A dynamic resource defragmentation scheme for virtualized SDN-enabled substrate networks

Armel Francklin Simo Tegueu, Slim Abdellatif, Thierry Villemur

A dynamic resource defragmentation scheme for virtualized SDN-enabled substrate networks

A. F. Simo Tegueu*†, Slim Abdellatif*† and Thierry Villemur *‡
*CNRS, LAAS, 7 Avenue du colonel Roche, F-31400 Toulouse, France
†Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
‡Univ de Toulouse, UT2J, LAAS, F-31100 Toulouse, France

***Abstract*—Virtual network embedding (VNE) was subject to extensive research which lead to the emergence of a large number of efficient VNE algorithms. When virtual networks (VNs) arrive and depart over time, the substrate network can easily drift into an inefficient configuration, where resources are increasingly fragmented causing a VN request rejection although cumulatively, there are enough available resources. The ability to reallocate running VNs clearly leads to a better resource utilization. In this paper, we propose "Garbage Collector"(GC), a novel network control program for dynamic and online resource management in virtualized SDN-enabled substrates. GC efficiently addresses the fragmentation problem by performing selective migration. Simulations show that GC clearly improves acceptance ratio of VNE algorithms. They also reveal that, it outperforms some existing works from the literature by increasing the VN acceptance ratio by more than** 10%**.**

I. Introduction

Network virtualization enables the co-existence of multiple concurrent VNs over the same substrate network (SN) in an independent and isolated. It relies on algorithms commonly known as "Virtual Network Embedding" algorithms to compute the substrate network resources that support each VN.

When VNs arrive and leave the infrastructure over time, the SN can easily drift into an inefficient configuration. Although cumulatively, enough resources are available, new VN requests may be rejected because these resources are too fragmented. The ability to reallocate running VNs allows to enhance resource utilization. This is why a defragmentation mechanism is usually used to complement one-line VNE algorithms to proactively or reactively trigger some VLs reallocations. Their objective is to evenly spread the load leading to a reduction of network resource fragmentation and, as a consequence, an improved admissibility for forthcoming VNs requests.

However, migrating reallocated VNs is non-trivial. It involves numerous operations to instantiate virtual nodes and redeploy their connecting VLs. Realizing these operations at hand is a time-consuming task as well as error prone. The emerging SDN paradigm has been recognized as a key solution to overcome these problems by enabling dynamic and automated configurations for a fast and reliable deployment. But, its use has also introduced some new constraints, namely, the limited capacity of forwarding tables, which is actually around a few thousands of entries [1] [2] in commodity SDN-compliant switches. These switching resources are not only requested by virtual nodes, but also required to embed VLs. In fact, a number of flow rules have to be installed on auxiliary nodes i.e., nodes that are not part of VN request, but are part of the physical paths that host the VLs.

In this paper, we propose a new dynamic proactive resource defragmentation scheme based on path migration, to address the fragmentation problem inherent to online virtual links mapping in virtualized SDN-enabled infrastructures. It relies on a proactive selection of the VLs to migrate, which is triggered on a VN departure, and in presence of congested entities, when the network is in a fragmented state. The VLs selection is loosen the congested substrate entities while limiting the number of VLs that can be profitably migrated. For increased defragmentation efficiency, the remapping of all selected VLs can be jointly performed.

The rest of this paper is organized as follows. Section II discusses existing SN resource management approaches. In Section III, we describe and formulate the problem of defragmentation, before introducing our proposed solution in Section IV. Section V presents the evaluation and discusses the obtained results. Last, Section VI concludes this paper.

II. Related work

Commonly, resource defragmentation algorithms based on VN reconfiguration are decomposed into three successive phases that we refer as: controlling reconfiguration, selecting candidate virtual components and remapping.

1) Controlling reconfiguration: As it is not conceivable to apply reconfiguration continuously, the objective is to determine when to trigger the process such as reconfiguration is effective. We distinguish two categories of approaches according to the trigger mode: periodic [3] [4] and event-based. This last category is decomposed into three subcategories: (1) reactively triggered on VN request rejection; (2) proactively [2] [5] triggered by events like VN departure and (3) hybrid which can be triggered when a VN request embedding fails or/and on a VN expiration. Some solutions [3][6][5] condition the reconfiguration by the level of congestion beyond which the reconfiguration happens.

