New Approach for Virtual Firewalls Orchestration in Edge-Cloud Continuum Infrastructure

1st Bartosz Kopeć-Persiński
Institute of Telecommunications and Cybersecurity
Warsaw University of Technology
Warsaw, Poland
bartosz.kopec-persinski@pw.edu.pl

2nd Andrzej Bęben
Institute of Telecommunications and Cybersecurity
Warsaw University of Technology
Warsaw, Poland
andrzej.beben@pw.edu.pl

***Abstract*—The evolution of the 5G systems into a virtualized infrastructure enables the deployment of virtual firewalls (vFW) to protect the system from undesirable traffic and other threats. The current orchestration algorithms usually provision vFW instances at network edges. We propose a new virtual Firewall Allocation and Traffic Distribution (vFATD) approach that flexibly allocates and scales vFW instances over the distributed edge-cloud continuum infrastructure to reduce the overall vFW system costs. We designed and evaluated the MILP-based exact and k-center-based heuristic orchestration algorithms. The experiments performed using actual data traffic from a mobile network operator confirm that the proposed vFATD algorithms are reasonable and may bring network providers significant gains.**

***Index Terms*—firewalls, orchestration, NFV, ECC, MILP, 5G/6G**

I. Introduction

Network security is one of the primary concerns of network providers who commonly use Intrusion Detection Systems (IDS) supported by threat intelligence and firewalls [1] to protect their infrastructure from undesirable traffic and other threats. Firewalls are typically deployed at the Points of Presence (PoP) to secure domain ingresses, aggregation points in the access layer, and inter-domain links, where traffic enters the domain. They monitor packet streams and filter out malicious packets based on source and destination IP addresses, port numbers, traffic profiling, or other complex signatures. Providers define their security policies by balancing protection costs with the level of security assurance.

Researchers continuously develop new detection and prevention methods to address emerging threat vectors. Techniques such as Deep Packet Inspection (DPI), threat intelligence, and Artificial Intelligence (AI)-based methods offer improved detection capabilities [2]. Advanced threat detection and intelligent methods are usually increasingly complex and computing-demanding. Therefore, network providers must continuously monitor the firewall load and upgrade it to more powerful platforms or deploy new devices to maintain network security. This process can be time-consuming and costly, particularly when network traffic changes.

This research was partially supported by the Ministry of Science and Higher Education, Poland, grant number KPOD.01.18-IW.03-0009/24, on "National Laboratory for Advanced 5G Research" financed by the European Union NextGenerationEU under National Recovery Plan.

The evolution of 5G and 6G networks into Virtual Network Function (VNF) or Cloud Native Function (CNF) architectures [3] enables the deployment of software-based Virtual Firewall (vFW) systems. Unlike traditional hardware firewalls, we deploy vFW appliances as containers or virtual machines of similar functionality. The main advantage of the vFW systems comes from the automation of life-cycle management and flexible deployment of vFW instances over the future Edge-Cloud Continuum (ECC) infrastructure [4]. The orchestrator adjusts the vFW system performance by horizontally scaling the number of vFW instances following daily and long-term traffic changes. This flexibility enables more efficient resource utilization, as other functions can use computing resources freed up during off-peak hours. Moreover, vFWs can be easily upgraded with new functionality, enabling fast reactions to emerging threats.

The vFW system requires designing an efficient orchestration algorithm that optimizes the allocation of vFW instances. Unfortunately, the standard orchestration algorithms, e.g., Horizontal Pod Autoscaling (HPA) in Kubernetes, focus mainly on CPU or memory utilization, which makes them ineffective in vFWs orchestration.

Our main contribution is the design of novel orchestration algorithms for vFW instance allocation, horizontal scaling, and traffic distribution within the network based on actual traffic demands. The proposed algorithms take into account: a) daily changes in traffic demands, b) the required computing resources of vFW instances and hardware accelerators for DPI or AI processes, and c) available network resources and data transfer costs. In the paper, (i) we formulate the virtual Firewall Allocation and Traffic Distribution (vFATD) problem as a Mixed Integer Linear Problem (MILP) [5], (ii) we propose both exact and heuristic algorithms to orchestrate vFW instances in the ECC infrastructure, and (iii) conduct experiments to evaluate the proposed vFATD algorithms using real traffic data collected from a mobile network.

