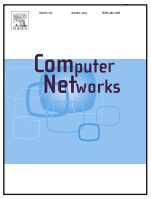
GreenShield-E2C: A Sustainable Energy-Aware Firewall Configuration Mechanism for Edge-to-Cloud Continuum

Computer Networks 282 (2026) 112279

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



GreenShield-E2C: A sustainable energy-aware firewall configuration mechanism for edge-to-cloud continuum

Daniele Bringhenti *, Fulvio Valenza

Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy

ARTICLE INFO

Keywords:
Firewall
Edge-to-cloud continuum
Network sustainability
Power consumption

ABSTRACT

The advent of the edge-to-cloud continuum paradigm has enabled a seamless integration of resources across different architectural layers, offering significant benefits in terms of scalability and flexibility. Due to the complexity of this environment, a key operation is to enforce adequate network security mechanisms to protect the continuum in an effective, efficient, and correct way. In this regard, a critical task is the configuration of distributed packet-filtering firewalls, because they are essential to protect the boundaries between the different layers. Unfortunately, their configuration is commonly performed manually and in an unoptimized way, raising pressing challenges from a sustainability perspective, due to the increasing power consumption related to the activation and operation of each firewall instance in the continuum. In order to address this problem, this paper proposes an approach, named GreenShield-E2C, which integrates automation, formal verification, and sustainability optimization for distributed firewall configuration into a unified methodology tailored explicitly for the continuum's heterogeneous and multi-layer nature. These achievements were reached by formulating the configuration problem as a Maximum Satisfiability Modulo Theories problem, ensuring security policy compliance while minimizing the firewall power consumption. The implementation of the proposed approach has been experimentally evaluated on scenarios derived from a realistic smart city use case, so as to showcase its energy efficiency effectiveness and performance.

1. Introduction

Originally, edge, fog, and cloud computing were conceived as distinct paradigms, each targeting different data processing needs. Cloud computing provides centralized and scalable computation and storage for data-intensive workloads [1], while edge and fog computing bring less computationally-demanding processing closer to data sources to support latency-sensitive applications [2]. As these paradigms have complementary objectives, they have often been adopted jointly, but in a loosely coordinated manner, leading to inefficiencies such as suboptimal resource utilization, workload fluctuations, and performance inconsistency. To overcome these limitations, the edge-to-cloud continuum paradigm has emerged as a unified model in which computational, storage, and networking resources (including IoT devices, sensors, and gateways) are seamlessly distributed across edge, fog, and cloud layers [3]. In this continuum, latency-sensitive, data-intensive, and context-aware applications can be dynamically orchestrated where they are most effective, according to performance, availability, or cost requirements, rather than being statically bound to a specific layer [4].

While this model increases flexibility and scalability, it also introduces significant challenges in terms of network management and security. In particular, the adoption of Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) has further increased the complexity of continuum networks [5]. The resulting infrastructures are large-scale, heterogeneous, and highly dynamic, making it difficult to ensure that communications between continuum components are both secure and efficient. Achieving a complete, yet safe integration among the different continuum components has thus become a crucial challenge, and it requires establishing which communications should be blocked, because potentially linked to a threat, or allowed to provide service continuity.

Therefore, distributed packet-filtering firewalls represent the main security mechanism for safeguarding the systems integrated in this continuum and their communications, thanks to their features of access control, workload isolation, and malicious traffic filtering. However, establishing their configuration is a complex security management operation by itself [6], because it consists of two tasks, i.e., the allocation of the distributed instances in the network service and the computation of their

* Corresponding author.
E-mail addresses: daniele.bringhenti@polito.it (D. Bringhenti), fulvio.valenza@polito.it (F. Valenza).

https://doi.org/10.1016/j.comnet.2026.112279
Received 25 August 2025; Received in revised form 9 March 2026; Accepted 29 March 2026
Available online 31 March 2026

filtering rules. Consequently, manual configuration approaches are no longer viable in large-scale and dynamic environments [7]. Moreover, in the edge-to-cloud continuum, firewall configuration raises an additional and often overlooked problem, i.e., sustainability. Activating and operating multiple firewall instances, especially on heterogeneous and resource-constrained platforms, can significantly increase overall power consumption, negatively impacting operational costs and environmental footprint.

Addressing energy efficiency is nowadays essential to reduce operational costs and optimize the security sustainability in such large-scale, distributed infrastructures. However, even if in literature several automatic firewall configuration solutions were proposed, aiming at avoiding human errors (e.g., exploiting formal methods) or configuration anomalies [8], and improving performance, almost all of them do not guarantee energy efficiency related to the activation and usage of firewall instances. The only exception is GreenShield [9], an approach that minimizes the power consumption related to firewalls activated in the network while ensuring that the security requested by the network administrator is guaranteed. However, its applicability is limited to traditional networks, because both the energy efficiency objective and the network models used in its approach lack all characteristics and features of the edge-to-cloud continuum (e.g., the power consumption of virtualization platforms and the firewall deployment across the heterogeneous layers of the continuum are overlooked).

Proposal and Contributions. In order to overcome the limitations of the current literature, this paper proposes GreenShield-E2C, an enhanced version of the original GreenShield methodology, specifically designed for the edge-to-cloud continuum. The proposed approach relies on a combination of policy-based management and constraint programming to automate the configuration of distributed packet-filtering firewalls while explicitly accounting for sustainability.

From a methodological perspective, GreenShield-E2C allows network administrators to express their connectivity requirements, in terms of isolation and reachability, through high-level network security policies. These policies are independent of vendor-specific configuration details and are written in a user-friendly language. The firewall configuration problem is then formalized as a Maximum Satisfiability Modulo Theories (MaxSMT) problem, which enables the automatic computation of the firewall allocation scheme and filtering rules while guaranteeing correctness by construction. Moreover, the MaxSMT formulation includes special clauses, named soft constraints, that can embed optimization objectives.

Based on this methodology, the main contributions of this paper can be summarized as follows:

- GreenShield-E2C is the first approach combining automation, formal verification, and green optimization, to address the distributed packet-filtering firewall configuration problem, explicitly tailored to the edge-to-cloud continuum, accounting for its multi-layer, heterogeneous, and virtualized nature;
- GreenShield-E2C adopts a formal MaxSMT-based formulation of the firewall configuration problem so as to provide correctness by construction, differently from the other papers in the literature. Specifically, the models presented in this paper include all the characteristics of their real counterparts that could possibly impact the solution itself, in such a way that formal assurance that the computed firewall configuration is compliant with the user requirements contributes to increasing the confidence in using this method;
- GreenShield-E2C pursues two energy-oriented optimization goals that were not targeted by any other alternative solution for firewall configuration. The first is the minimization of the overall power consumption due to all continuum components involved in firewall usage and deployment (e.g., physical firewalls, virtual network functions with firewalling functionality, physical servers hosting virtual firewalls, virtualization technology platforms). The second is the minimization of the path followed by traffic flows that must be blocked, so as to avoid unnecessary energy consumption by network nodes and firewalls that process them uselessly.

Paper structure. The remainder of this paper is structured as follows. Section 2 discusses the related work, highlighting its limitations in balancing network security and sustainability for automatic firewall configuration in the edge-to-cloud continuum. Section 3 describes the approach followed by GreenShield-E2C to reach all its objectives. Section 4 introduces all the formal models that are required for the problem definition. Section 5 formalizes the constraints of the MaxSMT problem modeling the automatic firewall configuration problem. Section 6 discusses how the framework implementing the GreenShield-E2C approach was developed and validated experimentally. Section 7 draws conclusions and outlines future work.

2. Related work

This section dissects the literature in two related research areas. The first is about studies that address the automatic distributed packet-filtering firewall configuration problem, but without pursuing green optimization (Section 2.1). The second concerns studies addressing edge-to-cloud continuum security problems while providing optimized sustainability (Section 2.2).

2.1. Automatic firewall configuration

The automatic firewall configuration problem started to be investigated in the early years of this century due to the importance of a policy-compliant, correct configuration as a prevention or response against cyberattacks in continuous, fast evolution. Due to the complexity of the problem, which involves two tasks (i.e., firewall allocation and rule computation), most of the related literature focuses on introducing contributions for only one of these operations.

Automatic Firewall Rule Computation: From the chronological point of view, the first task to be investigated was the firewall rule computation. The first proposal in this line is represented by Firmato [10], which uses a model compiler to refine a vendor-agnostic entity-relationship model of the network topology and security policies into a concrete firewall configuration. However, this toolkit could only be applied to centralized firewalls, i.e., it cannot introduce rules in different firewall instances of the same distributed architecture to satisfy a single set of policies. To overcome this limitation, first, some simple extensions of this initial proposal were introduced [11,12]. In [11], concrete firewall configuration rules are derived from high-level policies based on an Or-BAC model, whereas FACE [12] makes refinement decisions based on a trust model founded on the concept of network trustworthiness. Both of them were designed to be leveraged in traditional networks, and not in cloud-based environments. Later, this research line had a progressive boost in more recent years, thanks to the advent of network softwarization. In this context, new studies [13–16] also started to embed formal verification in their proposals, so as to provide network administrators with higher confidence in using them. However, the algorithms in [13,14] have a limited applicability, because they are exclusively designed to refactor already existing firewall configurations, so they cannot be used to create them from scratch. Instead, the approaches illustrated in [15,16] do not pursue any optimization goal, but simply try to reach a correct solution.

More recently, the problem of firewall rules computation has also been investigated from the perspective of increasing user-friendliness or integrating groundbreaking technologies such as artificial intelligence [17–21]. However, these approaches cannot formally ensure that the computed rules correctly enforce the security requirements. Moreover, they are not designed to support the native characteristics of the edge-to-cloud continuum, such as multi-layer deployments, virtualization-aware placement, and heterogeneous execution environments, nor do they

pursue sustainability or energy-efficiency objectives related to firewall activation and operation.

Automatic Firewall Allocation: In the research line addressing the more generic network service chaining or composition problem [22], some studies specifically focus on addressing the firewall allocation scheme definition, i.e., deciding where firewalls should be positioned in the logical network topology to satisfy user-specified security policies. However, most of the studies in this literature area [23–28] are limited in terms of firewall allocation features, because they can simply create a network chain or graph from scratch, firewalls included, but cannot cover most of the scenarios considered in this paper, i.e., when the network already exists, with already existing middleboxes such as NATs and load balancers, and when a distributed firewall architecture has to be designed so as to meet security requirements. The only exception is ConfigSynth [29], an approach that can establish the allocation scheme for firewalls and other functions, such as VPN gateways and intrusion detection systems, in a similar way as done in this paper. However, in addition to the fact that it cannot compute filtering rules, this tool cannot provide a guarantee of security requirement satisfaction because it considers them relaxable.

Complete Automatic Firewall Configuration: In literature, there are three studies whose proposals aim at solving the two firewall configuration tasks simultaneously (without pursuing sustainability optimization) [28,30,31]. However, all of them have limited applicability. The approach presented in [30] is limited to the synthesis of service function chains, which may include auto-configured firewalls, but its applicability to service graphs is not discussed. Instead, the method in [28] can automatically generate a service graph made only of firewalls interconnecting endpoints, minimizing their rule number, but without considering the possible presence of middleboxes that can alter the traffic. Finally, VEREFOO [31,32] goes beyond the limitations of the previous studies, as it can be applied to complex virtual service graphs. However, the models on which its approach relies are not characterized for the edge-to-cloud continuum, making it impractical for this environment.

In conclusion, there is no prior work addressing the automatic firewall configuration problem in the edge-to-cloud continuum in general, independently of the fact that no study among the discussed ones pursues sustainability optimization. Therefore, GreenShield-E2C represents the first approach aiming at casting the firewall configuration problem in this continuum through a unified methodology and formal models explicitly designed to capture its intrinsic characteristics. In particular, the proposed approach jointly considers firewall allocation and rule computation across heterogeneous edge, fog, and cloud layers, while accounting for the coexistence of physical and virtual firewall instances and the impact of virtualization technologies. By embedding energy-aware objectives within a formally verified, policy-based strategy, GreenShield-E2C enables the automated computation of correct and sustainability-oriented firewall configurations tailored to continuum-based network infrastructures.

2.2. Optimized sustainability of edge-to-cloud continuum security

In the literature, studies aiming to optimize the sustainability and energy efficiency of security solutions for the edge-to-cloud continuum were proposed in relation to multiple research problems.

A first problem where this trade-off has been investigated is workload scheduling in the edge-to-cloud continuum. When an orchestrator decides where a workload should be scheduled across the continuum, it should take into account both security requirements (e.g., reliability standards or the denial of execution on unreliable resources) and green optimizations (e.g., limiting energy consumption for constrained IoT devices). This vision has been followed by some recent studies [33–35], whose scheduling algorithms consider both factors, differently from earlier literature. The RT-SANE algorithm, introduced in [33], envisions the assignment of privacy labels to tasks, containing their deployment scheme jointly with energy-oriented costs such as the ones due to power and cooling for micro data centers of the fog layer and cloud data centers of the cloud layer. The two heuristic algorithms proposed in [34], i.e., RCSECH and RSECH, aim at optimizing workload scheduling by reaching a trade-off among rental cost, energy consumption measured as Thermal Design Power of multi-core processors, and task reliability, while considering deadlines and privacy requirements as constraints. Instead, the SCEAH algorithm discussed in [34] tries to optimize power consumption while ensuring security awareness, i.e., while guaranteeing that a task will be processed on a virtual machine that meets its security requirements.

A second problem is the optimization of communications in the edge-to-cloud continuum. In this scenario, IoT devices commonly suffer from limited spectrum efficiency. A possible solution is the adoption of the Cognitive Radio Non-Orthogonal Multiple Access (CR-NOMA) technology, but its broadcast nature makes transmissions vulnerable to eavesdropping [36]. The problem of improving spectrum efficiency while ensuring safe transmissions is formulated by the approach discussed in [37] as a non-convex multi-objective problem to maximize security energy efficiency (SEE), defined as the secure rate divided by total energy consumption, under constraints that guarantee different minimum rates for the users, depending on their role. This methodology also embeds another algorithm aiming at a security-energy trade-off, i.e., an optimized orthogonal antenna selection algorithm. In fact, increasing the number of antennas improves connectivity and security (e.g., through better beamforming and spatial diversity), but it raises energy consumption.

A third problem is disaster survivability and recovery. In the edge-to-cloud continuum, consolidation of containers implementing different microservices of an application allows reducing power consumption, as they are deployed on the same physical server, but increases the disaster recovery time (i.e., the time required to resume a minimum service level) because a higher number of containers must be rebooted if their physical server fails. This problem, i.e., achieving a balance between disaster recovery and power consumption, is addressed by the eDRECs (Energy-aware Dynamic Response and Efficient Consolidation Strategy) algorithm [38]. This strategy uses Generalized Stochastic Petri Net models to simulate and evaluate different recovery and consolidation configurations before deploying them in real systems. The computation of these configurations aims at reducing power consumption, while adhering to a desired value for the Recovery Time Objective.

A fourth problem is firewall configuration automation itself. Only GreenShield [9] tries to compute firewall configurations that minimize power consumption. However, the characterization of the green objective was exclusively done for physical firewalls, and did not include energy-related features of the edge-to-cloud continuum (e.g., the power consumption of the virtualization technology and of the virtual firewalls).