2) Selecting candidate virtual components: During this phase, critical virtual components (logical group of virtual links and/or nodes) potentially reconfigurable are selected. The number of virtual components being limited in order to reduce computational costs and reconfiguring duration, they should be relevantly chosen to achieve a better result. The selection of

each virtual component depends on some factors like, short-terms re-optimization objectives, substrate resources considered as fragmented and virtual entities consuming each type of these resources. It is often taken into account via a criterion (or criteria) that quantifies the impact of virtual component and the potential contribution of its migration for reducing the fragmentation.

3) Remapping: At this stage, the previously selected virtual components are reallocated. We distinguish criteria to classify remapping methods. The first one includes (1) integrated approaches [7] [8] that execute the selection and the remapping in a coordinated manner simultaneously; and (2) separated approaches where the selection and the remapping are performed independently. The second criterion includes interrelated approaches [9], which requires specific information from the initial embedding strategy, unlike the approaches that are able to operate with any initial embedding strategy. Finally, the last criteria allows to distinguish the approaches that are restrained to reallocate the selected virtual components individually (i.e sequentially), from those able to treat them simultaneously for a further efficient resource utilization.

The defragmentation scheme that we are proposing is a proactive in the extent that it is not triggered on a VN rejection, but typically on a VN departure. More precisely, to avoid frequent and useless reconfiguration, VN reconfiguration attempt is conditioned by the congestion level and the fragmentation level of substrates links and nodes. On a reconfiguration attempt, not all running VLs are remapped, neither all VLs belonging to a same VN. Instead, a subset of VLs that can be profitably migrated are selected. Our proposal is not related to any specific VNE algorithm. It can complement any suitable VNE algorithm requiring no modification and no specific information. Our approach is hence able to take advantage of existing VL mapping strategies by remapping all selected VLs jointly for a better resource allocation.

III. Problem description and formulation

In this study, we consider a scenario, in which a SN with fixed topology and capacity of links and nodes. The VNs consisting of a set of VLs come and leave the infrastructure dynamically. Their required resources are static during their lifetime. They are treated in sequence by an initial on-line embedding algorithm, with no information on future requests. We assume that its ultimate goal is to improve the acceptance ratio of VN requests. Only the lack of available resources may cause request rejection. A VN request is considered accepted, only if all VLs composing it, are successfully mapped onto the SN. We assume that all currently embedded VNs can be dis-embedded and re-embedded.

A. Network model

The SDN substrate network is modeled as a bidirectional graph $G = (N, L)$ where N is the set of SDN nodes and $L(L \subseteq N \times N)$ the set of physical links which operate in full-duplex mode. To each node $i \in N$, is associated a switching capacity U_i, which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node i flow table is denoted by U'_i. Each link $(i, j), i, j \in N$ is weighted by its bandwidth B_{ij}. Links are assumed to have the same characteristics in both directions, i.e. $B_{ij} = B_{ji}$. The bandwidth that is currently assigned at link (i, j) by already admitted virtual links is denoted by B'_{ij}.

B. Virtual network requests model

A VN request is composed of a set of K VLs. Each VL k is characterized by: a source node $s_k \in N$, and a set of destination nodes $T_k \in N \backslash \{s_k\}$ (when $|T_k| = 1$, the VL is point-to-point, otherwise it is point-to-multipoint); a bandwidth requirement of b_k.

C. Virtual link mapping model

The initial embedding algorithm maps each VL k to a substrate path denoted as F_k, which consists of a set of physical links and nodes. $f_k(i, j)$ refers to the amount of bandwidth used on link $(i, j) \in F_k$ by the VL k. Likewise, we denote by $l_k(i)$ the number of entries that are allocated at node $i \in F_k$ flow table to support VL k with the assumption that all entries consume the same amount of resources regardless of the complexity of the match operation and the related instructions to perform.

D. Resource defragmentation objectives

As stated in section III, the long-term objective is to enhance the embedding performance in terms of VN acceptance ratio. To this end, at each trigger of the defragmentation mechanism, the objective is to:

- minimize the number of congested substrate links and nodes while
- reducing the resources spent to map existing VLs, also known as embedding cost.