The organization of this paper is as follows. Section II defines the vFW orchestration problem and analyzes the state-of-the-art. Section III presents the proposed orchestration approach with exact and heuristic algorithms. Section IV describes the experiments and the results obtained. Finally, Section V summarizes the paper and outlines further work.

978-3-903176-75-1 ©2025 IFIP

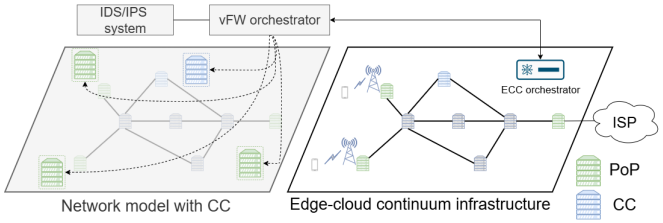


Fig. 1. Exemplary ECC infrastructure with PoPs and CC nodes

II. Problem statement & state of the art

Let us consider an exemplary 5G network with the vFW system deployed over the future ECC infrastructure as presented in Fig. 1. It comprises several Computing Centers (CC) and Point-of-Presence (PoP) connected by an underlying, over-provisioned transport infrastructure. PoPs cover gNB stations, provider edge routers, or ISP border routers, where traffic enters the domain. The computing centers correspond to all ECC infrastructure sites providing computing resources, such as far, near, edge, regional, and primary data centers. They differ in location, available resources, and costs, so the provider must choose the best site to deploy virtual appliances. The allocation of vFW instances closer to users at the network edges reduces transmission costs. Still, it may increase allocation costs because of the low aggregation level and higher infrastructure costs. On the other hand, the allocation of vFW instances in distant data centers reduces allocation costs but increases transmission costs. The orchestration algorithm deals with the discussed trade-off and should provide the optimum number of allocated vFW instances, their location, and traffic distribution between POPs and allocated vFW instances.

We formulate the presented problem as the Virtual Firewall Allocation and Traffic Distribution (vFATD) problem. The main goal is to find the least number of vFW instances, their optimal allocation, and the preferred traffic distribution in the ECC infrastructure to protect the 5G network and minimize the operational costs of the vFW system. The proposed algorithms focus on the trade-off between data transmission costs vs. costs of allocated computing resources. We leave other issues related to, e.g., the provider's lost image and customer trust due to service unavailability, as an open problem for further studies.

1) Literature review: The VNF orchestration has recently gained much attention. Most publications focus mainly on the VNF allocation or the automation of the deployment process while neglecting the proper traffic distribution problem.

There are also general VNF orchestration approaches, such as [6], focusing on the orchestration process of a generic VNF, including the optimization problem of traffic distribution in a given network graph. The authors propose a machine learning method to develop a real-time orchestrator, which uses offered traffic for a service s as an input argument. This algorithm returns what VNF instances should be activated in the time T at node N.

Each VNF has specific needs, so general orchestration methods may not be optimal. Therefore, we took a closer look at vFW orchestration and traffic distribution.

The authors of [7], [8] focus on an automatic optimal vFW allocation and configuration problem. They aim to minimize two factors: the number of vFW instances and the number of configured firewall policies on each. They formulate this problem as Maximum Satisfiability Modulo Theories (MaxSMT) and define two types of constraints: the hard one, related to the network graph and firewall policies, and the soft one, where vFW allocation and vFW configuration. To simplify the solution, the authors consider the probability of each vFW transmitting and receiving packets. Due to this, there is no need to model specific firewall functionality. In [8], the authors present the developed VEREFOO framework, which uses the MaxSMT formulation to solve the problem of the proposed network.

Another point of view had been presented at [9], where the SDN HyperFlex hypervisor was introduced. The orchestration algorithm uses the SDN north-bound interface to enforce the optimal allocation of vFW and the path between the network nodes. This framework roughly models QoS level. Based on changing latency on links, the algorithm allocates instances at possible nodes and sends new routes into OpenVSwitch flow tables. A similar problem of VNF placement and routing was introduced in [10], where the formal MILP formulation of the problem was discussed.

