Computer Networks 203 (2022) 108668

Contents lists available at ScienceDirect

Computer Networks

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

Deep data plane programming and AI for zero-trust self-driven networking in beyond 5G

Othmane Hireche [a], Chafika Benzaïd [a], Tarik Taleb [b,c,*]

[a] Aalto University, Espoo, Finland
[b] University of Oulu, Oulu, Finland
[c] Sejong University, Seoul, South Korea

ARTICLE INFO

Keywords:
AI
Federated learning
Blockchain
Zero trust
P4
Data plane programming
Self driving network
5G and beyond networks

ABSTRACT

Along with the high demand for network connectivity from both end-users and service providers, networks have become highly complex; and so has become their lifecycle management. Recent advances in automation, data analysis, artificial intelligence, distributed ledger technologies (e.g., Blockchain), and data plane programming techniques have sparked the hope of the researchers' community in exploring and leveraging these techniques towards realizing the much-needed vision of trustworthy self-driving networks (SelfDNs). In this vein, this article proposes a novel framework to empower fully distributed trustworthy SelfDNs across multiple domains. The framework vision is achieved by exploiting (i) the capabilities of programmable data planes to enable real-time in-network telemetry collection; (ii) the potential of P4 – as an important example of data plane programming languages – and AI to (re)write the source code of network components in a fashion that the network becomes capable of automatically translating a policy intent into executable actions that can be enforced on the network components; and (iii) the potential of blockchain and federated learning to enable decentralized, secure and trustable knowledge sharing between domains. A relevant use case is introduced and discussed to demonstrate the feasibility of the intended vision. Encouraging results are obtained and discussed.

1. Introduction

Along with the ongoing advances in the communications and networking technologies, highly-innovative mobile services have emerged, involving a potential number of users and a much further higher number of devices (e.g., Internet of Things — IoT devices). Configuring these extremely-large, extremely dynamic, and extremely-complex networks and constantly managing their lifecycle have consequently become much more challenging. In this regard, human interventions have lead to eventual errors: 80% of businesses claim to have experienced network errors caused by human mistakes on a regular basis [1].

Along with the emergence of Automation, Data Analysis, and Artificial Intelligence (AI) techniques, research efforts have been directed towards incorporating these techniques towards realizing the concept of Self-driving networks (SelfDNs) [2]. SelfDNs are networks which can monitor, analyze, and automatically maintain themselves, minimizing the intervention of network engineers to a large extent or completely getting rid of it. Networks can detect bugs, and define and enforce adequate policies to repair themselves. The road towards realizing this vision is still quite long, but the community of researchers firmly believes that this defines a new horizon for the future of smart network management.

The emerging distributed and cooperative AI techniques, including Federated Learning (FL), are envisioned to play a key role in empowering fully distributed self-managing capabilities in 5G and beyond networks while meeting their stringent isolation demands and data sharing regulations [3,4]. In fact, the FL paradigm enables knowledge sharing between distributed entities without exposing their local data, which allows improved learning accuracy, better data privacy preservation and reduced communication cost [5]. All these benefits have propelled the recent activities being progressed in different standardization organizations and alliances for adopting distributed AI in next generation autonomous mobile networks, including 3GPP [6,7], ETSI ZSM [8], ETSI ENI [9], ITU-T [10,11], and O-RAN [12]. Nevertheless, the utilization of AI, including distributed AI, to enable autonomic capabilities unleashes new attack vectors which may impair network performances and security if appropriate safeguard measures are not put in place [13]. For instance, the adoption of FL paradigm in SelfDN calls for effective mechanisms to prevent the manipulation of shared knowledge (i.e., local model updates) and guarantee that collaborative entities are not malicious. Blockchain, as a potential distributed ledger technology, comes into play to meet this goal, thanks to its intrinsic properties of decentralization, immutability, transparency, security and privacy [5].

* Corresponding author at: Aalto University, Espoo, Finland.
E-mail addresses: othmane.hireche@aalto.fi (O. Hireche), chafika.benzaid@aalto.fi (C. Benzaïd), tarik.taleb@oulu.fi (T. Taleb).

https://doi.org/10.1016/j.comnet.2021.108668
Received 29 May 2021; Received in revised form 30 September 2021; Accepted 30 November 2021
Available online 21 December 2021

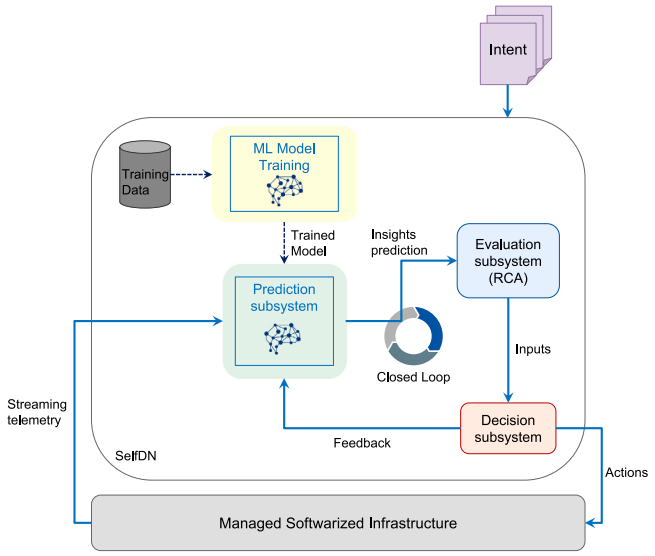

Fig. 1. A typical SelfDN architecture [2].

AI is expected not only to contribute to the vision of SelfDNs, but to also speed up its realization, particularly when intelligently leveraged with Data Plane Programming technologies (e.g., DPP – P4). Effectively, quite a number of devices in today's networks are programmable and P4 is a noticeable example of DPP languages, entirely orthogonal to the underlying architecture of the network. Jointly leveraging AI and P4 comes with numerous advantages, thanks to the variety of telemetry data that can be collected from network components using P4; i.e., not only information about network flows but also In-band Network Telemetry (INT). This forms an extremely large amount of information to be processed by AI.

Intuitively, in the context of SelfDNs, AI and DPP languages can be used to carry out actions and add flow rules in the match/action tables of network components with specific intents. In this paper, the intention goes beneath this simple vision, by leveraging AI to (re)write the source code of the devices in a fashion that the network becomes capable of automatically creating a match/action table, compiling the new code, and injecting it into a selected set of network components. Hereby, it is worth highlighting recent work on code writing using AI (e.g. [14,15]), which goes inline with the envisioned concept. This vision is further extended by defining a novel framework to empower fully distributed trustworthy SelfDNs across multiple domains, and this by relying on the potentials of blockchain and FL to enable decentralized, secure and trustable knowledge sharing.

The remainder of this article is organized in the following fashion. Section 2 introduces the basics of SelfDNs, while Section 3 motivates the need of empowering SelfDN in 5G and beyond networks. Section 4 provides background information related to key emerging technologies considered in this work, namely P4 language, genetic programming, FL and blockchain. Section 5 presents the proposed AI-powered trustworthy distributed SelfDN framework, describing its key functional blocks and highlighting the potential of the aforementioned technologies in realizing this vision. It particularly discusses the workflow for collecting network telemetry using P4 (i.e., as an example of DPP languages) and how AI can be used to write a P4 code. Section 5 also introduces our envisioned blockchain-based decentralized FL approach. Section 6 discusses a relevant use case and presents the related performance results. The paper concludes in Section 7.

2. Self-driving network basics

A self-driving network aims to empower intelligent closed-loop operation and management of the network. As shown in Fig. 1, the main steps in the closed-loop of a SelfDN include: (1) the collection of streaming telemetry from the network; (2) the use of Machine Learning (ML) to extract insights and make prediction; (3) the evaluation and decision making based on the intent specifying the desired network behavior [2].

2.1. Intent

Intents are a key enabler of automation, which is the foundation of self-driving networks. An intent is a high-level description of how the network should behave, without specifying the low-level configurations of network entities to fulfill the desired behavior. Thus, intents are about "*what* is required" rather than "*how* it is achieved". For instance, a network operator may specify that two devices must have the same quantity of traffic going through them, which will be automatically translated by the system into the deployment of a load balancer. In another scenario, the service model of a network slice may contain the high-level specification that the "HTTP traffic from slice X to the Internet needs a high security level", which will be automatically mapped to a service chaining of three virtual security functions (VSFs), namely a firewall, a Deep Packet Inspection (DPI), and an Intrusion Prevention System (IPS) [16]. While several languages have been proposed to describe the intents, an ultimate goal sought by SelfDN is to empower the automatic creation of intents from network operator's intentions expressed in a natural language. The contribution in [17] is a step towards this goal. In this work, the authors leveraged Machine Learning (ML) and feedback from the operator to extract the intent's main actions and target entities (e.g., network endpoints, middleboxes) from a pure natural language.

