Resource-Aware Placement of Softwarised Security Services in Cloud 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—**Virtualizing middleboxes as software for Cloud tenants can eliminate the monolithic processing and static deployment of legacy middleboxes and provide an efficient provisioning for security services. However, inefficient managing of the virtualized security services can reduce the gains of Cloud deployment. We propose a resources-efficient placement of the security functions in the infrastructure of a three-tier Cloud DC by modifying the Best-Fit Decreasing algorithm to solve the problem while satisfying the placement resources and traffic constraints.**

Keywords—Security services in Cloud Data Centers, Softwarised security, virtualised Security placement, Resource-aware allocation

I. Introduction

A typical security system for a Data Center (DC) usually consists of a combination of bespoke hardware-based (middleboxes) e.g., Firewalls and Intrusion Detection and/or Prevention Systems (IDS/IPS), and Deep Packet Inspection (DPI) appliances, deployed in fixed locations [1] which restricts the ability of the infrastructure to react rapidly to changes or respond to attacks [2]. These middleboxes are expensive high-performance vendor-specific appliances with limit extensible functionality and results in vendor lock-in. As ICT is increasingly outsourced to the Cloud, middleboxes start the transition with the emerge of increasing number of virtualized network appliances such as WAN optimizer [3] and IDPS [4], [5], and in-the-cloud network services offered by Cloud Services Providers CSP or third party companies [6]. Virtualized security services offer the same protection that hardware-based systems provide combined by the high performance and efficiency of Cloud services. For instance, virtualization of security will remove vendor lock-in and increase the rapid developing of more effective security solutions, resources provisioning in virtualized environment allows flexible scaling and increases the resource usage efficiency, the flexibility of deployment of virtualized services will offer a rapid response that raises the efficiency of the system to handle attacks and changes in traffic and infrastructure.

However, the lack of tools to efficiently manage in-Cloud middleboxes can introduce performance degradation and/or reduce the benefits of Cloud deployment [6]. For instance, the complex nature of traffic workload processing in middleboxes complicate the process of allocating and scaling resources in the Cloud. Furthermore, inefficient network placement or routing can affect network performance create bottlenecks and cause unnecessary scaling.

Research Challenge: We address the problem of allocating virtualized security in Cloud DC by introducing a *resources-aware placement methodology* to solve the placement problem of security modules in virtualized environment, while maintaining an efficient management of resources and satisfy the traffic constraint of modules' allocation. Our contribution is i) A classification of the security functions based on processed traffic constraints ii) Defining the placement problem of the stateless security class and its objective which based on maximising residual resources iv) A modified version of the Best-Fit Decreasing (BFD) algorithm to solve the problem and a comparative assessment with the Best-fit (BF), First-Fit Decreasing (FFD) and First-Fit (FF) algorithms.

The paper is structured as follows: Section II discusses related work while Section III introduces the placement problem of security functions and in Section IV, we adopt a modified version of the BFD and provide a performance and comparative assessment in Section V. Section VI concludes the paper.

II. Related work

With the emerging of virtualized middleboxes as a solution to mitigate the problem of hardware-middleboxes and with enterprises moving to the Cloud, Network Function Virtualization (NFV) technology proposed as a framework for implementing virtualized network function in virtualized environment [7].

Although much research attention has been drawn to the different aspects of managing dynamic in-network services in virtualized environments, yet only a few consider the distinct characteristic of the security functions in the process of managing services. The authors in [1] address the allocation of security services in virtualized environments and discuss their challenges. They model the allocation problem in ISP networks to minimise the cost of network operators as a mixed-integer linear programming (MILP) problem but no results are reported. Gamber et al. [6] suggest an orchestration layer for virtualized middleboxes that is network-aware and enforces network policies through chaining. It uses horizontal scaling to leverage performance which means dividing traffic flow over two paths to solve resource bottlenecks and SLA violations. However, it only considers functions that process traffic on the flow level.

What distinct our work from previous research in dynamic management of virtualized services is that (i) it addresses the unique characteristic of security services such as traffic constraints which impose a restriction on allocation and (ii) how to achieve efficient management of virtualized environment while (iii) maintaining the satisfaction of the security requirements.

