Towards the Dynamic Provision of Virtualized Security Services

Towards the Dynamic Provision of Virtualized Security Services

Cataldo Basile, Christian Pitscheider, Fulvio Risso,
Fulvio Valenza, and Marco Vallini

Politecnico di Torino, Dip. Automatica e Informatica, Torino, Italy
{cataldo.basile, christian.pitscheider, fulvio.risso,
fulvio.valenza, marco.vallini}@polito.it.

Abstract. When network operators want to offer security services to a large number of customers (potentially tens of million) with current technologies face several limitations in terms of infrastructure management and costs. Network Functions Virtualization (NFV) and Software-Defined Networks (SDN) paradigms try to overcome these limitations by allowing more flexibility, configurability and agility. Unfortunately, the problem of deciding which security services to use, where to place and how to configure them is a multi-dimensional problem that has not an easy solution. This paper provides a preliminary model that can be used to determine the best allocation for the security applications that are needed to satisfy, globally, the requests coming from users while minimizing the cost for the network operator, subject to the different levels of constraints expressed by the involved actors. This model can be exploited either to pursue an initial dimensioning and setup of the system infrastructure or to dynamically adapt it to support the security policies requested by users. Initial validation shows that allocations generated with our model have considerable advantages in terms of costs and performance compared to traditional approaches.

1 Introduction

In the current Internet, security services are usually active as a set of applications operating at the enterprise border, or through personal protection software installed on the user's personal device. Only recently, network operators are starting to be part of the game, with new service offers coming from major network players (e.g., [1]). However, when trying to offer security services to a large number of users (potentially tens of million) with current technologies, several limitations are immediately evident. Among the most important, the cost of the service can be prohibitive, as security services rely mostly on dedicated appliances operating on the traffic traversing through a given link, which implies to deploy a huge number of (expensive) middleboxes in different portions of the network. Furthermore, those middleboxes process all the traffic traversing a given link, hence limiting the possibility to offer different services (e.g., parental control to a first user, email content inspection to a second user) to different groups of users.

A possible solution to this problem is offered by the Network Functions Virtualization (NFV) [2] and Software-Defined Networks (SDN) paradigms [3]. The former transforms network functions into software appliances, executed in virtual machines on standard high-volume servers, which breaks the tight coupling between hardware and software in existing dedicated (hardware-based) appliances and allows security services to be executed on any server available in the network. The second introduces an unprecedented degree of agility in the network, allowing for a finely (and dynamically) selection of an arbitrary portion of traffic and force it to traverse different network paths. This can be used to give each user the security service he desires, as the traffic of different users can be dynamically redirected to a different set of security appliances.

While this new scenario is definitely appealing, it introduces new challenges as the problem of instantiating and configuring those security services, that are now software applications running in virtual machines (VMs), becomes a multi-dimensional problem. For instance, in order for a network operator to decide *which* and *where* to install the security services in its network infrastructure, at least the input from three different actors must be taken into consideration, as shown in Fig. 1. First, **users** are responsible for the selection of the security services they need (e.g., parental control), as well as the possible definition of some QoS parameter on the service itself and possible preferences with respect to the applications (e.g., use only open source software, use applications from a specific vendor). However, if we consider that future trends show a shift towards "human-friendly" policies (e.g., "*Allow Alice to get Internet access only from 6.00pm to 9.00pm*" or "*Block porn and gambling content to my children*"), it becomes immediately clear that the security services requested by each user are not evident from his policies and must be derived automatically by the system, which introduces additional complexity to the problem [4,5].

The second actor is represented by the **developers** of security applications, which need to specify the capabilities of each application and that are required to determine the possible sets of application that can be used to enforce the user policy. In addition, the developer has to specify also application-specific requirements, such as the amount of CPU/memory necessary to achieve a given throughput. Finally, **network operators** are in charge of defining both the network topology and the placement of the servers that will support the execution of the security applications, as well as possible constraints on the network infrastructure (e.g., the traffic of premium users must never traverse congested link, or their applications must be executed on unloaded servers).