2.2. Real-time telemetry

The operations of SelfDN are driven by telemetry data collected from the network. To achieve the SelfDN vision, to autonomously manage the network based on the continuous assessment of its status, a real-time gathering of relevant telemetry data is necessary. In order to reduce the amount of telemetry data to be collected, the type of data and the frequency of collection need to be specified. Moreover, a special attention should be paid to the quality of telemetry data. In fact, inaccurate or incomplete data can have a significant impact on the accuracy of decisions made by the SelfDN. The data quality refers to data that are fit for use by data consumers [18]. Accuracy, completeness, validity, timeliness and consistency are the key metrics to quantify the data quality. Finally, a common data model needs to be defined in order to normalize the data format, allowing its automatic parsing and use. OpenConfig Streaming Telemetry [19] initiative aims at defining a vendor-agnostic model for telemetry data.

2.3. Prediction

The prediction refers to the process of analyzing the streamed telemetry data to extract hidden insights, such as network anomalies and forecast of traffic load and service quality. To this end, mechanisms to process and correlate a large volume of multi-dimensional and time-varying data are required. ML is poised as an ideal candidate to meet this goal, thanks to its capabilities to perform multi-model learning on real-time and historical data to uncover hidden patterns and predict the future. Among various ML techniques, Deep Learning (DL) has recently emerged as a powerful technique having the ability to automatically learn useful features from multi-dimensional raw data, to capture dependencies in a spatio-temporal data space, and to scale its performance with the volume of data. The aforementioned capabilities combined with the technology development (i.e., memory and processing power) are the key drivers behind the recent success and popularity of DL in many application domains, including mobile and wireless networking [20]. In fact, DL has exhibited superior potential in

solving complex problems that are beyond the capabilities of traditional learning approaches. However, the efficiency of emerging ML models in delivering accurate and timely insights depends largely on the amount and variety of correlated data, but above all, on their quality [21]. As aforementioned, the use of poor-quality data that involve repeated, corrupted, missing, incomplete, inconsistent, or abnormal entries has a direct impact on the inferred insights, entailing the risk of increasing uncertainty and decreasing the reliability and optimality of decisions made by SelfDN. Therefore, a standardized procedure and advanced methods for cleansing and preparing data to fit for ML models are paramount to improving both training and inference accuracy. The ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) joint technical committee on artificial intelligence (ISO/IEC JTC 1/SC 42) is currently preparing a new series of four standards ISO/IEC AWI 5259[1] which addresses the quality of data for analytics and ML. The series aims, among other things, at describing criteria for data quality and a standardized procedure for data processing. Furthermore, ML is also recognized as a key enabler for improving and assessing the data quality in an automated way. Indeed, ML has proven to be an effective method for dimensionality reduction, feature extraction, de-duplication, outliers detection and removal, and training data valuation [22,23].

Besides data quality, model building is a vital stage to construct ML models that can achieve higher prediction/detection accuracy. This stage includes the selection of the appropriate learning algorithm and the optimization of its associated hyperparameters [21]. It is worth mentioning that hyper-parameter tuning is an expensive process requiring several training, validation and testing trials in order to find the optimal set of hyperparameters yielding the higher accuracy. Thus, innovative methodologies are required to automate and accelerate the design of high-performance ML models. The recently emerging AutoML (Automated ML) [24] and NAS (Neural Architecture Search) [25] approaches can be leveraged to automatically design high-scale and efficient ML models [26]. Moreover, transfer learning [21] and parallelization of hyperparameter optimization methods [27] are promising solutions to tackle the slow training issue.

Once the trained ML model is deployed, it runs the risk of becoming outdated over the time due to drift between the model training context and the current operating environment. In fact, the dynamicity of the operating environment can lead to changes in distribution of input data (i.e., data drift) and/or the target variable to predict (i.e., concept drift), resulting in degradation of the model's predictive performance [13]. Thus, continuous update of the deployed ML model is necessary to cope with drift and boost the autonomous driving capabilities. Model retraining and online learning are common strategies to deal with model drifts [28].

2.4. Evaluation

In SelfDN, the evaluation is a non-stop operation that constantly monitors the network status using telemetry data and extracted insights, compares the current status to the intended status stated in the intent, and takes necessary actions to restore compliance with the specified intent. The evaluation can either be a straightforward assessment of telemetry data (e.g., checking if the defined Service Level Agreements (SLAs) are met) or rely on extracted insights to conduct a root cause analysis (RCA) in order to identify the cause behind a failure, a dysfunction, or a security incident. ML, particularly DL, is deemed to play a pivotal role in fostering self RCA that can automatically identify the root causes by analyzing and correlating a large amount of high-dimensional data and provide timely and actionable feedback [13]. To meet the scalability and timeliness requirements of SelfDN, cascaded DL models can be leveraged to devise RCA mechanisms. It is worth mentioning that cascaded DL models can be trained in parallel with less training data [29].

Similar to the prediction stage, ensuring the data and model quality is a key requisite to empower accurate ML-driven RCA mechanisms.

2.5. Decision-making

Leveraging the evaluation results, the decision-making sub-system isolates the cause of the problem, and finds out the appropriate actions to apply in order to mitigate the problem. Due to the interconnected nature of the network and the potential problems that may stem from applying the new actions, it is pivotal to determine the right time of their execution. Moreover, backup and roll-back strategies need to be established, which allows to restore the network to its status before applying the action. Deep Reinforcement Learning (DRL) has shown to be a promising approach to enable self-adaptive and online decision-making in high-dimensional state–action space, which is essential for empowering SelfDN capabilities in complex and dynamic networks such as 5G and beyond networks [30].

3. SelfDN for 5G and beyond networks

While in the previous section we provided an overview of the main stages composing the operation and management closed loop of a SelfDN in general, here we specifically motivate the need of empowering SelfDN in 5G and beyond networks and discuss the key requirements and challenges that should be taken into account to enable self-driving mobile networks.

5G networks are being architected as highly programmable, extremely flexible and holistically-managed infrastructures that can meet the differentiated SLA (Service Level Agreement) requirements for distinct services and tenants [21]. Densification, softwarization, virtualization, edge computing and network slicing technologies are playing a key role in achieving this goal. Various applications and services, including enhanced mobile broadband, machine-to-machine communication, virtual and augmented reality, among others, have emerged demonstrating the potential of 5G in providing lower latency, higher data rates, as well as increased connectivity and energy efficiency. However, 5G capabilities are already envisioned to fall short to cater to stringent performance demands in terms of delay, reliability and efficiency of future use cases, such as holographic type communications, multi-sense networks, digital twins, and time engineered applications [31]. The foreseen limitations have spurred the recent efforts to evolve mobile networks beyond 5G to further enhance 5G capabilities by delivering ultra-low latency, ultra-high throughput, ultra massive connectivity while providing features that foster sustainability, such as ultra-low energy usage, ultra-high reliability, scalability and autonomy. Besides using advanced radio technologies, beyond 5G networks will pursue the shift towards a truly cloud-native, service-based architecture with a higher degree of virtualization and network programmability, and the support of ubiquitous intelligence [32].

The plurality of interconnected devices and services, coupled with the increasing demands in performance, flexibility, and cost-efficiency will inevitably pose significant complexity in managing and operating 5G and beyond networks. Thus, a shift towards SelfDN paradigm will play a key role in managing this complexity in order to fulfill 5G and beyond promises. Nevertheless, the realization of Self-driving mobile networks will not only inherit the challenges related to data and ML models quality as well as the standardization of data model and handling procedure as discussed in the previous section, but will come with some additional considerations that need to be taken into account.

