On the Optimality of Virtualized Security Function Placement in Multi-Tenant Data Centers

Abeer Ali
School of Computing Science
University of Glasgow, UK
A.Ali.4@research.gla.ac.uk

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK
Christos.Anagnostopoulos@glasgow.ac.uk

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow, UK
Dimitrios.Pezaros@glasgow.ac.uk

***Abstract*—Security and service protection against cyber attacks remain among the primary challenges for virtualized, multi-tenant Data Centres (DCs), for reasons that vary from lack of resource isolation to the monolithic nature of legacy middleboxes. Although security is currently considered a property of the underlying infrastructure, diverse services require protection against different threats and at timescales which are on par with those of service deployment and elastic resource provisioning. We address the resource allocation problem of deploying customised security services over a virtualized, multi-tenant DC. We formulate the problem in Integral Linear Programming (ILP) as an instance of the NP-hard variable size variable cost bin packing problem with the objective of maximising the residual resources after allocation. We propose a modified version of the Best Fit Decreasing algorithm (BFD) to solve the problem in polynomial time and we show that BFD optimises the objective function up to 80% more than other algorithms.**

***Keywords*—Data Centers, security virtualization, resource-aware allocation**

I. Introduction

Security in DCs is deployed as a sequence of in-network middleboxes such as firewalls and Intrusion Detection and Prevention Systems (IDS/IPS). To mitigate the problems of legacy hardware-based middleboxes such as, e.g., expensiveness, vendor lock-in, deployment inflexibility, and lack of resource scalability [1], virtualized middleboxes have emerged such as WAN optimizers [2] and IDPS systems [3], [4]). Moreover, as ICT is moving to the Cloud, more in-the-cloud network services are offered by Cloud Services Providers or third party companies [5]. Network Function Virtualization (NFV) is the technology used to deploy virtualized middleboxes as network functions (NFs). For security services, the flexibility of Virtualized Network Functions (VNFs) deployment and scaling will offer a rapid response such as deploying additional modules that raises the efficiency of the system to handle attacks and changes in traffic and infrastructure [6].

However, the inefficient management of in-cloud network functions can itself cause performance degradation and/or reduce turnover. For example, where the functions will be deployed and how this will effect the DC routing makes a difference, since such decisions highly affect the network performance and can result in resource wastage or cause bottlenecks [5]. Our work focuses on the efficient allocation of security services over multi-tenant virtualized DCs. A pool of security modules such as, e.g, firewalls, IDS/IPS, DDoS mitigation tools, and DPI engines, are available for tenants to request. Requested modules will be deployed in the DC as virtualized network functions where they can process traffic destined to the requesting tenant. The modules are offered on a per tenant basis to provide customised security, for example, an end-user will require basic security services while commercial or critical services can request more advanced service in the form of, e.g., multiple modules to ensure higher availability.

In this paper, we address the problem of allocation of different security modules over a virtualized DC infrastructure using a *resource-aware placement methodology*. Our contributions include the classification of the security modules based on the granularity of traffic. The ILP formalisation of the security function placement as an instance of the NP-hard bin packing optimisation problem with an objective function based on maximising residual resources. For a polynomial time solution, a modified version of the Best-Fit Decreasing (BFD) greedy algorithm is designed. Finally, through a comparative assessment, we demonstrate that BFD optimises the residual resources when compared to other algorithms.

The reminder of this paper is structured as follows: Section II discusses related work, and Section III introduces the problem of security modules placement. Section IV formalises the problem in ILP form and, in Section V, we adopt a modified version of the BFD algorithm and provide a performance and comparative assessment against different greedy algorithms in Section VI. Finally, Section VII concludes the paper.