Unfortunately, the excessive remapping of running VLs can cause network instability and can also induce significant computational costs, as well as bandwidth overhead due to the rerouting rules which are sent from the controller to the nodes. Consequently, our last objective is to restrict the whole number of VLs that will be migrated.

IV. Proposed solution

In this section, we present our proposal called "GC" (Algorithm 1), which is based on a heuristic approach. Acting as an event-condition-action engine, GC is proactively triggered when a VN leaves the infrastructure. It is structured into three successive phases. The first one denoted *Controlling reconfiguration* aims at determining at which conditions substrate resources are considered as fragmented and if, there is an urgent need for carrying out migrations. The next one, called *Selecting virtual links* aims at determining which VLs can profitably be migrated. And the last phase *Remapping* during which new mappings (hosting substrate paths and related assigned resources) are calculated, to reroute selected VLs.

Algorithm 1: Garbage Collector algorithm

Input : $G(N,L); X$ set of running Virtual Links; $F_k \; \forall k \in X$; θ; $N_{max}; \tau$

1 **begin**
2 Initialize $\aleph_\theta \longleftarrow$ set of substrate links $(i,j) : s(i,j) \geq \theta \cup$ set of substrate nodes i : $s(i) \geq \theta$; I_N and I_L; $Success \longleftarrow false; \zeta \longleftarrow \emptyset$
3 **if** $((\aleph_\theta \neq \emptyset)$ **and** $(I_N > \tau$ **or** $I_L > \tau))$ **then**
4 $\zeta \longleftarrow$ `Select VLs`$(G(N,L), X, F_k, \theta, N_{max})$
5 Revert currently assigned allocations to ζ
6 $Success \longleftarrow$ `Reallocate`$(\zeta, G(N,L))$
7 **if** *(Success)* **then**
8 Commit new assigned allocations to ζ
9 **else** Rollback currently assigned allocations to ζ
10

Algorithm 2: Select VLs

Input : $G(N,L); X$ set of running Virtual Links; $F_k \; \forall k \in X; \theta; N_{max}$
Output: ζ set of selected virtual links

1 **begin**
2 Initialize $\zeta \longleftarrow \emptyset; \Gamma \longleftarrow \emptyset; C \longleftarrow \emptyset; n_{max} \longleftarrow 0$
3 **repeat**
4 $C \longleftarrow X \backslash \zeta$; $n_{max} \longleftarrow N_{max} - |\zeta|$
5 $\Gamma \longleftarrow$ `MSS-MRu`$(G(N,L), C, F_k, \theta, n_{max})$
6 $\zeta \longleftarrow \zeta \cup \Gamma$
7 **until** $(|\zeta| = N_{max}$ **or** $\Gamma = \emptyset)$

A. Controlling reconfiguration

Unlike some approaches that systematically reconfigure when an event occurs, we propose (Algorithm 1 : line 3) to jointly combine two conditions that must be satisfied to launch a migration attempt: one on the congestion level of the substrate resources, and the other on their fragmentation level.

1) Detecting congestion: The congestion of only one substrate entity, may cause a rejection of a complete VN request or force mapping new VLs over more resource consuming (longer) paths. Hence, congestion is a fundamental situation that must be considered. On the other hand, it is important to spare some SN entity namely those with a central position in the SN topology that are likely to be frequently solicited. So, a SN entity is considered to be θ-congested if its stress s is greater than θ. Formally,

$$s(i) = w(i) * \frac{\sum_k (l_k(i))}{U_i} \geq \theta \tag{1}$$

$$s(i,j) = w(i,j) * \frac{\sum_k (f_k(i,j))}{B_{ij}} \geq \theta \tag{2}$$

where $s(i)$ and $s(i,j)$ represent respectively the stress of physical node i and link (i,j). $w(i)$ and $w(i,j)$ are the normalized node and link importance calculated offline based on notions such as centrality, communicability and betweeness [10]. θ is the congestion threshold from which an SN entity is considered congested.

