Cognition: A Tool for Reinforcing Security in Software Defined Networks

Emilia Tantar, Maria Rita Palattella, Tigran Avanesov,
Miroslaw Kantor, and Thomas Engel

SnT, Université du Luxembourg,
4, rue Alphonse Weicker, L-2721 Luxembourg, Luxembourg
{emilia.tantar,maria-rita.palattella,tigran.avanesov,
miroslaw.kantor,thomas.engel}@uni.lu

Abstract. Security is one of the most important requirements for networks and serious concerns for network providers and users. Software Defined Networking offers to network managers new opportunities for deploying efficient security mechanisms. By means of applications and controller functionalities, SDN is able to provide a highly reactive network security monitoring, to perform comprehensive traffic analysis, and to enforce fine-grained dynamic access policies. In the present work, we show how such security mechanisms can be further enforced by applying cognitive functions at the SDN application plane. The proposed approach that finds its foundation into the control loops applied in Autonomic Managers Networks (AMNs), can efficiently enable secure and safe SDN scenarios.

Keywords: SDN, security, cognitive mechanisms.

1 Introduction

In the last years we have witnessed the progressive emergence of a new network paradigm, namely Software Defined Networking (SDN), that by introducing a centralized and programmable way of designing networks, has revolutionized the classical distributed networking approach. In detail, SDN separates the data plane (i.e., traffic forwarding between network devices, such as switches, routers, hosts, etc.) from the control plane (i.e., the decision making about the routing of traffic flows) and allows programmability of the network by external applications [44]. By means of a Southbound (SBI) and a Northbound (NBI) Interface, the control plane (i.e., the SDN controller) can interact, respectively, with the data plane and the application plane. Since the SDN's birth, the SBI has been standardized and commonly identified in the OpenFlow protocol [38], while the NBI is still under development.

SDN seems to be the promising network paradigm of choice for the future; and thus, the Open Networking Foundation (ONF), which has the leadership in SDN standardization, has recently put some effort for designing how to migrate from classical networks to SDN [43].

Born with the intention of overcoming the drawbacks of traditional networks (e.g., manual configuration and further maintenance of every single device in the network, latency in path-recovery due to the distributed approach, etc.), SDN has soon gained the interest of companies, like Google, that have prefigured the advantages of applying it in their business networks [27].

SDN has quickly attracted the attention also of engineers and network designers in the standardization community (i.e., IETF, ITU-T), as shown by the newly ad hoc designed protocols for SDN, as the Forwarding and Control Element Separation Framework (ForCES) [65], and by the efforts conducted in the direction of extending and making classical protocols suitable for an SDN context. Ih this direction, it earths mentioning the Application-Layer Traffic Optimization Protocol (ALTO) [64], the Path Computation Element (PCE) Communication Protocol (PCEP) [62], the Bidirectional Forwarding Detection Protocol (BFD) [30] or the Extensible Messaging and Presence Protocol (XMPP) [53], to mention just a few.

Moreover, a new IEEE ComSoc Emerging Technology sub-committee focused on SDN and Network Functions Virtualization (NFV) has been recently created (December 2013), aiming to explore the aforementioned next generation networking technologies and their potential interaction with IPv6, cloud and mobility, which are fundamental IT design points [26].

Due to the new paradigm on which it is built, SDN has introduced several new challenges, in terms of performance, scalability, security and interoperability [55]. But, given (i) the importance for both end users and network providers to have a secure system, and (ii) the key-role played by security aspects in the maintenance and management of software defined networking, **security in SDN** has recently became one of the hottest arguments of discussion and investigation in the research community [31, 33, 48, 54, 57]. As pointed out in [41], the main aspects of interest from a security perspective are related to *authentication, identity management, monitoring, detection*, and *mitigation of threats*. To this aim, SDN controllers are enabled with a set of functions/modules that allow to monitor the current status of the whole network (e.g., network topology, switches' behavior, flows' path, etc.) and thus, to detect possible anomalies and thus, attacks from non-authorized malicious users.

In the present work, we have first outlined the main functions nowadays implemented in open-source SDN controllers, like OpenDayLight [45], FloodLight [18], POX [49], which in our humble opinion, can be efficiently exploited in monitoring, detection and (re)acting schemes for providing security in an SDN context. Our argument is supported by some related works (overviewed in the paper), that have already used some of these functionalities to develop SDN security solutions.

In the last decade, the increasing complexity of network infrastructures has raised up the conceptual need of building networks with autonomic management, see [36]. By following the same approach proposed for enabling Autonomic Network Management (ANM), i.e., by using intelligent mechanisms, such as bio-inspired techniques, or more generally particle methods in the application plane of an SDN architecture, we do believe it is possible to enhance the security that is achievable in a Software Defined Network, using only the controller functionalities. For instance, *classification techniques can play a role in building user profiles*, while other mechanisms (e.g., control loops, **cognitive algorithms**) *can be used in learning the various behavioral topologies and anticipating possible failures*, through rare events simulation.