II. Related work

As the use of virtualized middleboxes becomes more widespread, research is focusing on the different aspects of managing in-network functions in virtualized environments [5], [7]–[9]. Yet only a few consider the distinct requirements and constraints related to security functions such as, e.g., the granularity of the processed traffic. Gamber et al. [5] suggest a network-aware orchestration layer for virtualized middleboxes. It uses horizontal scaling to leverage performance, however, it only considers functions that process traffic at the flow level. The authors in [10], address the allocation of security services in virtualized environments and discuss their challenges. They model the allocation problem for ISP networks to minimise the cost of operators as a Mixed-Integer Linear Programming (MILP) problem but no results for the implementation are reported.

What distinguishes our work from previous research in the management of virtualized services is that it explicitly

addresses the requirements and constraints of security services such as traffic constraints which impose a restriction on allocation. While most previous work propose to place NFs on DC servers [5], [7], [11], the proposed architecture reduces the overhead of having to resort to non-shortest-path routing by deploying the security modules on the actual traffic path. Moreover, virtualized functions used to be implemented in a similar fashion to their hardware counterparts, i.e., as high-speed high-capacity appliances. They process traffic for different users in the network within their capacity. However, in multi-tenant environments where services are offered on a per-tenant basis, sharing security functions among different tenants will increase the complexity of managing them. Furthermore, specific security functionality that is based on building a behavioural model for traffic such as, e.g., anomaly detection modules, cannot be shared since each tenant will have a different normal behavioural model. Besides, tenants should be allowed to configure their security functions themselves. Yet, if the same binaries are shared among different tenants, then access control becomes cumbersome and there are real risks for illegitimate access that can cause security policy violations. Our work reduces the complexity and the security risks of deploying shared modules by adopting non-sharing strategy.

III. Security Management

In multi-tenant virtualized DCs, users run different applications, each with different security requirements. For instance, a typical web server may require security modules to detect and mitigate HTTP flood attacks and SQL injections, while critical servers may require a firewall, IDS and/or DPI to guarantee high availability and data integrity. We focus on orchestrating security services in multi-tenant virtualized DCs where they are offered as modules that are allocated throughout the infrastructure to process the required traffic. The modules are offered on a per-tenant basis and each request will result in deploying a module to process flows of the requesting tenant. A placement algorithm is responsible for selecting allocations for security requests to satisfy the request requirements and constraints, and maintain an efficient usage of the DC resources. Such arrangement offers the customisation to fulfil diverse tenants needs for different security services and levels of protection, and reduce the complexity of having to manage shared security modules.

A. Architecture

A multi-rooted tree is one of the most common virtualized DC network architectures [12]. For example, a k=4 fat-tree topology shown in Figure 1 with three layers of switches (ToR, aggregation, and core). We assume routing is flow-based Equal Cost Multiple Path (ECMP) [13], where a flow is identified by the typical five-tuple (source and destination IP address, source and destination transport layer ports, and IP_PROTO type), and flows are distributed over equal cost links. To reduce the detour length of the path that traffic has to take to pass through a security function, our approach allocates security modules to points collocated with switches at all layers. Traffic is rerouted from switches to the security function and back as shown in Figure 1. While it is common to deploy network functions at host VMs, our approach reduces the overhead imposed by the security system by deploying the security function on the actual traffic path and avoid redirecting traffic. The security function abstraction can be implemented as a distinct namespace within a software switch or on a separate, virtualized commodity x86 architecture that physically connects to a traffic-forwarding switch.

B. Placement of Security Modules

The placement problem is selecting an allocation for the requested module that satisfies the tenant request for a security service. The placement must ensure that the module requirements and constraints are satisfied. For the security function, the traffic constraints and the resource requirements are considered in the selection process.

1) Traffic Constraints: Traffic can be processed at different levels such as per-packet, per-flow, or flow-aggregate. Each level provides protection against different types of security vulnerabilities. For example, per-packet or per-flow processing cannot detect threats that span multiple flows such as, e.g., DDoS flooding. We classify security functions to two classes based on how traffic is processed to detect threats and subsequently the granularity of the traffic required for each. This classification will determine the traffic constraints of the modules and help the placement algorithm select the location where a security function can accurately capture the required traffic.