This paper provides an initial view of this problem, presenting a preliminary model that, starting from the inputs presented above, can be used to determine the best allocation for the security applications that are needed to satisfy, globally, the requests coming from users while minimizing the cost for the network operator, subject to the different levels of constraints expressed by the involved actors. Particularly, the model aims at providing an answer to the problem of (*a*) **which security applications** are needed, given that several applications may be available for the same task (e.g., content inspection) and that complex services

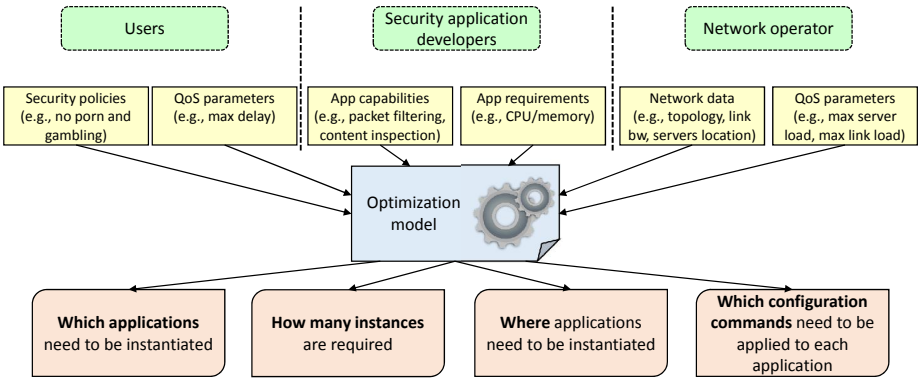

Fig. 1: Model overview.

may require the cooperation of several applications; (b) **how many instances** of them are required as we can range from having a single instance processing the traffic of all users to multiple instances each one operating on the traffic of a single user; (c) **where** each instance needs to be allocated, given the resources available in the network; (d) (possibly) **which configuration** commands are needed by each application to provide the requested service.

This model can be exploited either to pursue an initial dimensioning and setup of the system infrastructure or to dynamically adapt it to support the security policies requested by users. In the former case, the network operator can estimate the requests that may come from its users and engineer the network infrastructure in order to minimize its cost, which includes defining hardware and software resources as well as the location of the computing servers in charge of executing the NFV services (and possibly some dedicated hardware appliances, if needed). In the latter (which is left to our future work), the initial dimensioning is coupled with a run-time optimizer that adapts the workload (in terms of applications and their location) on the different computing nodes based on the request coming from the users. The dynamic adaption may provide substantial benefits in presence of nomadic users that require the network to deliver always the same service from different locations (e.g., on the home ADSL when the user is at home or on the 4G network infrastructure when traveling).

This paper is structured as follows. Section 2 presents a small example which highlights the advantages gain by the optimization. Section 3 presents the mathematical model this solution is based on. Section 4 presents the related work. Section 5 summarizes the paper and presents the future work.

2 A motivating example

This section presents a motivating example based on the backbone network of a small Internet Service Providers (ISP). The example shows our approach and

highlights the advantages, in terms of resources required and execution delay, of the use of an optimization model over the usual (non-optimized) approach when allocating resources to security services.

Let us consider the case where an ISP would like to offer to its customers three different categories of security applications: anti malware (AM), parental control (PC), and traffic filter (TF). Each security application category has a basic ($\mathrm{AM}_b, \mathrm{PC}_b, \mathrm{TF}_b$) and an advanced version ($\mathrm{AM}_a, \mathrm{PC}_a, \mathrm{TF}_a$), the advanced version has better security features but is more demanding in terms of resources. The ISP has a tree-like network infrastructure, where the users are connected to the leafs and the root of the tree represents the network core. The non-optimized approach instantiates at each edge node all application and all user-traffic is processed by all applications, as shown in Fig. 2.