In [11], the authors deal with the problem of dynamic control of routing and VNF orchestration in the SDN network to prevent service failure due to power loss. It is achieved by establishing multiple power-disjoint communication routes between nodes. They propose an optimal model to maximize the ratio of power-disjoint routes to VNF orchestration cost.

The main conclusion from the analyzed papers is that while there are methods for orchestrating VNFs, including virtual firewalls (vFW), the existing solutions are either too general or focus primarily on simplified QoS indicators rather than actual traffic volume and network resilience. The primary motivation for this paper is to design and evaluate an effective orchestration algorithm for a virtual firewall system that can adapt its performance to changing traffic conditions. The proposed solution aims to enhance the resilience of the operator's network by enabling traffic distribution across multiple computing centers.

III. Proposed orchestration method

This section presents the considered vFW orchestration system and describes the designed exact and heuristic vFATD algorithms.

A. The orchestration system

In Fig. 2, we present the proposed vFW orchestrator. It manages the life cycle of vFW instances by performing allocation, instantiation, scaling in/out, rolling updates, and termination actions. Moreover, it schedules the management events and adjusts the traffic distribution between PoPs and allocated vFW instances.

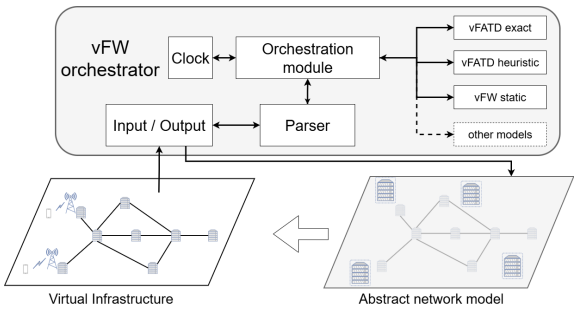


Fig. 2. The vFW orchestrator architecture

The vFW orchestrator handles management events triggered in discrete time slots T_n. At the beginning and after each topology change, the orchestrator activates the *Input / Output module* to collect data about the infrastructure topology, the available computing resources, and transmission characteristics between nodes. It uses that data to create an abstract infrastructure model as a graph with vertices representing computing resources and edges characterizing connections. Then, the vFW orchestrator periodically schedules orchestration events. Each event triggers the orchestrator to collect information about current traffic demands and actual traffic distribution. These data pass through the *Parser* module to one of the *Decision algorithms*: exact vFATD, heuristic vFATD, static vFW, or other. The algorithm analyzes traffic demands and the current allocation of vFW instances in the abstract infrastructure model. The algorithm returns a new vFW allocation vector and recommended flow distribution table as the output. The vFW allocation vector describes how many vFW instances should run at each node. The flow distribution table is a three-dimensional array that determines the number of flows incoming in PoP $x \in X$ and belonging to a given class $c \in C$ that will be analyzed by vFW instances running in CC $y \in Y$.

B. The vFATD exact algorithm

The vFATD exact algorithm uses an MILP model to derive the preferred solution. We have implemented the model in AMPL and solved it using the CPLEX solver. The model derives the preferred solution based on the abstract infrastructure model, vFW descriptor, and current demand description, considering the boundary condition, constraints, and the assumed objective function corresponding to vFATD goals.

1) The model of infrastructure: We model the virtual infrastructure as a directed graph $G(N, E)$, where N represents the set of nodes, while E corresponds to connections between them. Each node $n \in N$ belongs to one of the following types: PoP, compute center, or transit node. They play a specific role in the vFW system and differ in the available computing resources, e.g., CPU or RAM. In particular, we assume that traffic enters the system only at PoP nodes. They provide limited computing resources, allowing running just a few vFW instances, while compute center can handle many vFW instances. Transit nodes provide connectivity and no computing resources. We assume that a single vFW instance may process a given traffic volume independently of the number of flows or packet size. If more processing power is needed, more vFW instances should be allocated because the vFW system must analyze every transferred packet. For the sake of simplicity, we assume that vFW works without additional security features, i.e., DPI, IPS, IDS, etc. The algorithm decides whether it is better to run a new vFW instance or to transfer the flow to another node running vFW instances, depending on the overall vFW system costs. We assume virtual infrastructure provides connectivity, and a path $p \in P_{xy}$ exists between each pair of nodes.

