Application-Aware Adaptive Provisioning in Virtualized Networks

Rafael Pereira Esteves, Lisandro Zambenedetti Granville
Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Brazil

***Abstract*—Network virtualization is a feasible solution to tackle the so-called Internet ossification by enabling multiple virtual networks (VNs) running simultaneously on top of a shared physical infrastructure. Network management with virtualization support, however, poses challenges that need to be addressed in order to fully achieve an effective and reliable networking environment. One of the main aspects related to the management of network virtualization environments is virtual network provisioning. Unfortunately, current provisioning solutions focus on a single or a limited set of objectives that may not simultaneously match the requirements of an increasing number of applications deployed in networks everyday.**

In this thesis, we formulate the Application-Aware Virtual Network Provisioning Problem (AVNP) and propose an adaptive provisioning framework for virtualized networks that takes into consideration the characteristics of multiple applications and their distinct performance objectives. The proposed framework is based on the concept of *allocation paradigm*, which is defined as a set of provisioning policies that guide the resource allocation process. A paradigm translates objectives from both Infrastructure Providers (InPs) and Service Providers (SPs) to individual allocation actions that actually provision VNs. To determine the efficiency of a particular paradigm, we propose a virtual network performance computation model to quantify the performance of allocated VNs and guide paradigm changing decisions. Simulation results show the feasibility of allocation paradigms in helping network providers to select the best provisioning strategy given a set of InP/SP objectives.

I. Introduction

Network virtualization has been considered a viable alternative to guide the development of new network architectures and to overcome the Internet ossification [1] [2] [3]. In a network virtualization environment (NVE) the underlying physical infrastructure, commonly referred to as substrate, is shared among different users who create customized virtual networks (VNs). Among the technical challenges to enable NVEs, management has a special importance. Management of NVEs is crucial to guarantee the proper operation of the physical infrastructure, of the hosted VNs, and of the services supported by the VNs. Management in NVEs can be classified in provisioning, monitoring, and interfacing [4]. Provisioning consists of allocating VNs to SPs by defining the mapping of VN resources to the physical counterparts. Monitoring involves gathering updated status of physical resources and their associated VNs. Interfacing is required for InPs and SPs to respectively access, operate, maintain, and administer the physical and virtual nodes and links.

Typically, in an NVE, an infrastructure provider (InP) offers virtual network resources to multiple service providers (SPs) that deploy a variety of applications on top of virtual networks (VNs). The provisioning of VNs must consider requirements of both InPs and SPs. While the main objective of InPs is to generate revenue by accommodating a large number of VNs, SPs, on the other hand, have specific needs, such as guaranteed bandwidth among virtual machines, load balancing, and high availability. Inefficiencies in the provisioning process can lead to disastrous consequences for infrastructure providers including reduced number of SPs, monetary penalties (*e.g.*, financial credit) when Service Level Agreements (SLAs) are not satisfied, and low utilization of the physical infrastructure.

Current NVE provisioning systems allow SPs to request different resource configurations (*e.g.*, CPU, memory, disk) to build a VN. The SP is the main responsible for requesting resources that will better fit its applications. The InP then either allocates resources for the VN in the physical network or rejects the allocation if there are not enough resources to satisfy the SP's request. InPs run allocation algorithms to find the best mapping of VNs onto the physical substrate according to well-defined objectives, such as minimizing the allocation cost, reducing energy consumption, or maximizing the residual capacity of the infrastructure. Mapping virtual to physical resources is commonly referred to as *embedding* and has been extensively studied in recent years [5] [6] [7].

Despite the existence of a variety of VN embedding strategies, there are common limitations shared by most approaches. First, current embedding strategies are static. InPs cannot switch from an embedding strategy to another to satisfy a particular objective, such as energy efficiency or load balancing, nor they can support specific application requirements, such as fault-tolerance or exclusivity over resources. Second, current VN provisioning approaches consider VN embedding as an atomic operation, *i.e.*, the complete mapping of virtual to physical resources is defined prior to VN deployment, which in some cases may not capture the dynamics of the substrate. For example, when a VN expires, it releases resources that could be used by an ongoing VN request because they are better options towards a given objective (*e.g.*, reduce communication cost). In current embedding approaches, if the InP wants to optimize resource allocation and leverage the existence of better mapping alternatives, it has to migrate already deployed VNs to new locations, which increases operational costs.