Although 5G and beyond networks will be characterized by a large amount of data produced by different sources (e.g., services/1u applications, network elements) from different network domains (e.g., Radio Access Network (RAN), Core Network (CN), Transport Network (TN)), it is challenging to have enough high-quality data available for training

[1] https://www.iso.org/committee/6794475.html.

and validating ML models to be involved in the self-managing tasks. In fact, the network-typical and network characteristic datasets for 5G and beyond networks are actually scare since the roll-out of 5G networks has just started and the beyond 5G networks are still under specification [33]. Moreover, tighter privacy regulations (e.g., GDPR) may limit the useful data to be collected and shared. The use of Generative Adversarial Networks (GANs) and Federated Learning are two promising directions to address this data scarcity in compliance with regulation. GANs can be applied to efficiently create realistic synthetic network data based on limited historical data [34]. Federated Learning is an emerging distributed ML approach that has the advantage of learning a global model from local data and limiting the information exchange to the model parameters, which results in privacy-preservation and reduced communication cost [35].

The variety of data sources and types calls more then ever for a unified framework for handling network data. The recent initiatives by ITU-T and ETSI ENI are concrete steps in that direction. The ITU-T Y.3174 recommendation provides a data handling framework for enabling ML in future networks including mobile networks [36]. The proposed framework supports requirements related to ML data collection, ML data processing and ML data output. ESTI ENI introduces a high-level reference framework that describes some technical methods for generating high-quality actionable data efficiently and in a timely manner to support data-driven intelligent networks [37].

Finally, this paradigm-shift towards network intelligentization and full autonomy will raise new security concerns stemming from the use of AI. Indeed, ML techniques have been proven vulnerable to several adversarial attacks that can influence them to learn wrong models or make erroneous decisions/predictions, which may endanger not only the performance expectations of 5G and beyond networks but also the people's lives [13]. Hence, building robust ML models is paramount to foster trust in SelfDNs. Different defenses can be adopted to enable ML models robustness including input validation, adversarial training and moving target defense [13]. Being aware of the importance of AI security, the topic has recently gained attention from standardization bodies. For instance, ISO/IEC JTC 1/SC 42 has published two technical reports providing an overview of trustworthiness in AI [38] and an assessment of the robustness of neural networks [39]. ETSI Industry Specification Group on Securing Artificial Intelligence (ISG SAI[2]) has released three reports on the security threats, the mitigation strategies and the security of the data supply chain for ML-based systems and solutions in ICT field.

4. Preliminaries

In this section, we present the preliminary concepts that are required to follow the rest of the paper, namely P4 language, genetic programming, Federated Learning, and Blockchain.

4.1. P4 language

P4 is a domain-specific language for programming protocol-independent packet processors [40]. It allows to express how packets are processed by the data plane of programmable network elements such as hardware or software switches, network interface cards (NICs), or routers. P4 is devised to meet three main goals:

- *protocol-independence*, meaning that the network element should not be tied to any particular network protocols. P4 enables the specification of new header formats with new field names and types;
- *target-independence*, which means that the P4 program is independent of the network element details; and
- *reconfigurability*, providing the ability to reprogram the packet processing logic of the network element at runtime.

A typical P4 packet processing pipeline includes a *parser* for extracting the header fields, a set of *match-action tables* to process and modify the extracted headers, and a *deparser* for reassembling the processed headers into the outgoing packet [41]. Each match-action table matches specific keys from the header fields and computed metadata and invokes the corresponding actions on the packet upon matching. The order in which the match-action tables are applied to a packet is determined by an imperative control flow. The table entries are typically populated at runtime by the control plane.

P4 provides stateful memories (i.e., counters, meters and registers) for maintaining state across multiple packets when they are traversing the switch. The stateful memories can be *global* (i.e., they can be referenced by any match-action table) or *static* (i.e., bound to one table).

4.2. Genetic programming

Genetic Programming (GP) [42] is an evolutionary approach that extends Genetic Algorithms (GAs) for automatic writing of computer programs. It consists in applying GAs to a population of randomly created candidate programs to solve a target problem. A computer program (i.e., phenotype) is encoded as a genome (i.e., genotype). The population is progressively evolved over a series of generations. The programs in each generation are evaluated against a fitness function and the ones who achieve a better fitness score are selected for breeding in the new generation using crossover and mutation operations. The population evolvement process stops once the program that can solve the problem at hand is produced; i.e., the program with a satisfactory fitness value.

4.3. Federated learning

As stated in previous section, Federated learning (FL) is an emerging distributed ML approach that aims to train a global model while preserving the privacy of the distributed data. The traditional FL follows a client–server architecture in which a set of clients collaboratively train a global model by aggregating their locally-computed model updates through a central server while keeping their training data localized [43,44]. Once updated, the new improved global model is sent back to the clients and the procedure is repeated until the global model converges. By limiting the information exchange to the model parameters, FL fosters the sharing of locally learned knowledge with better control of data privacy and reduced communication cost. These benefits have prompted the recent growing interest in applying FL for 5G and beyond networks to meet their stringent isolation demands and data sharing regulations [4,45]. However, the reliance of traditional FL on a central unit poses both single point of failure and scalability issues. These limitations are serious obstacles to cater to the high scalability and reliability needs of 5G and beyond networks.

To mitigate the aforementioned issues, recent research works have been investigating the decentralization of the FL process to enable the aggregation of model updates in a distributed way; that is without requiring a central server [3,46,47]. For instance, Daily et al. [48] devise a collaborative learning strategy based on decentralized Asynchronous Gradient Descent (SGD) and using gossip communication for mutual exchange of model updates. In [49], a serverless FL approach that uses consensus for mutual interactions is developed. The authors in [50] propose a fully decentralized federated method that is based on interplanetary file system (IPFS). The work in [51] introduces the teacher–student concept whereby student models are learning from the outputs of teacher models on auxiliary samples generated from the locally available training data. While this approach eliminates the exchange of model parameters, it requires the sharing of auxiliary data.

In this paper, we devise a novel method for enabling decentralized FL in SelfDN with multi-domain setting.

[2] https://www.etsi.org/committee/sai.

4.4. Blockchain

The Blockchain technology provides a zero-trust mechanism that allows secure transactions without relying on a trusted third party; it achieves this by recording all the information exchanges in a set of blocks as digitally-signed transactions [52]. The blocks are chained together by adding in each block the hash of the previous block. Each block is first validated using a consensus protocol among nodes participating in the blockchain network before being added to the chain. Blockchains can be *permissionless* where anyone can join the blockchain and participate in the consensus process, or *permissioned* where its participants are only authorized entities. The key features of blockchain – decentralization, immutability, transparency, security and privacy – make it a potential candidate to empower zero-trust in 5G and beyond networks [5].

In this paper, we leverage blockchain to foster trust in data and communications in a multi-domain SelfDN.

5. AI-powered trustworthy distributed SelfDNs

Software-defined, programmable networking fabric is recognized as a key enabling technology to support 5G and beyond networks in achieving their promises of high-level scalability and flexibility with lower CAPEX and OPEX [21]. Going beyond 5G will be also characterized by pursuing deep programmability of the network fabric both vertically (i.e., control and data plane) and horizontally (i.e., end-to-end from the radio access network to edge and core network) [53] to fulfill the stringent performance requirements. In fact, endowing the next-generation mobile networks with data plane programming capabilities will bring various benefits, including dynamic traffic engineering at wire-speed, INT for latency-critical services, slicing and multi-tenancy, in-network security, and offloading of 5G VNF to data plane [54].

In this section, we propose a novel framework that leverages the potential of emerging technologies, namely Blockchain, Programmable Data Planes (PDP), and AI, to empower fully distributed trustworthy SelfDN across multiple domains. Fig. 2 illustrates the high level architecture of the proposed framework. We consider a multi-domain SDN-enabled network with PDP. We consider a multi-domain network that supports both control-plane and data-plane programmability using SDN and PDP, respectively. In addition to flexibility introduced by control-plane programmability, data plane programmability enables customized network functions/protocols to be deployed, stateful packet processing at wire speed as well as dynamic reconfiguration of the processing logic at run-time using domain-specific networking description language (e.g., P4). The aforementioned capabilities make PDP a key enabler to realize SelfDN.

In each domain, self-managing capabilities (e.g., self-configuration, self optimization, self-healing, self-protection) are enabled through ML-powered closed loops. The deployed closed loops leverage ML techniques to identify/predict performance and/or security issues based on constant processing of telemetry data produced by the data plane, and thus derive the corrective/preventive strategy to address the issue in order to meet the desired performance and security levels. The corrective/preventive strategy is issued as an intent that is automatically translated into a set of executable actions (e.g., P4 code) to be enforced on the programmable switches via the SDN controller. The translation is a complex process since the set of actions will depend not only on the new intent but also on various other parameters, such as the currently enforced intents and network topology. Thus, an AI-assisted solution is required to tame this complexity and generate optimal data-plane codes that are context-aware.