The **Stateless class** represents modules that process traffic at the individual flow or packet level. The typical operation of a module of this class is to match patterns of the packet or flow specification against a set of signatures/access lists and take action on finding a match, e.g., block, pass, alert and/or log. Therefore, detection or mitigation depends on the state of a single packet or flow. Since this packet/flow matching on a given signature is done independently at different links, replicated instances of this class can be distributed across multiple network locations where traffic destined to a tenant is being split. Examples of this equivalence class include Access Control List (ACL)-based stateless firewalls that evaluate packet contents statically; firewalls that keep track of the bidirectional state of network connections [14]; signature-based IDS and DPI such as, e.g., Snort [3] and Suricata [4].

The **Stateful class** includes security modules that process traffic to extract anomalies based on a coarser granularity than the stateless class such as, e.g., flow-aggregation. They use techniques based on different features of traffic such as, e.g., changes in traffic volume (Change Point Detection [15]), deviations in a given traffic feature distribution (Entropy, Histograms, etc. [16]), or use more complex machine learning techniques. They mine information from flow aggregate features to construct a model of normal behaviour and detect anomalies based on deviations from such normality. Therefore, a security module of this class cannot be duplicated and the intended monitored flows must all be steered to one instance of this type to capture an accurate behaviour model.

The traffic constraints of a security module will limit the available locations where the required traffic granularity is satisfied. For example, in a three-tier architecture, all traffic destined to a server is routed through the ToR switch connecting the server to the DC. Hence, modules can be deployed at this layer as a single instance satisfying all traffic

granularity levels. While at higher layers of the architecture, traffic destined to a tenant traverses multiple links at each level and only a fraction of the flows is observed at each parent switch. Therefore, stateless class modules can employ replicated instances across the parent switches of the requested tenants at certain level as long as the entire traffic is monitored. On the other hand, stateful class modules need to coordinate between duplicate instances to accurately capture the traffic features, otherwise traffic would need to be rerouted through a single switch where a single instance of the security module would be deployed.

2) Resource Requirements: Locations available to host software security modules/binaries have a predefined capacity of computing resources to allocate. This capacity is defined by a resource vector (CPU, RAM, I/O bandwidth, Storage). Each request from a tenant has a similar vector of the expected required resources which is the estimation of the amount of resources the modules will consume to process the requesting tenant traffic. However, computing the resources required by a security module is a complex process since, for example, it involves estimating the highly bursty temporal traffic characteristics of the corresponding tenant that have a major impact on the amount of the required resources.

To determine how to estimate the resources required for a security module request, we considered the widely-used IDS Snort as a case-study. Snort captures and inspects packets to detect malicious activities through DPI on the packet payload against attacks signatures represented as a rule set. While many traffic features affect resource consumption such as, e.g., traffic load, traffic type, packet size, present or absence of attack, number of packet fragmentation [17]–[19] , yet traffic intensity is the main factor to consider for initial deployment. Thus, each security module in the pool will be associated with two resource vectors that represent a *baseline requirement* (resources required for initial deployment of the module) and a *traffic requirement* associated with each traffic type (e.g. TCP, HTTP, mixed traffic,etc.), and represents the estimated amount of resources required to process a unit of traffic of this type. Besides, tenants associate their requests with an expected rate(s) for each traffic type(s) to be processed. The placement must ensure satisfying the resource requirements of the request such as the resources available at the chosen location(s) must be greater than or equal to the sum of the baseline and traffic resource required by the requested module.