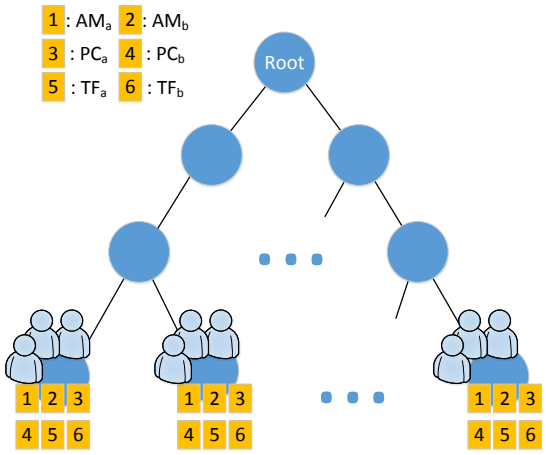

Fig. 2: Non-optimized security applications distribution.

To use our approach and perform the optimization we use more information. The ISP has three different customers types: Fearful, Smart and Unaware. The policies defined by Fearful users are complex and overprotective, consequently several security applications are required to enforce their policies. Smart users define specific policies for only the services that they really need. Unaware users select only the default (basic) security policy. Therefore, not all users require all possible security applications and this is where the optimization occurs.

Our approach considers additional inputs: the distribution of users and their desired security applications, the resource requirements and the execution delay of each security application, and the maximal resources available at each node.

We assume there is a *refinement process* that analyses the user policies and predicts the percentage of security applications required to enforce customers policies. The results of this prediction task for our example are reported in

		Fearful users	Smart users	Unaware users
anti-malware	basic	0%	40%	100%
	advanced	100%	60%	0%
parental control	basic	0%	0%	20%
	advanced	100%	40%	0%
traffic filter	basic	0%	0%	0%
	advanced	100%	100%	0%

Table 1: Percentage of users subscribing to a specific service.

the Table 1. To compute these numbers we assume that every edge node has 100 connected users with a distribution of 30% Fearful, 20% Smart and 50% Unaware.

We also assume that each security application can handle the traffic of at most 100 users (which is the same as the non optimized case to allow comparison) and that AM requires four times as much resources as PC, and PC requires four times as much resources as TF. Furthermore the advanced version of a security application requires twice as much resources than the basic version, in formulas:

$$32 \cdot \mathcal{R}(\mathrm{TF}_b) = 16 \cdot \mathcal{R}(\mathrm{TF}_a) = 8 \cdot \mathcal{R}(\mathrm{PC}_b) = 4 \cdot \mathcal{R}(\mathrm{PC}_a) = 2 \cdot \mathcal{R}(\mathrm{AM}_b) = \mathcal{R}(\mathrm{AM}_a)$$

where $\mathcal{R}(x)$ returns the resources needed by the security application x.

The total resources required at the node n_i are given by the sum of all instantiated security applications : $\mathcal{R}_{n_i} = \sum_j^k \mathcal{R}(x_j)$, where x_j are the security applications at n_i. Then, the total resource used to satisfy the policy are $\mathcal{R}_{tot} = \sum_{n_i} \mathcal{R}_{n_i}$.

Fig. 3 shows the a security application allocation that satisfy user policies obtained with our optimization model. By comparing it to results in Fig. 2, it is evident that the total number of required security application is reduced, because there are some security applications shared with several users that belong to different edge nodes. Moreover, with our approach, the points of presence where security applications need to be allocated is decreased of 40%, indeed, in the non-optimized solution $\mathcal{R}_{tot} = 63 \cdot \mathcal{R}(\mathrm{TF}_b) \cdot N$ are needed, while in the optimization solution $\mathcal{R}_{tot} = 37.5 \cdot \mathcal{R}(\mathrm{TF}_b) \cdot N$, where N represents the number of edge nodes.

In addition, our solution has positive impact on the quality of service, because the traffic only passes through the required security applications. For simplicity, we assume that the performance of a security application depends linearly on the number of the users. Therefore, the execution delay introduced by security applications can be computed as $\mathcal{T}_x = \mathcal{T}(x) \cdot \mathcal{N}(x)$ where $\mathcal{T}$ returns the processing time of a security application with a single user and $\mathcal{N}$ returns the number of users that security application. Therefore, the delay experienced by users is computed as $\sum_{x_k} \mathcal{T}_{x_k}$ where x_k are the security applications used by a user.