2) vFW descriptor: The processing power (capacity) of a single vFW instance has been modeled based on actual commercial appliances. We analyze Cisco ASA, and Fortinet Fortigate appliances to assess the vFW capacity η. A single vFW instance requires 4 CPU cores and 6 GB RAM. It can perform stateful inspection of up to 50 Gbps of aggregated traffic when additional security techniques, such as IPsec, DPI, IPS, etc, are disabled.

3) Demands description: The proposed model uses volumetric data that enters the network separated into flow classes. We measured data on the edges of a Polish mobile network. For simplicity, we split traffic just into ten flow classes: C1, C3, ..., C10 - where class C1 represents 10% of the lower-throughput flows, class C2 represents 10% of the higher-throughput flows, and so on. Finally, class C10 collects the highest-throughput flows. Afterward, the vFATD exact algorithm uses the number of flows within each class, assuming the same rate for flows in a given class. We may increase the number of flow classes to get more accurate traffic distribution at the cost of increased model complexity.

TABLE I
PARAMETERS USED IN THE EXACT VFATD ALGORITHM

Constants		
δ_{xc} - input traffic (traffic demands)	P_{xy} - path between two nodes	i - vCPU used by vFW instance
i_y^{max} - no. of available vCPU	j - vRAM used by vFW instance	j_y^{max} - available vRAM
η - processing capacity of vFW instance	ξ - allocation cost of vFW instance	ζ_e - transmission cost
κ - costs of lost traffic	β_c - bit rate in class c	l_y^{max} - no. of licenses

4) Notations: We use the following notations:

- $x \in X$ indicates PoPs with the ingress traffic,
- $y \in Y$ points out nodes where we allocate vFW instances,
- $c \in C$ denotes flow classes,
- $e \in E$ indexes edges in the network graph.

5) Variables: We define the following variables:

- n_y - number of vFW instances running at node y,
- F_{xyc} - traffic distribution (flow table).

6) Constraints and boundary conditions: Constraint (1) limits the allocated flows to the traffic demand incoming in a given PoP. Constraints (2) and (3) ensure that our algorithm can assign a new vFW instance only if it does not exceed the computing resources available on the considered node.

Constraint (4) assures the availability of hardware accelerators for the vFW instances. The last constraint (5) ensures that the traffic flows handled by the allocated vFW do not exceed their overall processing capacity.

$$\sum_{y} F_{xyc} \leq \delta_{xc}, \quad \forall x \in X, \forall c \in C \tag{1}$$

$$n_y \cdot i_y \leq i_y^{max}, \quad \forall y \in Y \tag{2}$$

$$n_y \cdot j_y \leq j_y^{max}, \quad \forall y \in Y \tag{3}$$

$$n_y \leq l_y^{max}, \quad \forall y \in Y \tag{4}$$

$$\sum_{xc} F_{xyc} \cdot \beta_c \leq n_y \cdot \eta, \quad \forall_y \in Y \tag{5}$$

7) Objective function: The objective function in (6) covers i) the allocation cost of each vFW (cost of running each single virtual appliance at CC resources), ii) the transmission cost of the traffic between nodes in the infrastructure, iii) the cost of the lost, non-analyzed traffic, and iv) the cost of rapid changes in vFW allocation.

$$\begin{aligned} minimize: f(y,n) = \sum_{y} n_y \cdot \xi \\ + \sum_{c} [\beta_c \cdot \sum_{e} [\zeta_e \cdot (\sum_{x} \sum_{y} F_{xyc} | e \in E)]] \\ + \kappa \cdot \sum_{xc} \beta_c (\delta_{xc} - \sum_{y} F_{xyc}) + \sum_{x} abs(n_y^{old} - n_y) \end{aligned} \tag{6}$$

After solving the MILP model, the algorithm finds the optimal vFW allocation (including minimization of the number of vFW instances) described by the vector n_y and finds the optimal routing of flows F_{xyc} between the CCs.