2) Detecting fragmentation: As the resource fragmentation is due to the departure of a VN which releases resources, the congestion may occur even if the network is in a clean state (for example when requests come and leave following LIFO (Last In First Out)). The resource fragmentation level allows to detect such situation (not only), hence avoiding the useless triggering reconfiguration. The level of SN resources fragmentation is measured by two indexes formally,

$$I_N = \frac{Max(s(i),\ \forall i \in N)}{\frac{\sum_{i \in N}(s(i))}{|N|}} \geq \tau \tag{3}$$

$$I_L = \frac{Max(s(i,j),\ \forall (i,j) \in L)}{\frac{\sum_{(i,j) \in L}(s(i,j))}{|L|}} \geq \tau \tag{4}$$

Where $I_N (1 \leq I_N \leq |N|)$ and $I_L (1 \leq I_L \leq |L|)$ represent respectively the fragmentation index of physical nodes and links. The more the indexes are close to 1 the more allocations are fairly distributed on the SN. The parameter τ is the threshold from which fragmentation is detected.

B. Selecting virtual links

Unlike some approaches proposing to select all VLs of the VNs that are mapped over congested entities, we propose to limit the number of VLs to migrate to N_{max}. For example, it may be fixed to the number of VLs composing the departing VN. Algorithm 2 shows how the selection is performed. This iterative algorithm selects at each iteration (line $3-7$) a set of n_{max} (n_{max} ensures that the total number of selected VLs will not exceed N_{max}) VLs noted Γ, among not yet selected ones noted C. The algorithm iterates until the number of selected VLs reaches N_{max} or stops when there is no more potentially suitable VL in C. We rely on a primitive function noted MSS-MRu for "Minimum Spanned Set with Maximum Resource utilization", in order to select a subset of VLs at each iteration.

1) Minimum Spanning Set (MSS) with Maximum Resource utilization: We define a Spanning Set as a set of VLs denoted Γ, such as each θ-congested substrate nodes and links from the set denoted $\aleph_\theta$, hosts at least one VL of Γ. Selecting a spanning set of VLs allows to consider all θ-congested entities. A Minimum Spanning Set is a Spanning Set with the minimum number of VLs. A Minimum Spanning Set with Maximum Resource utilization is a MSS such that the sum of assigned resources is maximum. It typically includes, the most consuming VLs, i.e those mapped on a longer substrate path or over multiple substrate paths.

Initially, referring to Algorithm 3 : line 2, the set of selected VLs denoted Γ is empty. The variable $\aleph_\theta$ refers to the set of θ-congested substrate links, as well as nodes (The load of each node is calculated without considering, the load of the

Algorithm 3: Minimum Spanning Set with Maximum Resource utilization (MSS-MRu)

Input : $G(N, L); C$ set of candidate Virtual Links; $F_k \; \forall k \in C; \theta; n_{max}$
Output: Γ set of spanning virtual links

1 **begin**
2 Initialize $\Gamma \longleftarrow \emptyset; \aleph_\theta \longleftarrow$ {Set of congested substrate links and nodes (the load of each node is calculated without considering, the load of the VLs for which it is an end node) }; $\overline{\aleph}_\theta \longleftarrow \emptyset; \forall k \in C$ calculate $A_k, \overline{A}_k, R_k$; a priority queue $PQ \longleftarrow \emptyset$
3 **repeat**
4 $PQ \longleftarrow \emptyset$
5 **foreach** $k \in C \backslash \Gamma$ **do**
6 $PQ.enqueue(k)$ using $Comparator$
7 $k \longleftarrow PQ.dequeue()$
8 $\Gamma \longleftarrow \Gamma \cup \{k\}$
9 $\overline{\aleph}_\theta \longleftarrow \overline{\aleph}_\theta \cup (\aleph_\theta \cap (F_k \backslash \{\{s_k\} \cup T_k\}))$
10 $\forall k \in C \backslash \Gamma$ Update $\overline{A}_k$
11 **until** $((|\Gamma| = n_{max})$ **or** $(|\overline{\aleph}_\theta| = |\aleph_\theta|))$

VLs for which it is end node). The variable $\overline{\aleph}_\theta$ refers to the set of spanned θ-congested entities. For each VL k in C, the algorithm maintains three metrics that are used to evaluate the suitability of migrating the VL.