III. Security Management

As users run different applications in Clouds DCs, they require different levels of security per application. For example, a typical web server may require protection against HTTP flood attacks and SQL injection, while critical servers may require a combination of solutions to guarantee availability and data integrity such as firewall, IDS and/or DPI. We focus on managing security services over multi-tenant Cloud DCs. Tenants can request security services which are offered as a set of deployed modules from a pool of security modules, e.g, firewalls, IDS/IPS and DPI. They can be deployed in the DC where they can process traffic destined to 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 management of the DC resources. As the security services are offered per tenant basis, it will provide the customization to fulfil tenants need for different security services and level of protection.

A. Architecture

Most Cloud DCs follow the three-tier network architecture as a multi-rooted tree with three layers of switches (TOR, aggregation, and core) [8] such as the k=4 fat-tree topology shown in Figure 1. We assume routing is flow-based Equal Cost Multiple Path (ECMP) [9] The security modules are deployed at points collocated with switches in all layers. Traffic is steered from switches to the security modules and back as shown in Figure 1. This approach is opposite of how VNF commonly deployed in rack hosts, However, our approach reduce the detour length that traffic have to go through to pass the virtualized middlebox and consequently reduce traffic delay impose by the security function deployment. The security function abstraction can be implemented as a distinct namespace within a software switch or on a separate, virtualized commodity x86 architecture physically connected to a traffic-forwarding switch.

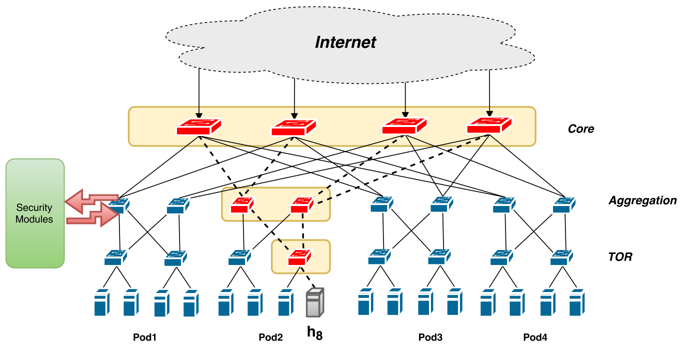


Fig. 1. Fat Tree Cloud Data Center size k=4

B. Placement of Security Modules Problem

The placement problem can be defined as selecting a location to fulfil a tenant's requests for a security module. The placement must guarantee that the module is working correctly by satisfying the module requirements and constraints such as the required traffic is passed to the module, enough computing resources are available at the location for the module to process the traffic. The cost of allocating the modules varies based on the location point(s) selected, e.g., one allocation may require traffic to be steered to a non-shortest path while another may require multiple instances to be deployed. Therefore, when many allocations satisfy a module's allocation constraint, the placement methodology is responsible for selecting one of them based on the certain objectives which make it a resource allocation problem. While each location collocated with a switch has limited resources and the cost is different from one to another, the objective can combine computing resources, power consumption and/or communication cost.

1) Traffic Constraints: Security modules process traffic in different ways depending on how threats are being detected. Traffic can be processed on a per-packet, per-flow, or per-flow-aggregate basis where each level can provide protection of different types of the security vulnerabilities. For example, per-packet or per-flow processing cannot detect threats that span multiple flows such as (e.g. DDoS flooding, worm spreading, probes). Thus, each module requires different granularity of traffic processing and the distribution of traffic over the DC will constrain the allocation of security modules to locations where this granularity can be satisfied. for instance, in a three-tier architecture, all traffic destined to a server is passing by a ToR switch. While in higher layers of the architecture traffic destined to a host traverses multiple (parent) switches and only a fraction of the flows is observed at each switch. Based on that we have produced a set of equivalence classes of security functions based on the detection method of different attacks and subsequently the granularity of the traffic being processed.

Stateless (packet-based) class: Represents modules that process traffic at the individual flow or packet level. Detection or mitigation decisions are made based on the state of a single packet or flow. Therefore, replicated instances of this class can be distributed across multiple network locations. This can be achieved by per-flow routing and by placing duplicate detection modules at diverse network locations where traffic matching a certain specification is being split due to ECMP routing.