The effectiveness and efficiency of the deployed self-managing closed loops is contingent on the accuracy of the incorporated ML models, which in its turn heavily relies on the availability of a large amount of high-quality training data. Such data may not be available in one domain and collaboration between domains is essential to improve accuracy and enable cross-domain self-managing operations. Nevertheless, the exchange of raw data among domains may not be possible due to strict business, security and privacy policies and regulations established to prevent leakage of sensitive information. To deal with these restrictions while fostering fully distributed collaboration between domains, we adopt a decentralized FL approach. This results in increased training efficiency and allows the sharing of learning knowledge without compromising data security and privacy. Furthermore, the adoption of a fully decentralized FL enables greater communication efficiency compared to the vanilla FL, as a central entity is not required to exchange the model updates.

Despite its benefits in preserving data privacy, FL is proven vulnerable to adversarial attacks [13]. In fact, malicious participants may launch poisoning attacks by uploading false or low-quality local model updates to undermine the accuracy of the global model. To overcome this challenge, we employ blockchain to enable secure and trustable decentralized FL. The blockchain ensures the integrity of the local model updates to prevent their manipulation. Furthermore, the quality of the local model updates is assessed using a smart contract, allowing to deter poisoning attacks by considering only local models with high performance in the update of the global model.

In the remainder of this section, we discuss the potential of programmable data plane in enabling real-time in-network telemetry collection, demonstrate how GP can be leveraged to automatically translate an intent into a P4 code, and describe the envisioned blockchain-based decentralized FL mechanism.

5.1. Programmable data plane for telemetry

The envisioned high dynamicity and large-scale of 5G and beyond networks make traditional techniques to monitor network state insufficient to entitle self-driving capabilities. In fact, the conventional sampled and probe-based techniques use poll-based strategy to fetch data through data plane, cover a limited range of data, lack the capability of aggregating data from multiple sources, and result in network resource consumption [55]. Programmable data plane, using e.g. P4 language, comes as a solution to overcome these limitations by enabling In-band Network Telemetry (INT), which consists in collecting and reporting the network state at the data plane level, without needing the involvement of the control plane [56]. Three operation modes are distinguished for an INT application based on the level of packet modifications, namely [56]:

- INT-XD (eXport Data) mode, known also as Postcard mode, whereby the metadata are exported directly from the data plane to the monitoring system without packet modification;
- INT-MX (eMbed instruct(X(ions))) mode, whereby INT instructions are incorporated in the packet header. Each receiving device sends the metadata directly to the monitoring system based on the embedded instructions;
- INT-MD (eMbed Data) mode, whereby both INT instructions and metadata are embedded in the packet and forwarded to the next hop until reaching the last network device before destination. The last network device removes the INT instructions and aggregated metadata from the packet and sends them to the monitoring system.

Using P4, we can generate different types of telemetry reports, including tracked flow reports, dropped packet reports, and congested queue reports.

The capabilities of P4-based programmable data plane to provide real-time telemetry with the ability to specify which information need to be collected make it a key driver to enable SelfDN. In the next subsection, we demonstrate how we leverage P4's registers and INT for real-time detection of Application-layer DDoS attacks.

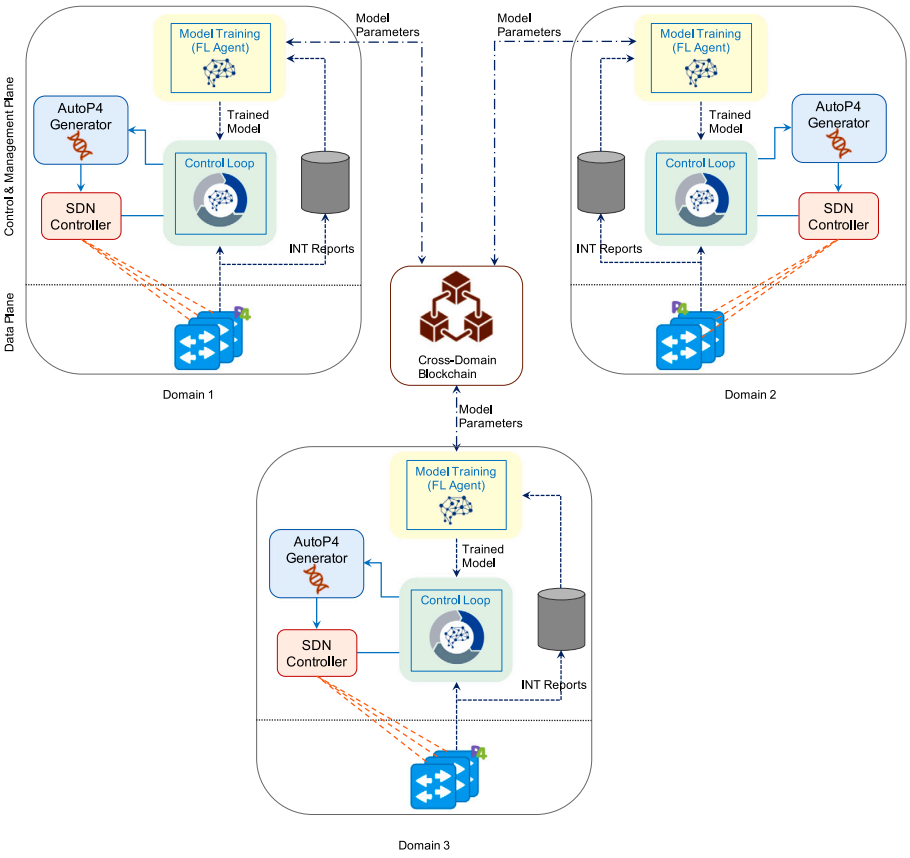

Fig. 2. High level architecture of fully distributed trustworthy SelfDN.

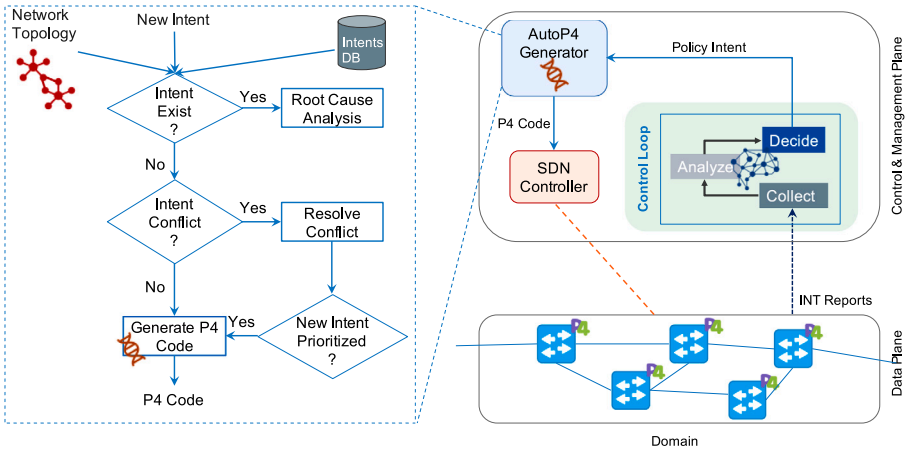

Fig. 3. P4 code generation process.

