Automated Security-focused Network Configuration Management: State of the Art, Challenges, and Future Directions

Jinglu Xu[1] and Giovanni Russello[2]
[1,2]The University of Auckland, Auckland, New Zealand
jxu671@aucklanduni.ac.nz, g.russello@auckland.ac.nz

***Abstract*—With the continuous advancements and innovation in Information Technology (IT), new vendors and products constantly emerge to provide networking services and solutions. Meanwhile, the multi-vendor environment and the huge diversity of siloed devices require new approaches to tackle the complexity and heterogeneity in network configuration management. However, the traditional way of using configuration scripts requires domain expertise with the target system. In recent years, many automated solutions have emerged in academia and industry. This paper presents a literature review on the state-of-the-art of them. It focuses on the background that led to the development of automated techniques, summarizes domain-specific c hallenges, a nd d iscusses r elated s tudies p ublished to date. Finally, five r esearch g aps w ith r esearch d irections are identified, i ncluding 1) i ntent t ranslation, 2) i ntelligent network configuration m anagement, 3) a utomated p lanning, 4) intent-based solutions for I2NSF, and 5) reliable network performance.**

***Keywords*—network, security, configuration, automation.**

I. Introduction

With the fast development of the networking industry, many vendors have been developing new devices and services. Accordingly, various protocols and technologies have emerged rapidly, bringing along diverse new terminologies and conceptualizations. As a result, network configuration management has become a complex process. Meanwhile, security plays a vital role in network configuration a nd n eeds t o b e addressed throughout the management life cycle, which further increases the complexity. In recent years, many research efforts have been put into developing automated solutions to simplify configuration m anagement t asks. S ome researchers proposed policy-based approaches to guide network behaviour; others developed methods based on autonomic or intent-based networking; others again investigated constraint-based approaches. There are also several efforts from the industry [1], [2], [3]. This plethora of approaches makes it difficult to get a clear picture of the fundamental properties of automated network configuration management. To the best of our knowledge, this field lacks a comprehensive review of existing work. This paper aims to address this gap and provide researchers interested in this subject with a solid starting point. Our main contributions are as follows:

1) We review, classify, and compare automated security-focused network configuration management techniques in academia and industry.
2) We thoroughly discuss the remaining challenges and issues in existing automated security-focused network configuration management techniques.
3) We identify several research gaps in this field and point out future research directions.

The rest of this paper is organized as follows. Section 2 gives an overview of security-focused network configuration management. Section 3 describes the significant contributions of academia and industry while providing insights into the remaining challenges. Section 4 summarizes the findings and identifies research gaps with research directions. Finally, we conclude this paper in Section 5.

II. An Overview of Security-Focused Network Configuration Management

Before examining different techniques in this subject, we should clarify the meaning and understand its value. This section establishes a standard definition of security-focused network configuration management and provides a historical view of its development.

A. Security-Focused Network Configuration Management Definition

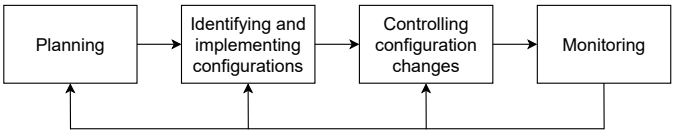

Fig. 1. Security-Focused Configuration Management Phases. Source: Adapted from [4].

The National Institute of Standards and Technology (NIST) defines security-focused configuration management as *"the management and control of configurations for systems to enable security and facilitate the management of information security risk."* [4] As shown in Fig. 1, they summarize the life cycle consists of four phases: 1) planning, 2) identifying and implementing configurations, 3) controlling configuration changes, and 4) monitoring. In NIST's definition, configuration items can be any elements that compose the information system. It is such a broad topic that we cannot cover everything in a single paper. Therefore, we adopt NIST's management life

2767-6684/22/$31.00 ©2022 IEEE
DOI 10.1109/DSA56465.2022.00061

cycle but limit the scope to network components. Security-focused network configuration management in this context is defined as *"a process for planning, establishing, maintaining, and monitoring network configuration with the focus on network security."*

B. Script-Based Network Configuration Management

The most traditional way of configuring network settings is through scripts. Most enterprise networks these days are built from devices manufactured by different vendors. However, the move to a network configuration standard is long overdue. So far, various terminologies and frameworks have been developed by vendors to allow device-to-device communication. Accordingly, different programming languages have been created for human-to-device interaction. Writing scripts has become challenging as their syntax and semantics are specific to each configuration environment. To address the heterogeneity problem, the Internet Engineering Task Force (IETF) proposed the Simple Network Management Protocol (SNMP) standard [5] that could collect information about network devices and services in a manner not bound to a particular manufacturer. In SNMP, every piece of information is an object with a unique identifier called Object Identifier (OID). The Management Information Base translates an OID into human-readable information, including names, definitions, and object descriptions. SNMP has been widely adopted due to its simplicity, but the problem is not many objects support the write operation. In practice, SNMP is mainly used to monitor devices and gather operational statistics rather than manage network configurations. After the IETF became aware of this situation, they published a multi-vendor configuration protocol NETCONF [6] with a programming language YANG [7]. NETCONF is designed explicitly for network configurations and can push configurations on devices from different vendors in a transactional manner. It can also test and validate configurations before the final commit. Although NETCONF has been proposed for more than ten years, its adoption has been slow in the network industry. The main reason is that the cost of replacing existing protocols is massive, and device vendors and service providers tend to create products based on existing vendor-specific protocols and languages. Overall, current configuration standards are not as effective as expected, requiring administrators to learn different scripting languages and get accustomed to product manuals.

> Take Away 1: Script-based network configuration management requires a lot of human interpretation and expertise. It does not scale and limits organizational agility.

C. Automated Network Configuration Management

Where possible, automated tools are used to simplify network configuration management through its life cycle. At the highest level, they can form the knowledge base of domain experts captured in product manuals and automate the operations that a human operator would have performed. At the lowest level, they do not require administrators to code in device-specific languages but allow them to express configuration requirements in a more natural and intuitive way. Over the years, with the evolution of autonomic networking, Software-Defined Networking (SDN), Network Function Virtualization (NFV), Artificial Intelligence (AI), and Machine Learning (ML), several automated techniques have appeared in academia and industry. Autonomic networking-based configuration management achieves self-managing by design via building autonomic networking architectures. The innovation lies in continuously monitoring the network state and adapting to changes. As goals and needs vary from business to business, companies and organizations face challenges translating best practices into situated practices. Policy-based configuration management enables such translation as business rules can be embedded within the policies to govern the behaviour of network devices. This is especially helpful for making configuration changes as it can be done by modifying policies instead of re-coding. Constraint-based configuration management abstracts away the procedure of implementing a configuration requirement. It automates the implementation phase by reducing requirement analysis to a constraint satisfaction problem. In this way, the computation of configuration solutions is left to the system. Configuration conflicts and inconsistencies caused by changes can be automatically detected by examining the mathematical results. The recent advances in intent-based configuration management provide the highest level of abstraction on protocols and hardware. It ties intents directly to device-native configurations, and thus the need to manually deploy policies device-by-device goes away. In addition, it can automate administrative tasks across a network based on changing business goals. Despite being called by different terminologies, all these techniques share the ultimate goal of minimizing human intervention. On the other hand, they still have some challenges and issues. In the next section, we will discuss them in more detail.