C. The vFATD heuristic algorithm

The heuristic algorithm solves the vFATD problem based on the vertex k-center problem [12]. We have implemented the GOn algorithm [13] (greedy k-center approximation) with the possibility of setting the weights of each vertex. Then, we use it to approximate the location of the CCs. The heuristic algorithm defines the network graph, calculates each node's weight, determines the k-center locations, assigns the flow table, and calculates the required number of vFW instances.

1) Calculation of vertices weight: Weights are used in the k-center algorithm to select compute centers closer to the PoP nodes with the higher flow density. An algorithm calculates weights as follows. For each PoP, the algorithm sums the product of the ingress flow number and the class cost. Then, each value is normalized to the percentage share of all ingress traffic. As a result, we obtain a vector with percentage weights representing the participation of flow numbers at each PoP in the network.

2) GOn k-center algorithm: GOn algorithm as input requires the distance matrix between vertices, the weight vector, the number of CCs, and parameter alpha, which determines the impact of weights on the result. The algorithm randomly selects the first k-center from the PoP nodes. The algorithm adds it to the k-center list. Then, the algorithm calculates the distance matrix using weights and selects the farthest vertex from each k-center on the current list as the next compute center. The process repeats k times until the algorithm selects all k-centers.

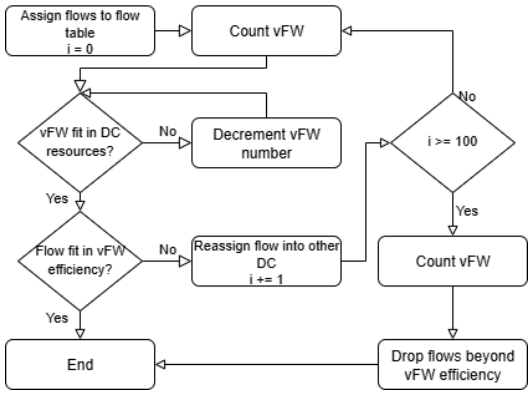


Fig. 3. The heuristic vFATD algorithm

3) Flow table & allocation vector: Flow assignment to the compute node and the vFW instance allocation algorithm are presented in Fig. 3. The algorithm works iteratively. Initially, it allocates flow distribution based on a random choice of the CC with probability given as the inverse of distances between PoPs and CCs. Then, it starts a loop with $i = 100$ iterations: the algorithm calculates the number of required vFW instances for each compute center based on assigned traffic and then checks if the resources consumed by the vFW instances do not exceed the CC resources; then, the algorithm checks whether the flow's traffic does not exceed the summarized vFW instances capacity at each CC. If so, the algorithm randomly assigns the excess flows to another compute center and starts the calculation of the required vFW instances again. The loop stops after 100 iterations, recalculates the number of vFW instances for the last time, and leaves the fraction of traffic (lost traffic) that does not fit the CC capacity without inspection.

IV. Experiments & results

Our experiments focus on the performance evaluation of the proposed vFATD approach. We compare its effectiveness to current providers' practices, where vFWs are usually placed in the PoPs. As the reference, we consider: i) vFW static allocation (vFSA), where vFWs are pre-provisioned for the busy hour, and ii) vFW allocation (vFA), where vFWs are horizontally scaled in/out following daily traffic changes. First, we analyze the benefits of the traffic distribution integrated into the vFATD algorithm. Then, we focus on resource and cost optimization, showing the vFATD advantages. Finally, we evaluate the accuracy vs. complexity of the proposed exact and heuristic vFATD algorithms.