In summary, the first three problems are related to the need to provide optimized security sustainability, but they are different problems from the one addressed in this paper. Instead, concerning the fourth, GreenShield-E2C is the first approach in the literature to pursue this kind of optimization for firewall configuration in the edge-to-cloud continuum. By doing so, it bridges the gap between energy-aware security optimization and formally verified firewall configuration in heterogeneous continuum-based infrastructures.

3. The proposed approach

This section presents a comprehensive overview of the approach followed by GreenShield-E2C for automatic firewall configuration in the edge-to-cloud continuum. First, it describes the inputs that must be fed to the methodology by the network administrator (Section 3.1). Then, it discusses how constraint programming is used to formulate a MaxSMT problem which embeds green optimization objectives (Section 3.2). Finally, it outlines the components of the produced output (Section 3.3),

and it discusses additional operational and applicability aspects of the proposed approach (Section 3.4).

The approach presentation is paired with the description of an exemplifying use case, representing the computer network of a smart city, which spans across all three layers of the edge-to-cloud continuum. This scenario represents a relevant study case, because smart cities rely on a heterogeneous network of interconnected devices, from resource-constrained sensors and IoT nodes at the edge (e.g., traffic lights, pollution monitors), to fog nodes managing local processing tasks (e.g., roadside units, local control centers), up to powerful cloud data centers hosting global city analytics and long-term storage. This layered and distributed nature poses unique challenges in terms of dynamic network configuration, security enforcement, and energy efficiency, i.e., precisely the issues that GreenShield-E2C aims at addressing. In addition, smart cities demand strict compliance with high-availability requirements for critical services such as public safety, transportation management, and emergency response. Hence, the smart city use case highlights the technical challenges introduced by the continuum and emphasizes the societal relevance of developing sustainable and reliable firewall configuration mechanisms.

3.1. Inputs of GreenShield-E2C

The network administrator is expected to specify four inputs to GreenShield-E2C: i) the physical network, ii) the logical network, iii) the firewall Virtual Network Function (VNF) types that may be deployed, and iv) the connectivity Network Security Policies (NSPs) to be enforced.

3.1.1. Physical network

The first input is a description of the physical network, representing the infrastructure spanning the whole continuum. In this description, the user must provide both the information necessary for the configuration problem itself and the details relevant to the energy efficiency objectives.

Concerning the information for the firewall configuration problem itself, it must specify the different functionalities and roles of the nodes composing the physical layer, as they vary depending on the layer.

All physical nodes in the *edge* layer of the continuum are endpoints, i.e., the source or destination for the traffic flows that the network administrator would like to control by specifying NSPs. An endpoint may be a single network host (e.g., a surveillance camera, a noise sensor) or a sub-network representing multiple hosts (e.g., a set of temperature sensors associated with the IP address range 210.6.33.0/24).

The physical nodes in the *fog* layer provide intermediate functionalities for network and security management, so they are not the source or destination of traffic flows. They are divided into three classes:

- Some nodes are physical middleboxes providing service functionalities such as network address translation, load balancing, or traffic inspection. For them, the administrator must provide a description of their configuration. For instance, she must specify how each network address translator or load balancer translates the IP addresses of the received packets.
- Other nodes are physical firewalls, which are already positioned in the network but are still *unused*. Specifically, they have been installed but not configured yet, because GreenShield-E2C will be in charge of deciding if they are actually needed for NSP enforcement and, in that case, of computing their rule sets. However, the administrator is given the possibility to force the usage of some firewalls, independently of their actual utility. Such a constraint may impact the global optimality of the final solution, but ensures higher flexibility for the users of the proposed methodology. For example, it supports the case where a firewall must necessarily be used due to a corporate policy.
- The remaining nodes are general-purpose fog servers, whose objective is to host Virtual Network Functions (VNFs). Some of these servers can also be considered candidates for deploying firewalling VNFs established as output by GreenShield-E2C. For each server, the administrator specifies the adopted virtualization technology platform (e.g., KVM, Docker).

Table 1
Average power consumption of the physical firewalls.

Firewall ID	Implementation	Power consumption (W)
m_{20}	FortiGate FG-7060E	2330
m_{21}	Palo Alto PA-450	33
m_{22}	FortiGate FG-7081F	6100

The nodes in the *cloud* layer represent cloud data centers or server farms where VNFs are or may be deployed. Also for these nodes, the administrator specifies the virtualization technology platforms installed there by the cloud providers.

Concerning the information required for the energy efficiency objectives, the input must include details related to power usage. On the one hand, each physical firewall is associated with a weight representing a possible value of power consumption. This value can be specified by the network administrator as an average consumption, or as a value closer to the minimum or maximum power consumption supported by that firewall. The choice depends on the operational scenario that the administrator intends to model, such as typical operating loads, peak traffic situations (e.g., traffic bursts), or low-traffic conditions, respectively. The weights associated with firewall instances are commonly different from one another, because each firewalling product has different energy requirements, as indicated in their data sheets. On the other hand, each fog server or cloud data center, together with its associated virtualization technology platform, is also characterized by a weight representing their combined power consumption when the node is not idle. Similarly to physical firewalls, this weight can be chosen to reflect average, high-load or low-load operating conditions, depending on the level of traffic conditions that the administrator wants to model. Also these values can be retrieved from vendor data sheets, energy-efficiency benchmarks, or empirical measurements available to the network administrator.

Fig. 1 depicts the physical network for the smart city use case. In the edge layer, there are endpoints with the role of traffic sources or destinations, such as surveillance cameras (e_1), an SOS call station (e_3) for emergency reporting, humidity (e_5) and noise sensors (e_7) monitoring environmental conditions, and a subnetwork (e_8) representing distributed environmental sensors. There are also more traditional computer network components, such as an incident response node (e_2), an environmental office (e_4) and a traffic management office (e_6), both with multiple computers. Then, the fog layer includes both processing units and service functions. For instance, fog nodes (from s_{23} to s_{29}) serve as general-purpose servers or middleboxes. Physical middleboxes such as NATs (m_9, m_{12}), routers (from m_{14} to m_{18}), a load balancer (m_{19}), and an IDS (m_{13}) provide key service functionalities. Three unconfigured, not yet active physical firewalls (from m_{20} to m_{22}) are also included. Instead, in the cloud layer, centralized computing facilities (c_{30}, c_{31}) represent data centers available for VNF deployment.

Finally, supposing that the network is working under normal traffic load, concerning power usage information, Table 1 reports the average power consumption of the three physical firewalls. In particular, the power consumption values for the FortiGate FG-7060E and FG-7081F firewalls are taken from the Fortinet FortiGate 7000 Series datasheet [39], while the value for the Palo Alto PA-450 firewall is derived from the PA-400 Series datasheet [40]. Instead, Table 2 reports the combined average power consumption of each fog server and its related virtualization platform. Values about server power consumption are derived from the most recent report about United States data center energy usage produced by the Lawrence Berkeley National Laboratory in December 2024 [41], reporting the average power consumption for different server types (from conventional to AI servers, with varying numbers of processors). Alternatively, they may be obtained from the SPECpower

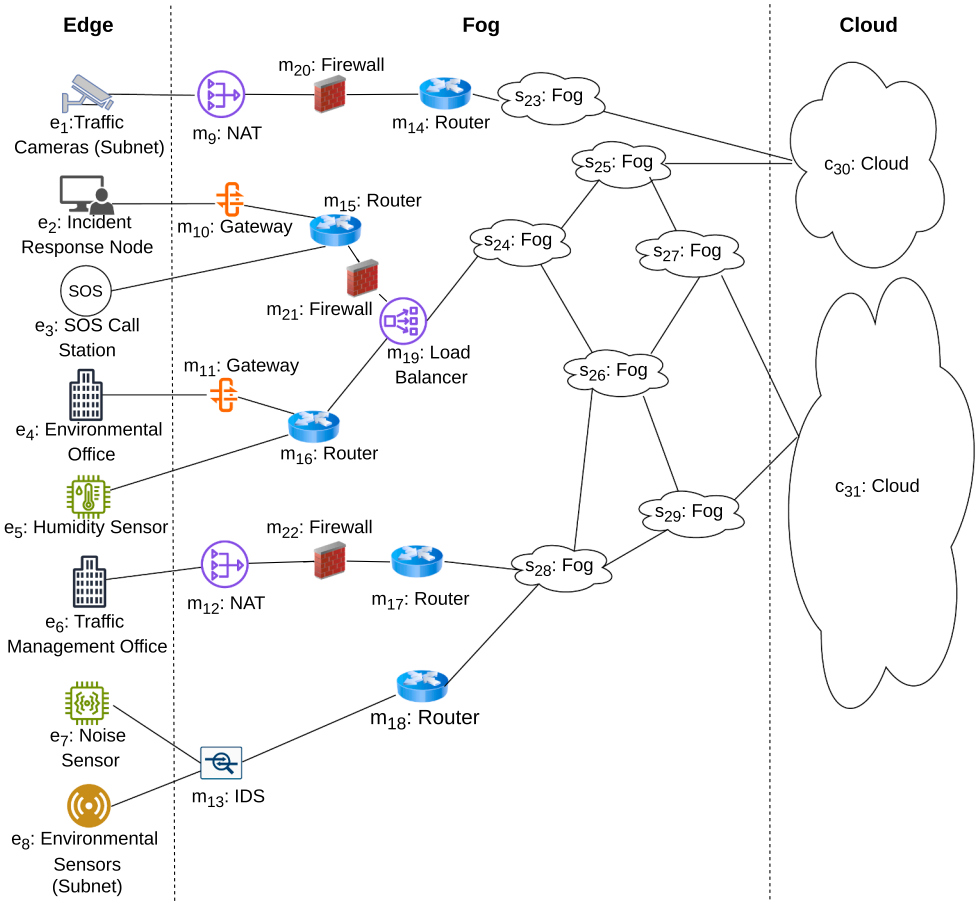


Fig. 1. Physical network of the use case.

benchmark suite, a widely adopted industry standard for server energy efficiency evaluation [42]. Instead, the power consumption of modern virtualization technologies, which is commonly negligible compared to server consumption, is obtained from the results of experimental studies reported in [43] and [44].

3.1.2. Logical network

The second input is a description of the logical network, also referred to as the service graph. A logical network is an abstract representation of how network service functions and endpoints are interconnected. It focuses on the functional relationships between components rather than their physical placement describing which services are provided, how they interact, and the logical paths that traffic follows between them. In practice, the logical network specifies the sequence and composition of functions that process the data flows, along with their configurations if already available. This abstraction is essential for service orchestration, as it enables administrators and automation frameworks to plan, optimize, and enforce service deployments without being constrained by the underlying hardware details.

In the GreenShield-E2C approach, each node of the logical network is mapped to a node in the physical network, either in a one-to-one correspondence if it represents a physical element or through a deployment action if it represents a Virtual Network Function (VNF).

All the elements of the *edge* layer in the logical network are the same as the ones in the physical networks, because they are IoT sensors or traditional computers interacting with remote services. Instead, some elements of the *fog* and *cloud* layers are VNFs, in the form of virtual machines or containers, deployed on physical servers. Moreover, these two layers also include special nodes, named Allocation Candidates (ACs). They represent possible positions of the service graph where the administrator is willing to accept the usage of firewall VNFs, if GreenShield-E2C deems them necessary for security enforcement. The user also has the possibility to force the use of a firewall VNF in some of the possible ACs.

Table 2
Combined average power consumption of fog servers and their virtualization platforms.

Server ID	Virtualization Platform	Power consumption (W)
s_{23}	Docker	180
s_{24}	KVM	410
s_{25}	Xen	760
s_{26}	LXC	135
s_{27}	KVM	1220
s_{28}	Docker	360
s_{29}	Xen	980

Fig. 2 illustrates the logical network corresponding to the smart city scenario. The edge layer directly mirrors its physical counterpart. The fog layer combines both physical functions (from m_9 to m_{22}) and already deployed service function VNFs, such as two IDSs (m_{23}, m_{24}). The fog also includes several ACs (from a_{25} to m_{30}), representing potential deployment points for firewall VNFs, depending on the NSPs and optimization decisions. In the cloud layer, a virtual machine hosting AI-based video analysis services for traffic management (e_{35}) and containers providing environmental data aggregation (e_{36}) and processing (e_{37}) functions interact with fog and edge devices. This layer also includes a proxy VNF

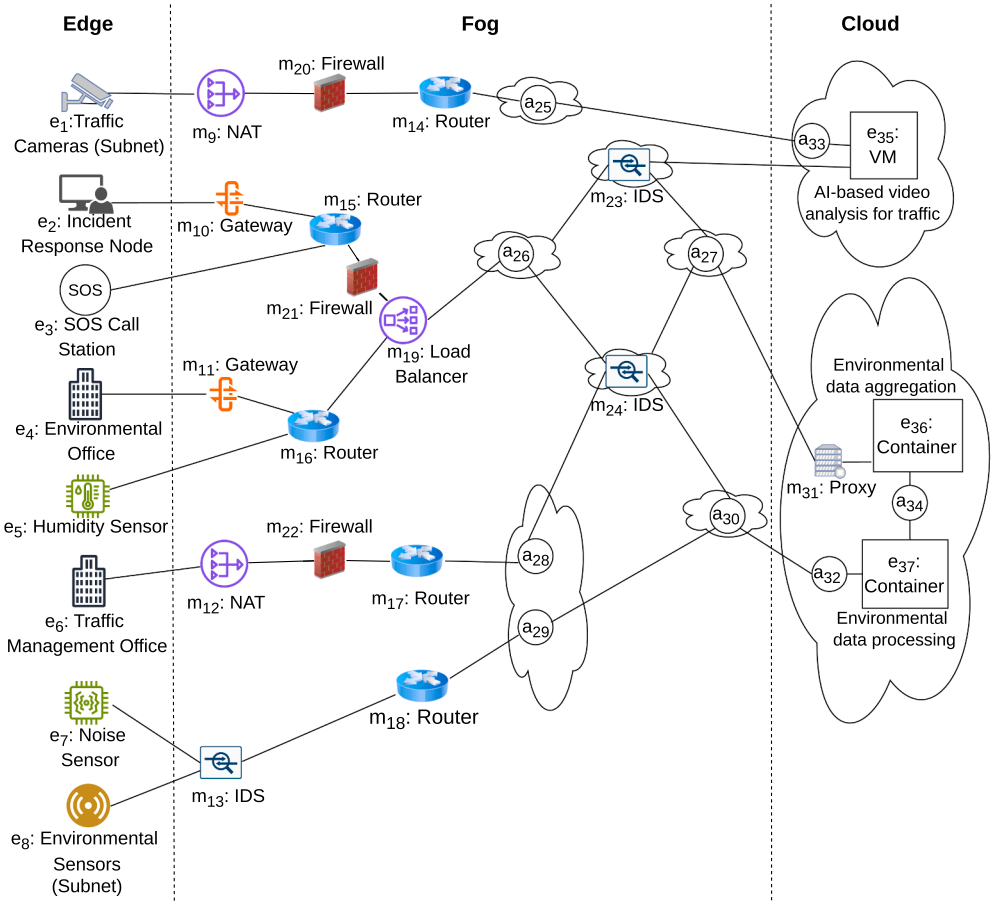


Fig. 2. Logical network of the use case.