IV. Problem Formulation

A. Problem Rationale

The placement of security modules is a resource allocation problem, where modules requested by tenants in virtualized DCs are allocated in a set of distributed locations each with limited capacity. We focus on the initial placement of the two security equivalence classes introduced in section III. To satisfy the traffic constraints, the *stateless class* modules will independently duplicate over links in case of allocations where traffic is distributed over multiple links. The *statefull class* modules for simplicity will be allocated to ToR switches where they can capture all traffic flows destined to the requested tenant. For example, in the k=4 fat-tree shown in Figure 1, a stateless class module deployed for a tenant in server 8 (in grey) has three available locations (shown in blue blocks) that satisfy the traffic constraints where each allocation will have redundant instances in each switch. on the contrary, a statefull module will be directly deployed at the corresponding ToR switch. When more than one allocations satisfy the resource requirements and traffic constraints of a request, the one that optimises the placement objective will be selected. We design the placement algorithm to achieve maximum allocation ratio using minimum amount of resources. We accomplish that with two objectives, the first objective is to maximise the *resources placement ratio* to achieve maximum allocation ratio of resources. The second objective is to maximise the *residual resources* of the framework which represent the spare resources after the placement has been completed.

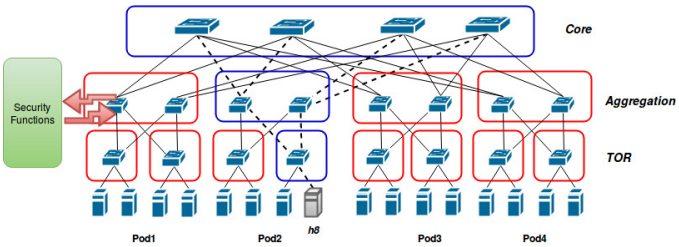


Fig. 1. locations rearrangement for ILP in k=4 Fat-Tree DC

B. Integer Linear Program (ILP) Formulation

The placement of security modules is an instance of a variable cost – variable size bin packing problem (VSBPP) [20] where switches can be represented as bins, the security modules are the items, bin size is the resource capacity of the switches, and the price is the resource consumption of modules to be allocated. To represent the problem as VSBPP in ILP form, every request must be allocated to only one location. Therefore, switches are rearranged in a new location schema as the blocks shown in Figure 1. The new locations schema combines switches with distributed traffic to one location. Consequently, the number of locations in the structure will be changed, for instance in the k=4 fat-tree shown in Figure 1, the number of locations will be reduced to 13 instead of 20. Besides, locations combining more than one switch will have a capacity equal to the sum of the capacity of the combined switches. Also, the cost of allocating a module to a combined location will be equal the cost of duplicating the module across the switches in the new location. For example, server 8 will have three valid locations to allocate modules that are shown in blue blocks. The ToR blue block location with a total cost equal the cost of the deployment of one instance. The second location will be the blue block in the aggregation level with the cost of two instances, and the third location in the core level will have a cost of four instances deployment.

The formulation of problem is as follows: Let the overall framework include a set of q requests Q= $\{r_1, r_2, ...r_q\}$ with each request r representing a module requested by a tenant, a set of p locations L= $\{l_1, l_2, ...l_p\}$, each with attribute $l.s$ as the resources' capacity at location l. A matrix v of size pxq representing the traffic constraint where $v_{r,l} = 1$ if location l can satisfy request r; 0 otherwise. For instance, for the stateful class, only the ToR level will be valid for allocation. While, for the stateless class, there are three parent locations that are

valid. To combine the two objectives mentioned earlier, we use an approach that allows us to represent them as a linear function to maximise the *residual resources* of the allocation. Our approach adds an extra virtual location to the actual locations with enough capacity to accommodate all requests, however, the cost will be more than the cost of allocation in the other real locations. The resource cost of allocating modules are represented by variable c of size pxq where $c_{r,l}$ represent the resources required to place request r in location l. After the placement is complete, all requests allocated to this virtual location will be considered unallocated requests. The allocation is represented as a binary variable x of size pxq where $x_{r,l}$=1 when request r is allocated to location l; 0 otherwise., i.e.:

$$x_{r,l} \in \begin{cases} 1, \\ 0 \end{cases} \qquad \forall r \in Q, \ \ \forall l \in L$$