III. Solutions for Automated Security-Focused Network Configuration Management

In this section, we first explain the sources and methodology that we used to select relevant literature. After that, we classify existing solutions into different categories and discuss them in detail.

A. Source Selection and Categorization

Automated network configuration management has received increasing attention from both academia and industry. Therefore, we used Google Scholar and Google as primary sources to search for relevant academic papers and technical articles. In addition, we searched research databases (e.g., IEEE Xplore, ACM Digital Library, Springer Link) and peer product review websites (e.g., Gartner Peer Insights, G2Crowd, IT Central Station) with the following keywords: *((network OR security) AND automation) OR ((autonomic OR policy OR solver OR intent) AND (network OR security))*. Also, we have limited

our scope to recent work from 2000. We then discarded papers clearly unrelated to the subject based on the paper abstract or product description. Particularly, we focused on papers discussing network security topics such as packet filtering. After the filtering process, 36 publications matched our requirements. We classify the resulting literature into five categories: *autonomic networking-based*, *policy-based*, *constraint-based*, *intent-based*, and *other industrial solutions*.

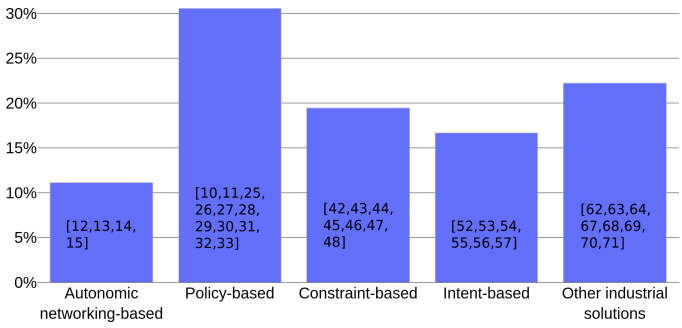

Fig. 2. Percentage of Solutions by Type of Technique.

As shown in Fig. 2, the autonomic-networking based technique accounts for 11% of all solutions. Autonomic networking was introduced more than 15 years ago. Still, not many related academic papers have been published because 1) the concept of autonomic computing was firstly introduced in the industry [8], and 2) autonomic networking requires changes to the overall network architecture (e.g., standards, protocols). This topic is too big to be carried out by independent researchers, and thus all associated architectural solutions were large research projects funded by federal states or standardization organizations. The policy-based technique takes up 31% of the total solutions. Policy-based management has also been around for a long time since its first appearance [9]. Over the years, this technique has been widely used in the network security domain, such as firewalls. It is interesting to note that we found two works [10], [11] combining the autonomic networking-based and policy-based techniques. We believe this was in response to autonomic networking lacking a way to tie business goals to low-level configurations. As this approach uses policy-based management in the autonomic networking architecture, we consider it falls into the policy-based management domain. The constraint-based technique is based primarily on computer science fundamentals, so it was mainly conducted by independent researchers from the computer science discipline. Intent-based solutions are relatively new, but this technique has received much attention (i.e., it accounts for 17% of the total solutions). This is because the new spring of AI and ML made this approach more practical. Other industrial solutions make up 22% of all solutions. Network automation has always been a popular topic in the industry, and we believe there are more products on the market. Due to space limitations, we only focus on solutions proposed by leading companies in this sector. The rest of this section will discuss each category's related work separately.

B. Automated Security-Focused Network Configuration Management Techniques

1) Autonomic Networking-Based Configuration Management: To reduce the complexity of the IT components caused by the industry's exploitation of new technologies, IBM [8] first presented the vision of autonomic computing in 2001. It refers to the ability of computers to manage themselves automatically through adaptive technologies and minimize the time needed by IT professionals to solve system difficulties and other maintenance problems. A computing system is considered autonomic if it supports self-configuration, self-healing, self-optimization, and self-protection (self-CHOP). Accordingly, they [16] introduced an autonomic system containing five building blocks: 1) an autonomic manager implements the Monitor-Analyze-Plan-Execute over a shared Knowledge control loop, 2) a knowledge source provides access to shared knowledge based on the interfaces prescribed by the architecture, 3) a touch point exposes the state and management operations for a resource in the system, 4) a manual manager provides the user interface where an IT professional can perform management functions manually, and 5) an enterprise service bus integrates other building blocks by directing the interactions among them. IBM's blueprint provided a general architecture for autonomic computing and inspired other research fields such as autonomic networking. Several research initiatives have emerged on building generic autonomic network architectures. Table 1 summarizes the domain-specific challenges and related work.

In 2007, Tschudin et al. developed a framework called autonomic network architecture (ANA) [12]. It contains a set of ANA nodes. Each node consists of a MINMEX, a playground, and a hardware abstraction layer that provide various services to facilitate network management tasks. The highlight of their work was that they introduced the compartment abstraction that allowed the division of communication networks into smaller and more easily manageable units. Compartments contain three abstractions types: Functional Blocks, Information Channels, and Information Dispatch Points. All members agree on some common set of operational and policy rules assigned to a specific compartment. Each compartment has autonomy regarding the communication principles and protocols to handle inter- and intra-communication. In [17], they gave details about the resilience and security framework. Chaparadza et al. proposed a conceptual architectural reference model called the Generic Autonomic Network Architecture (GANA) [13]. Self-management is achieved via instrumenting the network components with autonomic Decision-making-Elements (DEs) that collaboratively work together. It contains a Decision Plane that makes all decisions towards a node's behaviour and network-wide control. The Decision Plane consists of a hierarchy of DEs split into protocol, function, node, and network levels. Higher-level DEs manage lower-level DEs through control loops. Neves et al. proposed SELFNET framework [14] that aimed to combine autonomic management with SDN and NFV technologies. SELFNET framework is

TABLE I
AUTONOMIC NETWORKING-BASED CONFIGURATION MANAGEMENT CHALLENGES AND RELATED WORK.

Challenge	Solution
Self-configuration: To automate the process of configuring network components under varying and unpredictable conditions without disturbing the network performance.	[12], [13], [14], [15]
Self-healing: To detect and fix failure components, predict issues, and take proactive actions to discourage errors from impacting the system.	[13], [14]
Self-optimization: To improve performance by optimizing resource allocations and workloads dynamically at all times while keeping the complexity hidden.	[12], [13], [14]
Self-protection: To proactively detect, identify, and protect users and data against malicious attacks and threats.	[12], [14], [15]

a six layers architecture. The Self-Organizing Network layer achieves autonomic management. It consists of four sublayers: Monitor and Analyzer, Autonomic Management, VNFs Onboarding, and Orchestrator. The Autonomic Management sublayer uses the Tactical Autonomic Language to define the autonomic strategies to guide the automatic actions. It uses AI, data mining, and stochastic algorithms to handle network problems in both reactive and proactive manners. Despite years of research, existing autonomic networking frameworks have no large-scale adoption. The need for standards is obvious as autonomic networking is too big to be solved by individual initiatives. Under such circumstances, in 2015, the Internet Research Task Force (IRTF) proposed the definitions and design goals for autonomic networking [18] intending to define a common Autonomic Networking Infrastructure (ANI). They briefly described an autonomic network wherein autonomic nodes communicated through an Autonomic Control Plane. An autonomic node implements the ANI and contains several autonomic functions built from Autonomic Service Agents (ASAs). It requires no configuration and can derive all required information through self-knowledge and knowledge discovery. Following the IRTF's document, Cisco [19] launched Internetwork Operating System software with the ANI feature, which supported the autonomic discovery of connectivity and identity of new devices. Started with RFC 7575 [18], the IETF formed the Autonomic Networking Integrated Model and Approach (ANIMA) Working Group [20]. In 2021, they completed the final version of the autonomic networking reference model [15] as the first phase of their standardization process. Meanwhile, they published a series of RFCs to give more details about the building blocks of their reference model, including ASAs [21], GeneRic Autonomic Signaling Protocol (GRASP) [22], Autonomic Control Plane (ACP) [23], and Bootstrapping Remote Secure Key Infrastructure (BRSKI) [24]. With these the IETF standards, we hope standardized deployment practices will appear in the industry soon.