In this work, we address the problem of provisioning VNs considering multiple (possibly conflicting) InP and SP objectives to define how virtual resources are mapped in the substrate. To enable such flexible resource allocation,

978-3-901882-76-0 @2015 IFIP

we propose an architecture of a provisioning system for virtualized networks that allows InP and SPs to express high-level requirements for the requested VNs, which ultimately influence how VNs should be allocated in the physical substrate. The proposed provisioning approach is based on the concept of allocation *paradigms*. A paradigm encompasses goals associated with the high-level objectives of the InP and of the SPs. Each goal is realized by allocation actions executed within a window (one per goal), which is defined in real-time according to the current status of the substrate. This approach allows rapid adaptation of the provisioning process to the dynamics of the substrate and to the characteristics of the deployed applications.

The rest of this paper is organized as follows. Sections II and III describe the concept of allocation paradigms and a methodology to determine the efficiency of allocation paradigms, respectively. Section IV presents representative results from our evaluation of the proposed provisioning approach. We then conclude in Section V.

II. Application-Aware Virtual Network Provisioning using Allocation Paradigms

In this section, we begin by defining the basic concepts and the formal problem formulation of our paradigm-based adaptive provisioning approach. Next, we then describe a policy language used to describe allocation paradigms.

A. Basic concepts

A paradigm defines how VNs are allocated in the physical substrate. Each paradigm comprises a set of goals associated with application requirements. A goal is derived in actions, which, in turn, allocate individual VN resources. The main concepts of the paradigm-based provisioning approach are described below.

- **Paradigm:** a paradigm $\mathcal{P}$ represents a group of goals $(G_1, G_2, ..., G_n)$ that are considered in the provisioning of VNs;
- **Goal:** a goal G is a single high-level objective that is defined by the InP or requested by the SP in the provisioning of VNs;
- **Action:** an action A is a single provisioning operation executed to achieve a goal G.
- **Window:** a window W is a set of allocation Actions $(A_1, A_2, ..., A_n)$ executed sequentially according to a goal G. A paradigm $\mathcal{P}$ is thus realized by a set of windows $(W_1, W_2, ..., W_n)$ associated with the objectives $(G_1, G_2, ..., G_n)$ of the paradigm;
- **Allocator:** an allocator $Alloc$ executes provisioning of VNs through an allocation window W defined by a goal G. There is one allocator $Alloc$ entity associated with each goal G of a paradigm $\mathcal{P}$.

When provisioning VNs, each goal G is translated into a list of actions $(A_1, A_2, ..., A_n)$ that will be executed sequentially within a window W. Several goals can be combined together in the provisioning of a VN. The choice for specific allocation actions depends on the InP objectives and on the characteristics of the applications to be deployed over the requested VN, on the active paradigm $\mathcal{P}$, and on the current status of the physical substrate (*e.g.*, available servers/links). Unlike current multi-objective VN provisioning proposals, allocation paradigms allow InP operators to modify the VN allocation "on-the-fly" by using different goals for each request. This flexibility is important to make VN provisioning systems adaptable to a large diversity of VNs, where each one may run different applications. Figure 1 depicts the relationship between paradigms, goals, and actions.