Stateful (flow-based) class: Represent security modules that process traffic to extract anomalies based on the deviations from normal behaviour which require processing traffic on coarser granularity than a single flow. Steering all the intended monitored flows to one instance of this type results in an accurate construction of the behaviour model. However, some modules may be able to periodically share meta-information between distributed instances that work together which can be an alternative to steering the entire flows to one instance.

Network-Wide class: Represent modules detecting network-wide threats such as probes and worms which cause distribution changes in traffic features that can be observed at high aggregation levels [10]. Therefore, this class monitors aggregate packets/flows and consequently they can be allocated to centralised/core locations where aggregate traffic to multiple destinations can be processed.

2) Resource Constraints: While each location has limited computing resources defined by a vector (CPU cores, Memory, I/O bandwidth and Storage, etc.), each request from a tenant must be associated with a similar vector of the estimated resource required which is the resources required for the modules to process the required amount of traffic. The chosen allocation must then satisfy the resource requirements of the request where the resources available at the chosen location(s) must be greater than or equal to the resources requested, and only the allocations that satisfy the traffic and resource constraints are considered in the selection process. Although the resources required by a security module depend on many factors such as (hardware, configuration, platform, the rule set, traffic rate, etc), traffic intensity is the main factor to consider under the same platform [11]–[16]. Thus, each security module available to be requested by tenants will be associated with a resources vector that represents i) *baseline resources* which is the amount of resources required for initial deployment of the module and ii) *traffic resources* which is resources per traffic unit associated with each traffic type such as (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, each request will be associated with estimated rate(s) of each traffic type(s) to be processed for the requested tenant.

C. Placement Objective

The placement of security modules, which are requested by tenants over a Cloud DC in distributed locations collocated with network switches, casts as a resource allocation problem. We introduce the initial placement problem of the stateless equivalence class. In the fat-tree k=4 in Figure 1, switches and links carrying the traffic for host h_8 (in gray) are shown in dashed lines, and a security module requested for tenants residing on this host will have three locations (shown in yellow blocks) satisfying the traffic constraints to be considered: i) single instance of the security function is deployed at the ToR switch of this server, covering all traffic destined to/originating from that host ii) two instances of the function are deployed at the aggregation switches routing traffic to/from this host iii) four instances of the function are deployed in the four core layer switches.

Since there will be more than one allocation that will satisfy the resource and traffic constraints of allocating a request, one that optimises the placement objective will be selected. For the stateless equivalent class of modules, we design the placement objective to achieve i) *an efficient management of resources* while keeping maximum ii) *request satisfaction ratio*. For efficient managements of resources, maximising the *residual resources* of the framework after placement will save resources to accommodate more requests in the future. While to achieve maximum request satisfaction ratio, a simple minimizing to the number of unsatisfied requests can be used, however, minimizing *resources of the unsatisfied requests* will leverage the utilization of resources in case of requests are more than the system can accommodate and satisfying requests with more resources will increase profits where Cloud services are offered on pay-per-use basis.

IV. Placement Methodology

The placement of security modules is an instance of a variable cost – variable size bin packing problem (VSBPP) [17] 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. As bin-packing problems are shown to be NP-hard [18], many gready algorithms proposed to solve it. Best-Fit Decreasing (BFD) are the most widely applied. In BFD, items are sorted in decreasing order and the sorted items are allocated in the best location to fit them as a minimum empty space will be left after the allocation [19]. The BFD algorithm uses the best fit strategy to utilise resource consumption. We have adopted a modified version of BFD called *power-aware best fit decreasing* that is used to reduce power consumption in VM placement by allocating VM to the allocation that causes the least increase in power consumption [17]. We replace power consumption of VM placement by resources consumption as the cost function in security modules placement.