- We define the impact of a VL on a given set, as the number of elements of its substrate paths belonging to this set except its source and destination nodes. We distinguish two variants of the impact: impact of a VL k on $\aleph_\theta$ noted A_k and its impact on $\aleph_\theta \backslash \overline{\aleph}_\theta$ noted $\overline{A}_k$. Formally,

$$A_k = |\aleph_\theta \cap \{F_k \backslash \{\{s_k\} \cup T_k\}\}| \quad (5)$$

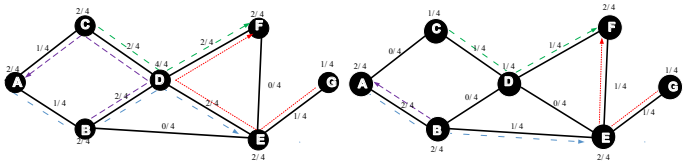
$$\overline{A}_k = \left|(\aleph_\theta \backslash \overline{\aleph}_\theta) \cap \{F_k \backslash \{\{s_k\} \cup T_k\}\}\right| \quad (6)$$

- We also define resource utilization R_k of a VL k as total resources assigned to VL. It reflects how much the VL affects overall resource utilization on congested substrate links and nodes. It is calculated as follows:

$$R_k = \alpha * \sum_{(i,j) \in F_k} (f_k(i,j)) + \beta * \sum_{i \in F_k} l_k(i) \quad (7)$$

It is important to note that, this formulation considers both, link bandwidth resource, as well as switching resources in term of flow table entries. Parameters α and β allows to configure the relative importance of each type of resource.

We also introduce a priority queue PQ to sort VLs based on a comparison logic called $Comparator$ which works as follows: a VL $k1$ is more suitable (of higher priority) than another one $k2$ if $\overline{A}_{k1} > \overline{A}_{k2}$; else if $\overline{A}_{k1} = \overline{A}_{k2}$, then $k1$ has priority if $A_{k1} > A_{k2}$; else if $A_{k1} = A_{k2}$ then $k1$ has priority if $R_{k1} > R_{k2}$. Otherwise one will be arbitrary designated as having priority. The algorithm iterates (algorithm 3: line 3 - 11) until the number of selected VLs reaches n_{max}, or stops when all congested entities are spanned. At each iteration, the PQ is refreshed, and the suitable VL is selected. The congested entities hosting selected VL are marked as spanned except for its source and destination nodes, and the impact of the rest $(C \backslash \Gamma)$ of VLs on not yet spanned entities, is updated.



(a) Network state before (b) Network state after

Fig. 1. Virtual Links Selection and Migration example

TABLE I
MSS-MRu IN ACTION

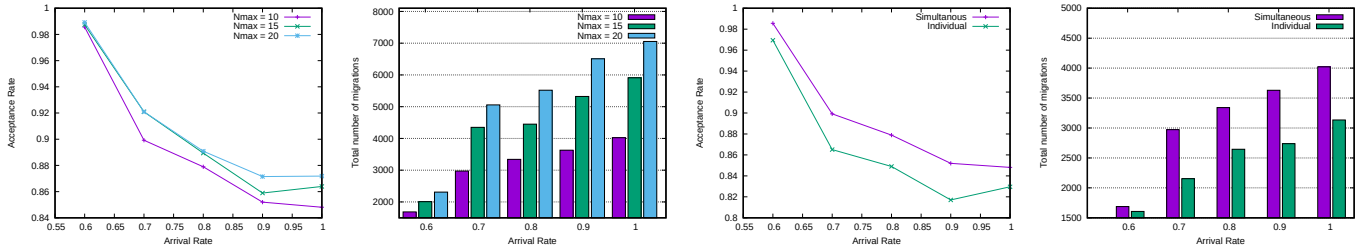
Iteration	Γ	$\aleph$	$\overline{\aleph}$	C
Initialization	$\emptyset$	$(B,D),(D,E)$ $(C,D),(D,F)$ C,D,B,E	$\emptyset$	$k_1(4,4,7), k_2(4,4,7)$ $k_3(3,3,5), k_4(4,4,7)$
1	k_1	~~(B,D)~~, (D,E) ~~(C,D)~~, (D,F) ~~C~~, D, ~~B~~, E	$(B,D),(D,E)$ D,B	~~$k_1(4,4,7)$~~, $k_2(2,4,7)$ $k_3(2,3,5), k_4(2,4,7)$
2	k_1, k_2	~~(B,D)~~, ~~(D,E)~~ $(C,D),(D,F)$ ~~C~~, ~~D~~, ~~B~~, E	$(B,D),(D,E)$ (C,D) D,B,C	~~$k_1(4,4,7)$~~, ~~$k_2(2,4,7)$~~ $k_3(1,3,5), k_4(2,4,7)$
3	k_1, k_2, k_4	~~(B,D)~~, ~~(D,E)~~ ~~(C,D)~~, (D,F) ~~C~~, ~~D~~, ~~B~~, ~~E~~	$(B,D),(D,E)$ $(C,D),(D,F)$ D,B,C,E	~~$k_1(4,4,7)$~~, ~~$k_2(2,4,7)$~~ $k_3(1,3,5), k_4(2,4,7)$