5.2. AI-assisted data plane code generator

To fully reap the benefits of programmable data plane in realizing SelfDN, the network should be able to automatically generate the code of the network devices based on intents. The flow-chart in Fig. 3 illustrates the proposed process for translating an intent into a P4 code. Once an intent is received, the code generator starts first by checking if this intent is already enforced in the network. In such case, a root cause analysis (RCA) is triggered to determine the actual cause of non-fulfillment of the intent. The RCA can be carried out either manually by the domain administrator or automatically leveraging the potential of ML [13]. The next step consists in identifying and resolving potential conflicts between the newly created intent and the existing intents. If no conflict is detected or the conflict resolution procedure prioritizes the newly created intent, the P4 code generation is triggered. AI can be a potential game-changer for making automatic data plane code generation a reality. In fact, recent works (e.g., [15,57,58]) have demonstrated how machine learning techniques can be leveraged to generate code source. For instance, the authors in [15] used neural networks to write Java code. In [58], GP is used to automatically repair buggy java programs. The work in [58] presents GP4P4, a first proposal for automatically creating P4 code for small network functions (e.g., firewall, Network Address Translation (NAT), and router) leveraging GP. Although GP4P4 shows the feasibility of autonomous P4 code creation, it exhibits some limitations. In fact, the generated code

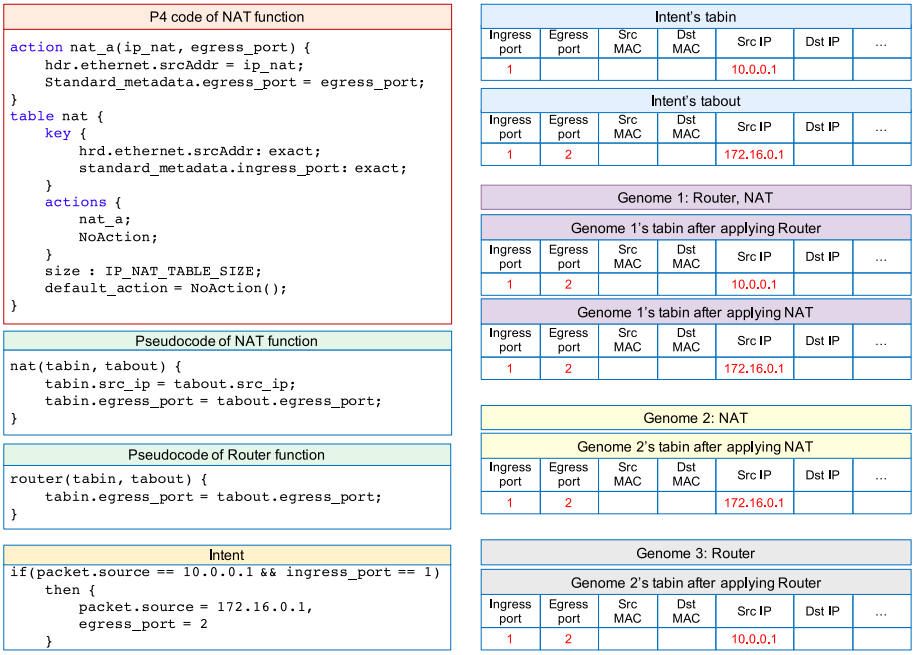

Fig. 4. Illustrative example of intent and genomes.

is in the form of a sequence of IF–THEN statements rather than match-action table (MAT) abstractions, which can largely increase the lookup time. Moreover, starting the code generation from evolving sequences of basic one-line declarations in P4 increases considerably the search space and time for GP algorithm.

To overcome the aforementioned limitations, we propose AutoP4; a new GP-based P4 code generation solution that instead of starting from scratch by evolving sequences of basic one-line declarations in P4, it takes as a starting point sequences of MATs. It is worth mentioning that using a high-level abstraction like MATs reduces notably the search space, and consequently the code generation time. Nevertheless, finding the optimal combination of MATs to generate the P4 code that satisfies the newly created intent as well as the intents that are already enforced on the switch with a reduced number of MATs is still a complex problem. To solve this problem, we adopt GP to automatically generate P4 code. Thus, a genome in the population represents the set of MATs in the candidate P4 program to satisfy the specified intents. A genome is encoded as a binary string of 0's and 1's, whereby a binary bit is set to 1 to indicate that the corresponding MAT is selected and to 0 to denote that the MAT is removed.

As illustrated in Fig. 4, an intent is expressed as a conditional statement that relates the changes (i.e., output) to be performed on the packet to the conditions (i.e., input) that should be fulfilled by its metadata and header fields. The intent in Fig. 4 states that if the packet is coming from port 1 and its source IP address is 10.0.0.1, the packet should be forwarded through port 2 after changing its source address to 172.16.0.1. This intent represents the implementation of a NAT function. To facilitate the assessment of intent satisfaction by a genome, we describe each MAT by a pseudo-code that represents the changes to be applied on a packet when it traverses this MAT. Fig. 4 shows an example of the P4 code of NAT (Network Address Translation) function and the corresponding pseudo-code. Furthermore, we define two arrays *tabin* and *tabout* that contains the values of the packet's metadata and packet fields instantiated from the intent's input and output, respectively. An intent is satisfied by a genome if providing the intent's *tabin* as an input to this genome, we get the intent's *tabout* as an output. As shown in Fig. 4, the *Genome 1*, composed of two MATs for NAT and routing functions, satisfies the intent in the example. This is the case also for *Genome 2*. However, *Genome 3* fails to fulfill the intent.

The population is progressively evolved over a series of generations until a termination criterion is met. In each generation, the GP process evaluates each genome against a fitness function, defined as:

$$ft = \frac{1}{m}\sum_{k=1}^{m} verf(intent_k) \tag{1}$$

where m is the total number of intents to satisfy by a genome. $verif(intent_k)$ is equal to 1 if the kth intent is satisfied; otherwise, it is set to 0. Thus, the fitness score assigned to each genome represents the proportion of intents satisfied by the genome relative to the total number of intents that need to be satisfied. The two genomes with the highest fitness scores are selected for breeding in the new generation by first applying the crossover operator to the selected genomes, then the mutation operator. The GP process terminates when it produces a genome that can satisfy all intents (i.e., its fitness score is equal to 1) or when the maximum number of iterations is reached. The P4 code corresponding to the genome with the highest fitness and the lowest number of MATs is generated and pushed on the network device.

5.3. Blockchain-based decentralized federated learning in SelfDN

To foster collaboration between domains for enhanced learning accuracy while preserving data privacy, we adopt a fully decentralized FL approach. Let consider a training task (e.g., anomaly detection, resource allocation) that is carried out cooperatively by a set of N domains $\{D_1, D_2, \ldots, D_N\}$. Each domain D_i uses its own dataset $S_i = \{(x_k, y_k)\}_{k=1}^{n_i}$ to train its local model (e.g., a DL network), where x_k, y_k and n_i are, respectively, the features vector, the label and the number of training samples. Let S denotes the global input space which satisfies $S = \cup_{i=1}^{N} S_i$ and $|S| = \sum_{i=1}^{N} n_i = N_S$. The goal of the cooperative learning is to minimize the following objective function

$$\min_{\omega}\left\{ F(\omega) \triangleq \frac{1}{N_S}\sum_{i=1}^{N}\sum_{(x_k,y_k)\in S_i} f(\omega;(x_k,y_k))\right\} \tag{2}$$

where $F(.)$ denotes the global loss function and $f(.;.)$ is the local loss function (e.g., mean-squared error) on one data sample. ω represents the global weight vector.

The local model updates (i.e., weights) ω_i of the domain D_i are uploaded to the blockchain for integrity assurance and a notification

is broadcast to inform the involved domains about the availability of new updates. A permissioned blockchain (e.g., Hyperledger Fabric) is considered to guarantee not only security and privacy, but also to provide performance in terms of network throughput and latency. It is worth noting that a permissioned blockchain is accessible only by authorized entities, allowing to form a consortium blockchain.

The blockchain is also leveraged to counteract poisoning attacks against FL models. To this end, and similar to [59], we use smart contracts to assess the quality of model updates and automatically identify malicious participants in the learning process. The smart contract is created by the initiator of the training task (e.g., a domain administrator) and includes a validation dataset and an evaluation criterion (e.g., accuracy performance). The $Verify\_Updates\_Quality()$ function of the smart contract is invoked each time the updates of a local model are uploaded to the blockchain. It allows to assess the local model quality against the validation dataset to decide its trustworthiness. A local model is considered trustable if the evaluation accuracy is above a desired threshold. The use of smart contract for model quality verification brings in a higher transparency which is essential to trust the participant domain.

Upon receiving a model update notification from $\mathcal{D}_j$, a domain $\mathcal{D}_i$ proceeds as follows:

- If $\mathcal{D}_i$ is training its local model, it broadcasts a *pause* message to inform the other domains that an update is ongoing in order to wait its new model before performing the aggregation. A timer T_{pause} is then triggered at the destination.
- If this update notification is the first since the last aggregation operation, $\mathcal{D}_i$ triggers a timer T_{notif} to wait for more updates before starting the aggregation process.
- Once the waiting timers expire, $\mathcal{D}_i$ retrieves the model updates that are qualified trustable by the smart contract and computes the global model locally using the Federated Averaging (FedAvg) technique.