The ILP formulation is as follows:

$$\max . \Big[\sum_{\forall l \in L} l.s - \sum_{\forall r \in Q} \sum_{\forall l \in L} x_{r,l} \cdot c_{r,l} \Big] \tag{1}$$

$$\text{s.t.} \sum_{\forall r \in Q} x_{r,l} \cdot c_{r,l} \ \leq l.s \quad \forall l \in L \tag{2}$$

$$x_{r,l} = 0, \quad \forall r \in Q, \ \forall l \in L \qquad if \ v_{r,l} = 0 \tag{3}$$

$$\sum_{\forall l \in L} x_{r,l} = 1, \quad \forall r \in Q \tag{4}$$

Note: the objective function of the ILP formulation is represented in (1) as maximising the residual resources after the placement. The constraint in (2) represents the locations capacity constraints which are enforced by the resources requirements of the problem. The constraint in (3) represents the location validity for requests which enforce the traffic constraint of the problem. The constraint in (4) ensures that each request is allocated to only one location.

V. Placement Methodology

As bin-packing problems have been shown to be NP-hard [21], many heuristic algorithms have been proposed to solve it. The Best Fit Decreasing (BFD) algorithm is the most widely applied, since it is proven to use not more than $\frac{11}{9} \cdot OPT + 1$ bins, where OPT is the number of bins in an optimal solution [22] within polynomial time [23]. In BFD, the items are sorted in decreasing order and the sorted items are allocated such that minimum empty space will be left after the allocation The BFD algorithm saves resources by selecting the best fit location. Moreover, The decreasing order of requests will result in allocating modules with high resource demand in more efficient locations that will reduce the total resource consumption and ultimately will result in accommodating more requests. We have adopted a modified version of BFD called *power-aware best fit decreasing* where virtual machines (VM) are allocated to locations that cause the least increase in power consumption [20]. We substitute power consumption of VM placement with resource consumption in security function placement as the cost function, as illustrated by the following algorithm:

Input: Set of requests Q, set of locations L
Output: Set of requests allocated to locations A

$A \leftarrow \emptyset$ // initialisation
$Q^* \leftarrow sort(Q)$ // sort request w.r.t. resources
for all $r \in Q^*$ **do**
  for all $l \in L$ **do**
    if $((capacity(A, r, l) = \texttt{TRUE}) \bigwedge (validation(r, l) = \texttt{TRUE}))$ **then**
      $l^* = \arg\min_{l' \in L} cost(r, l')$
    end if
    if $(l^* \neq 0)$ **then**
      $A \leftarrow A \cup \{(r, l^*)\}$ // allocate request r to location l^*
    end if
  end for
end for
return Set of allocated requests A

The set A refers to the set of allocated requests in certain locations, Q is set of initial requests, L is set of locations. The function $sort(Q)$ sorts the requests from Q by a decreasing order of resources required; $capacity(A, r, l)$ ensures the resources required in location l in allocation A is enough to accommodate a give request r; $validation(r, l)$ constrains the location to those who satisfy the traffic constraints such as for a stateful class request r, only parent location l that is in ToR level is valid, while $cost(r, l)$ calculates the cost of allocating the request r to location l. The algorithm sorts requests and then allocates each request to the *Best-Fit (BF)* location. First, it sorts requests by the amount of required resources to deploy one instance for each in a decreasing order. Then it allocates the sorted modules one by one by calculating the cost of placing the module to each valid location. The algorithm only considers locations with enough resources to accommodate the module, and finally selects the *BF* allocation by selecting the location that causes the minimum increase in total resource consumption/cost. The proposed resource-aware BFD algorithm achieves the objectives of maximising the allocation ratio using the minimum amount of resources.