Legend: $(A \longrightarrow E) = k_1; (B \longrightarrow A) = k_2; (C \longrightarrow F) = k_3; (G \longrightarrow F) = k_4$
$k(x,y,z) = k(\overline{A}_k, A_k, R_k)$

2) Application: Let's consider a SN $G(N, L)$ with 4 VLs $((A \longrightarrow E), (B \longrightarrow A), (C \longrightarrow F), (G \longrightarrow F))$. When activating GC, the MSS-MRu algorithm determines which VLs can profitably be migrated. Assume that each VL requests 1 unit of bandwidth and requires 1 flow table entry at each node. Also, the congestion threshold θ is set to 0.5. Given these considerations, there are 10 (6 nodes and 4 links) congested entities and 26 (11 units of bandwidth and 15 units of flow table entry) units of resource are currently consumed. When MSS-MRu algorithm is executed with $n_{max} = 3$, the VLs $(A \longrightarrow E), (B \longrightarrow A), (G \longrightarrow F)$ are selected and an example of their remapping is shown in Figure 1b. This solution leads to a better network state with 5 (4 nodes and 1 link) congested entities , and 18 (11 units of bandwidth and 7 flow table entries) units of resource consumed. Table I details step by step the behavior of the MSS-MRu algorithm.

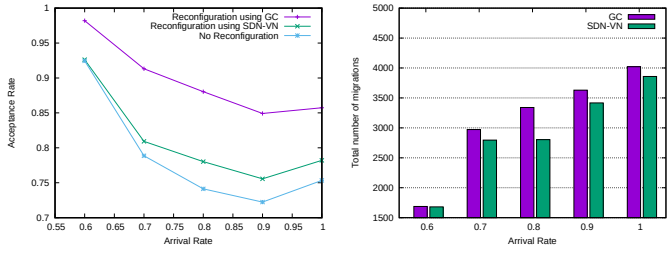
C. Remapping

The last step (Algorithm 1 : line 6) of our strategy consists in finding new substrate paths to reroute each previously selected virtual links. The performances of our solution, rely not only on the efficiency of the first two phases, but also on the efficient remapping algorithm. This latter must be chosen in adequacy with the targeted objectives of the defragmentation algorithm in order to avoid counterproductive effects. In our



(a) Impact of N_{max} - A. R. (b) Impact of N_{max} - M. C. (c) Sim. vs ind. mapping - A. R. (d) Sim. vs ind. mapping - M. C.

Fig. 2. Impact of parameters ($\theta = 0.9, \tau = 2$)



(a) Comparison - Acceptance Rate (b) Comparison - Migration Costs

Fig. 3. Comparisons ($N_{max} = 10, \theta = 0.9, \tau = 2$)

case, the consideration of link bandwidth and switching resources and the objective of allocating resources efficiently and fairly. These objectives are usually considered in many VNE algorithms amongst [11] [12] [13].

V. Performance Evaluation

We firstly introduce the simulation settings, before presenting our main results.