A. Assumptions

We perform our experiments using a representative single-domain network, illustrated in Fig. 4, which reflects the ECC infrastructure of a Polish mobile network. Each node represents a Computing Center (CC), where the provider deploys virtual firewalls (vFWs) and other 5G/6G network functions or

transit nodes, used only for data transmission. These CCs are located in major cities and act as aggregation points for traffic from PoPs. Its size indicates the available computing resources at each CC; e.g., the largest CCs are in Warsaw (WAW-1) and Poznan (POZ-1), while the others serve as regional centers. The distances between CCs range from 10 to 300 km. For simplicity, we assume that traffic arrives only at four exemplary PoPs (POZ-1, GDA-2, WAW-3, and RZW-1).

We used data records collected from the mobile operator in March 2024 for our experiments. The dataset comprises 18.5 million flows captured throughout a single day. We sampled the most significant flows hourly to derive the traffic envelope in the daily cycle. The dataset was subsequently anonymized by replacing IP addresses and port numbers and scaling traffic volumes by a constant factor K.

B. vFATD vs. current vFW allocation approaches

In this experiment, we analyse how vFATD and reference vFSA and vFA algorithms allocate vFWs under daily changing traffic. Fig. 5(a) presents reference vFSA and vFA approaches, while Fig. 5(b) presents the vFATD behaviour. Each line shows changes in the vFW instances allocated in different CCs. We also show aggregated traffic volume as a dashed line in both figures to illustrate how traffic changes. Note that the vFSA algorithm allocates a constant number of vFW instances based on the busy hour prediction, so we represent it as a solid line on Fig 5(a).

Starting from t_0 at midnight, we observe that the vFA and vFATD algorithms reduce the number of vFW instances following the traffic decrease. Starting at t_1 at 5 a.m., traffic grows, so both algorithms allocate more and more vFWs. They allocate new instances of vFWs in the nearest CC to the PoPs to handle increased traffic. When resources of RZW-1, POZ-1 became exhausted as t_2 and t_3, the vFA algorithm cannot allocate new vFW, so their number remains constant. As a consequence, new flows are left unchecked or dropped. On the contrary, the vFATD algorithm can still allocate new vFW instances in other locations thanks to traffic distribution. So, the number of vFW increases in POZ-2, where resources

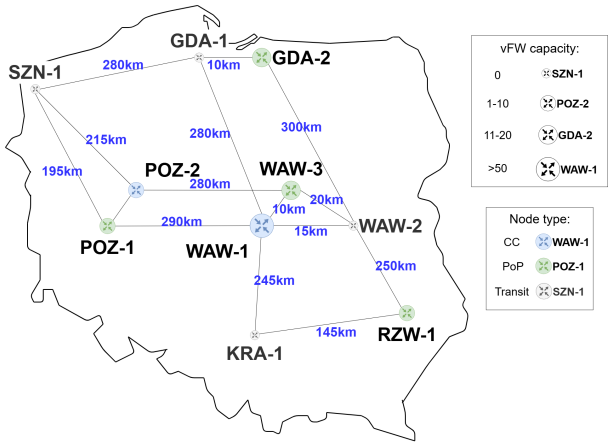
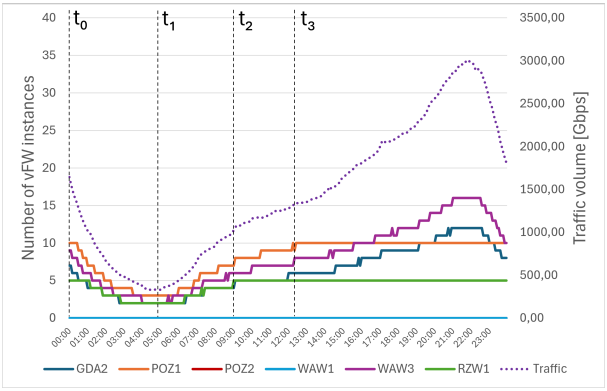
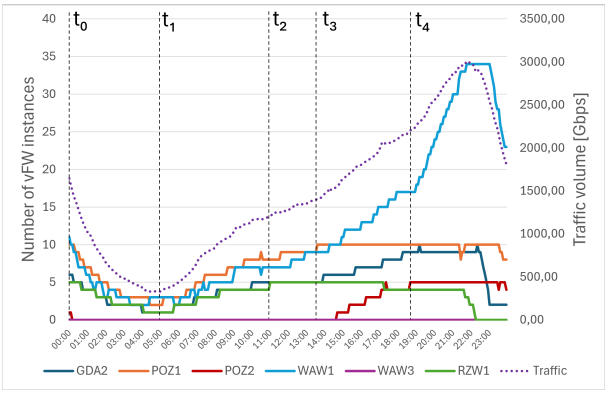