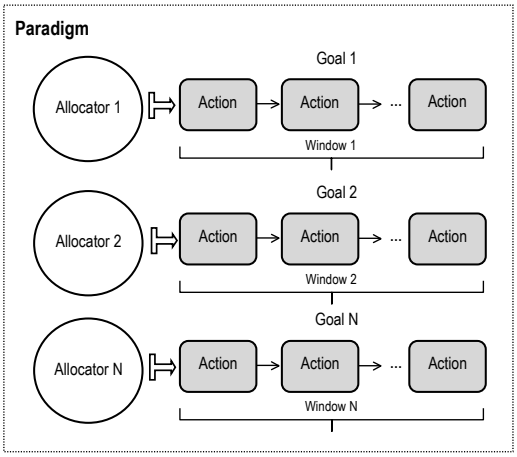


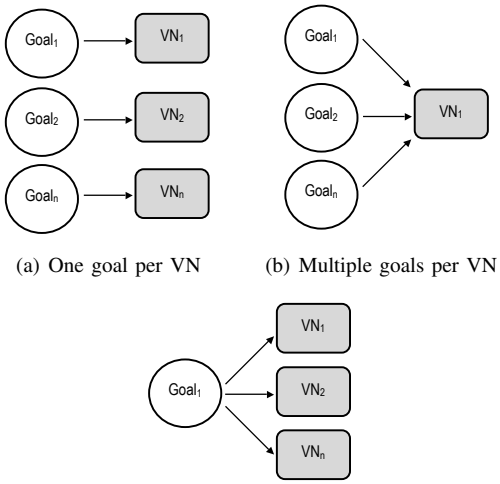
Fig. 1. Allocation paradigms, goals, and actions

B. Design characteristics

Allocation paradigms have two main design characteristics that make them suitable to guide resource allocation in NVEs:

1) Application Awareness: The paradigm model allows the InP to define the relationship between VNs and the applications that ultimately will run on them through three basic procedures illustrated in Figure 2. The first procedure (Figure 2(a)) allows only one goal to be associated with a VN. Different VNs may support different goals. Such procedure is better suited when there is one application per VN or when multiple similar applications share a single VN. In the second procedure (Figure 2(b)), all goals of an allocation paradigm are applied simultaneously during the allocation of a VN, resulting in *hybrid* VNs tailored for different goals. It is important to note that, in this procedure, some goals can predominate over others, depending on how the corresponding policies are defined [8]. Such policy adjustment can be used to handle conflicts between different goals. The third procedure (Figure 2(c)) reflects how VN allocation is tackled by current provisioning systems, where one goal is applied to all VNs. In summary, allocation paradigms aim to cope with the diversity of applications inherent in NVEs that require a variety of strategies to coexist under a single provisioning framework.

2) Adaptive Provisioning: Typical VN embedding schemes map the whole VN on the physical substrate, at once, upon receiving a VN request. Mapping all resources of a VN in an atomic operation is straightforward because the InP has



(a) One goal per VN (b) Multiple goals per VN

(c) One goal, multiple VNs

Fig. 2. VN allocation procedures

the complete view of the substrate network and the associated capacities. However, some approaches assume that individual resources (*i.e.*, virtual nodes) of the same VN request cannot share a physical one and have to be mapped at distinct locations [5] [9]. This limitation may reduce the chances of a successful embedding. In addition, most VN embedding approaches do not properly tackle the case where multiple VN requests arrive simultaneously, which is the typical case in realistic scenarios. Therefore, two or more ongoing VN requests can compete for the same physical resource increasing the chances of failed VN requests.

To overcome the aforementioned problems and allow rapid adaptation of current provisioning approaches to the dynamics of the physical substrate, we argue that a VN request should be mapped *in parts*. To realize such concept we propose the use of allocation *windows* and *rounds*. One allocation window encompasses a fixed number of individual allocation actions defined in real-time by the current allocation paradigm, such as virtual machine creation. The execution of the actions within an allocation window is called a round. Several rounds may be needed in order to complete the full allocation of a VN. In each round, the corresponding window executes the appropriate allocation actions defined by the active paradigm. Figure 3 illustrates how a VN would be mapped using the concepts of windows and rounds. The numbers represent the order virtual resources (*i.e.*, nodes and links) are allocated. The order actions are executed is defined by the policies included in the active paradigm.

The benefits of this partial and multi-iteration mapping is threefold. First, it allows multiple virtual resources of the same VN request to be mapped on the same physical asset. Second, windows allow rapid adaptation to changing network conditions. Two consecutive allocation rounds can result in different mappings compared to mapping all resources at once. For example, after the first round, the mapping of virtual machines can be modified dynamically to select a physical server that turned out to be a better mapping option for a given objective (*e.g.*, reduce number of active physical servers) and that was not available at the first round. Finally, a window can run actions from different VN requests making the problem of managing multiple ongoing VN requests more tractable.