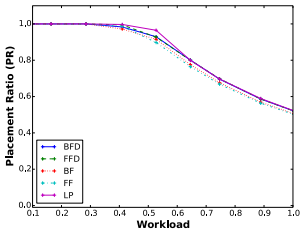
VI. Performance Evaluation

A. Performance metrics

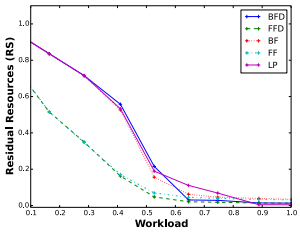
To demonstrate the efficiency of the resource allocation algorithms, we introduce two metrics that represent our objectives presented in Section IV. The first metric is the overall *Placement Ratio* (PR), which represents the ratio of allocated resources out of the total amount of resources requested. For example, PR=1 will indicate meeting all requests by finding the allocation that satisfies their constraints, while PR<1 indicates a failure ratio where not all requests are satisfied. Furthermore, it indicates the utilisation level of resources, where an efficient algorithm will result in higher amount of allocated resources. The second metric is the *Residual Resources* (RS) of the network, which is the ratio of the spare resources (after placement) to the total amount of resources available and is calculated by adding the residual resources at each location after placement. For a specific PR, RS indicates the efficiency of resource usage as the ratio of saved resources after placement.

B. Models Comparison

We compare the proposed BFD algorithm with three resource allocation algorithms: First Fit (FF), Best Fit (BF), and First Fit Decreasing (FFD) [24]. Specifically, in FF, the

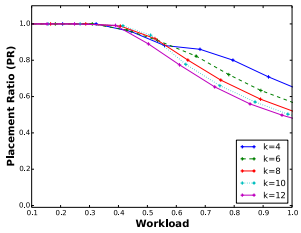


(a) Placement Ratio

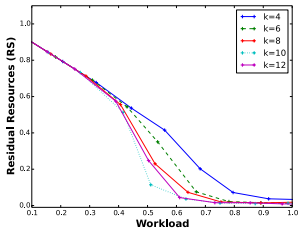


(b) Residual Resources

Fig. 2. Placement Ratio (PR) and Residual Resources (RS) BFD, FFD, BF and FF algorithms and ILP when p=20 and k=8



(a) Placement Ratio



(b) Residual Resources

Fig. 3. Placement Ratio and Residual Resources of BFD algorithm when p=20 and k=4, 6, 8, 10, *and* 12

unordered requests are allocated to the first level that will fit them. In the BF, the unordered requests are allocated to the best fit location where the total cost is minimised. In FFD, the requests are ordered in decreasing order based on resource consumption, and are allocated to first fit (module types with high resource consumption allocated first). In BFD, the requests are ordered the same way as in FFD, and then are allocated to the best-fit location. In addition, an ILP solution modelled using the Gurobi optimizer [25] is used as baseline for comparison.

C. Performance Assessment

Without loss of generality, we assume traffic is uniformly distributed on all servers and each server represents one tenant. All switch locations have an equal initial capacity of available resources to accommodate security modules. Modules are simulated as different-sized families based on the baseline resources required by each module which is the minimum resources to deploy the modules. The families' sizes are distributed evenly across the location capacity size. For example, for $p = 3$, i.e., the number of families is 3, the sizes of the first family will be a number between 0 and 0.33% of the capacity of each location, the second family size will be between 33% and 66%, and the third family size will be between 66% and 100% of the capacity of each location. The resources required by a request are calculated as shown in Section III.

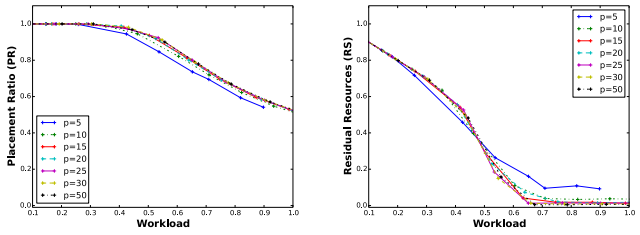
We simulated the *workload* as a ratio of resources requested out of the total resources available to the allocation, and this percentage is distributed evenly over the tenants where a tenant can request modules with total resources less than or equal to its share of the workload. Tenants request modules at random with maximum one from each family. We have evaluated the performance the algorithms over simulated $k \in \{4, 6, 8, 10, 12\}$ fat-tree DC topologies. We consider resources as a one-dimensional vector. All results are computed over an average of 20 runs. We present results of the PR and RS for the three different experimental scenarios.