Autonomic networking-based configuration management achieves automation in terms of self-CHOP properties. Its self-healing and self-protection nature can secure the network at the infrastructure level. Still, it entails a few issues and concerns. Existing architectural frameworks, including the IETF's standards, have not been fully implemented yet. Cisco is among the first to apply autonomic networking features in their products, but manual configuration is greatly needed in the management process. Besides, existing approaches mention little about how to fulfil organizational configuration requirements. These requirements can significantly impact security control implementation. As a result of lacking the link between requirements and implementation, it is impossible to directly reconfigure network components in response to new security needs. We believe the cause of this problem is because autonomic networking is a technical concept and all these frameworks are technology-specific. In real-world settings, configuration management is not only driven by best practices but also by business objectives and goals. The following section will discuss the policy-based network management introduced to solve this issue.

> Take Away 2: Autonomic networking-based configuration management realizes self-CHOP properties at the infrastructure level. However, it does not tie up low-level configuration with high-level requirements.

2) Policy-Based Configuration Management: The concept of policy-based management has been around for a long time since its first appearance [9]. In 1998, the IETF formed the Policy Framework Working Group [34] and published standards for defining a policy framework and information model [35], [36]. In this technique, organizations define policies to govern the behaviour of network components in response to business objectives and environmental changes. Policies are typically expressed in a policy language that provides systematic means to create, implement, and enforce policies for managed network resources. Policy languages formalize the definition of different network elements' operations in response to configuration changes. In its simplest form, a policy-based management system contains a policy repository for storing policies, a Policy Decision Point (PDP) for making decisions, and a Policy Enforcement Point (PEP) for enforcing decisions. The major challenges addressed by this approach are referred to as the policy refinement [37] process, as shown in Table 2.

In 2000, Yemini et al. [28] developed a network self-management system named NESTOR. It seeks to unify configuration management tasks requiring changes in heterogeneous elements at different network layers. They introduced the Resource Definition Language to express configuration models. Network configuration management is automated by executing policy scripts towards a Resource Directory Server. Boutaba et al. [29] proposed a similar approach, but their system

TABLE II
POLICY-BASED CONFIGURATION MANAGEMENT CHALLENGES AND RELATED WORK.

Challenge	Solution
Policy composition: To express network configuration management procedures in the form of a comprehensive set of policies using policy languages.	[25], [26], [27]
Policy translation: To automatically translate high-level policies into low-level device-native configurations.	[10], [11], [25], [26], [27], [28], [29], [30], [31]
Policy ratification: To validate whether the resulting configurations conform to the policies and find configuration conflicts.	[10], [30], [31], [32], [33]

was based on the Directory Enabled Networks specification. Burns [30] developed a configuration tool for automatically managing security policies. The policy engine inside the tool can pre-compute configuration settings and validate policies. NESTOR [28] is used to monitor the network and configure network devices based on the engine's output. Guttman and Herzog [25] proposed rigorous automated security management focusing on packet filtering and IPsec. It requires four steps: networks and packets modelling, expressing security policies, deriving algorithms to enforce security goals, and implementing the algorithms. They used two-location filtering statements to express the combined security policies of packet filtering and IPsec. Enck et al. [26] proposed a network configuration management system called PRESTO that stored predefined policy configlets in databases. The configlets define the services supported by the target router devices. After receiving configuration requests for related routers, PRESTO extracts the mapping information from databases and transforms configlets into complete device-native configurations. Chen et al. [31] introduced a management framework called PACMAN. It contains a central component called Active Document (AD), which is a graph representation that encodes network management primitives and composition derived from the method of procedure documents. PACMAN examines network-wide policies to guarantee that the execution of ADs does not violate network-wide constraints. Anderson et al. proposed a policy language called NetKAT [27] which could describe the behaviour of SDN switches mathematically. NetKAT is based on Kleene algebra with tests wherein Kleene algebra and Boolean algebra are used to reason about global network structure and the predicates defining switch behaviour, respectively. Prakash et al. introduced a way of expressing static Access Control List (ACL) policies on SDN endpoints using a graph model called Policy Graph Abstraction (PGA) [32]. It enables expressing policies independent of underlying network infrastructure and detects conflicts and errors via policy composition. In the PGA system, administrators write policies as graphs and submit them to the Graph Composer. The composer composes individual policies into a combined conflict-free policy set. Abhashkumar et al. extended PGA [32] by introducing a system called Janus [33]. In addition to PGA's capabilities, Janus allows the expression and composition of (Quality of Service) QoS and dynamic ACL policies based on their extended policy graph model.

Additionally, two studies used policies to govern the control loop in autonomic networking to combine the advantages of both techniques. The IBM research group introduced a Management for Autonomic Computing (PMAC) platform [10] supporting the IBM autonomic computing architecture [16]. PMAC is built on the standard Common Information Model (CIM) policy model [38], and each policy is written in the Autonomic Computing Policy Language. Strassner et al. proposed the FOCALE architecture [11] based on policy-based network management system specified in [39]. FOCALE contains a Policy Manager sitting above the Autonomic Manager. The Policy Manager is responsible for translating high-level business requirements into low-level network configurations. First, it uses ontologies to capture the semantics and behaviour of network entities. After that, the model-based translation layer translates policy actions into vendor-specific commands.

More and more service providers have provided cloud-based security solutions in recent years. The IETF realized the increasing adoption of cloud-based security services to replace on-premises security tools and the technical challenges enterprise customers face. Thus, they formed the Interface to Network Security Functions (I2NSF) [40] Working Group, which aimed to standardize software interfaces and data models for controlling and monitoring aspects of physical and virtual NSFs. They proposed a framework for I2NSF [41] that had a Consumer-Facing Interface, which contained a security controller [42] allowing enterprise customers to define, manage, and monitor security policies. The controller has a Policy Translator consisting of three components: 1) Data Extractor uses state transitions to extract data from a policy in XML format, 2) Data Converter searches through NSF Database for target NSFs with the extracted data and converts the policy into NSF-specific format, and 3) Policy Generator uses context-free grammar to generate low-level rules based on the YANG data model [7].

Although policy-based configuration management simplifies management tasks by using policies to define management procedures, it has some limitations in real-world settings. Firstly, these policy languages are very close to programming languages and require a steep learning curve for average users. Secondly, this technique can only handle configuration implementations and changes by examining predefined policies. But in real-life deployment, it is unfeasible to design a comprehensive policy repository that covers all possible configuration scenarios. Last but not least, as it does not enable any declarative representation of system logic, administrators need to explicitly declare the complete control flow of solving a configuration requirement in terms of a series of policies.