Table 3
Mapping between endpoint IDs and IP addresses.

Endpoint ID	IP Address or Subnet
e_1 – Traffic Cameras (Subnet)	10.0.0.0/24
e_2 – Incident Response Node	10.0.1.10
e_3 – SOS Call Station	10.0.1.20
e_4 – Environmental Office	10.0.2.5
e_5 – Humidity Sensor	10.0.3.10
e_6 – Traffic Management Office	10.0.2.6
e_7 – Noise Sensor	10.0.3.11
e_8 – Environmental Sensors (Subnet)	10.0.4.0/24
e_{35} – AI Video Analysis VM	192.168.1.10
e_{36} – Environmental Aggregation Container	192.168.1.20
e_{37} – Environmental Processing Container	192.168.1.21

(e_{31}) and some additional ACs (from a_{32} to a_{34}). Table 3 lists the IP addresses assigned to the endpoints of the smart city use case, while

3.1.3. Firewall VNF types

The third input is a set of firewall VNF types that the administrator is willing to place in the ACs of the logical networks, and consequently deploy on the corresponding servers. Each VNF type is characterized by a virtualization format (e.g., container or virtual machine), a set of requirements regarding memory and CPU resources, a power consumption value, and the virtualization platforms it is compatible with.

Table 4 provides representative examples of firewall VNF types considered in the smart city use case. Each VNF in Table 4 reflects a different trade-off between performance and energy consumption. For instance, `fw_iptables` is a lightweight solution suitable for edge nodes or resource-constrained fog servers, while `fw_suricata` also provides advanced inspection capabilities and is better suited for deployment in high-capacity cloud environments. The compatibility with virtualization platforms ensures that GreenShield-E2C only considers valid deployment candidates during constraint resolution. The power consumption values reported in Table 4 are inspired by empirical measurements related to energy-consumption analyses of VMs and containers available in the literature [45,46].

Table 4
Examples of firewall VNF types.

VNF Type	Form	CPU (cores)	RAM (MB)	Power (W)	Platform(s)
`fw_iptables`	Container	1	512	5	Docker
`fw_pfsense`	VM	2	2048	10	KVM
`fw_snort`	Container	2	1536	12	Docker
`fw_suricata`	VM	4	4096	18	KVM
`fw_opnsense`	VM	3	3072	15	KVM, VMware

3.1.4. Network security policies

The fourth input is a set of Network Security Policies (NSPs), describing which traffic flows must be discarded because they are potentially malicious, and which other ones must be able to reach their destination to guarantee the availability of some end-to-end communication services. The input set of NSPs should be anomaly-free, i.e., devoid of conflicts and sub-optimizations. However, this requirement is not a restriction because anomalies can be easily eliminated by means of

Table 5
Network security policies (NSPs).

ID	Type	Src IP	Dst IP	Src Port	Dst Port	Protocol
P1	Isol	10.0.1.10	192.168.1.10	*	22	TCP
P2	Isol	10.0.4.0/24	10.0.2.5	*	*	TCP
P3	Reach	10.0.0.0/24	192.168.1.10	*	80	TCP
P4	Reach	10.0.3.11	192.168.1.20	*	443	TCP
P5	Isol	10.0.1.20	*	*	123	UDP

multiple anomaly analysis techniques available in the related literature [31,47–49].

Each NSP is characterized by an action and a condition. The action specifies how the firewalling architecture must manage packets satisfying the condition, and it also discriminates NSPs into two classes. In particular, an *isolation NSP* is characterized by a *deny* action, while a *reachability NSP* is characterized by an *allow* action. Instead, the condition is used to identify the packets to which the action must be applied, specifying the IP 5-tuple of the prohibited or allowed flows. As possible values of the IP 5-tuple fields, the $*$ symbol can be used to specify value aggregation. For example, the value 170.24.31.$*$ used as source or destination IP address of the policy condition represents the address range 170.24.31.0/24. Instead, the value $*$ used as source or destination ports expresses all possible numbers that field may have, i.e., from 0 to 65535.

For each isolation NSP, the administrator can specify if the traffic identified by its condition must be blocked as close to its source, so as to avoid being processed by a larger number of middleboxes, with consequent higher energy consumption.

Table 5 presents exemplifying NSPs requested by the network administrator. P1 blocks SSH traffic between the incident response node and the AI-based traffic analysis VM, to prevent potential misuse. P2 ensures that bulk sensor traffic from the environmental subnet (e_8) does not reach the administrative office (e_4), possibly for privacy or load reasons. For P1 and P2, the administrator also requests that their related traffic be blocked near the sources. P3 and P4 allow web and secure access to cloud applications for city services. P5 blocks outbound NTP traffic from the SOS call station (e_3), mitigating amplification threats.

3.2. MaxSMT problem formulation and green optimization objectives

After being fed with these inputs, GreenShield-E2C formulates the firewall configuration problem as a particular type of constraint satisfaction problem, named partial weighted MaxSMT. Unlike the more traditional SAT problems, MaxSMT involves using first-order theories, e.g., the integer theory, thus allowing for higher expressiveness, required for the representations of all the problem components, from the network topologies to the NSPs. It also generalizes the SMT problem due to its partial and weighted nature. On the one hand, it is partial because it is characterized by two types of clauses: hard constraints that must always be satisfied in an output correct solution, and soft constraints that instead should only be satisfied as far as possible. On the other hand, it is weighted because each soft constraint is given a weight representing its priority. Consequently, the goal of a MaxSMT solver is to find an assignment for all the free decision variables of the problem, i.e., all the variables that are not bounded to predefined constant values, such that all hard constraints are satisfied simultaneously, while the sum of the weights assigned to the satisfied soft constraints is maximized.

The MaxSMT problem formulation has been adopted as it provides all three features that characterize the proposed approach, enriching the way it produces the output firewall configuration in a significant way. First, security automation is obtained by applying state-of-the-art MaxSMT solvers to the formulated problem, so that they can use their internal algorithms to search for a correct solution efficiently. Second, formal correctness of the firewall configuration is ensured by construction, as long as all the input models used in the MaxSMT clauses adhere to the main characteristics of their real counterparts and capture all the information that may influence the solution correctness. Third, optimization is reached thanks to the presence of soft constraints, which are relaxable and do not impact the search for a correct solution.

In more detail, GreenShield-E2C uses hard constraints to check for the satisfaction of all the input NSPs. For each NSP, a hard constraint is introduced to state that the NSP must be satisfied by activating and configuring some physical or virtual firewalls. It also introduces a set of hard constraints for each middlebox in the logical network topology to specify the way they can forward traffic flows, because this behavior may impact the NSP satisfaction (e.g., a traffic flow may never be able to reach a destination, thus preventing the satisfaction of a corresponding reachability NSP). Instead, it defines soft constraints to express two green optimization objectives: i) the minimization of the average power consumption of active/deployed firewalls, and ii) the blocking of traffic flows that match the condition of selected isolation NSPs as close as possible to their sources.

In defining these constraints, some predicates are intentionally left unspecified, i.e., they are not assigned fixed values, because the goal of solving the firewall configuration problem is to determine appropriate assignments for them. Examples of such free predicates include those that represent whether a firewall should be deployed or capture decisions about which filtering rules the firewall should implement. Moreover, the formulated problem is always decidable, i.e., a solution can always be found if it exists. This characteristic is provided by the fact that the theories selected for the problem formulation (i.e., the Boolean and integer theories, including only relational operators, without quantifiers) do not include operations such as the multiplication one, which would have made the MaxSMT problem undecidable.

3.3. Output of GreenShield-E2C

GreenShield-E2C tries to solve the formulated MaxSMT problem automatically by using a state-of-the-art solver, which investigates the solution space to search for a correct solution, i.e., a solution satisfying all the hard constraints. The reason for the adoption of such an approach is two-fold. First, modern MaxSMT solvers, which embed formally proven algorithms and strategies to speed up the performance, are able to find out if a correct solution exists in polynomial time on average, despite the NP-completeness of MaxSMT problems in terms of worst-case computational complexity. Second, any solver that follows the MaxSMT semantics can be used as-is, without needing to change the problem formulation or affecting the correctness and consistency of the solutions.

The result of this search carried out by the solver may be i) that no solution exists, or ii) the solution representing the optimal firewall configuration.

In the former case, a non-enforceability result is provided to the network administrator, as the solver could not find any solution satisfying all the hard constraints. A possible reason for unsatisfiability may be that the input VNF types have requirements not supported by the available ACs. However, the administrator can benefit from this information by changing the inputs accordingly, e.g., by introducing new VNF types, so that the solution space of the new MaxSMT problem is bigger, and the chance of finding a correct solution is higher.

In the latter case, the identified solution consists of two elements. One is the firewall allocation scheme, identifying which physical firewalls have been allocated and which ACs have been used to allocate firewall VNFs. The other is the filtering rule set associated with each activated physical firewall or allocated firewall VNF. Both elements have been produced by ensuring that all requested NSPs are satisfied, and that the previously outlined green optimizations are met as much as possible.

3.4. Operational and applicability aspects

This subsection discusses operational and applicability aspects of the proposed approach, with the goal of clarifying its practical use in edge-to-cloud scenarios. In particular, it addresses input availability and specification, discusses how the continuum dynamism can be handled, and explains the applicability of the green optimization.

Input availability and specification. The MaxSMT solver assumes that all inputs provided to GreenShield-E2C are specified correctly, i.e., they accurately reflect the characteristics of their real counterparts, such as the network topology and its configuration. If errors are introduced in the input specification, unexpected results may be obtained from the problem resolution. Although this may occur, it is important to note that the inputs required by GreenShield-E2C are based on information commonly available to network administrators in operational environments, and their specification is relatively straightforward.

On the one hand, concerning availability, the description of the network topology, the deployed services, and security requirements reflects information that administrators typically manage during routine network operation activities, which can be easily derived with automated tools based on standardized interfaces such as OpenC2 [50]. Also power consumption values for physical firewalls, servers, and virtualization platforms can be easily obtained from vendor datasheets, benchmark reports, or empirical measurements, as already discussed. On the other hand, concerning specification, the required inputs can be obtained by translating the information, extracted in their own formats, into the representation expected by GreenShield-E2C, and this simple translation step can be supported by automation mechanisms, including AI-based tools.

Handling of dynamism. The formal model on which GreenShield-E2C is based does not explicitly include variables or functions correlated to time. Instead, the proposed approach is conceived as a configuration mechanism where firewall configurations are computed with respect to a given set of operating assumptions. Despite this characteristic, GreenShield-E2C can still be employed to handle dynamic scenarios. In particular, network administrators can compute multiple configurations corresponding to different operational scenarios, e.g., by associating different power consumption values with the firewall models to represent varying traffic load conditions. These alternative configurations can then be rapidly deployed when the corresponding operating conditions arise. Furthermore, if new circumstances arise, such as changes in traffic patterns, security policies, or network topology, the GreenShield-E2C approach can be applied again to compute an updated configuration that reflects the new scenario.

From an operational perspective, GreenShield-E2C may also be integrated with SDN/NFV orchestration systems by exposing its computed firewall allocation schemes and rule sets through interfaces of the framework implementing it, enabling orchestration frameworks to deploy, update, or replace firewall configurations at runtime.

Applicability of the green optimization. Achieving the green optimization objectives pursued by GreenShield-E2C, i.e., the minimization of the power consumption associated with firewall usage in the edge-to-cloud continuum and the minimization of the traversal path of traffic flows that must be blocked, is a complex task by itself. These goals require reasoning on firewall allocation, rule configuration, and traffic propagation across heterogeneous network layers jointly, while accounting for the energy impact of both physical and virtualized components. Even if they are pursued by GreenShield-E2C in isolation, without accounting for trade-offs with parameters such as latency and reliability, their achievement and effective application represent a significant step forward with respect to the current literature about security sustainability, as already described in Section 2, which is the main focus of this study.

4. Model

This section describes the formal models defined for the main elements of the MaxSMT problem, to provide correctness by construction in its automatic resolution. Specifically, it illustrates the physical and logical network models (Section 4.1), the firewall VNF model (Section 4.2), the power and resource consumption model (Section 4.3), the traffic and network function models (Section 4.4), the NSP model (Section 4.5), and the firewall configuration model (Section 4.6). Finally, other useful model elements are presented (Section 4.7).

Table 6 includes the main formal notations (symbols, functions, predicates, operators) used in the next sections.

4.1. Network models

4.1.1. Physical network model

The first input of GreenShield-E2C, i.e., the physical network topology, is modeled as a directed graph $G_p = (N_p, L_p)$.

The vertex set N_p includes all the physical nodes such that each $n_i \in N_p$ is identified by a unique non-negative integer number i through the $index_p^N \colon N_p \to \mathbb{N}_0$ function. The edge set L_p is the set of directed links among physical network nodes. For that, each $l \in L_p$ is identified by the two non-negative integers identifying the nodes at the extremity of l, and this other association is represented by the $index_p^L \colon L_p \to \mathbb{N}_0^2$ function.

Regarding the vertex set, this is further modeled as the tuple $N_p = (E_p, O_p, C_p)$, where each subset represents a different layer of the continuum. In particular, E_p is the set of all physical endpoints (e.g., traffic cameras, humidity sensors, or environmental office personal computers) in the edge layer, O_p is the set of all physical elements of the fog layer, while C_p is the set of remote servers in data centers of the cloud layer. For what concerns the fog layer, O_p can contain different node types, namely, $O_p = (M_p, W_p, S_p)$, where M_p is the set of physical middleboxes, firewalls excluded (e.g., NATs, traffic monitors)[1], W_p is the set of physical packet-filtering firewalls which may be activated by GreenShield-E2C if deemed useful, and S_p is the set of physical fog servers where some virtual functions can be deployed.

Moreover, each physical server in $S_p \cup C_p$ is associated with a virtualization technology platform (e.g., KVM, Docker), which is used for VNF deployment and management. The set of all technologies used in the input G_p is denoted as K. Instead, the function that maps a server to the installed technology is $\kappa \colon S_p \cup C_p \to K$.

4.1.2. Logical network model

The second input, i.e., the logical network service topology, mapped to the physical infrastructure, is modeled as another directed graph $G_l = (N_l, L_l)$, where nodes and links are identified through the $index_l^N \colon N_l \to \mathbb{N}_0$ and $index_l^L \colon L_l \to \mathbb{N}_0^2$ functions, working on different inputs but in the same way as their counterparts named with p subscript. Here, the model definition for N_l and L_l requires taking into account their relation to the elements of the physical network topology. As concerns the N_l set, it is modeled as $N_l = (E_l, F_l, C_l)$, where each subset again refers to a different continuum layer (edge, fog, and cloud).

In the edge layer, E_l contains the same physical endpoints as E_p, as this network part is mainly composed of physical IoT systems communicating with remote services or with each other. The two edge sets therefore coincide, i.e., $E_l = E_p$.