The current work encloses an overview on how cognition enabled countermeasures can be applied for security failures in a centralized autonomous management perspective in a SDN scenario. Based on the advances available in literature, and with no pretension of being exhaustive, in the present work we have investigated the various facets and the intersection existing between autonomous networking, cognitive networks, and SDN, with a particular focus on security challenges. As compared with the autonomous networking context, SDN policies are decoupled from the physical layer and can be applied based on application-layer attributes.

In our work we have focused on detailing the main characteristics a cognitive module can have in the context of overcoming security failures in SDN, assuming it is implemented in the application plane, interacting with the controller through the northbound interface. While addressing the dynamic specificity of the autonomous SDN management, two major components emerged: the control loops (for which the literature acknowledges a panoply of answers) and the cognitive mechanism, that can be depicted through various particle like methods. A preliminary study has been conducted in understanding the ways of integrating cognition in an SDN context from the self-optimization perspective of the autonomous SDN management.

In order to illustrate the feasibility of cognition integration, a dynamic multi-objective context has been considered and a real use case has been identified as a possible scenario where the suggested approach can be applied in the dynamic context. Finally, the possible factors triggering dynamic changes in an SDN context have been identified and ways of including them in a multi-objective formulation also proposed.

The rest of the paper is organized as follows. Section 2 first provides an overview of the main security challenges raised up in SDN. Then, the non-cognitive functions/methods that may help in monitoring and detecting possible anomalies and attacks, are presented. Also, some security solutions built on these functions, already available in literature, are presented. Section 3 introduces the concept of cognition, and the use of cognitive mechanisms in conjunction with control loops in addressing the security concerns raised in the autonomous SDN management for a centralized SDN context. Section 4 details the use of multi-objective dynamic formulations for improving SDN security. To show the potentiality of the suggested approach, we illustrate how it can be applied in a real use case. Finally, Section 5 concludes the paper and presents possible paths for future research.

2 SDN Non-cognitive Functions for Enabling Security

Software Defined Networking with its new paradigm has introduced new edges and facets in the network, that may open a way to spiteful attackers [33]. In detail, the weaker points of the SDN architecture identified as being easily exploitable by attackers were: (i) the software applications-based control of the network, and (ii) the fully centralized control (i.e., single controller) of the network that represents *a single point of failure*.

Extensive analysis studies of

The newly introduced attack vectors where subject to extensive analysis studies, see [33]. Among the newly introduced attack vectors, it is worth to mention: Denial of

Service (DoS) attack on the controller from the SBI, attacks on SDN-enabled switches (in fact, the new protocols they support may imply new vulnerabilities), attacks using control channel (i.e., communication between controller and switches), attacks on the controller itself as well as attacks on the applications (with deployment of malicious applications) running on the controller.

Beyond the aforementioned security challenges that it has introduced, it has to be emphasized that SDN has also brought a plus to the whole system, i.e., the ability (of the SDN controller) to react in *real-time* to anomalies [41, 54] thanks to the programmability of the network. Several factors can trigger anomalies. In detail, based on the type of dynamic changes that take place in the network, and the location of the security breaches, anomalies can occur at various levels of the network. We can distinguish changes in the network configuration, in the traffic's behavior, and finally in the users' profile. It has to be noticed that behind every anomaly/change taking place in the network, an attack could be hidden.

SDN controllers provide functionalities which enable to cope with such, very often undesirable, anomalies. A set of useful features, able to enhance the overall security of the system [57], are already available in open-source controllers, like Floodlight [18], OpenDayLight [45], POX [49]. Hereafter we refer to them as *non-cognitive functions* because they provide basic functionalities for supporting security, without the use of cognition.

In the following sections we first introduce the non-cognitive functionalities, which from our point of view, are potential important building blocks for SDN security applications. Then, we survey some security solutions that have been already proposed in literature, and built on the aforementioned type of functionalities.

2.1 SDN Controller Functionalities

In this section we list a set of OpenFlow-based SDN features that can be exploited by the controller applications for dealing with failures and anomalies, and thus, for providing network security. It is worth to notice that these features are ready to use in most of the open-source SDN controllers.

Network Topology. One of the most outstanding assets of SDN is the global view of the network. The information about all network elements, the links established among them, and the changes occurring in the network, are constantly monitored. By doing so, the controller tries to keep an up-to-date imagine of the network graph.