Each step in the control flow must be executed in the given order, or the requirement can never be fulfilled. To this end, we need a more declarative, dynamic and scalable technique. In the next section, we will explore solutions built on constraints.

> Take Away 3: Policy-based configuration management uses policies to simplify management tasks. On the other hand, it requires administrators to learn new policy languages and manually define a large set of policies, making it significantly constrained by human expertise.

3) Constraint-Based Configuration Management: Constraint-based configuration management provides automated modelling and reasoning about configuration requirements. Unlike the policy-based technique explicitly specifying all steps to achieve a requirement, the constraint-based approach only identifies the properties of a configuration solution to be found. Studies using this technique typically take the logic representation of domain knowledge and constraints as input and then dynamically compute solutions to satisfy these constraints. Each constraint can contain one or more configuration requirements, such as security, performance, and reliability. The representation of domain knowledge is formal while descriptive so that it can be shared and reused by different vendors. Also, it needs to hide the underlying process of solving constraints from users. This can be achieved using an off-the-shelf solver, or new solving algorithms are required to suit more specific domains. Finally, it provides some mechanism to prevent network-wide misconfigurations. We summarize this technique's challenges and related work in Table 3.

Narain [43] developed a Requirement Solver that could determine general network configurations (e.g., addressing, routing, security) based on given network components and configuration requirements expressed as first-order logic constraints. The Requirement Solver is implemented in a logical system called Alloy [50] and solutions are found by the Alloy model finder. Later, Narain et al. proposed another Requirement Solver called ConfigAssure [49] which was implemented with an SAT-based model finder called Kodkod [51]. ConfigAssure stores network component information as tuples in a database. Still, configuration requirements are represented as constraints. In addition to configuration synthesis, it supports configuration error diagnosis and repair via analysis of proof of unsolvability and removing unsolved constraints. Based on their studies, Homer et al. proposed MulVAL [44] to provide suggestions for configuration changes that could mitigate threats. It reasons about potential attack paths using an attack graph written in Datalog, transforms it into a Boolean constraint, and solves it using a min-cos SAT solver. Nelson and Barratt proposed a firewall configuration management tool called Margrave [45]. In Margrave, administrators send a query to the user interface, which returns a set of scenarios satisfying the queried behaviour. Margrave maps ACL, NAT, routing policies, and user queries into first-order logic constraints. Again, Kodkod [51] is used to generate device configuration satisfying the query. Delaet and Joosen [46] introduced an object-oriented policy language named PoDIM that supported the modelling of cross-machine constraints. It consists of a rule language and a domain model describing configuration constraints and the domain of the rules, respectively. In PoDIM, network components such as devices and interfaces are defined as different object classes in the domain models. A translation controller is responsible for generating objects for rules, creating configuration files from objects, and deploying the files on target devices. Chen et al. proposed a management framework named COOLAID [47]. It uses a logic-based language based on Datalog to formalize domain knowledge from both device vendors and service providers, which is then applied on top of a database-like abstract data model representing network information. It uses constraints to detect and prevent network-wide misconfigurations. Network changes are carried out by modifying tables in the database. Soulé et al. presented a framework named Merlin [48] to generate configurations for SDN switches, middleboxes, and end hosts. They used logical formulas to represent physical topology and constraints. The Merlin compiler maps configuration policies into a constraint problem and solves them using parameterizable heuristics. It contains run-time components called negotiators that can communicate amongst themselves to adjust configurations to dynamically changing resource demands.

Constraint-based configuration management allows administrators to provide the least amount of input by leaving the solving process to the system. However, it is unclear how to generate network knowledge and constraints in an automated way. Existing solutions require users to have sufficient mathematical knowledge and manually create many supporting files to construct them. This is not practical for most companies as they do not have computer scientists work on-site. Also, solving constraint problems is a computationally expensive process that often leads to unaffordable computation times when the network becomes large. In recent years, due to the maturity of SDN, intent-based networking [52] has attracted significant attention from both industry and academia. We will discuss solutions falling in this category in the next section.

> Take Away 4: Constraint-based configuration management turns complex configuration tasks into constraint satisfaction problems. Nevertheless, it becomes computationally expensive when the network scales. Also, it requires users to have sufficient mathematical knowledge.

4) Intent-Based Configuration Management: The concept of intent has existed in Natural Language Understanding (NLU) for a long time. Intent is a high-level declarative policy that describes the outcome of an action. In intent-based configuration management, human operators do not focus on individual devices. Instead, they can define the desired

TABLE III
CONSTRAINT-BASED CONFIGURATION MANAGEMENT CHALLENGES AND RELATED WORK.

Challenge	Solution
Domain knowledge reasoning: To define a formal specification to describe knowledge of the configuration domain, including network information and constraints on configurations.	[43], [44], [45], [46], [47], [48]
Configuration synthesis: To automatically solve configuration problems given network knowledge and configuration constraints.	[43], [44], [45], [46], [47], [48], [49]
Misconfiguration prevention: To detect and prevent misconfiguration in the network, including configuration conflicts and unsolvable constraints.	[45], [47], [48], [49]

outcomes and high-level operational goals in terms of intents, such as *"communications between network a and network b need to be secured."* Upon receiving desired outcomes, the system determines how to accomplish the highest-level goal by consistently changing, tuning, substituting, or adapting sub-level policies and implementing them across the network. Also, it continuously monitors the network state to verify whether the actual network status matches the one expected by the intent. As the system has complete knowledge of the entire network, it can adjust network operations when the intent starts to drift. It also optimizes the current configurations to improve the reliability of network service. The primary function concerns and studies associated with this technique are summarized in Table 4.

The latest network management trend is building intent-based networking upon SDN. SDN greatly improves network programmability by navigating a network infrastructure using software applications. Intent-based networking raises the abstraction level on the Northbound Interface (NBI). It avoids the laborious manual coding inside NBI to deal with different configuration scenarios. Popular controllers including Open Network Operating System (ONOS) [59] and OpenDaylight (ODL) [60] have supported intent-based NBI. On the standardization side, the IETF proposed a protocol language called Intent-Based Network Modeling [53] (IB-NEMO) language, which was also part of ODL's Network Intent Composition project. IB-NEMO is a protocol language used for interactions between an application and an SDN controller. It is specifically designed to satisfy common use cases, including virtual wide-area networks, virtual data centres, bandwidth on demand, and service chaining. Han et al. proposed an intent-based management platform [54] for managing virtual network technologies in SDN. They defined each intent object to require resources, conditions, priority, and instruction attributes. They also introduced the concept of a vocabulary store for mapping intent entities and low-level configurations. Kiran et al. developed an intent renderer application named iNDIRA [55] as an intermediate layer between SDN NBI and users. It can translate network provisioning and file transfer requirements into network commands and execute them on the Network Service Interface and Globus [61] data transfer tools. In iNDIRA, users interact with a chatbot to send configuration intents in natural language via the user interface. It uses natural language processing (NLP) to extract intent entities and constructs ontologies to represent relationships between services and their arguments. Intent ontologies are then rendered into network-specific commands via the SDN controller. Jacobs et al. proposed a similar chatbot application for configuring SDN rules [62]. It uses NLP and a sequence-to-sequence learning model to construct configuration requirements in an intent language called NILE. Based on NILE, they used Named Entity Recognition for intent entities extraction and represented another intent translation application called LUMI [56]. LUMI is served as an intermediate layer between natural language and policy language Merlin [48]. To detect configuration contradictions, they used a Random Forest Classifier to correlate the NILE intents and search for incremental intent deployment.

