 0.6, the relative enhancement is higher than 6% in all TEs. For density 0.12, there is a clear pattern of enhancement increase with the substrate size in all TEs. For density 0.06, TE4 develops a pattern of small increases (up to 3%) with the increase of substrate size. In particular, the best relative enhancement was obtained with TE4, for which with 250 substrate nodes networked with density 0.6, achieved more than 15% relative online VNE acceptance ratio enhancement. In general, TE4 presents a clear pattern of relative VNE acceptance ratio enhancement, which is more accentuated for larger networks and higher densities. Results show that the proposed migration is effective for TE1, TE2, and TE4. To the best of our knowledge, this work is the first attempt that considers both additive and non-additive constraints in links and nodes in virtual network scenarios, considering a holistic migration approach controlled by specific triggers and policies. Our results demonstrate the effectiveness of our migration approach, specially for large networks.

IV. Conclusion

We have proposed an effective VN migration approach driven by the the virtual link migration problem. Migration policies have been proposed to define the type of physical resources that would host the to-be-migrated VLs considering additive (delay and hop count) and non additive (CPU and BW) QoS requirements. We have proposed and evaluate the effectiveness of our approach considering concrete migration events to trigger the migration process systematically. Our results demonstrate that, in general, our approach enhances VNE acceptance ratio with higher increases in large networks. Our future work will be directed to maximize the enhancements by means of optimal configuration parameters of the migration policies. We will also develop self-adjusting mechanisms for online VNE where the above triggers and policies can be adjusted dynamically as a function of QoS requirements, counter-offer elaboration and other important management aspects.

REFRENCES

[1] S. Lo, M. Ammar, and E. Zegura, ” Design and analysis of schedules for virtual network migration,” IFIP Networking Conference, pp. 1-12, May 2013.

[2] C. Aguilar-Fuster, “Evaluation of metaheuristics for online virtual network embedding in virtual network environments,” unpublished.

[3] R. Bradford , K. Evangelos, F. Anja , and S. Harald, "Live wide-area migration of virtual machines including local persistent state," in Proc. of the 3rd international conference on Virtual execution environments, pp. 169-179, ACM, 2007.

[4] X. Chang, X. MI, and J. Muppala, “Performance evaluation of artificial intelligence algorithms for virtual network embedding,” Engineering Applications of Artificial Intelligence, vol. 26, pp. 2540–2550, Novemebr 2013.

[5] N. M. K. Chowdhury, and R. Boutaba, “ Network virtualization: state of the art and research challenges,” IEEE Communications Magazine, vol. 47, pp. 20–26, July 2009.

[6] Ch. Clark, K. Fraser, S. Hand, J. Gorm Hansen, E. Jul, Ch. Limpach, I. Pratt, and A. Warfield,” Live migration of virtual machines,” NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vlo. 2, pp. 273-286, 2005.

[7] K. Durkota, “Implementation of a discrete firefly algorithm for the qap problem within the seage framework,” May 2011.

[8] I. Fajjari, N.A. Aitsaadi , G. Pujolle, and H. zimmermann, “vne-ac: virtual network embedding algorithm based on ant colony metaheuristic,” IEEE International Conference on Communications (ICC), pp 1-6, Jun 2011.

[9] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and , X. Hesselbach, “Virtual network embedding: A survey,” IEEE Communications Surveys & Tutorials, vol. 15, pp.1888-1906, 2013.

[10] Z. W. Geem, J. H Kim., and G. Loganathan, “A new heuristic optimization algorithm: harmony search simulation,” vol. 76, pp. 60-68, 2011.

[11] A. Hartman, “Software and hardware testing using combinatorial covering suites,” Graph Theory, Combinatorics and Algorithms, vol. 34, pp. 237-266, 2005.

[12] W. H. Hsu, Y. P. Shieh , C. H. Wang , and S. C. Yeh, “Virtual network mapping through path splitting and migration,” Advanced Information Networking and Applications Workshops (WAINA), pp. 1095-1100, March 2012.

[13] Y. Jiang, J. Lan, Z. Wang, and Y. Deng, “ Embedding and reconfiguration algorithms for service aggregation in network virtualization,” International Journal Of Communication Systems, vol. 29, pp. 33-46, May 2014.

[14] A. Khan, S. Herker, and X. An, “Survey on survivable virtual network embedding problem and solutions,” In International Conference on Networking and Services, pp. 99-104, March 2013.

[15] W. Qiang, F. Sheng, and L. Wu, “ Heuristic survivable virtual network embedding based on node migration and link remapping,” in Proc. ITAIC, Chongqing, pp. 181–185, 2014.

[16] A. A. Shahin, "Memetic multi-objective particle swarm optimization-based energy-aware virtual network embedding," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 6, April 2015.

[17] M. Yuy, Y. Yiz, J. Rexfordy, and M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration,” ACM SIGCOMM Computer Communication Review. vol. 38 , pp. 17-29, April 2008.

[18] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an Inter- network,” in Pro. IEEE INFOCOM, vol. 2, pp. 594-602, 1996.

[19] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, “A unified enhanced particle swarm optimization-based virtual network embedding algorithm,” International Journal Of Communication Systems, vol. 26, pp. 1054-1073, 2013.

[20] Y. Zhu and M. Ammar, “Algorithms for assigning substrate network resources to virtual network components,” in Proc. IEEE INFOCOM, Barcelona, pp. 1-12, April 2006.

[21] H. L. Zhu, “A virtual network reconfiguration algorithm to improve acceptance ratio,” in Proc. BWCCA, Krakow, pp. 602-604, 2015.

[22] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, ”Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive,” In Proc. ACM SIGCOMM, Seattle, WA, vol. 38, pp. 231-242, October 2008.

[23] T. Korkmaz, and M. Krunz, “Multi-constrained optimal path selec-tion,” in INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 834 – 843, 2011.