By also assuming that:

$$4 \cdot \mathcal{T}(\mathrm{TF}_b) = 2 \cdot \mathcal{T}(\mathrm{TF}_a) = 2 \cdot \mathcal{T}(\mathrm{PC}_b) = 2 \cdot \mathcal{T}(\mathrm{AM}_b) = \mathcal{T}(\mathrm{PC}_a) = \mathcal{T}(\mathrm{AM}_a)$$

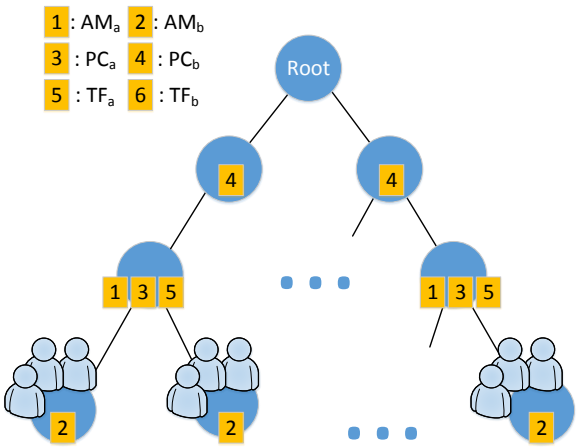

Fig. 3: Optimized security applications distribution.

in our example, the delay added to Smart users is reduced of 60,8%, Fearful users of 68,7%, while for Unaware users of 87,7%, indeed, in the non-optimized solution the delay is $1200 \cdot \mathcal{T}(\mathrm{TF}_b)$ for all users, while with our approach, Smart user delay is $470 \cdot \mathcal{T}(\mathrm{TF}_b)$, Fearful users delay is $376 \cdot \mathcal{T}(\mathrm{TF}_b)$ and Unaware users delay is $148 \cdot \mathcal{T}(\mathrm{TF}_b)$.

3 The model

In this initial work, we abstract the network as a set of terminal nodes, network nodes, allocation places, and connections. Terminal nodes are nodes where users access the ISP network. Network nodes are generic nodes within the ISP network. Allocation places are network nodes where ISP can setup the security controls needed to satisfy users security requirements (i.e., points of presence).

3.1 Users and policies

The set with all the users is $U = \{u_i\}$. We name here the security controls to be the security applications (SA) are known: $S = \{\mathrm{SA}_i\}_i$. Without loss of generality we can state that all the policies that can be selected by users are known and in $P = \{\mathrm{p}_i\}_i$, e.g., as they are provided by the ISP as a set of predefined security services.

Every user specifies his own policy $P_{u_i} = \{p_i\}_{i<n_i} \subseteq P$ and each policy is associated to a set of SA sequences that can be used to enforce them:

$$\mathrm{sa}(p) = \left\{(\mathrm{SA}_{1,j}, ..., \mathrm{SA}_{k_i,j})_i\right\}_j$$

where $(\mathrm{SA}_{1,j}, ..., \mathrm{SA}_{k_i,j})_i$ is the i-th policy implementation. A policy implementation is an ordered sequence of SAs that can be used to enforce the policy. In this model, SAs are considered to be chained but the model easily extends to other arrangements that can be modelled as graphs (but it is outside of the scope of this paper).

We assume that all the policy implementations are correct as they are provided by the trusted and correct refinement process. The refinement of all the available policies into sequences of SAs needed to enforce them can be automated or can be manually done by administrators at the network operator. Note that the refinement process also considers possible incompatibilities between SAs when generating the policy implementations.

Moreover, policies are refined in different ways depending on the user due to user preferences, decisions or other limitations. Thus not all the sequences are applicable to all the users (e.g., a user does not want SAs from a specific vendor). Therefore we introduced the following function:

$$\mathrm{sa}(p, u_k) = \left\{(\mathrm{SA}_{1,j}, ..., \mathrm{SA}_{k_i,j})_i\right\}_j$$

where $\mathrm{sa}(p, u_k) \subseteq \mathrm{sa}(p)$, that is, user u_k can select only a subset of the available policy implementations.