In the industrial space, several commercial products have influenced the adoption of intent-based networking. As a leading networking company, Cisco introduced Cisco Digital Network Architecture (DNA) [57] software that could translate high-level business intent into zero-trust policies. It can achieve several fundamental management tasks, including managing software updates, discovering network devices, monitoring, and troubleshooting. Cisco devices are inherently aware of Cisco DNA. It also includes multi-vendor software development kits that allow interactions with other vendors' network devices. ML techniques are used to provide accurate insights into network deployment and predict future network performance. Juniper Networks acquired an intent-based networking startup Apstra [58] in December 2020. In May 2021, they released Apstra 4.0 software which injected intent-based networking and automated closed-loop validation into its data centre networking portfolio. Juniper's Apstra solution provides a deployment method called connectivity templates, which allow administrators to create and reuse validated templates to set up multi-vendor networks. It supports multiple device operation systems, including Cisco NX-OS, Nvidia Cumulus, and Juniper Junos OS.

Intent-based configuration management is perhaps the most human-friendly solution due to its highest level of abstraction from any protocol and vendor. In addition, it can automate administrative tasks across a network under changing organizational goals and needs. However, just like the beginning of any other network innovation, a few challenges remain. So far, academic studies have mainly focused on proposing intent renderers on top of NBI. Due to the lack of standardization of NBI API or intents, the ways of specifying intents are defined separately, and thus these renderers are limited to their own use cases. Meanwhile, commercial products typically provide a more comprehensive framework containing mature

TABLE IV
INTENT-BASED CONFIGURATION MANAGEMENT CHALLENGES AND RELATED WORK.

Challenge	Solution
Intent translation: To extract entities from configuration intents in natural language utterances and break them down into a set of non-conflicting policies or configurations.	[53], [54], [55], [56], [57], [58]
Network state awareness: To create a system knowledge base including information about the current network state and continuously observe it.	[57], [58]
Intent assurance: To continuously diagnose and adjust configurations as needed to ensure the original configuration intents are always fulfilled.	[57], [58]

APIs and rich capabilities. They all offer a Graphical user interface (GUI) where administrators can centrally configure their network resources. GUI makes the configuration job much more user-friendly by hiding the underlying commands. However, the configuration itself is still carried out at the device level. Although these products are claimed to support vendor-agnostic management, a great percentage of the functionalities are not working on third-party devices. For example, Cisco DNA only provides very basic visibility, interaction and monitoring for non-Cisco devices. This is not ideal given the heterogeneity of networks in real-world production environments. Despite the high price tag, they have limited intent-based networking capabilities and still put humans in the centre of the control loop.

> Take Away 5: Intent-based configuration management uses organizational goals and needs to drive low-level configuration. Compared to other techniques, it creates the highest level of abstraction and thus greatly improves the ease of managing configuration.

5) Other Industrial Solutions: In recent years, several industrial all-in-one configuration management tools have been developed to support multi-vendor network operations. These tools focus on automating repetitive day-to-day configuration tasks, including basic provisioning, performance monitoring, upgrades and patches, and configuration backups. They typically come with a web-based user interface containing provisioning configlet and script templates across multiple vendors. Administrators can use them directly or customize and then execute them on devices. Some of these tools were developed by network device vendors [63], [64], [65]. They usually provide full control of their own physical devices while integrating third-party configuration components into the control panels. For example, FortiManager can centrally manage the configurations of multiple Fortinet devices from a single console. Although it does not support direct configuration on devices from other vendors, its security fabric connector allows integration and automation with third-party vendors such as Cisco pxGrid [66] and Clearpass [67]. Meanwhile, some IT solution companies have proposed several cross-vendor software [68], [69], [70], [71], [72]. Overall, these industrial solutions are still script-based, but they simplify the tasks by providing off-the-shelf templates and a centralized control panel.

IV. DISCUSSION AND FUTURE DIRECTIONS

Table 5 summarizes existing solutions according to the category, specialization, and management phase (discussed in Section 2). For the sake of readability, we use numbers to denote the phases. Having discussed different techniques, in this section, we compare them using the *specialization* and *phase* criteria to gain further insights. Note that we do not include solutions belonging to the *other industrial solutions* category. The main reason is it is difficult to examine their functionalities in detail as they are not open-source. After that, we suggest some research directions based on our findings.

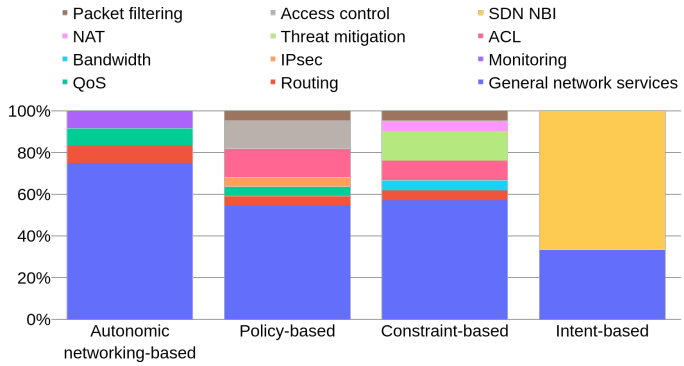

Fig. 3. Percentage of Solutions to Different Specialization by Type of Technique.

First, we studied various specializations that each technique covered. Looking at the specialization criterion in Fig. 3, solutions for general network services played a dominant role across all of these techniques. General network services refer to any type of network configuration task. Solutions fell into this specialization attempted to generalize network services. They did not specify their work domain and aimed to solve various configuration jobs as a whole. This is especially the case for autonomic networking-based solutions as they typically focus on the entire network architecture. The policy-based technique demonstrated a strong ability to solve domain-specific problems in network security such as ACL. ACL uses a set of rules defining conditions of permitting or denying actions and thus can naturally fit into the policy-based paradigm. The constraint-based technique was also used in various domains thanks to its ability to represent different types of knowledge using logic. The intent-based approach mainly focused on SDN NBI as SDN has already provided vendor-independent control over the entire network from a

TABLE V
SUMMARY OF AUTOMATED NETWORK CONFIGURATION MANAGEMENT SOLUTIONS.

Solution	Category	Specialization	Phase
[12], [14], [15]	Autonomic networking-based	General network services	3, 4
[13]	Autonomic networking-based	Routing, QoS, Monitoring	3, 4
[10], [11]	Policy-based	General network services	2, 3, 4
[25]	Policy-based	Packet filtering, IPsec	2
[26]	Policy-based	General network services	2
[27]	Policy-based	Access control, Routing	2
[28], [29]	Policy-based	General network services	3
[30]	Policy-based	Access control	3
[31]	Policy-based	General network services	2, 3
[32]	Policy-based	ACL	3
[33]	Policy-based	QoS, ACL	3
[43], [47], [49]	Constraint-based	General network services	2, 3
[44]	Constraint-based	Threat mitigation	2, 3
[45]	Constraint-based	ACL, NAT, Routing	2, 3
[46]	Constraint-based	General network services	2
[48]	Constraint-based	Bandwidth, ACL, Packet filtering	2, 3, 4
[53], [54], [55]	Intent-based	SDN NBI	2
[56]	Intent-based	SDN NBI	2, 3
[57], [58]	Intent-based	General network services	2, 3, 4
[63], [64], [65], [68], [69], [70], [71], [72]	Other industrial solutions	General network services	2, 3, 4

single logical point. Building an intent-based layer upon an SDN architecture enables abstracting control commands at a higher level, which delivers more value to both domains.