The first experiment shows the *effect of the workload* on the performance metrics in the case of p=20 families. The results of PR and RS metrics for a k=8 fat-tree are shown in Figure 2. Figure 2(a) shows the *objective function* placement ratio near 1 for low workloads. While beyond 50% workload, PR starts decreasing linearly with the workload, which occurs as a result of the requested resources starting to exceed resource capacity at each location. While all algorithms suffer from such reduction, decreasing-order algorithms (FFD and BFD) show slightly less reduction in PR than BF and FF and overlap with the optimal ILP solution at higher workloads. This can be attributed to that decreasing-based algorithms are allocating larger modules first and save resources to accommodate more requests, and subsequently lead to higher allocation ratio. Figure 2(b) shows the residual resources *objective function* after the allocation is complete and, similar to the PR case, it exhibits a reduction as workload increases where the increasing of requested resources will result in a reduction in spare resources. Furthermore, it shows that when $workload$<0.5 and while PR near 1 for all algorithms, Best-Fit algorithms (BFD and BF) result in more RS than their Fist-Fit counterparts (FFD and FF). Best-fit algorithms utilise resources by selecting locations which cost the least increase in resource consumption, allowing more resources to the allocation process, and leading to an increase in RS that can reach 80% of other algorithms. However, when $workload$>0.5 and PR starts dropping, Best-Fit algorithms still show significant more RS than the rest till all algorithms reach saturation when workload is beyond 0.8 and no RS is left to accommodate any more requests.

The second experiment explores the *scalability* of the BFD allocation algorithm with different network sizes. We simulated the placement problem when p=20 over fat-tree with $k \in \{4, 6, 8, 10, 12\}$. The results of PR and RS metrics for the BFD algorithms are shown in Figure 3. The two metrics degrade as workload increases for all networks, however, this degrading is increasing with network size. This is attributed to the increase in the number of switches that traffic is distributed over in each level which results in more duplication and lower values for PR and RS. However, in the fat-tree architecture, this increase is reduced in higher network sizes as shown in $k = 8, 10$ and 12.

The third experiment illustrates the *scalability* of the algorithms with different numbers of module families. PR and RS are shown in Figure 4 for the BFD algorithm when p=5, 10, 15, 20, 25, 30, 50. PR and RS in Figures 4(a) and 4(b), respectively, show the scale properties of the BFD algorithm where identical results for PR are observed in case of p>10 and the same for RS. However, for smaller numbers of module families, for example, when p=5, PR and RS show a decrease that can be explained by the sizes of modules being widely spread and not fit to all available remaining locations which results in resource fragmentation.

Based on the above results, the BFD algorithm exhibits better resource utilisation. The Best-fit strategy in the BFD algorithm reduces the resources consumed by the placement



(a) Placement Ratio (b) Residual Resources

Fig. 4. Placement Ratio and Residual Resources of BFD algorithm when p=5, 10, 15, 20, 25, 30, 50 and k=8

through selecting allocation with minimum duplication and consequently cause the least increase in resource consumption. This can be observed in the higher percentage of the objective function RS when PR=1. At the same time, the decreasing order strategy of requests before placement results in increasing the amount of allocated resources in case of PR<1. This is demonstrated by the higher value of the second objective PR relative to the other algorithms. We conclude that the BFD algorithm optimises both our objectives. Moreover, it scales well with network sizes and increasing number of modules families.