The proposed BFD resource-aware allocation algorithm sorts requests and then allocates each request to the *Bets-fit (BF)* location. Firstly, it sorts requests by the amount of required resources to deploy one instance for each in a decreasing order. Then it calculates the cost of allocating the request to each level, the algorithm only consider the allocations that have enough resources to accommodate the request, and finally selects the *BF* allocation by selecting level that causes the least increase in total resource consumption (min cost) as illustrated by the following algorithm:

BFD Placement Algorithm

```
Input: $H, M, S, Q, L$
Output: $A, Unsatisfied\_Requests$
1: $Sorted\_Q \leftarrow sort(Q)$ // sort by requested resources
2: for all $h, m \in Sorted\_Q$ do
3:     $min\_cost \leftarrow MAX$
4:     $level\_found \leftarrow 0$
5:     for all $l \in L$ do
6:         if $(A.Enough\_Resources(h, m, l) == TRUE)$ then
7:             $cost \leftarrow calculate\_cost(A, h, m, l)$
8:             if $(cost < min\_cost)$ then
9:                 $level\_found \leftarrow l$
10:                $min\_cost \leftarrow cost$
11:            end if
12:        end if
13:        if $(level\_found \neq 0)$ then
14:            $A \leftarrow A.Allocate(h, m, l)$
15:        else
16:            $Unsatisfied\_Requests \leftarrow h, m$
17:        end if
18:    end for
19: end for
20: return $A$, $Unsatisfied\_Requests$
```

Note: The function $sort(Q)$ sorts the request list Q by the decreasing order of resources required to satisfy each request. The function $Enough\_Resources(h, m, l)$ ensures the resources required in level l is enough to accommodate request (h, m). The function $calculate\_cost(A, h, m, l)$ calculates the resources cost of allocations A after allocating request (h, m) to level l.

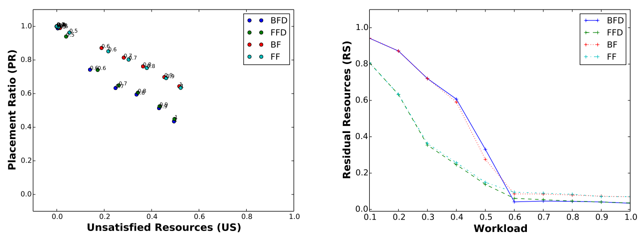
V. Performance Evaluation

A. Performance metrics

We introduce three metrics for evaluating our placement methodology: i) *Placement Ratio* (PR) achieved by the allocation algorithms, which represents the ratio of satisfied requests out of the total number of requests. For example, PR with the value of 1 will indicate satisfying all requests by finding the allocation that satisfies their constraints, while a less than 1 PR indicates a failure ratio where not all requests been allocated ii) *Residual Resources* (RS) of the network, which is the ratio of the spare resources (after placement) to the total amount of resources available to the allocation and calculated by adding the residual resources at each location after placement. RS is a normalisation of our first objective presented in Section III and will indicate the efficiency of resources usage as the ratio of saved resources after allocation in case of PR is close to 1 iii) *Unsatisfied Resources* (US) which equals the sum of resources of the unsatisfied requests out of the total resources requested by modules. US indicates the utilisation level of resources in case of PR is less than 1, where an efficient algorithm will result in less amount of unsatisfied resources. US is normalization for our second objective.

B. Models Comparison & Performance Assessment

We compare the proposed BFD algorithm with three resource allocation algorithms: First-Fit (FF), Best-Fit (BF), and First-Fit Decreasing (FFD). Specifically, in the FF, the unordered requests are allocated to the first level that fit them. In the BF, the unordered requests are allocated to the best fit location, where the total cost is minimised. In the FFD, the requests are ordered in a decreasing order based on the resource consumption and are allocated to first fit (module types with high resource consumption allocated first). In the BFD, the requests are ordered with the same way as in the FFD, and then are allocated to the best-fit location.



(a) Unsatisfied Resources (b) Residual Resources

Fig. 2. Residual Resources and Unsatisfied Resources of BFD, FFD, BF and FF algorithms when p=10 and k=6

We assume traffic is uniformly distributed on all servers and each server represents one tenant. All switch locations have an equal initial capacity of resources for the allocation of modules. Modules are simulated as different-sized families based on the baseline resources The families' sizes are distributed evenly across the location capacity size. The resources required by a request are calculated as described in Section III with the traffic part equals 20% of the baseline part. We simulated the *Workload* as a ratio of resource requested out of the total resources available to the allocation, and that these percentage is distributed evenly on the tenants where a tenant can request modules with total resources less than or equal its share of the workload. Tenants request modules in random with maximum one of each family. For simplicity, we consider resources as a one-dimensional vector. All results are computed over an average of 50 runs.