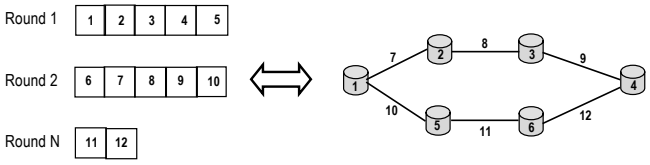


Fig. 3. Allocation windows and rounds

Possible disadvantages of this paradigm-based allocation may appear in small-sized static scenarios (*i.e.*, VNs having long duration, InP or SP objectives not changing over time) because in such scenarios it is unlikely that allocating VNs in parts will produce a different (and better) result compared to a single round allocation and can potentially increase provisioning times. Besides, it may be not feasible to accommodate conflicting goals in a single paradigm without harming application performance. Partial allocation can also produce incomplete VNs if there are not available resources to complete the request in the subsequent rounds.

C. AVNP Problem Formulation

In this subsection we formulate the application-aware virtual network provisioning (AVNP) problem. We model the physical network, the virtual network request, and the allocation paradigm, respectively.

1) Physical Network: We model the physical network as a weighted undirected graph $N^p = (M^p, R^p, L^p, O^p)$, where M^p is the set of physical machines, R^P is the set of physical network elements (*e.g.*, routers and switches), L^p is the set of physical links used to connect physical machines and network elements, and O^p is the set of InP objectives that can be considered during VN provisioning. Each physical machine $m^p \in M^p$ has an associated CPU capacity $c(m^p) \in \mathbb{R}^+$. Each physical link $l^p_{ij} \in L^p$ connecting two physical machines $i, j \in M^p \cup R^p$ has an associated bandwidth $b(l^p_{ij}) \in \mathbb{R}^+$. An objective $o^p \in O^p$ is an overall goal that can be chosen by the InP. Each objective is associated with target index $t(o^p) \in \mathbb{N}$. Possible values that $t(o^p)$ can take are listed in Table I. New objectives can be defined by the InP resulting in additional o^p and $t(o^p)$ values.

2) Virtual Network Request: In our model, a virtual network (VN) request is defined as a weighted undirected graph $N^v = (M^v, L^v, P^v)$, where M^v is the set of virtual machines, L^v is the set of virtual links, and P^v is the set of properties desired for the applications running on the VN. Different than the physical network, a virtual network has no intermediate nodes for routing or switching; virtual links are requested, however, in order to have allocated bandwidth between key virtual machines. Similar to the physical network, each virtual

machine $m^v \in M^v$ requests an amount of CPU capacity $c(m^v) \in \mathbb{R}^+$, and each virtual link $l_{ij}^v \in L^v$ connecting two virtual machines $i, j \in M^v$ has a bandwidth requirement denoted by $b(l_{ij}^v) \in \mathbb{R}^+$. A property $p^v \in P^v$ is a non-functional requirement defined by the applications running on the VN. Each property has a corresponding target index $t(p^v) \in \mathbb{N}$ that is associated with a high-level requirement requested for the VN. For now, the values that $t(p^v)$ can take are listed in Table II. These values are used as reference. The model can be easily extended to include as many properties as supported by the InP. In this case, if the InP supports a new VN property (*e.g.*, , low price) a new p^v and a corresponding $t(p^v)$ value have to be included in the model.