6. Case study - DoS attack detection and mitigation at data plane

This section presents a practical study that demonstrates a self detection and mitigation of Denial of Service (DoS) attacks leveraging the INT capability of PDPand the potential of deep learning, and using the AutoP4 generator to automatically generate the P4 code needed to enforce the mitigation policy. It is worth mentioning that the main aim of this study is to show the feasibility of implementing a fully autonomous self-defense closed-loop that integrates intelligence at both anomaly detection stage and mitigation policy generation stage.

6.1. Testbed architecture

Fig. 5 illustrates the test-bed setup. We used ten (10) virtual machines (VMs) hosted on the same physical server. The first VM (Services) hosts a streaming service and a web server, each of them deployed on a container. The second VM (Controller) acts as the brain of the system, with the capabilities of analyzing the received telemetry data and deciding the actions to enforce. The Controller relies on a DL model built using Multi-Layer Perceptron (MLP) algorithm to identify application-layer DDoS attacks [60]. The MLP network consists of one input layer, two hidden layers with 64 neurons each, and a two-class softmax output layer. The model takes as an input the flow features extracted from the collected from the network traffic and provides as an output the traffic class; that is, legitimate traffic or DDoS traffic. It is trained on the DDoS dataset we created in [60] and achieved an accuracy of 99.65%. Two VMs are used to simulate, respectively, three legitimate clients accessing the streaming service through the web server and four attackers performing a high-rate HTTP-based flooding attack against the server web to make it inaccessible. The DDoS attack is launched using Hulk[3] tool. The aforementioned VMs are connected through six P4-enabled BMv2 [61] switches, each of them deployed on a separate VM. We used Linux bridges to link the VMs' network interfaces directly to the switches.

Our goal is to first detect whether a DDoS attack is occurring or not based on the analysis of the telemetry received from switches. Upon attack detection, the system issues a new mitigation policy in the form of intent. The mitigation policy corresponds to adding a new firewall rule to drop packets from the attacker. If the firewall function is already deployed on the switch, the controller populates the firewall table with the new drop rule; otherwise, a new P4 code is first generated using GP in order to add the firewall function to the switch.

6.2. Telemetry data

A key metric to detect the high-rate HTTP-based flooding attack is the large number of packets sent by the attacker in a short period of time. To this end, we created a register ($Flow\_Count$) on the switch to save the number of packets for each network flow. It is worth mentioning that a network flow is defined by the set of packets sharing the same 5-tuple (SRC IP, DST IP, Proto, SRC Port, DST Port), where SRC IP, DST IP, Proto, SRC Port and DST Port denote the source IP address, the destination IP address, the protocol, the source port and the destination port, respectively.

We defined two strategies to read the telemetry data. In the first strategy, called *Classical Telemetry*, the information in $Flow\_Count$ register is read periodically by the Controller. The collection period is set to $10s$ in our case study. The shortcoming of the first strategy is that the information is collected even if no security issue is happening. To overcome this limitation, the second strategy adopts INT telemetry approach, where the Controller is notified only if an abnormal flow is detected. To this end, we added two other registers ($Time\_Reg$ and $Flow\_Reg$) and a set of actions in the ingress step of the packet processing in order to check if the number of packets received over a period of time exceeds the defined threshold. Algorithm 1 illustrates the actions code. In the conducted study, the Controller is notified when the time interval during which 100 packets have been received is less than 300 ms. Once notified, the Controller requests and monitors the content of $Flow\_Count$ register over a period of time. If the Controller notices that the number of packets is considerably increasing, it will take the decision to drop the packets received from (SRC IP, SRC Port). The new intent is enforced on the switch as explained above.

Algorithm 1 Updating Registers when receiving packets.

```
if (packet.ipv4) then
    Flow_Count[packet.id] ++;
    if (Flow_Count[packet.id] == 1) then
        Time_Reg[packet.id] = Timestamp();
    end if
    if (Flow_Count[packet.id] > Threshold_Packet) then
        Flow_Count[packet.id] = 0;
        if (Timestamp − Time_Reg[packet.id] < Threshold_time) then
            Flow_Reg ++;
            if (SEND_Msg == false) then
                ▷ forward the packet to the controller to specific
                  application port
                set_packet(packet, IP_CONT, PORT_CONT);
                SEND_Msg = true;
            end if
        end if
    end if
end if
```

[3] https://github.com/grafov/hulk.

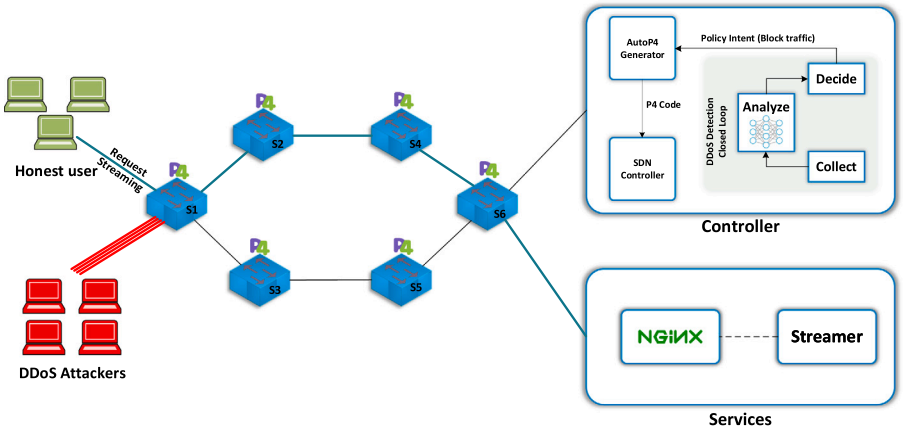

Fig. 5. Testbed architecture.

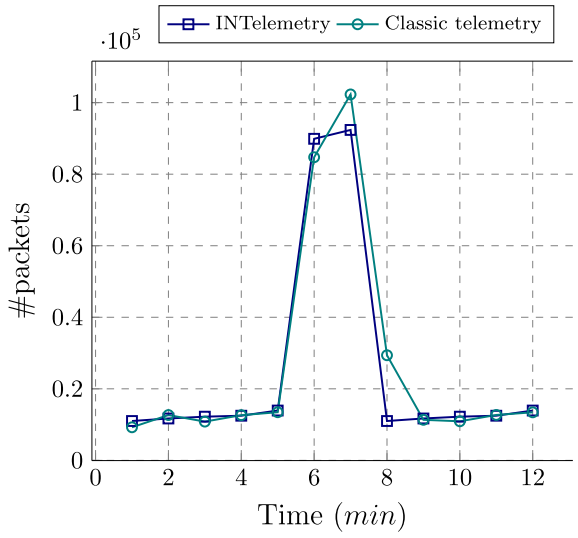

Fig. 6. Number of packets traversing the BMv2 switch when web server is under DoS attack.

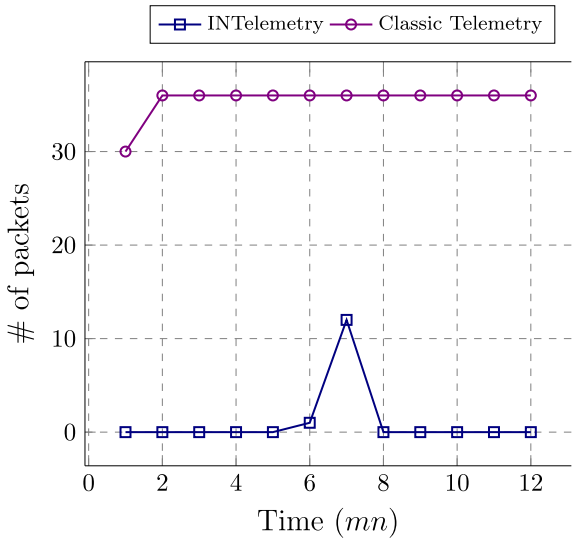

Fig. 7. Number of telemetry reports sent by BMv2 switch to the controller.

6.3. Performance evaluation

To assess the performance of the INT strategy compared to the *Classical Telemetry* strategy, we investigated their impact on the number of packets traversing the switch, and the number of telemetry reports sent to the Controller. Furthermore, the efficiency of the GP algorithm is evaluated in terms of the time needed to generate the P4 code.