Fig. 4. The ECC infrastructure assumed for the experiments



(a) currently used vFSA and vFA approaches



(b) proposed vFATD exact allocation approach

Fig. 5. Comparison of vFW allocation approaches: a) currently used vFSA and vFA approaches, b) proposed vFATD exact approach

are still available. When resources in POZ-2 become fully booked at t_4, traffic is distributed to WAW-1, where new vFWs are allocated. The analyzed example shows the main benefit of vFATD, which can allocate new vFW instances if there are still available resources in any CC. So, the vFATD adjusts the vFW system performance to the actual traffic conditions, preventing flow losses. This feature protects the network from sudden traffic fluctuations or deliberate Denial of Service attacks and relieves the operator's responsibility for accurate vFW provisioning.

C. Resource optimization

Proper orchestration of vFWs may bring noticeable operational cost savings. In this experiment, we consider how effectively the algorithms use allocated vFWs. We analyze the unused vFW capacity indicated by the vFW over-provisioning factor (OF), defined as the complement of the ratio of the aggregated traffic offered in PoPS to the sum of servicing capabilities of all allocated vFWs. In Fig. 6 we present the results for vFSA, vFA, and our vFATD algorithm.

The results say that: 1) most of the time, vFSA ineffectively uses resources due to assumed static provisioning. The best situation is during the busy hour (at 10 p.m.), when the OF drops to 10%; 2) vFA is much more effective because it adapts

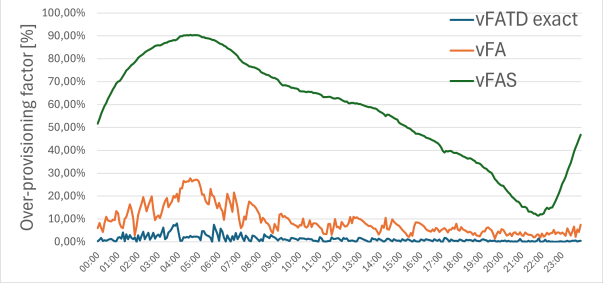
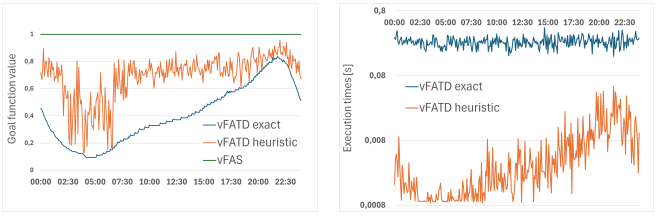


Fig. 6. The evolution of vFW over-provisioning factor (OF)



(a) accuracy (b) execution time

Fig. 7. The accuracy vs. complexity of proposed vFATD algorithms

vFWs to current traffic. However, vFA must keep at least one over-provisioned vFW at each PoP, leading to OF up to 30% observed in our case at 5 a.m. If there were more PoPs in the system, OF would increase; 3) The vFATD keeps the lowest OF, which never exceeds 10%. In the worst case, it runs a single spare vFW instance for the entire system because it can redirect traffic to an available vFW instance.

D. Accuracy vs. complexity of vFATD algorithms

We compare the accuracy and complexity of the proposed exact and heuristic vFATD algorithms. The exact vFATD is the NP-hard allocation problem of $O(2^n)$ complexity. The heuristic uses the GOn k-center algorithm and assigns flows randomly in m iterations, resulting in a total complexity of $O(k^2n + kn\log n + m)$. Hence, the heuristic grows linearly with the number of variables n and iterations m, and quadratically with the number of CCs. Fig. 7 compares the objective function values and calculation time of the exact and heuristic vFATD algorithms. We normalized the objective function to the vFSA results to show the vFATD gain. We conclude that the exact vFATD algorithm optimizes the vFW system, but requires relatively long calculations due to its complexity. The heuristic approach solves the vFATD problem quickly, but with reduced accuracy. The observed instabilities highlight the need for further investigation and improvement.