3.2 SA instantiation modes

To further optimize allocation, we consider that SAs can be used in three modes:

- *individual*, when the SA is only used and allocated for a single user;
- *shared*, when the SA is shared among several users. This case is approximated with a single process enforcing a common configuration for all users (single rules can be labelled to be distinguishable). Malicious users can insert rules into the common configuration so that the overall SA performance is affected;
- *multitenant*, when the SA shared among several users but each user has a separate "enforcement engine". This case is approximated with a single coordinating process plus several per-user processes that perform the enforcement. Each per-user process uses a fixed amount of resources. Malicious users cannot affect the performance of other users.

SAs can be used in more than one mode. Shared and multi-tenant can be also used as individual SAs. We do not expect that multi-tenant can also be used as shared ones but we cannot exclude it. The modes are provided in a "SA manifest file" by the application developers.

3.3 Network model and allocation

The abstract network allocation model is represented as a graph G:

$$G = ((U \cup N) \cup A, E)$$

Nodes represent the entities: users (in U), network devices (in N), or allocation places (in A). The users are the terminal nodes (i.e., they only have outgoing nodes). Edges in E are only between a node in $(U \cup N)$ and a node in A. Thus G is bipartite.

Allocation places are associated to a set of constraints $\rho(a) = \{c_i\}_{i<m}$ used during the allocation. Examples of constraints are the number of SAs that can be deployed in a given places or relations with the HW resources (e.g., RAM or available CPU cycles). Other constraints are considered during the allocation phase, e.g., shared and multi-tenant should/must not allocated where only a user can be protected.

3.4 Metrics

Taking into account the capabilities and requirements of each security application, specified by its developers, (CPU, memory, throughput, etc.), the metrics can be used to evaluate the impact of the enforcement of a policy on a given SA, they are in the set $M = \{\mu_k\}_k$:

- *SA metrics* that provide a fixed value for each SA, i.e., regardless of the policy and users:
$$\mu_k^f(\text{SA})$$
Examples of these metrics are the cost to buy a SA, the memory size needed to initialize a SA (without configuring anything).
- *policy metrics* that provide a value for each SA-policy pair, i.e., regardless of the users or allocation graph:
$$\mu_k^p(\text{SA}, p)$$
Examples of these metrics are the number of rules to enforce the policy p on SA.
- *policy implementation metrics* whose value depends on the specific user policy implementation and the allocation graph.:
$$\mu_k^i(p, u_i, j)$$
Examples of these metrics use formulas to estimate the overall delay due to the enforcement of a policy p with the policy j-th policy implementation on the abstract allocation graph G.

3.5 Problem definition

The optimization problem we want to solve is defined as follows:

"*Given an abstract network allocation model G, a set of users U with their policies $P^{(u_i)}$, their policy implementations* $\text{sa}(p, u_k) = \left\{(\text{SA}_{1,j}, ..., \text{SA}_{k_i,j})_i\right\}_j$ *and an optimization function based on the metrics μ_k, find a set of policy implementations that satisfy user policies.*"

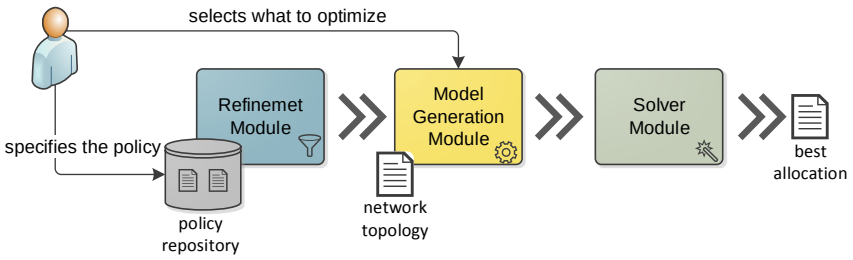

Fig. 4: Prototype architecture.

We assume there is a function that returns the allocation places that can be used to enforce the j-th policy implementation of user u_k:

$$A\left(u_k, \mathrm{j}\right) = \{a_1, \ldots\} \subseteq A$$

This information can be produced during the refinement process. We propose to automatically build an optimization program from the inputs (users, network, policies) based on the selected optimization function. The built optimization problem is then solved using off-the-shelf solvers, as presented in Section 3.6.