TABLE I
INP OBJECTIVES EXAMPLES

Target	Goal	Description
0	Green	Virtual machines and links should be mapped on the smallest set of physical assets
1	Load balancing	Virtual machines and links should be mapped in distinct locations and cannot share the same physical resource
2	Low communication cost	Virtual machines with more capacity should be placed close to each other
3	High server utilization	Virtual machines of the same request should be grouped in the same server
4	Random	Picks random physical resources to host virtual ones. Used for testing

3) Allocation paradigm: An allocation paradigm $\mathcal{P}$ is defined by a set of goals $(G_1, G_2, ..., G_n)$ that are considered in VN provisioning. Each goal $G_i \in \mathcal{P}$ reflects an InP objective or a characteristic desired for an application running in the VN, having the same meaning of an objective $o^p \in O^p$ supported by the InP or a property $p^v \in P^v$ defined in a VN request, respectively. An individual goal G_i is realized by a set of allocation actions $(A_1, A_2, ..., A_n)$ executed sequentially within a window W_i. Each window W_i has a size attribute $s(W_i) \in \mathbb{N}^+$ corresponding to the number of actions that are executed in each round. An allocator entity $Alloc$ is responsible to trigger each window W. Multiple allocators can run in parallel to speed up the provisioning process.

The provisioning of a VN is thus a function of: the number of resources (*i.e.*, virtual machines and virtual links) that need to be allocated for the requested VN, the number of allocators deployed, and the maximum size of the window of each allocator, which can be dynamically adjusted in each round.

The size of allocation windows can vary according to the current provisioning status of the requested VNs. If a VN is already deployed and no changes are expected in the short run, the size of the allocation window for that VN is zero. On the other hand, if the VN provisioning has just started or modifications on a previously allocated VN are scheduled, then the size of the window is greater than zero. The size of an allocation window can also be adjusted to prioritize one goal over the others. The higher the priority of a goal, the larger the size of its corresponding allocation window because more actions of the goal are executed in a single round. There is a clear tradeoff between the size of the allocation windows and the provisioning time. A large paradigm window requires fewer rounds to allocate a whole VN, but it is unlikely to take advantage of a better allocation option that becomes available. On the other hand, a small paradigm window is more adaptable to dynamic environments at the price of higher overhead, which can result in larger provisioning times.

TABLE II
VN PROPERTIES EXAMPLES

Target	Property	Description
0	Reliability	Replicas of allocated resources should be placed in different locations
1	Security	A virtual machine should not share the same physical machine of another SP
2	Best-effort	Virtual machines can be placed at any location
3	Cheap	cheap physical machines should be selected to host virtual ones
4	Low latency	Virtual links should be mapped on physical paths with small hop number

D. Paradigm Policy Specification

As a part of our paradigm-based provisioning approach, we have developed a policy language to allow InP operators to specify the relationship between paradigms, objectives, and allocation actions. Although there are many policies available [10] designed for a variety of purposes, none of them define adequate constructs to allow InP operators expressing allocation paradigms. The main constructs of the proposed paradigm policy language are presented below and an example of paradigm policy is given in Figure 4.

- **objective:** an objective o defines a customized allocation objective for a paradigm p. Each objective o has a corresponding *window* attribute w specifying the identifier of the allocation window hosting the actions $(a_1, a_2, ..., a_n)$ associated with the objective;
- **action:** an action a specifies an individual allocation. Each action a is defined by an identifier, a list of conditions $(c_1, c_2, ..., c_n)$ that trigger the action, and an operation that actually implements the action, such as *SelectServer* or *MoveToServer* that are used to place or migrate a VM to a given physical machine, respectively;
- **window:** a window w is a logical structure hosting actions that are executed sequentially. A window w has a *size* attribute s that defines the number of actions that are executed before the next scheduling. A window also defines the *order* on which the actions are verified in each turn.

An action is triggered when a set of associated *conditions* is satisfied. The action is then realized by a low-level *operation* (*e.g.*, create a VM) supported by the substrate. The *window* construct defines the size of allocation windows and the

```
Protected_redundant {
      objective Protected {
            action Protected-CreateVM-new {
                  conditions  = {VMs_toAllocate > 0}
                  operation = {SelectServer : empty(all)}
            }
            action Protected-CreateVM-shared {
                  conditions  = {VMs_toAllocate > 0}
                  operation = {SelectServer : used_servers(request)}
            }
            window Protected {
                  size = 2
                  order = {Protected-CreateVM-new, Protected-CreateVM-shared}
            }
      }
      objective Redundant {
            action Redundant-CreateVM {
                  conditions  = {VMs_toAllocate > 0}
                  operation = {SelectServer : empty(all)}
                  operation = {SelectServer : unused_servers(request)}
            }
            window Redundant {
                  size = 3
                  order = {Redundant-CreateVM}
            }
      }
}
```