In the fog layer, the logical node set F_l is the tuple $F_l = (M_p, W_p, M_l^F, A_l^F)$. M_p and W_p are the same physical middlebox and firewall sets of O_p, as each one provides a service by itself from the logical point of view. M_l^F is the set of VNFs exercising service functions other than firewalling, already deployed on a specific server $s \in S_p$. Instead,

[1] In the reminder of the paper, when we use the term “middleboxes“, we refer to “service functions, firewall excluded“, to avoid excessive repetitions of the full clause.

Table 6
Notation.

Notation Element	Definition
Symbols	
$\mathbb{B}$	Set of Boolean values {true, false}
$G_p = (N_p, L_p)$	Directed graph modeling the physical network
$N_p = (E_p, O_p, C_p)$	Physical nodes: endpoints, fog, and cloud nodes
$O_p = (M_p, W_p, S_p)$	Fog layer nodes: middleboxes, firewalls, and servers
$K = \{k_1, k_2, \dots, k_{\lvert K \rvert}\}$	Virtualization technology platforms
$G_l = (N_l, L_l)$	Directed graph modeling the logical network
$N_l = (E_l, F_l, C_l)$	Logical network nodes by layer: edge, fog, cloud
$F_l = (M_p, W_p, M_l^F, A_l^F)$	Fog layer logical nodes: middleboxes, physical firewalls, VNFs, and ACs
$C_l = (M_l^C, A_l^C, E_l^C)$	Cloud layer logical nodes: VNFs, ACs, and service endpoints
$M_l = M_l^C \cup M_l^F$	Set of all middlebox VNFs
$A_l = A_l^C \cup A_l^F$	Set of all ACs
$S = S_p \cup C_p$	Set of all servers
$V = (v_1, v_2, \dots, v_{\lvert V \rvert})$	Firewall VNF types
$T = (t_1, t_2, \dots, t_{\lvert T \rvert})$	Packet classes
$t = q_{t,1} \vee q_{t,2} \vee \dots \vee q_{t,n_t}$	Single traffic
$q_{t,i} = (IPSrc, IPDst, pSrc, pDst, tProto)$	Single traffic condition
$F = (f_1, f_2, \dots, f_{\lvert F \rvert})$	Traffic flows
$P = (p_1, p_2, \dots, p_{\lvert P \rvert})$	NSPs
$p = (y, k)$	Single NSP: type, condition
$g_n = (d_n, \mathbb{F}_n)$	Filtering configuration: default action, rule set
Functions	
$index_p^N \colon N_p \to \mathbb{N}_0$	Maps a physical node to its index
$index_p^L \colon L \to \mathbb{N}_0^2$	Maps a physical link to its index
$\kappa \colon S_p \cup C_p \to K$	Maps a server to its virtualization technology platform
$index_l^N \colon N_l \to \mathbb{N}_0$	Maps a logical node to its index
$index_l^L \colon L_l \to \mathbb{N}_0^2$	Maps a logical link to its index
$\sigma \colon M_l \cup A_l \to S$	Maps a middlebox VNF or an AC to the related physical server
$\gamma \colon S \to \mathcal{P}(M_l \cup A_l)$	Maps a physical server to the related middlebox VNFs and ACs
$\pi : F \to (N_l)^*$	Maps a flow to the ordered list of traversed nodes
$\tau : F \times N_l \to T$	Maps a flow and a node to ingress traffic
$\mu : N_l \times T \to T$	Maps input traffic to its output form
$\iota : \mathbb{B} \to \{0, 1\}$	Maps a Boolean to an integer (false = 0, true = 1)
Predicates	
$supported : V \times S \to \mathbb{B}$	True iff the firewall VNF type is supported by the server
$deployed : V \times A_l \to \mathbb{B}$	True iff the firewall VNF is deployed at the AC
$deny : N_l \times T \to \mathbb{B}$	True iff the node drops input traffic
$near : P \to \mathbb{B}$	True iff traffic must be blocked as near as possible to the source
$used : W_p \cup A_l \to \mathbb{B}$	True iff a firewall or VNF is used
$allowlist : W_p \cup A_l \to \mathbb{B}$	True iff default action is deny (allowlist mode)
$rule : (W_p \cup A_l) \times T \to \mathbb{B}$	True iff the firewall has a rule matching the input traffic
Operators	
$\wedge, \vee, \neg$	used for conjunction, disjunction, negation
.	used to denote a tuple element

A_l^F is the set of the fog-located ACs where a packet-filtering firewall VNF may be potentially installed by GreenShield-E2C, if considered necessary to satisfy the input NSPs.

In the cloud layer, the logical node set is the tuple $C_l = (M_l^C, A_l^C, E_l^C)$. Similarly as for fog, M_l^C and A_l^C are the sets of middlebox and firewall VNFs, respectively, but possibly deployed in servers managed by a cloud provider. Instead, E_l^C is a set of virtual VNFs offering end services, which can be endpoints of communications with the elements in E_l of the edge layer.

For the sake of conciseness, the notations M_l, A_l, and S are introduced, such that $M_l = M_l^C \cup M_l^F$, $A_l = A_l^C \cup A_l^F$, and $S = S_p \cup C_p$, so as to simplify the formalism used in the remainder of the paper.

4.1.3. Physical-logical network relationship model

The elements of the M_l and A_l sets are associated to elements of the S set with the σ and γ functions, defined as $\sigma \colon M_l \cup A_l \to S$ and $\gamma \colon S \to \mathcal{P}(M_l \cup A_l)$.

The former maps a middlebox VNF or an AC to the physical server where it is or may be deployed, while the latter maps a physical server, either in the fog or cloud layer, to the set of already deployed middlebox VNFs and ACs. Instead, the latter maps a physical server to the set of already deployed middlebox VNFs and possibly deployed firewall ones.

4.2. Firewall VNF model

The third input of GreenShield-E2C is a set of possible firewall VNF types that may be deployed in fog and cloud servers, denoted as V. Each server in S is also characterized by a set V_s of firewall VNFs it supports, which mainly depend on the installed virtualization technology. As the relationships of each $v \in V$ with both ACs and the related physical servers will represent a central ingredient for the definition of multiple hard and soft constraints of the MaxSMT problem, they are here modeled with two predicates, named *supported* and *deployed*.

On the one hand, the *supported*: $V \times S \to \mathbb{B}$ predicate is true if the input firewall VNF type v is supported by the server s, i.e., if $v \in V_s$, false otherwise. The interpretation of this predicate is thus directly derived from the inputs, and does not require any subsequent reasoning from the MaxSMT solver.

On the other hand, the *deployed*: $V \times A_l \to \mathbb{B}$ predicate is true if an instance of the firewall VNF type v is deployed on the AC a (and, physically, on the related server $\sigma(a)$), false otherwise. Different from the previous predicate, the interpretation of *deployed* will represent an output of the methodology, as it will be fully established by the solver.

4.3. Power and resource consumption model

Each physical firewall $w \in W_p$ is associated with a power consumption cost c_w, representing the average power consumption when w is active (i.e., it has been powered up). It is also associated with the maximum number of configurable rules $r_w^{\max}$, related to the firewall available resources and filtering performance.

Each firewall VNF type $v \in V$ is associated with a power consumption cost c_v, related to the activation of an instance of that type on a server, and the maximum number of configurable rules $r_v^{\max}$. Besides, the deployment of $v \in V$ in a server requires a number of processing cores denoted as cpu_v, and an amount of memory denoted as ram_v.

Each physical server $s \in S$ and its related virtualization technology $\kappa(s)$ have a combined power consumption cost c_s when the node is processing an average traffic load (i.e., when the node is not idle). Overall, the actual power consumption of a server is the sum of c_s and the costs c_v of the allocated VNFs. In relation to this aspect, $s \in S$ is associated with a parameter l_s expressing the upper threshold over which the overall power consumption cannot go. Moreover, each $s \in S$ is associated with a parameter cpu_s expressing the number of cores that are available for VNFs in that server, and a parameter ram_s expressing the amount of

available memory[2] These limits allow modeling heterogeneous resource constraints of the devices located across the edge-to-cloud continuum. As examples, servers deployed closer to the network edge are typically characterized by tighter power and resource limitations, while servers in the core of the fog network or in the cloud can offer higher computational capacity and energy limit. Therefore, they will be used to define hard constraints in the MaxSMT problem formulation, establishing that the overall power and resource consumption related to the possibly multiple deployed VNFs does not exceed the respective limits.

4.4. Traffic and network function models

4.4.1. Traffic model

The modeling approach pursued in this study is not per-packet, but per-packet class. Namely, a packet class is defined as a group of packets sharing the same characteristics in terms of values characterizing their IP 5-tuple fields. This definition is enough to address the automatic configuration of packet-filtering firewalls, as they make decisions only based on those five fields. From here on, for the sake of conciseness, a packet class is referred to as traffic.

Let T be the set of all possible traffics received by at least a node $n \in N_l$. A traffic $t \in T$ is modeled as a disjunction of predicates $q_{t,1} \vee q_{t,2} \vee \ldots \vee q_{t,n_t}$. In turn, each $q_{t,i}$ sub-predicate is modeled as a conjunction of five predicates, imposing specific values for the IP 5-tuple, and concisely written as a tuple $q_{t,i} = (IPSrc, IPDst, pSrc, pDst, tProto)$. Each sub-predicate of $q_{t,i}$ may define a specific value (e.g., a single IP address such as 184.53.44.82 or a single port such as 22) or a grouping condition (e.g., an IP address range 10.15.12.0/24, or a port range [12300, 12500]).

A packet belongs to a traffic t if its IP 5-tuple fields have values satisfying at least one of the $q_{t,i}$ sub-predicates composing t.

4.4.2. Traffic flow model

Nodes in N_l can modify any received traffic t before sending it to the next hop. The concept of how a traffic crosses G_l, undergoing possible modifications before reaching its final destination, is defined as traffic flow.

Let F be the set of all possible traffic flows crossing G_l. A traffic flow $f \in F$ is modeled as a list of nodes and traffics, in an alternated way, i.e., $f = [n_s, t_{sa}, n_a, t_{ab}, n_b, \ldots, n_j, t_{jk}, n_k, \ldots, n_p, t_{pd}, n_d]$. The extremity elements of this list, n_s and n_d, are endpoints, i.e., $n_s, n_d \in E_l \cup E_l^C$. The other nodes appearing in the list are middleboxes, packet-filtering firewalls or ACs, i.e., they belong to the M_p, W_p, M_l^F, A_l^F, M_l^C and A_l^C sets. Instead, each traffic t_{ij} represents the packets possibly forwarded from n_i to n_j in the flow, after n_i has applied a possible transformation. This formalization allows to express how packets are modified and steered to pass through in case they were not stopped, and then the possibility that they could be dropped will be taken into account by the MaxSMT solver.

Relationships among traffic flows, single traffics, and network nodes are modeled with a pair of functions. On the one hand, the $\pi\colon F \rightarrow (N_l)^*$ function maps a flow to the ordered list of nodes crossed by that flow, including the destination, but excluding the source. On the other hand, the $\tau\colon F \times N_l \rightarrow T$ function maps a flow and a node to the traffic, belonging to that flow, received by that node.

All possible traffic flows F crossing G_l can be pre-computed before the MaxSMT problem formulation, by knowing the input NSPs and the network function behavior (later formalized in this section). Indeed, multiple algorithms exist in the literature for this purpose. In this study, we adopt the Atomic Flow computation strategy, based on the Atomic Predicate idea, originally proposed by Yang and Lam in [51,52] and recently reused by other studies [53,54]. As we did not apply changes to this algorithm, we do not report it in this paper.

4.4.3. Network function model

Concerning network service functions (e.g., network address translators, load balancers, simple forwarders, and firewalls themselves), only their forwarding and modification behaviors require modeling, as they are the only behavior components impacting the resolution of the firewall configuration problem.

On the one hand, the forwarding behavior specifies if a traffic is blocked by the network function and it is modeled by *deny*: $N_l \times T \rightarrow \mathbb{B}$. This predicate maps a network function installed on the node $n \in N_l$ and a traffic $t \in T$ to true if n drops all the packets included in the class t (i.e., satisfying at least one of the $q_{t,i}$ sub-predicates of t), to false otherwise. On the other hand, the modification behavior defines how a traffic is modified by a network function (e.g., by changing the source IP address) and it is modeled by $\mu\colon N_l \times T \rightarrow T$. This function maps a network function installed in the node $n \in N_l$ and an input traffic t to the traffic that may be produced by n as output. Note that μ may also be the identity function, if no modification is ever applied by the corresponding network function.

4.5. Network security policy model

The fourth input of GreenShield-E2C is a set of NSPs that must be enforced in the network.

Let P be the set of the input NSPs. $p \in P$ is modeled as a tuple $p = (y, k)$, where y is the NSP type, k is the policy condition, identifying the traffic to which the NSP is related. For the former, y is a constant that is assigned with the i or r value, depending on whether the NSP is requesting isolation or reachability, respectively. For the latter, k is modeled in the same way as t as it expresses a condition on a traffic, i.e., $k = (IPSrc, IPDst, pSrc, pDst, tProto)$. In greater detail, k provides information on how the matching traffic appears at the source and at the destination of the crossed network path. This means that the predicates *IPSrc* and *pSrc* specify conditions on the traffic generated by the source, while the predicates *IPDst*, *pDst*, and *tProto* specify conditions on the traffic received by the destination. The set of all the flows satisfying these conditions for an NSP $p \in P$ is denoted as $F_p \subseteq F$.

Additionally, the predicate *near*: $P \rightarrow \mathbb{B}$ maps an NSP p to true if it is an isolation policy for which the user requested that the identified traffic must be blocked as close as possible to its source.

4.6. Firewall configuration model

Two aspects of the firewall configuration are modeled: i) the usage of the firewalling instances composing the distributed architecture; ii) the filtering configuration of each used firewall.

The usage of each firewall is modeled with the *used*: $W_p \cup A_l \rightarrow \mathbb{B}$ predicate, which is applicable to both physical firewalls and firewall VNFs allocated in the ACs of the fog and cloud layers. Specifically, this predicate maps a physical firewall $w \in W_p$ to true if that firewall must be activated to enforce some input NSPs, to false if it can be left unused or removed from the physical infrastructure. Instead, it maps an AC $a \in A_l$ to true if a firewall VNF must be used there, to false otherwise. In this second case, the *used* predicate will have to be linked to the *deployed* predicate through some constraints. Additionally, the GreenShield-E2C user can impose that a firewall $w \in W_p$ must be necessarily included in the final configuration, e.g., because of some organizational policy. Similarly, the user may want to enforce the usage of a firewall VNF in an AC $a \in A_l$. In this particular case, the *used* predicate maps the related input firewalls to true. For all the other firewalls, the *used* predicate is left free, i.e., there is no imposition on the Boolean value to which the predicate maps those firewalls, as it will be computed by the solver as output.

[2] In the case of $s \in C_p$, these parameters are to be meant as the amount of resources that the cloud provider is willing to offer to a consumer for their VNFs in that server.