3.6 Model implementation: the prototype

Fig. 4 presents the simple linear workflow of the tool. User policies are stored into a repository that is accessed by the Refinement Module. The user is allowed to select from a fixed list of predefined policies. As it is not the main target of this work presented in this paper, current implementation of the Refinement Module is simplified; there is a limited number of policies each one mapped with a fixed function to the SAs that are needed to implement it.

The Model Generation Module reads the mapping of users and security applications and generates the Mixed-integer linear programming (MILP) problem to be optimized by the solver. More in details, this component uses the following inputs when building the optimization problem:

- the target function, which specifies the desired optimization outcome (e.g., the maximization of the network throughput or minimization of the deployment cost);
- the resource limits and the other constraints ρ associated to the current network topology, custom allocations (e.g., Bob's security application must be allocated on node x), which are translated into optimization constraints, mainly inequalities, by the optimization model builder;
- the allocation generation algorithm to use, part of the optimization model builder, which places user implementations $\mathrm{sa}(p, u_k) = \left\{\left(\mathrm{SA}_{1,j}, ..., \mathrm{SA}_{k_i,j}\right)_i\right\}_j$

in the valid allocation places for user implementations. Moreover, this algorithm also determines valid allocations by combining an individual allocation for each user. This algorithm does not generate all the possible combinations, but it internally uses metrics to build dominance properties to limit the solution space.
- metric combination functions, which combines metric functions to permit the estimation of the target function for each valid allocation.

We have implemented in our prototype a simple module that generates a linear optimization problem that minimizes the number of used SAs. The optimization model builder has been implemented using a rule engine for complex event processing (Drools [6]).

Finally, we have used the IBM CPLEX solver to solve the generated problems. Given the size of the problems that we have built, the entire optimization process (generation and actual optimization) lasts less than 1 s. Further tests on scalability are needed, however, given that the performance bottleneck is the optimization performed at the solver, relying on off-the-shelf products whose performance is excellent and the size of linear problems that are able to manage is very large, we expect to be able to easily manage real world example.

4 Related work

Recently, several works have been proposed to adopt NFV and SDN. However, the provisioning of new services by using these paradigms brings up the resource allocation problem. More precisely, in the NFV domain this is known as Virtual Network Function Placement (VNF-P).

An interesting contribution to manage VNF-P is offered by [7]. This work presents and evaluates a formal model for resource allocation within the NFV environments. In particular it considers services that can be allocated on hybrid scenarios, i.e., part of services are instantiated on physical hardware and the others on a virtualized environment. The allocation problem typically has a wide set of solutions, therefore an optimization criteria must be provided to choose among them. Moens et al. propose an Integer Linear Programming (ILP) model to minimize the number of used servers.

Multi-objective Resource Scheduling Algorithm (MORSA) [8] proposes to optimize the resource of the NFV domain by addressing requirements of the infrastructure and the stakeholder policies. In particular, it focuses on the combination of conflicting objectives solving them by using Multi-objective Genetic Algorithm (MOGA). This makes it possible to obtain approximate solutions in a reasonable computation time. The provided architecture follows a plug-in approach that increases the flexibility.

Similarly to the others, Clayman et al. in [9] addresses the dynamic placement of virtualized functions and services. However, this contribution argues that to automatically manage this type of infrastructure a high-level orchestration is needed. In particular, the orchestrator manages the configuration, creation and

removal of the virtual nodes and related services. This component is supported by a monitoring system, that collects and reports on the behaviour of the resources.

Beloglazov et al. in [10] propose a strategy to manage live migration (i.e., switching off idles nodes) minimizing power consumption by considering CPU utilization. Finally, García-Valls et al. in [11] describe a graphical discrete event simulation tool for solving the virtual resource allocation in SDN domain.

Gember et al. presented the design, implementation, and evaluation of a network-aware orchestration layer for Middleboxes (MBs), named Stratos [12] . Stratos allows tenants to specify logical middlebox deployments by using a simple logical topology abstraction. The key components of Stratos are an application-aware scheme for scaling, a rack-aware placement and a network-aware flow distribution, which work in concert to carefully manage network resources at various time scales.