6.3.1. INT vs. classical telemetry

We consider a scenario with one attacker who launches a DoS attack against the web server during a video streaming session started by a legitimate client. The attack is launched at minute 5 and stopped at minute 7. Fig. 6 shows the number of packets traversing the BMv2 switch before and during the attack is launched and after its mitigation. It is observed that after detecting the attack and deploying the firewall code on the switch by the Controller, the traffic load goes back to normal.

Fig. 7 displays the number of telemetry reports sent by the BMv2 switch to the Controller, before and during the DoS attack and after its mitigation. The Classical Telemetry strategy sends a report each 10 seconds, regardless if an anomaly is detected or not. In fact, the controller receives 36 packets each minute; the switch $S6$ connecting directly to the controller sends 6 packets/min and forwards 30 packets from the other switches, $S4$ also sends 6 packets/min and forwards 12 packets from $S1$ and $S2$. Meanwhile, the INT strategy reports the telemetry data only when the web server is under attack. This allows to considerably reduce the communication overhead entailed by telemetry data collection, making the INT strategy more desirable in large-scale networks.

6.3.2. INT-based mirroring vs. classical mirroring

To show the advantage of combining INT and DL in timely detecting and mitigating the DDoS attack while reducing the communication overhead between the data plane and the control & management plane, we evaluate the number of packets forwarded to the Controller and the mitigation time of the DDoS attack considering two strategies, namely: (i) *classical mirroring* where all network traffic flowing from/to the web server is continuously mirrored to the Controller; (ii) *INT-based mirroring* where only network traffic from suspicious sources to the web server is forwarded to the Controller. A source is deemed suspicious (i.e., potential attacker) if the number of packets received from this source over a period of time exceeds the defined threshold as defined by the INT. In both scenarios, the mirrored traffic is analyzed by the MLP model to confirm that the DDoS attack is ongoing; in which case a "traffic blocking" intent is automatically translated into the corresponding P4 code by the AutoP4 Generator and enforced on the closest switch to the attacker(s). Fig. 8 depicts the results obtained with three (3)

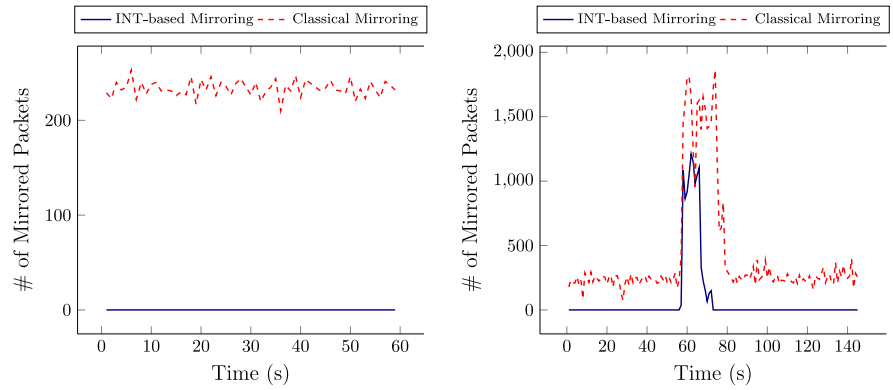

Fig. 8. INT-based mirroring vs. classical mirroring with three legitimate clients and four DDoS attackers.

legitimate clients and four (4) attackers launching simultaneous Hulk attack against the web server at 59 s.

The results demonstrate the superiority of INT-based mirroring over classical mirroring with regard to the number of packets exported to the Controller either when only legitimate traffic is received by the web server or during the attack period. In fact, INT-based mirroring strategy triggers the mirroring only when a suspicious traffic is observed and limits the forwarded traffic to the one considered suspicious. By doing so, the communication overhead between the data plane and the Controller is greatly reduced. Furthermore, the depicted results show the effectiveness of INT-based mirroring strategy in quickly detecting and mitigating the DDoS attack compared to the classical mirroring strategy. This can be explained by the fact that the selected mirroring performed by the INT-based mirroring strategy reduced the analysis time and improved the detection accuracy.

6.3.3. P4 code generation time

We assessed the efficiency of AutoP4 generator in terms of the search time required by the GP algorithm to create the MATs pipeline for satisfying a given set of intents. The experiments were conducted using a set of 50 MATs that allow to implement different network functions (e.g., firewall, NAT, router, and load balancer). Fig. 9 plots the average search time as a function of the number of intents that need to be satisfied. The search time is measured considering different sizes of the initial population. Each point in the plots is the average of 100 independent runs carried out on a machine with 4-cores Intel's 1.6 GHz CPU and 16 GB RAM.

The results reveal that the proposed GP algorithm succeeds in generating the P4 code in a reasonable time, i.e., in the order of tens to few hundreds of milliseconds, even when a high number of intents is considered. These results corroborate our claim that using a high-level abstraction like MATs as initial population will notably reduce the code generation time compared to evolving sequences of one-line declarations in P4. Furthermore, the results show that the increase in the population size allows to decrease the search time to some extent. In fact, increasing the number of individuals in the initial population improves the algorithm evolvability and hence increases the chance of finding the solution. Nevertheless, exceeding the given limit (40 individuals in our case) will lead to higher computational load due to the increased search space. Thus, defining a reasonable size of the population is vital for the algorithm convergence.

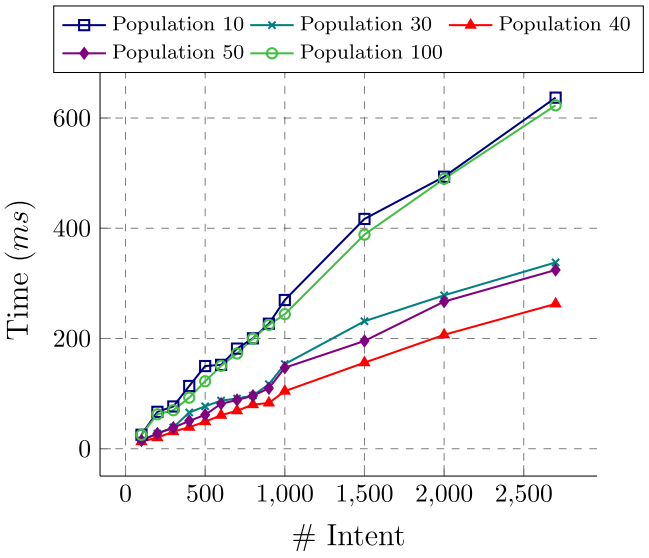

Fig. 9. Time that GP takes to generate a genome respecting all intents.

7. Conclusion

This article introduced the emerging concept of Self-Driving Networks (SelfDN). It presented a novel framework to empower fully distributed trustworthy SelfDNs across multiple domains. To achieve its vision, the framework leverages the potential of programmable data plane for enabling real-time In-band telemetry (INT); AI for automatic (re)writing of network device code; and blockchain and Federated Learning for enabling decentralized, secure and trustable knowledge sharing among domains. As a proof of concept, we demonstrated the effectiveness of INT, Deep Learning and P4 in autonomously detecting and mitigating the application-layer DDoS attacks at data plane level. In the light of the obtained results, we believe that leveraging AI for code writing will be a game-changer for achieving the SelfDN vision.

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.

Acknowledgment

This work was supported in part by the European Union's Horizon 2020 research and innovation programme under the MonB5G project (Grant No. 871780); and the Academy of Finland Project 6Genesis Flagship (Grant No. 318927).

References

[1] J. Networks, Four Reasons To Automate Your Network Right now, Juniper Networks. INC, 2016, URL https://www.juniper.net/assets/uk/en/local/pdf/whitepapers/2000634-en.pdf.

[2] K. Kompella, Self-Driving Networks, IGI Global, 2019, URL https://www.igi-global.com/chapter/self-driving-networks/214426.

[3] O.A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1342–1397.

[4] Y. Liu, J. Peng, J. Kang, A.M. Iliyasu, D. Niyato, A.A. Abd El-Latif, A secure federated learning framework for 5G networks, IEEE Wirel. Commun. 27 (4) (2020) 24–31.

[5] C. Benzaid, T. Taleb, M.Z. Farooqi, Trust in 5G and beyond networks, IEEE Netw. (2021).

[6] 3GPP TS 23.288 V17.0.0, Architecture enhancements for 5G system (5GS) to support network data analytics services (release 17), 2021.