The filtering configuration g_n of each firewall $n \in W_p \cup A_l$ is modeled as the tuple $g_n = (d_n, \mathbb{F}_n)$. $\mathbb{F}_n$ is a set of filtering rules, while d_n is the default action applied to any traffic that does not match the condition of any filtering rule in $\mathbb{F}_n$. The possible values that can be assigned to d_n are "deny" or "allow". In relation to this assignment, the *allowlist*: $W_p \cup A_l \rightarrow \mathbb{B}$ predicate maps a firewall $n \in W_p \cup A_l$ to true if it is configured in allowlist mode with "deny" as default action, and to false if it is configured in denylist mode with "allow" as the default action. Instead, each $\mathbb{f} \in \mathbb{F}_n$ is a specific rule defined as $\mathbb{f} = (\mathbb{a}_\mathbb{f}, \mathbb{k}_\mathbb{f})$, where $\mathbb{a}_\mathbb{f}$ is the rule action, while $\mathbb{k}_\mathbb{f}$ is the rule condition identifying matching packet classes. For a firewall n, each $\mathbb{a}_\mathbb{f}$ is "allow" if d_n is "deny", and vice versa. In view of this assumption, the *rule*: $W_p \cup A_l \times T \rightarrow \mathbb{B}$ predicate is introduced to map a firewall $n \in W_p \cup A_l$ and a traffic $t \in T$ to true if n is configured with a filtering rule whose condition is matched by the packets of t, and to false otherwise. In fact, the action can be easily derived from the interpretation of the *allowlist* predicate. In conclusion, both the *allowlist* and *rule* predicates are left free so that the solver can establish them in a way that satisfies the MaxSMT problem soft constraints, i.e., the ones optimizing power consumption.

4.7. Other model elements

Two auxiliary model elements, which cannot be reduced to a specific scope, are introduced here.

The "." notation is used to denote a specific tuple element. For instance, given a tuple $u = (a, b, c)$, $u.a$ identifies element a of tuple u.

The ι: $\mathbb{B} \rightarrow \{0, 1\}$ function maps the Boolean value *false* to the integer 0, and *true* to 1. It will thus be used in the problem constraints to move from the Boolean theory to the integer one.

5. MaxSMT problem

This section formalizes all the constraints composing the MaxSMT problem. On the one hand, it describes the hard constraints related to NSP enforcement (Section 5.1), network forwarding behavior (Section 5.2), firewall usage and deployment (Section 5.3). On the other hand, it presents the soft constraints related to power consumption due to firewall configuration (Section 5.4).

5.1. Hard constraints related to NSPs

The highest-level constraints are those requesting the enforcement of all NSPs, as they are strictly related to the user-specified security requirements. These constraints differ depending on the type of NSP that must be enforced, i.e., isolation or reachability.

The hard constraint defined for an isolation NSP $p \in P$ such that $p.y = i$ is formalized in (1). This clause imposes that all traffic flows F_p satisfying $p.k$ must be prevented from reaching their destination. This imposition is achieved if, for each possible flow $f \in F_p$, at least a (physical or virtual) firewall is used in the flow path $\pi(f)$, and it actively blocks the traffic $\tau(f, n)$ related to that flow.

$$\forall f \in F_p.\ \exists n \in W_p \cup A_l.\ (n \in \pi(f) \wedge used(n) \wedge deny(n, \tau(f, n))) \tag{1}$$

The hard constraint defined for a reachability NSP $p \in P$ such that $p.y = r$ is formalized in (2). This clause is the negation of the one expressed in (1). In fact, it imposes that, among all traffic flows F_p satisfying $p.k$, at least a flow $f \in F_p$ can reach its destination. This imposition is achieved if, for at least a flow $f \in F_p$, any (physical or active) firewall which is used in the flow path $\pi(f)$ does not block the traffic $\tau(f, n)$ related to that flow.

$$\exists f \in F_p.\ \forall n \in W_p \cup A_l.\ (n \in \pi(f) \wedge used(n) \implies \neg deny(n, \tau(f, n))) \tag{2}$$

In both (1) and (2), the firewall forwarding behavior is taken into account through the inclusion of the *deny* predicate. Therefore, the satisfaction of the NSP enforcement constraints requires that the lower-level hard constraints related to the network forwarding behavior and those related to firewall usage and deployment are also satisfied.

5.2. Hard constraints related to the network forwarding behavior

Some hard constraints are defined over the *deny* predicate, also appearing in (1) and (2), to express the network forwarding behavior, and depend on the specific function types present in the logical topology. In this study, two main function type classes are considered.

The first class includes all the network functions that can never block traffic flows, i.e., the endpoints in $E_l \cup E_l^C$ and the middleboxes in M_l (as the M_l does not include firewalls). For any of these network functions, the hard constraint formalized in (3) makes $deny$ maps that node and any received traffic to false.

$$\forall n \in (E_l \cup E_l^C \cup M_l).\forall f \in F.(deny(n, \tau(f, n)) = false) \tag{3}$$

The second class includes all physical firewalls in W_p and all virtual firewalls that may be possibly deployed in the ACs of A_l. Unlike the middleboxes in M_l, firewalls may block or allow the input traffic depending on their filtering rules. The hard constraint formalized in (4) expresses both possible cases of this more complex forwarding behavior. A firewall blocks an input traffic t if and only if the firewall is actually used and either (a) the firewall applies the "deny" default action due to the absence of "allow" rules matching t, or (b) the firewall applies a "deny" rule before t could be processed by the "allow" default action.

$$\begin{aligned} &\forall n \in W_p \cup A_l.\ \forall f \in F.\\ &(deny(n, \tau(f, n)) = (used(n) \wedge ((a) \vee (b))))\\ &(a) = allowlist(n) \wedge \neg rule(n, \tau(f, n))\\ &(b) = \neg allowlist(n) \wedge rule(n, \tau(f, n)) \end{aligned} \tag{4}$$

The *deny* and *rule* predicates appearing in these constraints are left free, with minor exceptions related to restrictions imposed by the administrator.

5.3. Hard constraints related to firewall usage and deployment

In general, the *used* predicate is left free, because its interpretation represents an output of GreenShield-E2C and should be determined by the solver automatically. However, the user may restrict the solution space by imposing that some physical or virtual firewalls must be necessarily used. The hard constraint formalized in (5) represents this imposition, by making the *used* predicate map firewall $n \in W_p \cup A_l$ to true.

$$used(n) = true \tag{5}$$

Other four hard constraints are related to firewall positioning in the logical ACs and their deployment in the corresponding physical servers.

First, if a firewall is used in an AC $a \in A_l$ belonging to the server $\sigma(a)$, it is required that an instance of a firewall VNF type $v \in V$, supported by the server itself, is deployed on a, as formalized by the hard constraint (6).

$$\begin{aligned} &\forall a \in A_l.\ \big(used(a) \implies\\ &(\exists v \in V.\ (deployed(v, a) \wedge supported(v, \sigma(a))))\big) \end{aligned} \tag{6}$$

Second, if an instance of a firewall VNF type $v \in V$ is deployed in an AC $a \in A_l$, the deployment of a different type $v' \in V$ is forbidden, because it would be meaningless and redundant, as formalized by the hard constraint (7).

$$\begin{aligned} &\forall a \in A_l.\ \forall v \in V.\ (deployed(v, a) \implies\\ &(\forall v' \in V \mid v' \neq v.\ (\neg deployed(v', a)))) \end{aligned} \tag{7}$$

Third, for each server $s \in S$, the number of available cores cpu_s cannot be exceeded by the overall number of processing cores demanded by the VNFs located in the ACs of the set $\gamma(s)$. Similar consideration applies to the amount of available memory ram_s. These two restrictions

are formalized by the hard constraints (8) and (9).

$$\forall s \in S. \left(\sum_{a \in \gamma(s)} \left(\sum_{v \in V} \imath\big(deployed(v,a)\big) \cdot cpu_v \right) \right) \leq cpu_s \tag{8}$$

$$\forall s \in S. \left(\sum_{a \in \gamma(s)} \left(\sum_{v \in V} \imath\big(deployed(v,a)\big) \cdot ram_v \right) \right) \leq ram_s \tag{9}$$

Fourth, a significant hard constraint, formalized in (10), is related to power consumption. Specifically, the upper power consumption limit l_s of any server $s \in S$ cannot be exceeded by the consumption of the deployed VNFs. In terms of formalization, the main difference of this constraints with respect to the ones presented in (8) and (9) is that the cost c_s is included, to take into account the combined cost of the running physical server s and its related virtualization technology $\kappa(s)$.

$$\forall s \in S. \, c_s + \left(\sum_{a \in \gamma(s)} \left(\sum_{v \in V} \imath\big(deployed(v,a)\big) \cdot c_v \right) \right) \leq l_s \tag{10}$$

Finally, some hard constraints impose the maximum number of configurable rules related to each used physical firewall or deployed VNF be respected. These two cases are formalized in (11) and (12), respectively.

$$\forall w \in W_p. \, used(w) \implies \left(\sum_{\substack{f \in F_p | \\ w \in \pi(f)}} \imath\big(rule(w, \tau(f,w))\big) \right) \leq r_w^{max} \tag{11}$$

$$\forall a \in A_l. \, used(a) \implies \left(\sum_{\substack{f \in F_p | \\ a \in \pi(f)}} \imath\big(rule(a, \tau(f,a))\big) \right) \leq \left(\sum_{v \in V} \imath\big(deployed(v,a)\big) \cdot r_v^{max} \right) \tag{12}$$

5.4. Soft constraints related to power consumption due to firewall usage

The first optimization goal of GreenShield-E2C is to minimize the overall power consumption related to firewalls used in the whole edge-to-cloud continuum. Three soft constraint classes are introduced for the achievement of this objective, related to:

- the minimization of the power consumption related to the usage of physical firewalls;
- the minimization of the power consumption related to the virtualization technology running on the servers associated with used ACs;
- the minimization of the power consumption related to the single firewall VNFs running in the server associated with the used ACs.

For the representation of the soft constraints, the notation $Soft(\varphi, w, g)$ is used. There, φ is the formula which should be satisfied if possible, w is the weight representing the penalty to be paid if φ cannot be satisfied, and g is the group of soft constraints to which φ appears. All these three soft constraint classes are associated with the same optimization group g_1, so that they have the same relative priority.

The goal of minimizing the power consumption related to the usage of physical firewalls is formalized in (13). This formula states that the *used* predicate should map each physical firewall $w \in W_p$ to false, if possible, so that only the ones that are really required are mapped to true.

$$\forall w \in W_p. \; Soft(\neg used(w), c_w, g_1) \tag{13}$$

The goal of minimizing the power consumption related to the virtualization technology running on the servers associated with used ACs is formalized in (14). This formula states that the *used* predicate should not map any AC associated with a server $s \in S$ to true. Otherwise, a penalty of c_s would be paid due to the necessity of running the virtualization technology to manage the deployed firewall VNFs.

$$\forall s \in S. \; Soft(\neg(\exists a \in \gamma(s). \, used(a)), c_s, g_1) \tag{14}$$

The goal of minimizing the power consumption of the single firewall VNFs running in the server associated with used ACs is formalized in (15). This formula states that no firewall VNF should be deployed in any AC, if possible.

$$\forall s \in S. \, \forall a \in \gamma(s). \, \forall v \in V. \; Soft(\neg deployed(v,a)), c_v, g_1) \tag{15}$$

Algorithm 1 computation of the weights for the constraint (16).

Input: an isolation NSP $p \in P$
Output: the value of all weights $c_{n,f}$

```
if near(p) = true then
    for f ∈ F_p do
        c_tmp ← 1
        for n ∈ π(f) do
            if n ∈ W_p ∪ A_l then
                c_{n,f} ← c_tmp
                c_tmp ← c_tmp + 1
```

5.5. Soft constraints related to power consumption due to processing traffic

The second optimization goal of GreenShield-E2C is to block packet classes, identified by selected isolation NSPs, as close to their source as possible so that fewer middleboxes must process the corresponding traffic flows and can consequently avoid consuming additional power. This objective is formalized in (16), whose formulas make the *deny* predicate map each pair composed of a firewall or AC and each input traffic to false, if possible. The weights $c_{(n,f)}$ associated with these soft clauses start from 1 for the first firewall or AC encountered in the path $\pi(f)$ and are progressively increased by 1 for each crossed firewall or AC, as shown in Algorithm 1. In this way, if the solver needs to make a firewall block a certain traffic identified by these selected isolation NSPs, it will choose it among the ones that are nearer to the source, because the penalty $c_{(n,f)}$ for making the formula of the constraint unsatisfied is lower.

$$\forall p \in P | \, near(p) = true. \; \forall f \in F_p. \; \forall n \in N_l | \, n \in \pi(f). \; Soft(\neg deny(n, \tau(f,n)), c_{(n,f)}, g_2) \tag{16}$$

5.6. MaxSMT problem resolution

The MaxSMT problem representing the automatic firewall configuration is built using all the previously described hard and soft constraints. An automated solver is then used to search for the optimal solution, after being configured in such a way to follow a lexicographic priority of objectives. Specifically, the constraints of group g_1 are declared to the solver before the ones of group g_2, so that it gives higher priority to satisfy the first ones, as the most impactful choice for the minimization of firewall power consumption is determining if a firewall is used or not.

The solution search is unsuccessful if the solver cannot find a model, i.e., an interpretation for all free variables and predicates that satisfies all hard constraints. In this case, the user is informed about the unsatisfiability of the formulated problem, and may decide to modify some inputs for the next run.

Instead, if at least a satisfying interpretation exists, the solver identifies the solution that optimizes the satisfaction of the soft constraints, while respecting all hard ones. In greater detail, the produced model provides information about both the allocation scheme and configuration rule set of the distributed packet-filtering firewall. The former derives from the interpretation of the *used* and *deployed* predicates, which inform about which physical firewalls must be activated, which ACs

should have an active firewall VNF type, and which VNF types should be actually deployed there. The latter is derived from the interpretation of the *allowlist* and *rule* predicates, which establishes the default action for each use firewall, and its set of specific filtering rules, with opposition action.

6. Implementation and validation

A Java-based framework was developed based on the proposed GreenShield-E2C approach. The implemented tool embeds Z3, a state-of-the-art theorem prover, as automated MaxSMT problem solver [55]. It exposes REST APIs for interaction with external users or software, and it supports both JSON and XML formats as message encoding.

Multiple validation tests for the developed framework were carried out on a 4-core Intel i7-6700 3.40 GHz workstation equipped with 32 GB RAM, while using version 4.8.8 of the Z3 solver. On the one hand, optimization tests allowed to assess the achievement of the originally set green goals, in terms of minimization of power consumption related to the usage and configuration of packet-filtering firewalls (Section 6.1). On the other hand, scalability tests contributed to assessing how the performance of the tool varies depending on parameters of the inputs, e.g., the numbers of ACs and NSPs (Section 6.2). Finally, all the achieved results are synthesized to highlight the key strengths of GreenShield-E2C demonstrated across the experiments (Section 6.3).