Next, we compare different techniques by network configuration management phases in Table 5. Surprisingly, we found that none of the above studies provided support for phase 1. We assume this is because the planning phase activities (e.g., investigating organizational risk tolerance, regulatory standards, service-level agreement (SLA), and ethical issues) are typically conducted at the business process level. Unlike network resources, they are not easy to be interpreted by machines. On the other hand, as shown in Fig. 4, phases 2 and 3 have received much attention from the research community. The reason is activities in these two phases contain most of the technical challenges in network configuration management. Notably, the intent-based technique was proposed to create the highest level of abstraction, so most of the solutions in this area focused on challenges in phase 2. Conversely, phase 4 did not get much attention from researchers as it is easier to implement, and network vendors have already developed many mature monitoring tools.

To summarize, there are still many challenges lying in existing techniques. The autonomic networking-based technique requires modification of the traditional Internet and OSI model. Thus, the way toward a complete architecture is long. The major drawback of the policy-based technique is the complexity of creating and maintaining a large set of policies. Also, existing policy languages are very close to programming languages. For the constraint-based technique, establishing network knowledge and constraints is too difficult for average network administrators. The intent-based technique is regarded as the most human-friendly technique so far. As the abstraction level is increased, the standardization of a more translatable intent language is needed. But to the best of our knowledge, little has been done in this respect. Meanwhile, most commercial products are only dedicated to vendor-specific network devices. Based on our findings, we select five research gaps where we think researchers' experience can play a fundamental role. For each gap, we point out current solutions' limitations and outline some research directions.

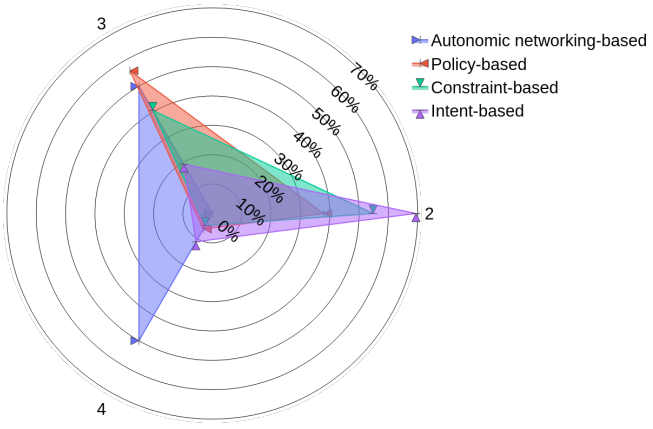

Fig. 4. Technique Comparison by Network Configuration Management Phases.

A. Gap 1: Intent Translation

The lack of a standard yet expressive language for specifying intents is still challenging. On the academic side, intent semantics are typically formalized into an abstract presentation layer (e.g., ontology, graph database) and then mapped to sub-layer policies or configurations. The construction and maintenance of such a presentation layer are cumbersome and labour-intensive. Moreover, it often requires support from various domain experts. For example, ontology and network experts must collaboratively review and verify network ontologies. In

the industry, all major networking vendors are independently developing their own intent-based solutions with private APIs. Each SDN-based solution uses a controller platform that has different approaches to intent. As conflicts of profits exist, defining a standard way of intent-to-configuration translation is mainly focused inside the IEFT. However, their intent standardization efforts are still in their infancy period.

Overall, due to the current complexity involved in network configuration, it is impossible to define a universal intent definition language that can be applied to any configuration scenario. We believe a more promising way is to abstract the intent representation further using AI-based techniques. The main idea is to allow users to express security-related operational goals in a natural language and then dynamically generate configuration scrips or sub-layer policies. Recent advances in AI and ML offer an opportunity to adopt this idea in the coming years. Particularly, NLP has demonstrated promising performance in translation between natural languages [73] and programming languages [74]. Generative Pre-trained Transformer 3 (GPT-3) [75], a neural-network-powered language model trained by OpenAI, represents a breakthrough in NLP. The API is claimed to be able to generate any kind of text that is indistinguishable from what a human can produce. It can potentially translate natural language into any programming language, such as Unix commands (e.g., Natural Language Shell Demo [76]) and JSX (e.g., Debuild [77]). What's more, it does not require further training for distinct language tasks since its training data are comprehensive. We believe using NLP is a promising way to convert natural language queries to various policy languages or even configuration scripts on the fly. One future work direction would be to verify the feasibility of using modern NLP technologies in intent translation.

B. Gap 2: Intelligent Network Configuration Management

When the desired secure state starts to drift due to environmental changes, the configuration management system should be able to resolve conflicts between desired outcomes and the current network state and then generate new configuration solutions without delay. Both autonomic networking-based and intent-based configuration management addressed such requirements via different approaches. The former focused on designing an autonomous closed control loop, whereas the latter suggested intent-driven configuration change control. We believe using intents to guide autonomic networking-based configuration management can take advantage of both techniques. In fact, [15] has addressed intents as a future feature for autonomic networking. The IEFT suggested intent refinement could be achieved via policies following the event-condition-action (ECA) paradigm. However, ECA-based policies have limited adaptability to changes. We think their approach may not fully tap the potential of the intent-based technique. To this end, we consider a more intelligent way to guide the control loop via ML. For example, Reinforcement Learning (RL) introduces intelligent agents that can decide what actions to take to achieve a long-term reward. This principle allows an RL-enabled configuration management system to have the potential to automatically learn and adapt to the time-changing network environment to assure the highest-level intent continuously. Another example is using Deep Learning (DL) to predict the actions of human experts for subsequent automation. Future studies may explore the potential of using these ML technologies to deliver a continuous learning system with configuration intelligence.

C. Gap 3: Automated Planning

In an organization, technical people usually think about planning at the system level. They focus on translating technical best practices into situated practices. On the other hand, non-technical people focus on planning at the organizational level. It includes translating laws, regulations, etc., from higher authorities into organizational documents and practices. It is cliche but true that a communication barrier exists between these two groups of people. Consequently, the resulting configurations may not reflect the overall organizational security needs and even violate business rules or official regulations.

The emergence of policy-based and intent-based configuration management aimed to solve this problem. However, translating business rules into technical requirements is often challenging and error-prone, and existing studies still depend on the knowledge base captured in technical documents. To the best of our knowledge, none of the current work focused on facilitating planning at the organizational level. Despite years of study, there is a gap between business requirements and actual implementations. An important future direction is to provide mechanisms to automate planning activities at both levels. We believe NLP again has the potential to help in closing this gap by interpreting high-level business rules and translating them into machine-actionable items. For example, regulatory compliance is critical to upholding business processes' integrity while protecting public and stakeholder interests. Future studies can investigate using NLP techniques to automatically extract configuration-related rules from legal texts and map them to technical implementations.