Meng et al. in [13] proposed an approach of manipulating VM placement to address the scalability concern in modern data center networks. The authors formulated this problem in a Traffic-aware Virtual Machine Placement Problem (TVMPP), proving its NP-hardness and proposing a two-tier approximation algorithm to solve it efficiently . Another result of this work is an analysis on how traffic patterns and network topology in data centers affect the potential network scalability.

Mehraghdam et al. in [14] have formulated an optimization problem for placing the chained VNF in an operators network with multiple sites, based on requirements of the tenants and the operator. The authors presented a formal model for specifying VNF chaining requests and requirements. In detail an investigation of the possible trade-offs among different optimization objectives with a Pareto set analysis was done.

5 Conclusions and Future Work

This paper presents an initial optimization model capable of selecting the most appropriated security application and determining the best allocation places for them. The model requires as input the number of connected users and their security requirements, all possible allocation places, and the resources available at each allocation place. The resulting deployment configuration has two advantages over a non optimized solution. At first, the deployment cost for a network operator is reduced because less security applications are deployed and therefore less resources are required. Secondly, the quality of the service is improved because less security applications are involved in traffic processing.

Future work will couple the initial dimensioning with a run-time optimizer that adapts the workload on the different computing nodes. The optimization is performed in terms of applications and their locations based on the user demand. This will have substantial benefits in presence of nomadic users, because they require the same services from different locations.

Acknowledgment

The research described in this paper is part of the SECURED project, co-funded by the European Commission (FP7 grant agreement no. 611458).

References

1. Vodafone: Rete sicura (2015) https://retesicura.vodafone.it/.
2. The European Telecommunications Standards Institute: Network function virtualization - White Paper 2. Technical report (October 2013)
3. Feamster, N., Rexford, J., Zegura, E.: The road to SDN: An intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review **44**(2) (April 2014) 87–98
4. Basile, C., Lioy, A., Scozzi, S., Vallini, M.: Ontology-based policy translation. In: CISIS2009: 2nd International Workshop on Computational Intelligence in Security for Information Systems, Burgos, Spain (September 2009) 117–126
5. Basile, C., Lioy, A., Pitscheider, C., Valenza, F., Vallini, M.: A novel approach for integrating security policy enforcement with dynamic network virtualization. In: NetSoft2015: 1st IEEE Conference on Network Softwarization, London (UK) (April 2014)
6. Proctor, M.: Drools: A rule engine for complex event processing. In: AGTIVE2011: 4th International Conference on Applications of Graph Transformations with Industrial Relevance, Berlin, Heidelberg (2012) 2–2
7. Moens, H., De Turck, F.: VNF-P: A model for efficient placement of virtualized network functions. In: CNSM2014: 10th International Conference on Network and Service Management. (November 2014) 418–423
8. Yoshida, M., Shen, W., Kawabata, T., Minato, K., Imajuku, W.: MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure. In: APNOMS2014 : 16th Asia-Pacific Network Operations and Management Symposium. (September 2014) 1–6
9. Clayman, S., Maini, E., Galis, A., Manzalini, A., Mazzocca, N.: The dynamic placement of virtual network functions. In: NOMS2014 : Network Operations and Management Symposium. (May 2014) 1–9
10. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: CCGrid2010: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. (May 2010) 577–578
11. García, A.J., Cervelló-Pastor, C., Jiménez, Y.: A modular simulation tool of an orchestrator for allocating virtual resources in SDN. International Journal of Modeling and Optimization **4**(2) (2014) 88–99
12. Gember, A., Krishnamurthy, A., John, S.S., Grandl, R., Gao, X., Anand, A., Benson, T., Akella, A., Sekar, V.: Stratos: A network-aware orchestration layer for middleboxes in the cloud. CoRR **abs/1305.0209** (June 2013)
13. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: INFOCOM2010, San Diego (CA) (March 2010) 1–9
14. Mehraghdam, S., Keller, M., Karl, H.: Specifying and placing chains of virtual network functions. In: CloudNet2014: IEEE 3rd International Conference on Cloud Networking, Luxembourg (October 2014) 7–13