[7] 3GPP TS 28.533 V16.7.0, Management and orchestration; architecture framework (release 16), 2021.

[8] ETSI GS ZSM 002, Zero-touch network and service management (ZSM); reference architecture, 2019.

[9] ETSI GR ENI 0018, Experiential networked intelligence (ENI); introduction to artificial intelligence mechanisms for modular systems, 2021.

[10] FG-ML5G, URL https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx.

[11] ITU-T Series Y, Supplement 55, ITU-t y.3170-series - machine learning in future networks including IMT-2020: Use cases, 2019.

[12] O-RAN.WG2.AIML-v01.02.02, O-RAN Working Group 2, AI/ML Workflow Description and Requirements, Technical Report, 2020.

[13] C. Benzaid, T. Taleb, AI for beyond 5G networks: A cyber-SecurityDefense or offense enabler? IEEE Netw. Mag. (2021) (to appear).

[14] M. Balog, A.L. Gaunt, M. Brockschmidt, S. Nowozin, D. Tarlow, Deepcoder: Learning to write programs, 2016, arXiv:1611.01989.

[15] V. Murali, L. Qi, S. Chaudhuri, C. Jermaine, Neural Sketch Learning for Conditional Program Generation, in: Proc. of 6th International Conf. on Learning Representations, 2018.

[16] C. Benzaid, T. Taleb, ZSM security: Threat surface and best practices, IEEE Netw. Mag. 34 (3) (2020) 124–133.

[17] A.S. Jacobs, R.J. Pfitscher, R.A. Ferreira, L.Z. Granville, Refining network intents for self-driving networks, in: Proceedings of the Afternoon Workshop on Self-Driving Networks, in: SelfDN 2018, Association for Computing Machinery, New York, NY, USA, 2018, pp. 15–21, http://dx.doi.org/10.1145/3229584.3229590.

[18] R.Y. Wang, D.M. Strong, Beyond accuracy: What data quality means to data consumers, J. Manage. Inform. Syst. 12 (4) (1996) 5–33.

[19] OpenConfig, OpenConfig, 2016, https://openconfig.net/.

[20] C. Zhang, P. Patras, H. Haddadi, Deep learning in mobile and wireless networking: A survey, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2224–2287.

[21] C. Benzaid, T. Taleb, AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions, IEEE Netw. Mag. 34 (2) (2020) 186–194.

[22] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, S. Schmid, Adaptable and data-driven softwarized networks: Review, opportunities, and challenges, Proc. IEEE 107 (4) (2019) 711–731.

[23] J. Yoon, S.O. Arik, T. Pfister, Data valuation using reinforcement learning, in: Proc. of the 37th International Conf. on Machine Learning, PMLR 119, 2020.

[24] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, AMC: Automl for model compression and acceleration on mobile devices, in: Proc. of 15th European Conference on Computer Vision (ECCV), in: Lecture Notes in Computer Science, vol. 11211 (2018) 815–832.

[25] B. Zoph, Q.V. Le, Neural Architecture Search with Reinforcement Learning, in: Proc. of the International Conf. on Learning Representations (ICLR), 2017.

[26] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE 107 (8) (2019) 1738–1762.

[27] J. Dodge, K. Jamieson, N.A. Smith, Open Loop Hyperparameter Optimization and Determinantal Point Processes, in: Proc. of AutoML, 2017.

[28] D. Le, N. Zincir-Heywood, A frontier: Dependable, reliable and secure machine learning for network/system management, J. Netw. Syst. Manage. 28 (2020) 827–849.

[29] A. Clemm, M.F. Zhani, R. Boutaba, Network management 2030: Operations and control of network 2030 services, J. Netw. Syst. Manage. 28 (2020) 721–750.

[30] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, L. Wang, Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges, IEEE Veh. Technol. Mag. 14 (2) (2019) 44–52,.

[31] T. Taleb, R.L. Aguiar, I.B. Yahia, B. Chatras, G. Christensen, al, White paper on 6G networking, in: 6G Research Visions, No.6, 2020, 2020.

[32] K. Samdanis, T. Taleb, The road beyond 5G: A vision and insight of the key technologies, IEEE Netw. 34 (2) (2020) 135–141.

[33] NetWorld2020, SRIA - smart networks in the context of NGI, 2020.

[34] H. Navidan, P.F. Moshiri, M. Nabati, R. Shahbazian, S.A. Ghorashi, V. Shah-Mansouri, D. Windridge, Generative adversarial networks (GANs) in networking: A comprehensive survey & evaluation, Comput. Netw. 194 (2021) 108149.

[35] O.A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1342–1397.

[36] ITU-T Y.3174, Framework for data handling to enable machine learning in future networks including IMT-2020, 2020.

[37] ETSI GR ENI 009, Experiential networked intelligence (ENI); definition of data processing mechanisms, 2021.

[38] ISO/IEC TR 24028, Information technology — Artificial intelligence — Overview of trustworthiness in artificial intelligence, 2020.

[39] ISO/IEC TR 24029-1, Artificial intelligence (AI) — Assessment of the robustness of neural networks — Part 1: Overview, 2021.

[40] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: Programming protocol-independent packet processors, ACM SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 88–95.

[41] P4 Language Consortium, P4_16 language specification, version 1.2.1, 2020.

[42] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.

[43] B. McMahan, D. Ramage, Federated learning: Collaborative machine learning without centralized training data. (2017), 2017, URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[44] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, Communication-efficient learning of deep networks from decentralized data, 2016, ArXiv E-Prints, arXiv–1602.

[45] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Blockchain and federated learning for 5G beyond, IEEE Netw. (2020).

[46] P. Kairouz, et al., Advances and open problems in federated learning, Found. Trends ®Mach. Learn. 14 (1) (2021).

[47] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, L. Barbieri, Opportunities of federated learning in connected, cooperative, and automated industrial systems, IEEE Commun. Mag. 59 (2) (2021) 16–21.

[48] J. Daily, A. Vishnu, C. Siegel, T. Warfel, V. Amatya, GossipGraD: Scalable deep learning using gossip communication based asynchronous gradient descent, 2018, CoRR, arXiv:1803.05880.

[49] S. Savazzi, M. Nicoli, V. Rampa, Federated learning with cooperating devices: A consensus approach for massive IoT networks, IEEE Internet Things J. 7 (5) (2020) 4641–4654.

[50] C. Pappas, D. Chatzopoulos, S. Lalis, M. Vavalis, IPLS: A framework for decentralized federated learning, 2021, arXiv preprint arXiv:2101.01901.

[51] T. Wittkopp, A. Acker, Decentralized federated learning preserves model and data privacy, 2021, arXiv preprint arXiv:2102.00880.

[52] D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Blockchain for 5G and beyond networks: A state of the art survey, J. Netw. Comput. Appl. (2020) 102693.

[53] N. Foster, N. McKeown, J. Rexford, G. Parulkar, L. Peterson, O. Sunay, Using deep programmability to put network owners in control, ACM SIGCOMM Comput. Commun. Rev. 50 (4) (2020) 82–88.

[54] F. Paolucci, F. Cugini, P. Castoldi, T. Osiński, Enhancing 5G SDN/NFV edge with P4 data plane programmability, IEEE Netw. 35 (3) (2021) 154–160.

[55] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, A. Wang, Network telemetry framework, in: OPSAWG, Internet-Draft, 2020.

[56] The P4.org Applications Working Group, In-band network telemetry (INT) dataplane specification, 2020, URL https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf.

[57] Y. Yuan, W. Banzhaf, ARJA: Automated repair of java programs via multi-objective genetic programming, IEEE Trans. Softw. Eng. 46 (10) (2020) 1040–1067.

[58] M. Riftadi, J. Oostenbrink, F. Kuipers, GP4p4: Enabling self-programming networks, 2019, arXiv:1910.00967.

[59] Y. Liu, J. Peng, J. Kang, A.M. Iliyasu, D. Niyato, A.A.A. El-Latif, A secure federated learning framework for 5G networks, IEEE Wirel. Commun. 27 (4) (2020) 24–31.

[60] C. Benzaid, M. Boukhalfa, T. Taleb, Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment, in: Proc. of the IEEE Wireless Communications and Networking Conference (WCNC), 2020.

[61] Behavioral model (bmv2), https://github.com/p4lang/behavioral-model.