Fig. 4. Example of paradigm policy - Protected + redundant

order that the actions of a policy are evaluated. In addition to the main constructs, the paradigm policy language can use a set of auxiliary functions provided by the underlying virtualization platform to realize provisioning actions such as virtual machine creation and virtual machine migration.

III. Determining the Efficiency of Allocation Paradigms

Evaluating the efficiency of an allocation paradigm and determining the associated conditions for paradigm switching is a critical task. That is highly dependent on the performance achieved by the applications running on top of a VN, which requires proper feedback from SPs. In this section, we propose a VN computation model that considers all the applications running in a VN and whose output is used to guide paradigm switching.

A. VN Scoring Methodology

The VN computation model is based on the concepts of *tiles* and *scores*, typically found in benchmarking systems [11] [12]. A tile is a fixed-size group of virtual machines running multiple applications. The score is a numerical value attributed to the VN reflecting the combined performance of all tiles (and applications). Our score calculation is adapted from the VMMark benchmarking system [11] [13], used to measure the performance of applications running in virtualized environments. The score metric $\mathcal{S}$ for a VN is calculated as follows:

$$\mathcal{S} = \sum_{i=1}^{\tau} \mathcal{T}_i \tag{1}$$

where $\mathcal{T}_i$ is the performance of the tile i and τ is the total number of tiles a VN supports. The total score of a VN is thus the sum of the performance of all its tiles. The performance of a individual tile $\mathcal{T}$ is defined by:

$$\mathcal{T} = (\prod_{j=1}^{n} \frac{App_j}{Ref_j})^{\frac{1}{n}} \tag{2}$$

where App_j refers to the performance achieved by the jth application in terms of metrics defined by VMMark, for example, *operations per minute* for Web 2.0 applications, Ref_j is the reference value for the application App_j, and n is the total number of applications of the tile. The $\mathcal{T}$ value is thus the geometric mean of the normalized performance of all applications of a tile.

B. VN Performance Prediction Model

In order to evaluate the efficiency of an allocation paradigm in terms of application performance, the InP operator needs to monitor the performance of the applications running on a VN. However, such reactive evaluation may result in excessive monitoring traffic in large NVE scenarios and in performance degradation of short-lived applications. Therefore, predicting the performance of the applications to be deployed over a VN and evaluating allocation paradigms *in advance* can improve overall VN performance. By using the R statistical package [14] to analyze the data submitted to the VMmark Web site [11] we found that the overall score of a VN grows linearly with the number of tiles [15]. Therefore, a simple linear regression model can be used to predict the performance of a VN given the number of tiles. The predicted score $\mathcal{S}'$ of a VN can be defined as:

$$\mathcal{S}' = 0.1573 + \tau \times 1.0206 \tag{3}$$

where τ is the number of tiles. The adjusted R-squared is 0.973.