VII. Conclusions

In this paper, we have studied the problem of placement of security services over multi-tenant, virtualized DC infrastructures. We have classified security functions based on the state keeping requirements of the intrusion detection and mitigation process. We have formulated the placement problem to meet both traffic constraints and computational resource requirements as an instance of the NP-hard VSBPP bin packing problem. We have defined the residual resources and placement ratio as the objective functions for the placement, and we have subsequently formulated the problem in ILP and proposed a modified version of the Best-Fit Decreasing greedy algorithm as a polynomial time solution. We have evaluated our approach against three other algorithms and the ILP solution. The results have shown that BFD significantly outperforms the other models up to 80% while satisfying the corresponding traffic and resource capacity constraints.

Acknowledgment

The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, and EP/P004024/1; by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication and services; and by the EU H2020 GNFUV Project RAWFIE-OC2-EXP-SCI (Grant No. 645220), under the EC FIRE+ initiative.

References

[1] A. Ali, R. Cziva, S. Jouet, and D. Pezaros, "Sdnfv-based ddos detection and remediation in multi-tenant, virtualized infrastructures," 2017.

[2] (2017) Silver peak sd-wan optimizer. [Online]. Available: https://www.silver-peak.com/

[3] (2017) Snort intrusion detection system. [Online]. Available: https://www.snort.org/

[4] (2017) The suricata open source ids, ips, and nsm. [Online]. Available: https://suricata-ids.org/

[5] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella, "Stratos: A network-aware orchestration layer for virtual middleboxes in clouds," *arXiv preprint arXiv:1305.0209*, 2013.

[6] D. A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 51–62.

[7] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015.

[8] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:15.

[9] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *2011 31st International Conference on Distributed Computing Systems*, June 2011, pp. 559–570.

[10] C. Basile, C. Pitscheider, F. Risso, F. Valenza, and M. Vallini, *Towards the Dynamic Provision of Virtualized Security Services*. Cham: Springer International Publishing, 2015, pp. 65–76.

[11] F. Wang, R. Ling, J. Zhu, and D. Li, "Bandwidth guaranteed virtual network function placement and scaling in datacenter networks," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Dec 2015, pp. 1–8.

[12] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[13] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," Internet Requests for Comments, Internet Engineering Task Force, RFC 2992, November 2000.

[14] M. G. Gouda and A. X. Liu, "A model of stateful firewalls and its properties," in *2005 International Conference on Dependable Systems and Networks (DSN'05)*, June 2005, pp. 128–137.

[15] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim, "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, Sept 2006.

[16] P. Bereziski, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, 2015.

[17] I. Karim, Q.-T. Vien, T. A. Le, and G. Mapp, "A comparative experimental design and performance analysis of snort-based intrusion detection system in practical computer networks," *Computers*, vol. 6, no. 1, 2017.

[18] K. Salah and A. Kahtani, "Performance evaluation comparison of snort NIDS under linux and windows server," *Journal of Network and Computer Applications*, vol. 33, no. 1, pp. 6 – 15, 2010.

[19] C. Sanders and J. Smith, *Applied Network Security Monitoring: Collection, Detection, and Analysis*, 1st ed. Syngress Publishing, 2013.

[20] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, 2012, special Section: Energy efficiency in large-scale distributed systems.

[21] D. S. Johnson, *Bin Packing*. New York, NY: Springer New York, 2016.

[22] M. Yue, "A simple proof of the inequality FFD (L) $\leq$ 11/9 opt (L) + 1, $\forall$L for the FFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, no. 4, pp. 321–331, 1991.

[23] K. Fleszar and C. Charalambous, "Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem," *European Journal of Operational Research*, vol. 210, no. 2, 2011.

[24] W. Leinberger, G. Karypis, and V. Kumar, "Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints," in *Parallel Processing, 1999. Proceedings. 1999 International Conference on*. IEEE, 1999, pp. 404–412.

[25] (2017) gurobi optimizer. [Online]. Available: http://www.gurobi.com/index/