6.1. Optimization validation

The optimization provided by GreenShield-E2C in terms of minimization of power consumption due to firewall activation was experimentally assessed by carrying out the following tests:

1. GreenShield-E2C was compared with manual naïve configuration strategies, which lack the advanced optimization allowed by automation and can therefore be used for a baseline analysis;
2. GreenShield-E2C was compared with a reduced version of itself, devoid of soft constraints, so that such ablation analysis can showcase the effectiveness of these clauses in achieving green optimization;
3. GreenShield-E2C was compared with the original GreenShield approach, which can minimize power consumption only of physical firewalls;
4. GreenShield-E2C was compared with a state-of-the-art approach for firewall configuration in virtual networks, VEREFOO, which only pursues the optimization goal of minimizing the number of allocated firewalls, independently of their power consumption;
5. GreenShield-E2C was applied to three different operational conditions characterized by different amounts of traffic, so as to assess how it can manage situations such as traffic bursts.

6.1.1. Comparison with manual configuration strategies

First, GreenShield-E2C was compared against two common manual firewall configuration strategies: (a) the worst-cost strategy that, for simplicity, used all physical firewalls, allocates a virtual firewall in each AC, configures all of them with *allow* as default action, and installs one *deny* rule for each isolation NSP in each used firewall; (b) a more optimized strategy that, for each isolation NSP, uses a physical firewall or allocates a virtual one, with *allow* default action, in the position that is closest to the source specified by the NSP, configuring a rule that enforces the NSP in it.

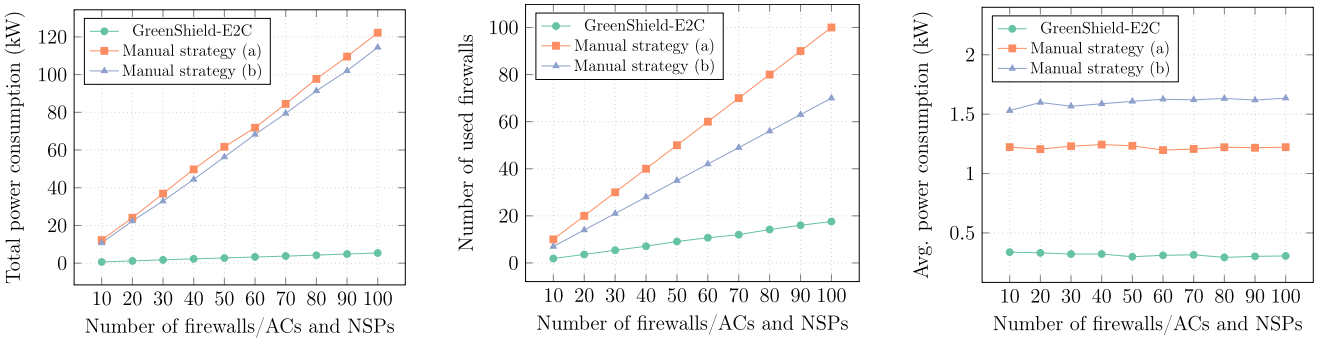
This comparison was carried out by running the GreenShield-E2C framework and by simulating the execution of these two manual strategies on 100 variations of 10 use cases of progressively increasing size of the firewall configuration problem. Through these tests, three comparative parameters, i.e., the total power consumption of the used firewalls, the number of used firewalls, and the average power consumption per used firewall could be experimentally computed as the average of their 100 corresponding values measured in the 100 use case variations.

Each use case of the 10 employed ones is based on a pair of network topologies (i.e., physical and logical ones) that are artificially extended versions of the smart city scenario introduced in Figs. 1 and Fig. 2, and on a set of NSPs that are generated as an extension of the ones shown in Table 5, in such a way that the numbers of isolation and reachability NSPs are the same. This choice is motivated by the fact that smart city infrastructures are widely recognized as a representative example of edge-to-cloud continuum environments, as they combine edge devices, intermediate fog nodes, and cloud data centers within a single, heterogeneous network architecture [56]. In each use case, the combined number of firewalls and ACs and the number of NSPs are the same, so as to provide a progressively increasing problem size, and they range from 10 to 100 in increments of 10 across the different test scenarios.

Moreover, the 100 variations created for each use case characterized by a specific number of firewalls/ACs and NSPs are automatically synthesized so that in each variation only the power consumption values assigned to physical firewalls, physical servers with their running virtualization technology and firewall VNFs are modified. Specifically, these values are randomly selected from three ranges, whose lower and upper bounds were determined based on an analysis carried out on the most common power consumption values of real-world solutions.

- The power consumption values for physical firewalls are randomly selected in the range between 30 and 6500 W. This range is broad because the different firewall solutions have varying power requirements depending on the features they can provide. Considering as examples just three firewalls of the same vendor, i.e., Fortinet, the average power consumption is 29.5 W for the 100F model, 2330 W for the 7060E-8 model, and 6100 W for the FG-7081F model.
- The combined power consumption values for physical servers and their running virtualization technology are randomly selected in the range between 100 and 1400 W. On the one hand, these values are derived from the most recent report about United States data center energy usage produced by the Lawrence Berkeley National Laboratory in December 2024 [41], reporting the average power consumption for different server types (from conventional to AI servers, with varying numbers of processors). On the other hand, the power consumption of modern virtualization technology is comparable and negligible. For example, in the experimental study described in [43], the consumption for Docker and LXC (i.e., container orchestrators) was measured to be around 1 W, while the consumption for Zen and KVM (i.e., VM orchestrators) was measured to be just slightly higher, around 4 W.
- The power consumption values for firewall VNFs are randomly selected in the range between 4 and 30 W. These values are derived from the empirical results presented in [43]. According to that analysis, in idle and CPU/memory stress test scenarios, the additional power drawn per VM or container with respect to the virtualization platform consumption is minimal, on average around 4–6 W. Instead, it can increase to around 25–30 W under network-intensive workloads.

All the findings of this comparison are reported in Fig. 3. Concerning the total power consumption of the solutions computed by GreenShield and the two manual configuration strategies, averaged on the 100 variations of each use case , Fig. 3a shows a significant reduction in average power consumption achieved by GreenShield-E2C, up to two orders of magnitude difference. This result confirms its ability to optimize sustainability while guaranteeing the satisfaction of all security policies, and shows that GreenShield-E2C is even more effective in addressing the firewall configuration problem in the bigger use cases, which are the ones commonly occurring in the edge-to-cloud continuum scenarios, such as smart cities. These achievements were possible thanks to the soft constraints that force the usage of firewalls in such a way as to minimize the power consumption. The impact of these constraints is also reflected in the number of used firewalls in the ten use cases by the three solutions, charted in Fig. 3b. Instead, GreenShield-E2C achieves a



(a) Total power consumption (b) Firewall usage status (c) Average power consumption per firewall

Fig. 3. Comparison between GreenShield-E2C and manual configuration strategies.

significant reduction in this parameter with respect to the two manual strategies. In particular, strategy (a) uses all possible firewalls, while strategy (b) uses fewer firewalls. However, the large majority of them are physical firewalls at the edge, with higher power consumption on average than virtual ones. This difference is also highlighted by Fig. 3c, which shows that the average power consumption per firewall required by the configuration computed by GreenShield-E2C is always one magnitude order less than that of the solutions produced by the two manual strategies. In this third chart, the plot related to GreenShield-E2C has a decreasing trend (i.e., it goes from 337W to 306W) because the slope of its number of activated firewalls (used as the denominator) is steeper than the slope of its total power consumption (used as the numerator), as was also expected.

6.1.2. Comparison with a GreenShield-E2C version without soft constraints

Second, an ablation analysis was carried out by comparing GreenShield-E2C with a reduced version, where all soft constraints, i.e., all clauses related to the green optimization, are removed. This special variant of the framework is based on a standard SMT formulation of the firewall configuration problem, as it is composed only of hard constraints. Therefore, even if it formally guarantees the satisfaction of the input NSPs, it does not pursue theoretically any optimization objective related to power consumption.

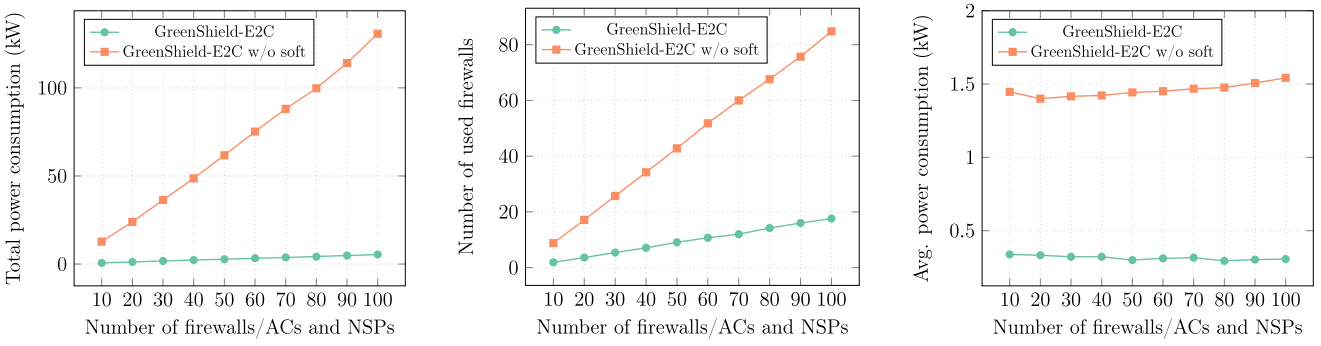
This comparison was conducted on the same 100 variations of the 10 use cases, which progressively increased in size, already used in the previous baseline analysis. The results of these comparative tests are reported in Fig. 4. As is already evident from Fig. 4a, removing the soft constraints results in a significant increase in the total power consumption of the computed firewall configuration. In particular, the total power consumption of the solutions obtained without soft constraints is consistently at least one order of magnitude higher than that achieved by GreenShield-E2C across all problem sizes. For the largest size, it becomes two orders of magnitude higher, and is only slightly inferior to the results achieved by manual configuration strategies. Fig. 4b further shows that the absence of soft constraints also impacts the number of activated firewalls. The GreenShield-E2C version without soft constraints tends to activate a significantly larger number of firewall instances, as the solver is no longer guided toward minimizing the usage of physical firewalls or the deployment of virtual ones. Moreover, Fig. 4b confirms that the average power consumption per activated firewall is also substantially higher when soft constraints are disabled.

The motivation of these results is that, without being driven by soft constraints, the solver simply outputs a solution compliant with all hard constraints, which is generally the first correct solution found in its search in the problem solution space. As a consequence, this solution may be heavily underoptimized: it may activate a higher number of firewalls than necessary, and it may also choose energy-intensive firewall instances, even when lower-power alternatives could be used to enforce the same input NSPs. In conclusion, this ablative analysis showed that the soft constraints introduced in GreenShield-E2C are essential to effectively achieve green optimization for distributed firewall configuration in the edge-to-cloud continuum.

6.1.3. Comparison with greenshield

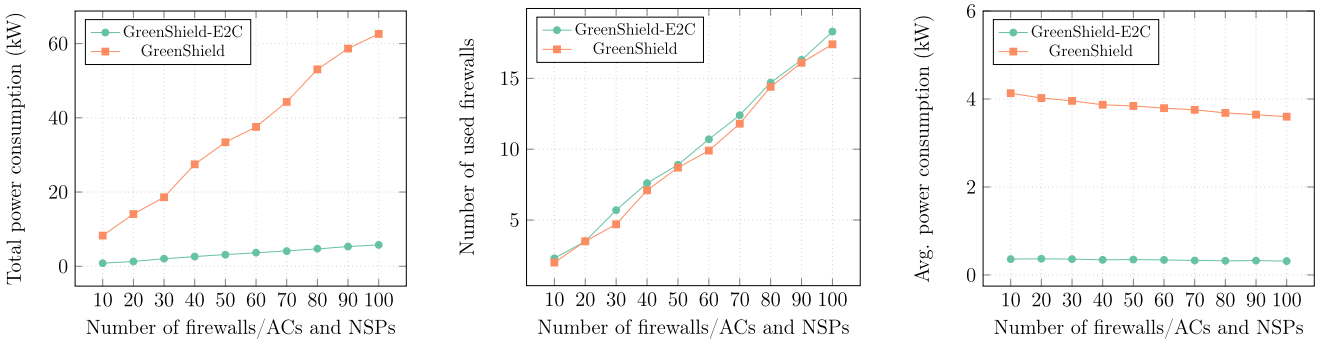
Third, GreenShield-E2C was compared with the original GreenShield approach from which it derives. The same 100 variations of the already described 10 use cases were reused, but with two slight modifications related to test execution and to the NSP generation, both due to the fact that GreenShield can only compute configurations for physical firewalls:

- Each use case is characterized by a physical network topology and a logical topology. In the execution of these comparative tests, both



(a) Total power consumption (b) Firewall usage status (c) Average power consumption per firewall

Fig. 4. Comparison between GreenShield-E2C with and without soft constraints.



(a) Total power consumption (b) Firewall usage status (c) Average power consumption per firewall

Fig. 5. Comparison between GreenShield-E2C and GreenShield.

are fed to GreenShield-E2C, while only the physical topology is given as input to GreenShield.
- The NSPs are generated ensuring that there is at least a physical firewall in the path of each flow identified by the condition of an NSP.

The results of these comparative experiments are reported in Fig. 5.

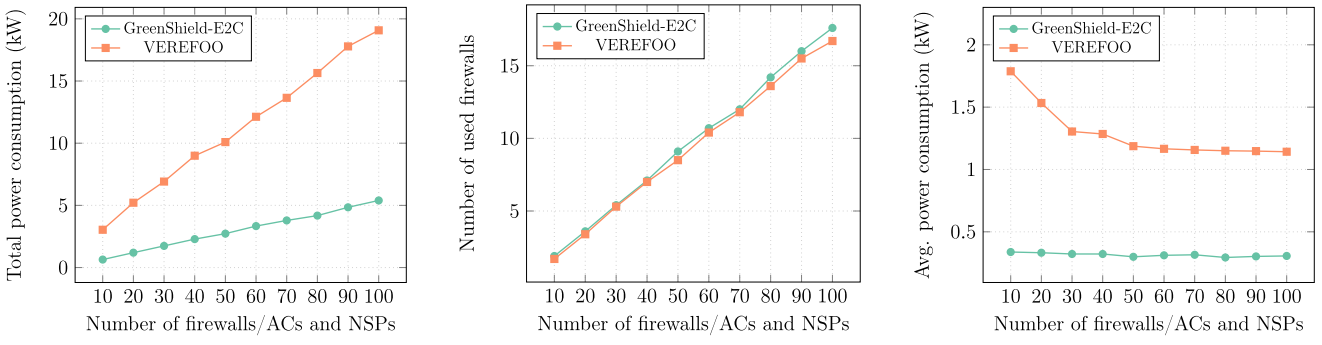
GreenShield consistently exhibits higher power consumption than GreenShield-E2C across all problem sizes, as shown in Fig. 5a. This behavior is explained by the fact that GreenShield is restricted to using only physical firewalls, which are typically less energy-efficient, for the resolution of the firewall configuration problem. In contrast, GreenShield-E2C is capable of exploring a broader solution space that includes virtual firewalls to be allocated on general-purpose servers, whose energy usage is inferior on average. As the problem size increases, i.e., with a higher number of firewalls/ACs and NSPs characterizing the use cases, the power consumption gap between the two approaches becomes more pronounced. This trend highlights the scalability advantage of GreenShield-E2C, which proves increasingly more effective in larger scenarios with an even more extended solution space.