C. Efficiency of an Allocation Paradigm

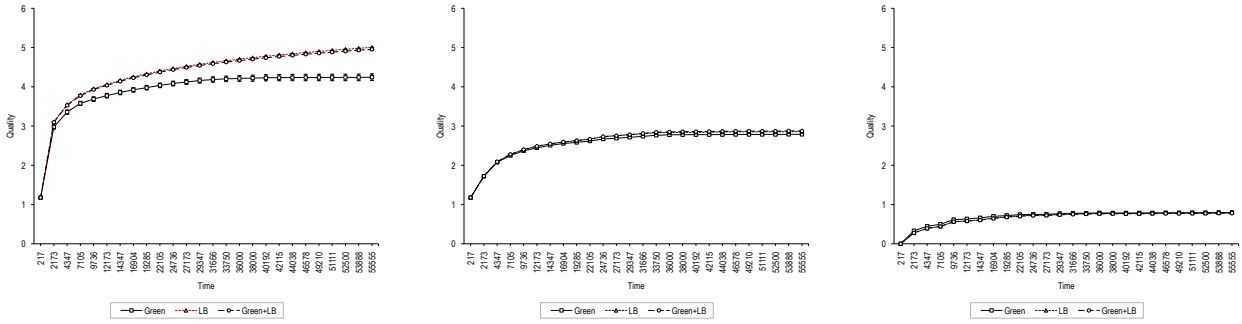
The efficiency of an allocation paradigm is influenced by two main factors: the number of rounds required to complete the provisioning of a VN and the predicted score of the allocated VNs. The number of rounds is directly related to the provisioning time of the VN. The score, in turn, reflects the quality of the allocation paradigm because VNs with high score indicate that hosted applications experience good performance and the VN is unlikely to change in the short run. The quality of an allocation paradigm $\mathcal{Q}$ is given by:

$$\mathcal{Q} = \frac{S'}{\mathcal{U}} \tag{4}$$

where $\mathcal{U}$ is the performance of the reference system obtained when applying Equations 1 and 2 to the reference values defined by VMmark guide [13].

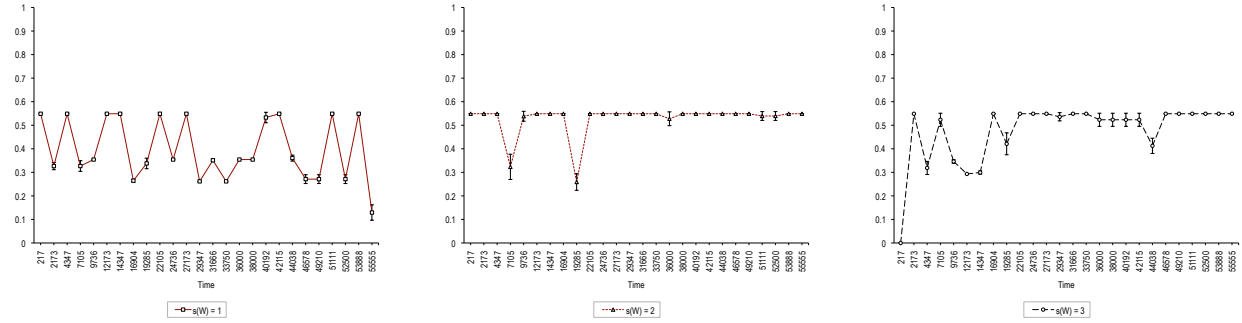
The ultimate goal of an allocation paradigm is to reduce the number of necessary rounds to allocate a VN, which impacts VN deployment time, and avoid excessive paradigm changes, which is related to the stability of the provisioning system. A paradigm change can occur when the score of the provisioned VNs are below a threshold defined by the InP operator. Therefore, the efficiency an allocation paradigm is defined as:

$$\mathcal{E} = \frac{1}{\alpha\mathcal{R} + \beta(1 - \mathcal{Q})} \tag{5}$$



(a) s(W) = 1 (b) s(W) = 2 (c) s(W) = 3

Fig. 5. Paradigm quality



(a) s(W) = 1 (b) s(W) = 2 (c) s(W) = 3

Fig. 6. Paradigm efficiency - Paradigm switching

where $\mathcal{R}$ is the number of rounds, $\mathcal{Q}$ is the paradigm quality, α and β are adjusting factors to weigh the influence of $\mathcal{R}$ and $\mathcal{Q}$ in the paradigm efficiency $\mathcal{E}$.

IV. Evaluation

We evaluate our paradigm-based provisioning in terms of *paradigm quality*, which reflects the quality of an allocation paradigm in terms of the score predicted for the provisioned VNs and *paradigm efficiency* that combines both paradigm quality and number of rounds according to Equation 5. For each experiment, we evaluate two allocation paradigms composed of single goals: Green and Load Balancing (LB) defined in Table I and a combination of the two. We also vary the size of the allocation window from 1 to 3. We also investigate the impact of switching one paradigm (Green) to another (LB) in the half of the simulation.