V. Summary

The paper focuses on virtual firewall (vFW) orchestration in future 5G/6G mobile networks deployed over the virtualized Edge-Cloud Continuum infrastructure. We proposed the novel orchestration approach, Virtual Firewall and Traffic Distribution (vFATD), for vFW instance allocation, horizontal scaling, and traffic distribution within the ECC infrastructure. We formally defined the vFATD problem as the MILP model and proposed the exact and heuristic algorithms to solve it. Our experiments compared the vFATD performance to currently used approaches such as static vFW allocation (vFSA) and dynamic vFW allocation (vFA), assuming a real traffic dataset collected from one of the Polish mobile operators. We analyze the benefits of the traffic distribution mechanism integrated into the vFATD algorithm, focused on resource and cost optimization. Moreover, we assessed the accuracy vs. complexity of the proposed exact and heuristic vFATD algorithms.

The results confirmed that the proposed vFATD approach outperforms the currently used orchestration strategies. In particular, vFATD can adjust the vFW performance to the actual traffic conditions until computing resources are available at any location. In contrast, other approaches focus on resources available in a local PoP. Thanks to this, vFATD may protect the network from sudden traffic fluctuations or Denial of Service attacks, and relieve the necessity of exact vFW provisioning. Moreover, the vFATD approach also uses available resources more effectively. The heuristic algorithm returns the feasible solution much faster but is slightly unpredictable.

Our further work will focus on improving the stability of the heuristics algorithm, exploiting reinforcement learning decision algorithms, e.g., PPO, LSTM-PPO, and performing experiments on the developed prototype.

References

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[2] S. Çalışır, R. Atay, M. K. Pehlivanoğlu, and N. Duru, "Intrusion detection using machine learning and deep learning techniques," in *4th UBMK*, pp. 656–660, 2019.

[3] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "VNF and CNF placement in 5G: Recent advances and future trends," *IEEE TNSM*, vol. 20, no. 4, pp. 4698–4733, 2023.

[4] A. Bocci, S. Forti, and A. Brogi, "Sustainable cloud-edge infrastructure as a service," in *12th MECO*, pp. 1–4, 2023.

[5] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM REVIEW*, vol. 57, no. 1, pp. 3–57, 2015.

[6] L. Gu, D. Zeng, W. Li, S. Guo, A. Y. Zomaya, and H. Jin, "Intelligent VNF orchestration and flow scheduling via model-assisted deep reinforcement learning," *IEEE JSAC*, vol. 38, no. 2, pp. 279–291, 2020.

[7] D. Bringhenti, G. Marchetto, and et. al., "Automated firewall configuration in virtual networks," *IEEE TDSC*, vol. 20, no. 2, pp. 1559–1576, 2023.

[8] D. Bringhenti, R. Sisto, and F. Valenza, "A demonstration of verefoo: an automated framework for virtual firewall configuration," in *IEEE ICNS*, IEEE, 2023.

[9] N. Đeric, A. Varasteh, A. Basta, A. Blenk, R. Pries, M. Jarschel, and W. Kellerer, "Coupling VNF orchestration and SDN virtual network reconfiguration," in *International Conference on Networked Systems (NetSys)*, IEEE, 2019.

[10] M. Gao, B. Addis, M. Bouet, and S. Secci, "Optimal orchestration of virtual network functions," *Computer Networks*, vol. 142, pp. 108–127, 2018.

[11] P.-Y. Kong and Y. Jiang, "VNF orchestration and power-disjoint traffic flow routing for optimal communication robustness in smart grid with cyber-physical interdependence," *IEEE TNSM*, vol. 19, no. 4, pp. 4479–4490, 2022.

[12] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph.," *Operations Research*, vol. 12, no. 3, pp. 450–459, 1964.

[13] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.