A. Simulation settings

Our algorithm is applied to a real network topology with different randomly generated VLs requests. The considered experimental set up is described hereafter.

1) Network model: We consider in this work a real network topology taken from the European Research Network GEANT with 41 network nodes and 60 links that connect the main European cities. Like [2], we assume that a flow table size of 125 entries (The remainder being reserved to the forwarding table of the virtual nodes) is dedicated at each node to VLs resource allocation. We consider that each flow rule requires 1 entry in flow table of traversed nodes.

2) Load model: We assume that requests arrive following a Poisson distribution with an arrival rate λ that is varied on $\{0.6, 0.7, 0.8, 0.9, 1\}$ i.e. the average number of requests varies from 60 to 100 requests per 100 units of time (UT) respectively. The requests lifetime conforms to an exponential distribution with an average of 125UT. The number of VLs per request is set according to a discrete uniform distribution, using the values given in $[6, 12]$. The bandwidth requirement is uniformly distributed between 100 and 300Mbps.

3) Algorithms settings: The embedding algorithm to optimally map VLs onto the SN is based on an ILP formulation taken from [11]. It is used for both: mapping in-coming VNs and remapping selected VLs. The Integer Linear model was implemented in C++ with CPLEX-12.06 solver [1]. The resolution time is set to a maximum of 15 seconds. A gap of less than 5% to the optimal solution is considered satisfactory. The simulation horizon is fixed to 1500UT. The parameters of GC are set as follows: $\tau = 2, \theta = 0.90, N_{max}$ is varied on $\{10, 15, 20\}$, $\alpha = 1$ and $\beta = 200$. These two latter parameters are calibrated to scale switching and bandwidth resources to the same magnitude. The remapping is performed in both modes, simultaneous (all selected VLs mapped jointly) and individual (VLs mapped one by one). GC will be compared to another efficient solution from the literature, called "SDN-VN" [2] that is a proactive approach also considering switching resources.

B. Performance metrics

- Acceptance Rate (A.R): the percentage of successful virtual links requests out of all the requests that arrived during the simulation time.
- Migration Cost (M.C): the total number of migrations occurred during the whole simulation.
- Maximum instantaneous link/node utilization: the greater percentage of assigned bandwidth/flow table entries at a given link/node, computed at a time instant t, i.e. $Max(\frac{B'_{ij}}{B_{ij}}\ \forall (i,j) \in L)$ for links and $Max(\frac{U'_i}{U_i}\ \forall i \in N)$ for nodes.

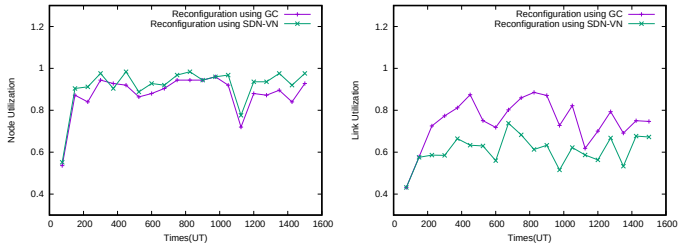
C. Evaluation results and analysis

1) Effect of N_{max}: We first examine the influence of the maximum number of VLs to migrate at each trigger of GC. Figures 2a and 2b show the performances of GC in terms of acceptance ratio and migration cost as a function of the arrival rate (λ). We evidently observe that, the acceptance ratio is improved when N_{max} increases. The reason is that, remapping more VLs gives more chance to unload congested nodes and links, which may cause requests rejection. We also observe that, at the high offered load, more gain can be achieved. This is because, at the low load, even though the available

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

resources are fragmented, they are sufficiently abundant to successfully process incoming requests, by migrating just a few VLs. It is also clear that, under excessively high load, the SN will be continually saturated, and VLs requests will be rejected independently of the value of N_{max}. Furthermore, while increasing N_{max}, improves the requests admissibility, it also leads to more reconfigurations which increase the migration cost as shown in Figure 2b.