When analyzing paradigm efficiency, the LB goal presents better performance in terms of paradigm quality (Fig. 5) when compared to the Green goal. This happens because LB tries to spread VN requests over a high number of physical resources, thus increasing the acceptance ratio, which reflects in the quality of the provisioned VNs. When s(W) = 1 the number of rounds required to provision VN is higher, which means that VNs will take longer time to be fully allocated.

When we increase the window size to 2 we observe that the quality of the provisioned VNs decreases. The explanation for lower quality of the allocated VNs when the window size is higher relies on the fact that simultaneous mapping of multiple resources increases the chances of failed requests. On the other hand, the number of rounds is smaller, which results in VNs that are rapidly deployed. This behavior is more evident when we increase the window size to 3.

Paradigm efficiency (Fig. 6) combines both paradigm quality and number of rounds. When s(W) = 1 the paradigm efficiency presents high variability and is exactly the efficiency obtained by the Green goal. This happens because the number of rounds also changes more often in this case. For s(W) = 2, the number of rounds is stable most of time, which reflects in the paradigm efficiency. The quality reduction observed for s(W) = 3 also impacts the overall paradigm efficiency, indicating that paradigm quality and number of rounds have to be considered together. After the paradigm switches from Green to LB, the efficiency resembles the values achieved by the LB goal. The same behavior can be observed for s(W) = 2 and s(W) = 3.

V. Conclusion

Accommodating multiple InP and SP objectives in VN provisioning is a challenging task. In this thesis we have proposed an adaptive provisioning framework that considers multiple InP and SP objectives and specific application characteristics. Our proposal is based on the concept of allocation paradigms, which are groups of application-related policies that guide resource allocation. In addition, we also proposed a virtual network performance computation model to evaluate the efficiency of paradigm-based allocations.

VI. Final Remarks

This thesis can be downloaded from http://www.inf.ufrgs.br/~rpesteves/thesis.pdf. The work conducted during the course of research has been published in [4], [8], and [15].

References

[1] N. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, July 2009.

[2] J. Carapinha and J. Jiménez, "Network Virtualization: a View From the Bottom," in *ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, August 2009, pp. 73–80.

[3] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network Virtualization: a Hypervisor for the Internet?" *IEEE Communications Magazine*, vol. 50, no. 1, pp. 136–143, January 2012.

[4] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the Management of Virtual Networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 2–10, July 2013.

[5] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, February 2012.

[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking Virtual Network Embedding - Substrate Support for Path Splitting and Migration," *ACM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, April 2008.

[7] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, "Virtual Network Embedding Through Topology Awareness and Optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797 – 1813, 2012.

[8] R. P. Esteves, L. Z. Granville, H. Bannazadeh, and R. Boutaba, "Paradigm-Based Adaptive Provisioning in Virtualized Data Centers," in *IFIP/IEEE International Symposium on Integrated Network Management*, May 2013, pp. 169–176.

[9] C. Guo, G. Lu, H. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet - A Data Center Network Virtualization Architecture with Bandwidth Guarantees," in *International Conference on emerging Networking EXperiments and Technologies*, December 2010.

[10] IETF-POLICY, *Policy Languages Review - Policy Languages Interest Group*, 2012. [Online]. Available: http://http://www.w3.org/Policy/pling/wiki/PolicyLangReview

[11] VMmark. http://www.vmware.com/a/vmmark/.

[12] Standard Performance Evaluation Corporation (SPECvirt_sc2010). http://www.spec.org/virt_sc2010/.

[13] VMware VMmark Benchmarking Guide. http://www.vmware.com/go/download-vmmark/.

[14] The R Project for Statistical Computing. http://www.r-project.org.

[15] R. P. Esteves, L. Z. Granville, M. F. Zhani, and R. Boutaba, "Evaluating Allocation Paradigms for Multi-objective Adaptive Provisioning in Virtualized Networks," in *IEEE/IFIP Network Operations and Management Symposium*, May 2014, pp. 1–9.