Interestingly, as shown in Fig. 5b, GreenShield generally activates slightly fewer firewalls than GreenShield-E2C. However, this does not translate into better energy efficiency, as illustrated in Fig. 5c. On the one hand, the average power consumption per activated firewall is significantly higher for GreenShield, once again due to its exclusive reliance on physical firewalls. On the other hand, GreenShield-E2C, despite using more firewalls, benefits from including virtualized ones. Hence, it derives that a firewall allocation scheme output by GreenShield-E2C may include more firewall instances than the scheme computed by GreenShield for the same use case, but both the total and per-firewall average power consumptions are less in the former than in the latter, because the consumption of a single physical firewall is higher than the sum of consumptions of more virtual ones.

6.1.4. *Comparison with VEREFOO*

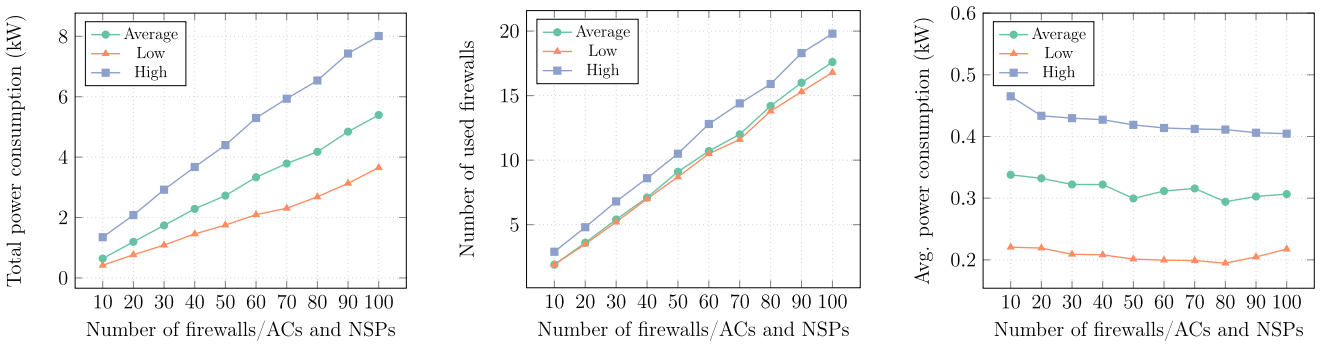
Fourth, GreenShield-E2C was compared with a state-of-the-art automatic firewall configuration mechanism, VEREFOO [31]. The reasons why VEREFOO was selected for this comparison are that, as already discussed in Section 2, it is the most feature-complete solution for normal virtual computer networks, it can jointly compute both the firewall allocation scheme and the filtering rule sets, and its code is available as open source [57]. The same use cases already employed for previous tests were reused for this comparison. However, concerning the physical and logical graphs characterizing those scenarios, only the logical ones were fed to VEREFOO. This simplification is motivated by the fact that VEREFOO is not designed to solve the logical-to-physical embedding problem. In fact, it does not model the relationship between the logical firewall allocation scheme and the underlying physical infrastructure. Therefore, it cannot reason about physical resource constraints, virtualization technologies, or power consumption aspects related to firewall deployment, and it operates exclusively at the level of virtual network topologies.

The results of these tests are plotted in Fig. 6. As the goal of VEREFOO consists of minimizing the absolute number of allocated firewalls in the virtual topology, it is always able to produce solutions with fewer firewalls than GreenShield-E2C in each use case, as can be noted from Fig. 6b. Nevertheless, the difference in the number of activated firewalls between VEREFOO and GreenShield-E2C is relatively small, and it does not exceed one firewall instance, even for the largest problem sizes. Most importantly, VEREFOO completely overlooks power consumption aspects when selecting which firewalls to allocate. As a consequence, it tends to prefer solutions that activate fewer firewall instances at the cost of relying on more energy-intensive ones. This behavior is clearly reflected in the results drawn in Fig. 6a and Fig. 6c. In particular, the total power consumption of the configurations computed by VEREFOO is consistently between three and five times higher than the one achieved by GreenShield-E2C across the different problem



(a) Total power consumption (b) Firewall usage status (c) Average power consumption per firewall

Fig. 6. Comparison between GreenShield-E2C and VEREFOO.



(a) Total power consumption (b) Firewall usage status (c) Average power consumption per firewall

Fig. 7. GreenShield-E2C results under low, average, and high operating conditions.

sizes. A similar trend is observed for the average power consumption per activated firewall, which is approximately four to six times higher for VEREFOO.

These results confirm that the strategy followed by VEREFOO, which involves minimizing the absolute number of firewalls, is insufficient to achieve energy-efficient configurations in edge-to-cloud environments, even without considering its limited applicability. Therefore, GreenShield-E2C achieves better results while addressing a more comprehensive optimization problem than existing automatic firewall configuration approaches.

6.1.5. Comparison among different traffic conditions

The GreenShield-E2C approach allows users to assign different power consumption values to the firewalls, so as to represent varying traffic load conditions and to tackle dynamic conditions. In order to experimentally assess this capability, we evaluated the proposed approach under three different operating conditions, representing low, average, and high traffic loads.

Starting from the use cases described in Section 6.1.1, these three conditions were modeled by instantiating power consumption values from three partially overlapping ranges derived from the same baseline intervals that were already defined for physical firewall power consumption, servers and virtualization platform consumption, and firewall VNFs consumption. The lower range represents low-load operating conditions, the upper range represents high-load or peak traffic situations (e.g., traffic bursts), and the central range represents average traffic conditions. The partial overlap among these ranges avoids unrealistic discontinuities between scenarios and captures moderate variability around typical operating points. For each one of the 10 use cases, 100 different power consumption values were extracted from these ranges, so as to compute averages of the results.

The developed framework was applied to all these use cases, and the test results are reported in Fig. 7. In general, GreenShield-E2C requires more firewalls and leads to higher power consumption for managing high-load situations. This evidence is explained by the different feasibility of optimal solutions under the considered traffic profiles. Under low and average traffic conditions, it is often possible to identify an optimal configuration in which a limited number of firewall instances, characterized by moderate power consumption, can be deployed on a single server (or a limited number of them), while combined power consumption remains below the server's threshold. Under high-load conditions, this same configuration may no longer be feasible. In fact, when firewall instances are associated with higher power consumption values, closer to their maximum operational levels, their deployment on a single server may violate the corresponding power cap. Consequently, the solver is forced to identify alternative configurations in which the firewalling functionality is distributed across a larger number of firewall instances deployed on different servers. This redistribution increases the number of activated firewalls and leads to higher overall power consumption, as observed in the high-load results.

Nevertheless, the observed differences among the three traffic profiles are relatively limited across all use cases, so GreenShield-E2C exhibits consistent behavior when transitioning across different operating conditions. Moreover, this characteristic suits the practical usage discussed in Section 3, in which a network administrator can precompute a small set of optimal firewall configurations corresponding to different traffic profiles and dynamically switch among them according to the current operating conditions, so as to combine adaptability with controlled configuration management.

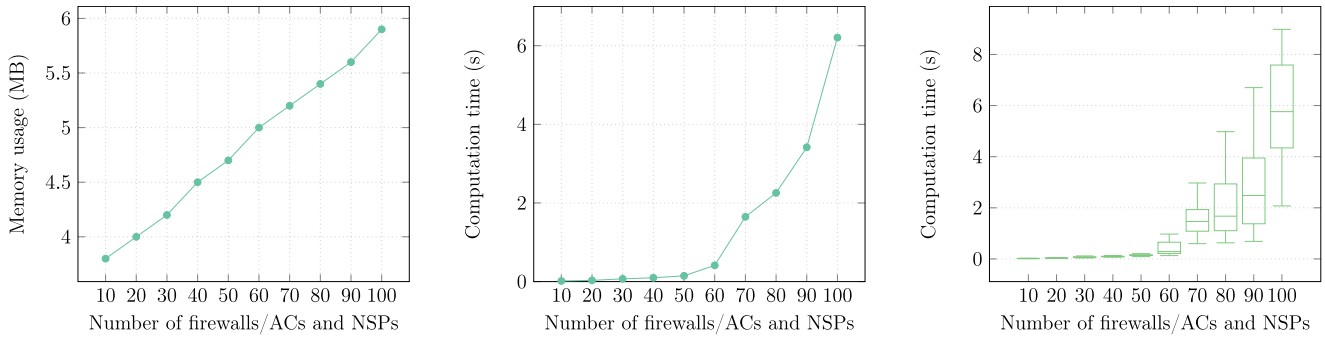
6.2. Scalability validation

To assess the practicality of GreenShield-E2C, we conducted a detailed analysis of its scalability in terms of memory and time performance. These aspects were crucial in determining whether the framework can support large-scale edge-to-cloud topologies and big NSP sets without introducing excessive configuration computation delays. This analysis was carried out by executing GreenShield-E2C on the same 100 variations of the 10 use cases employed for the comparison with the two manual configuration strategies, and by experimentally measuring the memory usage and computation time. The results are reported in Fig. 8.

Concerning memory usage, Fig. 8a reports the average estimate over the 100 iterations of each use case. The plot shows that its growth with respect to the configuration problem size remains linear and moderate, especially when considering that the most complex scenario (i.e., the one with 100 firewalls/Acs and 100 NSPs) required less than 6 GB of RAM. This key observation confirms that the memory footprint of GreenShield-E2C remains well within the capabilities of any computational device.

Concerning computation time, Fig. 8b similarly reports the average estimate over the 100 iterations of each use case. Execution time grows with increasing problem size, as expected. However, the trend remains within acceptable bounds. Even in the most complex scenario analyzed, GreenShield-E2C is able to compute a complete, formally correct, and optimized firewall configuration in less than 6 seconds. This demonstrates that the MaxSMT-based formalism, when paired with a state-of-the-art solver, is effective in delivering scalable automation without incurring excessive computational overhead. Moreover, the results validate two main design features of the proposed approach, i.e., the careful formal modeling, which avoids undecidable formulations, and the separation between hard and soft constraints.

To complement the average-case analysis, the box plots charted in Fig. 8c provide additional insight into the variability of GreenShield-E2C's computation time across the 100 test iterations for each use case. The distributions show limited variance, with most execution times concentrated near the median and narrow inter-quartile ranges. This confirms that the framework behaves consistently even in the presence of



(a) Memory usage (b) Time validation (c) Box plot

Fig. 8. Scalability validation.

randomized NSP sets, and is not prone to unpredictable performance spikes. Notably, no significant outliers were recorded even in the largest scenarios, indicating robust worst-case performance.

The time scalability of GreenShield-E2C is also in line with, and often better than, the ones characterizing state-of-the-art proposals of the literature. Looking at the results reported in the papers describing the most feature-complete alternative solutions, ConfigSynth [29] requires around 20 s just to establish the allocation scheme for 20 firewalls and other security devices without their operational rules, VEREFOO [31] needs approximately 90 s for computing the full, yet non-energy-optimized configuration of 100 firewalls, and the original Greenshield [9] takes up to 10 s to configure 100 physical firewalls. In comparison, GreenShield-E2C is able to find the optimal solution for a problem characterized by 100 ACs and 100 NSPs in slightly more than 6 s, while providing more green-oriented optimization features than alternative techniques.

6.3. Synthesis of the validation results

The experimental evaluation collectively confirms the effectiveness and practicality of GreenShield-E2C across optimization, comparison, and scalability dimensions.

First, the comparison with manual strategies shows substantial energy savings, with total power consumption reduced by up to two orders of magnitude in larger scenarios. The ablation analysis further demonstrates th significance of the soft constraints: without them, the solver still produces correct configurations, but with at least one order of magnitude higher power consumption.

Second, the comparison with GreenShield and VEREFOO showcases the superiority of GreenShield-E2C over them. Unlike GreenShield, which optimizes only physical firewalls, GreenShield-E2C exploits the full edge-to-cloud continuum and achieves consistently lower total and per-firewall consumption. Instead, unlike VEREFOO, which minimizes the number of firewalls without considering energy, GreenShield-E2C reduces total power consumption typically by a factor between three and five, despite activating a comparable number of instances. This shows that minimizing firewall count does not imply minimizing energy consumption in heterogeneous infrastructures.

Third, the evaluation under low, average, and high traffic conditions demonstrates that GreenShield-E2C adapts its allocation strategy coherently with varying power consumption profiles. Moreover, the differences across operating profiles remain contained, supporting the feasibility of precomputing a small set of optimized configurations and dynamically switching among them in response to traffic fluctuations.

Finally, scalability results indicate that the approach remains practical for large scenarios. Even with 100 firewalls/ACs and 100 NSPs, the framework computes a formally verified and energy-optimized configuration in about 6 seconds, using less than 6 GB of RAM.

7. Conclusions and future work

This paper presented GreenShield-E2C, an approach combining automation, formal verification and sustainability optimization to address the distributed firewall configuration problem in network topologies of edge-to-cloud environments. The adopted MaxSMT formulation allows finding the configuration solution that minimizes the power consumption related to the usage of physical firewalls and the allocation of virtual ones, while ensuring the satisfaction of all network security policies specified by the administrator.

The experimental validation of the framework implementing the GreenShield-E2C approach showcased its effectiveness in terms of sustainability optimization and scalability to large-scale scenarios, also compared with other manual and automatic configuration strategies. In particular, GreenShield-E2C achieves total energy savings of one to two orders of magnitude compared to non-optimized configurations (ablation without soft constraints), and a reduction by a factor of approximately 3–5 compared to VEREFOO, despite activating a comparable number of firewall instances. Concerning scalability, even in the largest evaluated scenarios (100 firewalls/ACs and 100 NSPs), the framework computes a formally verified and energy-optimized configuration in the order of a few seconds (about 6 s), with memory usage below 6 GB, confirming the practical applicability of the approach to large edge-to-cloud environments.

Future work will explore extending the approach to support multi-objective optimization as a complementary research direction, requiring the explicit modeling and resolution of trade-offs among heterogeneous and potentially conflicting goals, such as low latency, high reliability, and cost controllability. Moreover, GreenShield-E2C will also be extended to the automatic configuration of other network security functions such as VPN gateways.