D. Gap 4: Intent-Based Solutions for I2NSF

For many small and medium enterprises (SMEs), their size and budgets limit the investment in a security operation team. Therefore, SMEs tend to adopt an outsourcing stance regarding security solutions. The IETF's I2NSF addressed the growing demand of organizations for outsourcing security services to external service providers. Through the Consumer-Facing Interface, administrators can consume network security services hosted by one or more providers. The main problem is that the current I2NSF framework only supports policy-based configuration management, and each policy must be described using the Condition-Action paradigm in XML format. For people who do not have requisite security knowledge and programming skills, authoring policies is still quite challenging. To help people with different expertise levels, researchers may consider developing an approach similar to combining the intent-based technique and SDN. That is to say, building an

intent layer above the Consumer-Facing Interface allows users to use high-level intents to express security expectations.

E. Gap 5: Reliable Network Performance

Although the primary focus of these studies is to automate different management phases, they should also minimize disruption to network performance and user experience. Solutions based on problem-solving, AI, and ML technologies require excessive computational capacity, memory, and power, which can be challenging to deploy in devices with limited resources. High computational complexity can lead to long latency or even packet loss. It does not only impact system performance but also can create security vulnerabilities. These days, more and more businesses are moving to the cloud to reduce the Total Cost of Ownership (TCO). Since cloud computing is much more elastic than traditional networks, organizations can move the computation process to the cloud. However, security becomes a big concern because input data can include sensitive information such as organizational network topology. We recommend future work can investigate how to improve network performance and ensure security while developing automated solutions. For example, researchers can integrate encryption mechanisms into existing solutions.

V. Conclusion

This paper provides a structured literature review of state-of-the-art automated security-focused network configuration management techniques. We classify them into different categories, discuss the pros and cons of each technique, and compare them according to the specializations and phases they address.

Our study finds that the autonomic networking-based and constraint-based techniques were once hot topics in the research community. Unfortunately, both haven't shown great success in real-world implementations. To date, the policy-based technique is still the most popular approach among network administrators as it has been historically used in network security. The rapid development of AI and ML promoted the growth of the intent-based technique. We think it provides users with the most declarative and dynamic way of managing various configurations in a heterogeneous network. Still, AI-enabled network configuration management has a long way to go before fully delivering its potential. For example, intent translation is only in its infancy. Among all the techniques, we discover that no existing solution supports the planning phase. We recommend more research is needed in developing methods to translate business rules into technical requirements.

References

[1] (2021) What is network automation? Cisco. [Online]. Available: https://www.cisco.com/c/en/us/solutions/automation/network-automation.html

[2] (2021) What is network automation? Fortinet. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/network-automation

[3] (2021) Juniper automation. Juniper Networks. [Online]. Available: https://www.juniper.net/us/en/solutions/automation.html

[4] A. Johnson, K. Dempsey, R. Ross, S. Gupta, and D. Bailey, "Guide for security-focused configuration management of information systems," *NIST Special Publication 800-128*, vol. 800, no. 128, pp. 1–99, 2011.

[5] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," RFC 1157, 1990.

[6] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (NETCONF)," RFC 6241, 2011.

[7] M. Bjorklund, "YANG - a data modeling language for the NETCONF," RFC 6020, 2010.

[8] P. Horn. (2001) Autonomic computing: IBM's perspective on the state of information technology. IBM. [Online]. Available: https://homeostasis.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf

[9] M. Sloman, "Policy driven management for distributed systems," *Journal of Network and Systems Management*, vol. 2, no. 4, pp. 333–360, 1994.

[10] D. Agrawal, K. Lee, and J. Lobo, "Policy-based management of networked computing systems," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 69–75, 2005.

[11] J. Strassner, N. Agoulmine, and E. Lehtihet, "FOCALE: A novel autonomic networking architecture," *International Transactions on Systems Science and Applications*, vol. 1, pp. 1–13, 2006.

[12] C. Tschudin, C. Jelger, G. Bouabene, G. Leduc, L. Peluso *et al.*, "ANA project: Autonomic network architecture," 2007.

[13] R. Chaparadza, S. Papavassiliou, T. Kastrinogiannis, M. Vigoureux, E. Dotaro *et al.*, *Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering*. IOS Press, 2009.

[14] P. Neves, R. Calé, M. Costa, C. Parada, B. Parreira *et al.*, "The SELFNET approach for autonomic management in an NFV/SDN networking paradigm," *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, pp. 1–17, 2016.

[15] M. Behringer, B. Carpenter, T. Eckert, L. Ciavaglia, and J. Nobre, "A reference model for autonomic networking," RFC 8993, 2021.

[16] (2005) An architectural blueprint for autonomic computing. IBM. [Online]. Available: \urldef{\webmaccom}\url{https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf}

[17] J. Sterbenz, M. Schöller, A. Jabbar, and D. Hutchison, "ANA resilience framework," 2007.

[18] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter *et al.*, "Autonomic networking: Definitions and design goals," RFC 7575, 2015.

[19] (2017) Configuring autonomic networking. Cisco. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3650/software/release/16-12/configuration_guide/sys_mgmt/b_1612_sys_mgmt_3650_cg/configuring_autonomic_networking.pdf

[20] Autonomic networking integrated model and approach (ANIMA). The Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/wg/anima/documents/

[21] B. Carpenter, L. Ciavaglia, S. Jiang, and P. Peloso, "Guidelines for autonomic service agents," draft-ietf-anima-asa-guidelines-01, 2021.

[22] C. Bormann, B. Carpenter, and B. Liu, "GeneRic autonomic signaling protocol (GRASP)," RFC 8990, 2021.

[23] T. Eckert, M. Behringer, and S. Bjarnason, "An autonomic control plane," RFC 8994, 2021.

[24] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, "Bootstrapping remote secure key infrastructure (BRSKI)," RFC 8995, 2021.

[25] J. Guttman and A. Herzog, "Rigorous automated network security management," *International Journal of Information Security*, vol. 4, no. 1-2, pp. 29–48, 2005.

[26] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel *et al.*, "Configuration management at massive scale: System design and experience," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 3, pp. 323–335, 2009.

[27] C. Anderson, N. Foster, A. Guha, J. Jeannin, D. Kozen *et al.*, "NetKAT: Semantic foundations for networks," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 113–126, 2014.

[28] Y. Yemini, A. Konstantinou, and D. Florissi, "NESTOR: An architecture for network self-management and organization," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 5, pp. 758–766, 2000.

[29] R. Boutaba, S. Omari, and A. Virk, "SELFCON: An architecture for self-configuration of networks," *Journal of Communications and Networks*, vol. 3, no. 4, pp. 317–323, 2001.

[30] J. Burns, A. Cheng, P. Gurung, S. Rajagopalan, P. Rao *et al.*, "Automatic management of network security policy," in *DISCEX '01*, vol. 2. New York: IEEE, 2001, pp. 12–26.

[31] X. Chen, Z. Mao, and J. Merwe, "PACMAN: A platform for automated and controlled network operations and configuration management," in *CoNEXT '09*. New York: ACM, 2009, pp. 277–288.

[32] C. Prakash, J. Lee, Y. Turner, J. Kang, A. Akella *et al.*, "PGA: Using graphs to express and automatically reconcile network policies," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 29–42, 2015.