The result of PR versus US of BFD, FFD, BF and FF in Fat tree with k=6 and number of families p=10 at different workloads are shown in Figure 2(a). An efficient algorithm will result in high placement ratio and low unsatisfied resources. Figure 2(a) depict an efficient performance for the four algorithms in low workload, however, when workload is more than 0.5 the decreasing order algorithms exhibits less PR than other algorithms but less US where allocating different sizes of modules may result in not allocating more number of modules but resulting in less total of the unsatisfied percentage of requested resources. In the other hand, we shows the *effect of the workload* as on the performance metric RS in Figure 2(b). RS exhibits a reduction as workload increases where the increasing of requested resources will result in a reduction in the spare resources. When the workload is less than 0.5, the BF algorithms have more RS than the FF algorithms, where BF utilizes resources by selecting a location that causes the least increase in resources, allowing more resources to the allocation process and leading to increasing in RS that can reach 30% of other algorithms. While when workload is more than 0.5 and PR start dropping, decreasing based algorithms BFD and FFD exhibit less residual resources than non-decreasing order algorithms in spite of them having less PR as they allocate fewer but larger requests and end up with less remaining resources. Thus, the BFD algorithm demonstrates better utilisation of resources and satisfies both of our objectives. Our future-work include experiment to find the scalability of the algorithm with network size and increasing number of modules families.

VI. Conclusions

We study the problem of efficiently allocating modularized security services in Cloud DC to protect tenants against network threat such as DDoS, Worms and Probes. We provide a classification of security modules based on traffic processing in the detection process. We define the residual resources and the resources of the unsatisfied requests after the allocation as the placement objective and proposed a modified version of the Best Fit Decreasing algorithm for the solution. We evaluate our approach with three other algorithms and demonstrate that BFD significantly increases the residual resources objective, outperforming the other algorithms up to 30% while satisfying the constraints of the problem.

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, EP/P004024/1, and EP/L005255/1, and by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication services protecting end-user applications from disaster-based failures.

References

[1] 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.

[2] 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. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402966

[3] (2017) Silver peak sd-wan optimizer. [Online]. Available: https://www.silver-peak.com/

[4] (2017) Snort intrusion detection system. [Online]. Available: https://www.snort.org/

[5] (2017) The suricata open source ids, ips, and nsm. [Online]. Available: https://suricata-ids.org/

[6] 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.

[7] R. Cziva, S. Jouet, and D. P. Pezaros, "Gnfc: Towards network function cloudification," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov 2015, pp. 142–148.

[8] 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.

[9] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," Internet Requests for Comments, Internet Engineering Task Force, RFC 2992, November 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2992.txt

[10] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, Aug. 2005. [Online]. Available: http://doi.acm.org/10.1145/1090191.1080118

[11] J. S. White, T. Fitzsimmons, and J. N. Matthews, "Quantitative analysis of intrusion detection systems: Snort and suricata," pp. 875 704–875 704–12, 2013. [Online]. Available: http://dx.doi.org/10.1117/12.2015616

[12] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804509001040

[13] W. Bul'ajoul, A. James, and M. Pannu, "Improving network intrusion detection system performance through quality of service configuration and parallel technology," *Journal of Computer and System Sciences*, vol. 81, no. 6, pp. 981 – 999, 2015, special Issue on Optimisation, Security, Privacy and Trust in E-business Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000014001767

[14] 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. [Online]. Available: http://www.mdpi.com/2073-431X/6/1/6

[15] M. Fisk and G. Varghese, "Applying fast string matching to intrusion detection," DTIC Document, Tech. Rep., 2002.

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

[17] 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, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000689

[18] D. S. Johnson, *Bin Packing*. New York, NY: Springer New York, 2016, pp. 207–211.

[19] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various vm placement strategies in cloudsim," *Journal of Cloud Computing*, vol. 4, no. 1, p. 20, 2015. [Online]. Available: http://dx.doi.org/10.1186/s13677-015-0045-5