CRediT authorship contribution statement

Daniele Bringhenti: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization; **Fulvio Valenza:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R.B. Bohn, J. Messina, F. Liu, J. Tong, J. Mao, NIST Cloud computing reference architecture, in: World Congress on Services, SERVICES 2011, Washington, DC, USA, July 4–9, 2011, IEEE Computer Society, 2011, pp. 594–596. https://doi.org/10.1109/SERVICES.2011.105

[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: a complete survey, J. Syst. Archit. 98 (2019) 289–330. https://doi.org/10.1016/J.SYSARC.2019.02.009

[3] D. Rosendo, A. Costan, P. Valduriez, G. Antoniu, Distributed intelligence on the edge-to-Cloud continuum: a systematic literature review, J. Parallel Distributed Comput. 166 (2022) 71–94. https://doi.org/10.1016/J.JPDC.2022.04.004

[4] M.M. Hasan, T. Sultana, M.D. Hossain, A.K. Mandal, N.T. Thu, G.-W. Lee, E.-N. Huh, The journey to cloud as a continuum: opportunities, challenges, and research directions, ICT Express (2025). https://doi.org/10.1016/j.icte.2025.04.015

[5] M. Jangjou, M.K. Sohrabi, A comprehensive survey on security challenges in different network layers in cloud computing, Arch. Comput. Methods Eng. 29 (6) (2022) 3587–3608. https://doi.org/10.1007/s11831-022-09708-9

[6] A. Voronkov, L.H. Iwaya, L.A. Martucci, S. Lindskog, Systematic literature review on usability of firewall configuration, ACM Comput. Surv. 50 (6) (2018) 87:1–87:35. https://doi.org/10.1145/3130876

[7] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, Automation for network security configuration: state of the art and research trends, ACM Comput. Surv. 56 (3) (2024) 57:1–57:37. https://doi.org/10.1145/3616401

[8] D. Bringhenti, S. Bussa, R. Sisto, F. Valenza, Atomizing firewall policies for anomaly analysis and resolution, IEEE Trans. Dependable Secur. Comput. 22 (3) (2025) 2308–2325. https://doi.org/10.1109/TDSC.2024.3495230

[9] D. Bringhenti, F. Valenza, GreenShield: optimizing firewall configuration for sustainable networks, IEEE Trans. Netw. Serv. Manag. 21 (6) (2024) 6909–6923. https://doi.org/10.1109/TNSM.2024.3452150

[10] Y. Bartal, A.J. Mayer, K. Nissim, A. Wool, *Firmato*: a novel firewall management toolkit, ACM Trans. Comput. Syst. 22 (4) (2004) 381–420. https://doi.org/10.1145/1035582.1035583

[11] F. Cuppens, N. Cuppens-Boulahia, T. Sans, A. Miège, A formal approach to specify and deploy a network security policy, in: Proc. of Formal Aspects in Security and Trust: Second IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST), an Event of the 18Th IFIP World Computer Congress, August 22–27, 2004, Toulouse, France, 173, 2004, pp. 203–218. https://doi.org/10.1007/0-387-24098-5_15

[12] P. Verma, A. Prakash, FACE: A firewall analysis and configuration engine, in: Proc. of the 2005 IEEE/IPSJ International Symposium on Applications and the Internet (SAINT 2005), 31 January - 4 February 2005, Trento, Italy, IEEE Computer Society, 2005, pp. 74–81. https://doi.org/10.1109/SAINT.2005.28

[13] N.B. Youssef, A. Bouhoula, A fully automatic approach for fixing firewall misconfigurations, in: Proc. of the 11Th IEEE International Conference on Computer and Information Technology, CIT 2011, Pafos, Cyprus, 31 August-2 September 2011, 2011, pp. 461–466. https://doi.org/10.1109/CIT.2011.84

[14] K. Adi, L. Hamza, L. Pene, Automatic security policy enforcement in computer systems, Comput. Secur. 73 (2018) 156–171. https://doi.org/10.1016/J.COSE.2017.10.012

[15] D. Ranathunga, M. Roughan, P. Kernick, N. Falkner, The mathematical foundations for mapping policies to network devices, in: Proc. of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26–28, 2016, 2016, pp. 197–206. https://doi.org/10.5220/0005946201970206

[16] A. El-Hassany, P. Tsankov, L. Vanbever, M.T. Vechev, Netcomplete: practical network-Wide configuration synthesis with autocompletion, in: S. Banerjee, S. Seshan (Eds.), Proc. of the 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9–11, 2018, 2018, pp. 579–594. https://www.usenix.org/conference/nsdi18/presentation/el-hassany.

[17] M. Jiménez-Lázaro, J. Berrocal, J. Galán-Jiménez, Deep reinforcement learning based method for the rule placement problem in software-Defined networks, in: 2022 IEEE/IFIP Network Operations and Management Symposium, NOMS 2022, Budapest, Hungary, April 25–29, 2022, IEEE, 2022, pp. 1–4. https://doi.org/10.1109/NOMS54207.2022.9789906

[18] N. Wintering, E. Lanfer, N. Aschenbruck, Automating network perimeter threat prevention for decentralized network administration, in: P. Varga, P. Celeda, T. Wauters, M. Tortonesi, J. François, J. Galán-Jiménez (Eds.), 20th International Conference on Network and Service Management, CNSM 2024, Prague, Czech Republic, October 28–31, 2024, IEEE, 2024, pp. 1–7. https://doi.org/10.23919/CNSM62983.2024.10814436

[19] W. Zahwa, A. Lahmadi, M. Rusinowitch, M. Ayadi, In-Network ACL rules placement using deep reinforcement learning, in: IEEE International Mediterranean Conference on Communications and Networking, MeditCom 2024, Madrid, Spain, July 8–11, 2024, IEEE, 2024, pp. 341–346. https://doi.org/10.1109/MEDITCOM61057.2024.10621188

[20] H. Sun, X. Liao, J. Wang, Q. Qi, Z. Zhuang, J. Liao, D.O. Wu, Fast and scalable ACL policy solving under complex constraints with graph neural networks, IEEE/ACM Trans. Netw. 32 (5) (2024) 4175–4190. https://doi.org/10.1109/TNET.2024.3409529

[21] W. Zahwa, A. Lahmadi, M. Rusinowitch, M. Ayadi, Deep reinforcement learning for in-Network placement of ACL rules under constraints, in: 2025 21St International Conference on Network and Service Management (CNSM), Bologna, Italy, October 27–31, 2025, IEEE, 2025, pp. 1–7. https://doi.org/10.23919/CNSM67658.2025.11297518

[22] D. Bhamare, R. Jain, M. Samaka, A. Erbad, A survey on service function chaining, J. Netw. Comput. Appl. 75 (2016) 138–155. https://doi.org/10.1016/J.JNCA.2016.09.001

[23] Y. Han, J. Li, D. Hoang, J. Yoo, J.W. Hong, An intent-based network virtualization platform for SDN, in: Proc. of the 12th International Conference on Network and Service Management, CNSM, 2016, pp. 353–358. https://doi.org/10.1109/CNSM.2016.7818446

[24] E.J. Scheid, C.C. Machado, M.F. Franco, R.L. dos Santos, R.J. Pfitscher, A.E.S. Filho, L.Z. Granville, INSpIre: integrated NFV-based intent refinement environment, in: Proc. of the IFIP/IEEE Symp. on Integrated Network and Service Management (IM17), 2017. https://doi.org/10.23919/INM.2017.7987279

[25] Z. Hao, Z. Lin, R. Li, A SDN/NFV security protection architecture with a function composition algorithm based on trie, in: Proc. of the 2Nd Intern. Conf. on Computer Science and Application Engineering (CSAE18), 2018. https://doi.org/10.1145/3207677.3277992

[26] Y. Liu, Y. Lu, W. Qiao, X. Chen, A dynamic composition mechanism of security service chaining oriented to SDN/NFV-Enabled networks, IEEE Access 6 (2018). https://doi.org/10.1109/ACCESS.2018.2870601

[27] A.S. Sendi, Y. Jarraya, M. Pourzandi, M. Cheriet, Efficient provisioning of security service function chaining using network security defense patterns, IEEE Trans. Services Comput. 12 (4) (2019). https://doi.org/10.1109/TSC.2016.2616867

[28] M. Yoon, S. Chen, Z. Zhang, Minimizing the maximum firewall rule set in a network with multiple firewalls, IEEE Trans. Comput. 59 (2) (2010). https://doi.org/10.1109/TC.2009.172

[29] M.A. Rahman, E. Al-Shaer, Automated synthesis of distributed network access controls: a formal framework with refinement, IEEE Trans. Parallel Distrib. Syst. 28 (2) (2017). https://doi.org/10.1109/TPDS.2016.2585108

[30] N. Schnepf, R. Badonnel, A. Lahmadi, S. Merz, Rule-Based synthesis of chains of security functions for software-Defined networks, Electron. Commun. Eur. Assoc. Softw. Sci. Technol. 76 (2018). https://doi.org/10.14279/TUJ.ECEASST.76.1075

[31] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, J. Yusupov, Automated firewall configuration in virtual networks, IEEE Trans. Dependable Secur. Comput. 20 (2) (2023) 1559–1576. https://doi.org/10.1109/TDSC.2022.3160293

[32] F. Pizzato, D. Bringhenti, R. Sisto, F. Valenza, Automatic and optimized firewall reconfiguration, in: NOMS 2024 IEEE Network Operations and Management Symposium, Seoul, Republic of Korea, May 6–10, 2024, IEEE, 2024, pp. 1–9. https://doi.org/10.1109/NOMS59830.2024.10575212

[33] A. Singh, N. Auluck, O.F. Rana, A.C. Jones, S. Nepal, Scheduling real-Time security aware tasks in fog networks, IEEE Trans. Serv. Comput. 14 (6) (2021) 1981–1994. https://doi.org/10.1109/TSC.2019.2914649

[34] A. Taghinezhad-Niar, J. Taheri, Security, reliability, cost, and energy-Aware scheduling of real-Time workflows in compute-Continuum environments, IEEE Trans. Cloud Comput. 12 (3) (2024) 954–965. https://doi.org/10.1109/TCC.2024.3426282

[35] G.L. Stavrinides, H.D. Karatza, Security, cost and energy aware scheduling of real-Time IoT workflows in a mist computing environment, Inf. Syst. Frontiers 26 (4) (2024) 1223–1241. https://doi.org/10.1007/S10796-022-10304-2

[36] A. Akbar, S. Jangsher, F.A. Bhatti, NOMA And 5G emerging technologies: a survey on issues and solution techniques, Comput. Networks 190 (2021) 107950. https://doi.org/https://doi.org/10.1016/j.comnet.2021.107950

[37] R. She, M. Sun, Security energy efficiency analysis of CR-NOMA enabled IoT systems for edge-cloud environment, Int. J. Comput. Intell. Syst. 16 (1) (2023) 118. https://doi.org/10.1007/S44196-023-00273-Y

[38] I. Fé, T.A. Nguyen, M.D. Mauro, F. Postiglione, A. Ramos, A. Soares, E. Choi, D. Min, J. Lee, F.A. Silva, Energy-aware dynamic response and efficient consolidation strategies for disaster survivability of cloud microservices architecture, Computing 106 (8) (2024) 2737–2783. https://doi.org/10.1007/S00607-024-01305-X

[39] FortiGate 7000 Series Datasheet, Fortinet, 2025. Visited: 2025-12-27, https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_7000_Series_Bundle.pdf.

[40] Palo Alto Networks PA-400 Series Datasheet, Palo Alto Networks, 2023. Visited: 2025-12-27, https://www.paloaltonetworks.com/resources/datasheets/pa-400-series-pan-os-11-0.

[41] A. Shehabi, A. Hubbard, A. Newkirk, N. Lei, M.A.B. Siddik, B. Holecek, J. Koomey, E. Masanet, D. Sartor, et al., 2024 United states data center energy usage report, eScholarship Publishing (2024). https://doi.org/10.71468/P1WC7Q

[42] SPECpower™ Benchmark – Power Consumption and Energy Efficiency for Servers, SPEC.org, 2025. Visited: 2025-12-27, https://www.spec.org/30th/power.html.

[43] R. Morabito, Power consumption of virtualization technologies: an empirical investigation, in: I. Raicu, O.F. Rana, R. Buyya (Eds.), 8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, December 7–10, 2015, 2015, pp. 522–527. https://doi.org/10.1109/UCC.2015.93

[44] C. Jiang, Y. Wang, D. Ou, Y. Li, J. Zhang, J. Wan, B. Luo, W. Shi, Energy efficiency comparison of hypervisors, Sustain. Comput. Informatics Syst. 22 (2019) 311–321. https://doi.org/10.1016/J.SUSCOM.2017.09.005

[45] C. Xu, Z. Zhao, H. Wang, R. Shea, J. Liu, Energy efficiency of cloud virtual machines: from traffic pattern and CPU affinity perspectives, IEEE Syst. J. 11 (2) (2017) 835–845. https://doi.org/10.1109/JSYST.2015.2429731

[46] M. Warade, K. Lee, C. Ranaweera, J. Schneider, Monitoring the energy consumption of docker containers, in: 47th IEEE Annual Computers, Software, and Applications Conference, COMPSAC 2023, Torino, Italy, June 26–30, 2023, IEEE, 2023, pp. 1703–1710. https://doi.org/10.1109/COMPSAC57700.2023.00263

[47] E. Al-Shaer, H.H. Hamed, R. Boutaba, M. Hasan, Conflict classification and analysis of distributed firewall policies, IEEE J. Sel. Areas Commun. 23 (10) (2005) 2069–2084. https://doi.org/10.1109/JSAC.2005.854119

[48] H. Hu, G. Ahn, K. Kulkarni, Detecting and resolving firewall policy anomalies, IEEE Trans. Dependable Secur. Comput. 9 (3) (2012) 318–331. https://doi.org/10.1109/TDSC.2012.20

[49] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, D. Huang, Brew: a security policy analysis framework for distributed SDN-Based cloud environments, IEEE Trans. Dependable Secur. Comput. 16 (6) (2019) 1011–1025. https://doi.org/10.1109/TDSC.2017.2726066

[50] S. Tanzarella, M. Repetto, Context discovery for digital service chain with openc2, in: 11th IEEE International Conference on Network Softwarization, NetSoft 2025, Budapest, Hungary, June 23–27, 2025, IEEE, 2025, pp. 579–584. https://doi.org/10.1109/NETSOFT64993.2025.11080629

[51] H. Yang, S.S. Lam, Real-Time verification of network properties using atomic predicates, IEEE/ACM Trans. Netw. 24 (2) (2016) 887–900. https://doi.org/10.1109/TNET.2015.2398197

[52] H. Yang, S.S. Lam, Scalable verification of networks with packet transformers using atomic predicates, IEEE/ACM Trans. Netw. 25 (5) (2017) 2900–2915. https://doi.org/10.1109/TNET.2017.2720172

[53] P. Zhang, X. Liu, H. Yang, N. Kang, Z. Gu, H. Li, APKeep: Realtime verification for real networks, in: 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25–27, 2020, USENIX Association, 2020, pp. 241–255. https://www.usenix.org/conference/nsdi20/presentation/zhang-peng.

[54] D. Bringhenti, S. Bussa, R. Sisto, F. Valenza, A two-Fold traffic flow model for network security management, IEEE Trans. Netw. Serv. Manag. 21 (4) (2024) 3740–3758. https://doi.org/10.1109/TNSM.2024.3407159

[55] L.M. de Moura, N.S. Bjørner, Z3: An efficient SMT solver, in: Proc. of the Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008, 4963 of *Lecture Notes in Computer Science*, 2008, pp. 337–340. https://doi.org/10.1007/978-3-540-78800-3_24

[56] I.A.T. Hashem, A. Siddiqa, F.A. Alaba, M. Bilal, S.M. Alhashmi, Distributed intelligence for IoT-based smart cities: a survey, Neural Comput. Appl. 36 (27) (2024) 16621–16656. https://doi.org/10.1007/S00521-024-10136-Y

[57] VEREFOO: Open-source implementation. Visited: 2025-12-27, https://github.com/netgroup-polito/verefoo.