[33] A. Abhashkumar, J. Kang, S. Banerjee, A. Akella, Y. Zhang, and W. Wu, "Supporting diverse dynamic intent-based policies using Janus," in *CoNEXT '17*. New York: ACM, 2017, pp. 296–309.

[34] Policy framework working group. The Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/wg/policy/documents/

[35] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn *et al.*, "Terminology for policy-based management," RFC 3198, 2001.

[36] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy core information model," RFC 3060, 2001.

[37] J. Moffett and M. Sloman, "Policy hierarchies for distributed systems management," *IEEE Journal on selected areas in communications*, vol. 11, no. 9, pp. 1404–1414, 1993.

[38] Common information model. The Distributed Management Task Force. [Online]. Available: https://www.dmtf.org/standards/cim

[39] J. Strassner and J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation*. USA: Morgan Kaufmann, 2004.

[40] Interface to network security functions (i2nsf). The Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/wg/i2nsf/documents/

[41] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for interface to network security functions," RFC 8329, 2018.

[42] J. Jeong, P. Lingga, J. Yang, and C. Chung, "Security policy translation in interface to network security functions," draft-yang-i2nsf-security-policy-translation-08, 2021.

[43] S. Narain, "Network configuration management via model finding," in *LISA '05*, vol. 19. USA: USENIX Association, 2005, pp. 15–15.

[44] J. Homer, X. Ou, and M. McQueen, "From attack graphs to automated configuration management - an iterative approach," Kansas State University, Tech. Rep., 2008.

[45] T. Nelson, C. Barratt, D. Dougherty, K. Fisler, and S. Krishnamurthi, "The Margrave tool for firewall analysis," in *LISA '10*. USA: USENIX Association, 2010, pp. 1–18.

[46] T. Delaet and W. Joosen, "PoDIM: A language for high-level configuration management," in *LISA '07*. USA: USENIX Association, 2007, pp. 261–273.

[47] X. Chen, Y. Mao, Z. Mao, and J. Merwe, "Declarative configuration management for complex and dynamic networks," in *Co-NEXT '10*. New York: ACM, 2010, pp. 1–12.

[48] R. Soulé, S. Basu, P. Marandi, F. Pedone, R. Kleinberg *et al.*, "Merlin: A language for provisioning network resources," in *CoNEXT '14*. New York: ACM, 2014, pp. 213–226.

[49] S. Narain, G. Levin, S. Malik, and V. Kaul, "Declarative infrastructure configuration synthesis and debugging," *Journal of Network and Systems Management*, vol. 16, no. 3, pp. 235–258, 2008.

[50] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. USA: MIT press, 2012.

[51] E. Torlak and D. Jackson, "Kodkod: A relational model finder," in *TACAS '07*. Berlin: Springer, 2007, pp. 632–647.

[52] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura, "Intent-based networking - concepts and definitions," draft-irtf-nmrg-ibn-concepts-definitions-00, 2019.

[53] S. Hares, "Intent-based Nemo problem statement," draft-hares-ibnemo-overview-00, 2015.

[54] Y. Han, J. Li, D. Hoang, J. Yoo, and J. Hong, "An intent-based network virtualization platform for SDN," in *CNSM '16*. New York: IEEE, 2016, pp. 353–358.

[55] M. Kiran, E. Pouyoul, A. Mercian, B. Tierney, C. Guok, and I. Monga, "Enabling intent to configure scientific networks for high performance demands," *Future Generation Computer Systems*, vol. 79, pp. 205–214, 2018.

[56] A. Jacobs, R. Pfitscher, R. Ribeiro, R. Ferreira, L. Granville, and S. Rao, "Deploying natural language intents with Lumi," in *SIGCOMM Posters and Demos '19*. New York: ACM, 2019, pp. 82–84.

[57] (2021) Cisco DNA center 2.2.2.0 data sheet. Cisco. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/dna-center/nb-06-dna-center-data-sheet-cte-en.pdf

[58] (2021) Juniper Apstra system data sheet. Juniper Networks. [Online]. Available: https://www.juniper.net/content/dam/www/assets/datasheets/us/en/network-automation/apstra-solution.pdf

[59] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi *et al.*, "ONOS: Towards an open, distributed SDN OS," in *HotSDN '14*. New York: ACM, 2014, pp. 1–6.

[60] OpenDaylight user guide. OpenDaylight. [Online]. Available: https://nexus.opendaylight.org/content/sites/site/org.opendaylight.docs/master/userguide/manuals/userguide/bk-user-guide/bk-user-guide.pdf

[61] F. I, "Globus toolkit version 4: Software for service-oriented systems," *Journal of Computer Science and Technology*, vol. 21, no. 4, pp. 513–520, 2006.

[62] A. Jacobs, R. Pfitscher, R. Ferreira, and L. Granville, "Refining network intents for self-driving networks," in *SelfDN '18*. New York: ACM, 2018, pp. 15–21.

[63] (2021) Fortinet FortiManager data sheet. Fortinet. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortimanager.pdf

[64] (2020) Juniper Junos space network director data sheet. Juniper Networks. [Online]. Available: https://www.juniper.net/content/dam/www/assets/datasheets/us/en/sdn-management-operations/junos-space-network-director.pdf

[65] (2019) Huawei eSight data sheet. Huawei. [Online]. Available: https://e.huawei.com/en/material/esight/5a455ad5523f4bc49c3ee35e0b97fcd1

[66] (2018) Cisco pxGrid: Automate multi-platform communications through a unified architecture. Cisco. [Online]. Available: https://pubhub.devnetcloud.com/media/pxgrid-api/docs/overview/Cisco_pxGrid_White_Paper_09192018_JE.pdf

[67] (2021) Aruba Clearpass policy manager data sheet. Hewlett Packard. [Online]. Available: https://www.arubanetworks.com/assets/ds/DS_ClearPass_PolicyManager.pdf

[68] (2020) Solarwinds network configuration manager data sheet. SolarWinds. [Online]. Available: https://www.solarwinds.com/-/media/solarwinds/swdcv2/licensed-products/network-configuration-manager/resources/datasheets/ncm-datasheet.ashx?rev=aefce2796e44431a856ef1ba40a06f5a

[69] Network configuration manager data sheet. ManageEngine. [Online]. Available: https://download.manageengine.com/network-configuration-manager/datasheet.pdf

[70] (2019) BMC TrueSight automation for networks data sheet. BMC. [Online]. Available: https://documents.bmc.com/products/documents/96/96/469696/469696.pdf

[71] Infoblox NetMRI data sheet. Infoblox. [Online]. Available: https://www.infoblox.com/wp-content/uploads/infoblox-datasheet-netmri.pdf

[72] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan *et al.*, "A general approach to network configuration analysis," in *NSDI '15*. USA: USENIX Association, 2015, pp. 469–483.

[73] (2021) Cross-lingual transfer evaluation of multilingual encoders. Google Research. [Online]. Available: https://sites.research.google/xtreme

[74] M. Lachaux, B. Roziere, L. Chanussot, and G. Lample, "Unsupervised translation of programming languages," *arXiv Preprint:2006.03511*, vol. 1, pp. 1–21, 2020.

[75] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan *et al.*, "Language models are few-shot learners," 2020.

[76] (2020) Natural language shell demo. OpenAI. [Online]. Available: https://vimeo.com/427943407/98fe5258a7

[77] S. Shameem. (2020) Introducing Debuild. Debuild. [Online]. Available: https://debuild.co