2) Impacts of simultaneous vs individual remapping: Another important aspect of our approach is its flexibility to simultaneously or individually remap selected VLs. Figures 2c and 2d show that the simultaneous allocation is more efficient. It decreases the final rejection ratio compared to the case where each VL is reallocated in sequence. Since the remapping algorithm has information about all selected VLs and, it can achieve in one-shot, a better substrate resource arrangement. Also, reallocating each VL individually, is less efficient than when all selected VLs are dis-embedded to be reallocated simultaneously. This is why, as presented in Figure 2d, the migration cost is higher when VLs remapping is performed simultaneously than individually.



(a) max node utilization (b) max link utilization

Fig. 4. Instantaneous view ($\lambda = 0.8, N_{max} = 10, \theta = 0.9, \tau = 2$)

3) Comparative analysis : The final experiments show the gain of our proposal, compared to the ILP based VLs mapping with no reconfiguration and with reconfiguration using SDN-VN. As expected, the results from Figure 3a show that, when VLs embedding strategy incorporates a defragmentation mechanism, the acceptance ratio is improved. The admittance gain is on average about 10%. This is due to dynamic resources cleaning. Moreover, the results also show that GC significantly increases the acceptance rate in comparison to legacy SDN-VN. In fact, Figure 4 depicts the maximum instantaneous link and node utilization when $\lambda = 0.8$. We have observed that, GC is triggered at instant 163 when the first VN expiration occurs in presence of 0.90-congested nodes. GC takes advantage by minimizing maximum node utilization. Furthermore, more bandwidth are consumed, but, while maintaining maximum link utilization less than 0.9. Another key observation from Figures 2c and 3a, regarding acceptance reveals that, GC also outperforms SDN-VN even when VLs are individually remapped. We think that, one of the main reasons is the fact of removing source and destination nodes from "VL impact" (equations 6). This allows to prioritize VLs whose traversed intermediate nodes are congested. This increases chance to move a selected VL from its current substrate path to another one more profitable. Unfortunately, Figure 3b reveals that GC reconfigures more VLs than SDN-VN. This can be justified not only by the higher acceptance rate that causes bottlenecks, but also more departures.

VI. Conclusion

This paper has proposed and evaluated a new proactive solution called Garbage Collector or GC to deal with SN resources fragmentation in virtualized SDN environments. Our first contribution is a VLs selection algorithm, which avoids remapping all existing VLs, but only the most promising running VLs. Also, our solution does not impose to reallocate VLs individually, but precludes the remapping of selected VLs simultaneously.

Our method was evaluated on a real network topology under commonly used load model. The simulations show that the acceptance ratio increases with the maximum number of VLs allowed to be migrated each time that GC is triggered. We also have shown that our proposal improves the efficiency of the mapping algorithm and outperforms an existing solution from the literature.

Acknowledgment

This work was partially funded by the French National Research Agency (ANR) and the French Defense Agency (DGA) under the project ANR DGA ADN (ANR-13-ASTR-0024). and by European Unions Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960)

References

[1] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[2] R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. De Turck, and S. Latr, "Dynamic resource management in sdn-based virtualized networks," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 412–417.

[3] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–12.

[4] 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, no. 2, pp. 17–29, Mar. 2008.

[5] B. Wanis, N. Samaan, and A. Karmouch, "Substrate network house cleaning via live virtual network migration," in *2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 2256–2261.

[6] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vnr algorithm: A greedy approach for virtual networks reconfigurations," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011, pp. 1–6.

[7] P. N. Tran, L. Casucci, and A. Timm-Giel, "Optimal mapping of virtual networks considering reactive reconfiguration," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, Nov 2012, pp. 35–40.

[8] T. P. N. and T.-G. A., "Reconfiguration of virtual network mapping considering service disruption," in *2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 3487–3492.

[9] N. Farooq Butt, M. Chowdhury, and R. Boutaba, *Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 27–39.

[10] E. Estrada and D. J. Higham, "Network properties revealed through matrix functions," *SIAM Review*, vol. 52, no. 4, pp. 696–714, jan 2010.
[11] F. S. Tegueu, S. Abdellatif, T. Villemur, P. Berthou, and T. Plesse, "Towards application driven networking," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, June 2016, pp. 1–6.
[12] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of vne: A survey," *China Communications*, vol. 13, no. 6, pp. 48–62, June 2016.
[13] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.