For the controllers based on Beacon technology [15], like OpenDayLigth, Floodlight and Beacon itself, this function is performed by the Topology Manager bundle. Other components such as ARP handler, Host Tracker, Device Manager, and Switch Manager, provide information to complete the topology database for the Topology Manager. Also the POX controller includes a topology module dealing with network topology.

Switch Statistics. According to the OpenFlow standard [38], the switches can maintain counters for several entities, such as: flow table, flow entry, port, queue, group, etc. For instance, they may count the number of received and transmitted packets per port, number of packets lookups in a flow table, number of active entries of a flow table, etc. The counters provide useful statistics that can be used for profiling network elements. The profiles can further be used in anomaly detection techniques. In OpenDaylight and Floodlight, the statistics are available through a *StatisticManager* class.

Flow Table Manipulation. The OpenFlow standard [38] allows to manipulate packets on per-flow basis [47]. Thus, it is possible to add/remove VLAN tag of the packet, set the transport port, set source and destination data layer and network addresses, set the next hop, drop the packet, read flow information, etc. In the OpenDayLight API, an Action class allows to manipulate flows, and take actions on them, through methods of a Flow class.

The per-flow manipulation is an important SDN feature that can be exploited in several way, for taking security countermeasures. For instance, it allows to:

– *Monitor flows*
  A controller application may store and analyze the aforementioned flow fields. For example, it is feasible to keep track of all ports on which given host tries to access another, thus, giving a possibility to detect port scanning which is usually used in cyberattacks [59]. Another example is to aggregate the incoming flows to a given host: abnormally high number of connections from different sources may identify a DDoS attack. Alternatively, the flow monitoring may be used to build host traffic profile for further anomaly detection.

– *Mirror traffic*
  To the end of Deep Packets Inspection (DPI), the controller may dynamically enable mirroring of a suspected flow at any switch and forward it to a specialized host.

– *Operate firewall at a switch*
  Basic firewall functionalities, such as Network Address Translation (NAT) [58], denying/allowing traffic by destination port or by source address can be executed directly in the switch, by installing the corresponding rules into the device. If the network manager or an SDN application needs to block traffic coming from a particular source, it is interesting to notice the possibility of blocking an unwanted traffic (e.g., from a malicious host) at the source (by installing the blocking rule directly at the switch to which the malicious host is connected) in order to avoid unnecessary additional network load; this is thanks to the global view of the network.

Direct Packets Injection. To verify whether the behavior of some switches is different from the one expected, it might be useful to inject a packet to the network and track its path/content. OpenFlow already supports such functionality that allows the controller to send a packet out through the datapath. For instance, in OpenDayligth there is a transmitDataPacket function within the DataPacketService class.

2.2 SDN Security Solutions Built on the Controller Functionalities

In literature some security solutions based on the security-oriented functionalities, presented in the previous section, have been already proposed. Hereafter, we overview them, underlying which SDN features they employed the most.

One of the classical solution for enabling a secure network consists in the implementation of enhanced monitoring mechanisms, which besides providing measures on the status of the traffic flows, are able to detect anomalous network behavior. For instance, the Distributed DoS scheme (DDoS) proposed in [8], employs, on top of the switches monitoring system, Self Organizing Maps by using *flow statistics* and *flow manipulation* functions, to detect potential anomalous flows. Another DDoS detection system, namely Defense4All [51] has been built into the OpenDaylight controller. By using a combination of *flow statistics, flow manipulation*, and *network topology* functionalities, it generates a standard traffic profile that compared with the current one allows to detect suspicious traffic. The latter is sent to a middle-box for more detailed analysis.

In turn a cost optimized monitoring mechanism based on genetic algorithms has been proposed in [6]. This method, built on top of *network topology* and *flow manipulation* functions, exploits the fact that DPI engines can be virtualized and dynamically deployed as pieces of software on commodity hardware. Jin et al. [28], by taking advantage of *flow table manipulation* and especially *flow monitoring* features, have provided a mobile malware detection system. The solution that they have proposed is able both to identify suspicious network activities through real-time traffic analysis, as well as to impose new security rules in real time, when needed.

To the aim of securing a software defined network, solutions involving more advanced hardware components, as programmable switches have been also proposed, as the Resonance system [42], which applies dynamic access control. In Resonance, the programmable switches are used to enforce dynamic access control policies based on both flow-level information and real-time alerts. In other words, *flow manipulation*, *flow monitoring* and *firewall* features are being used in order to reinforce SDN security.

SDN controllers are enabled also with a set of functions that allows the redirection of affected flows (e.g., flows manipulated by attackers) and the application of changes in the rules of faulty network elements. Solutions making use of these functions can be based on *network topology, flow manipulation* and *flow mirroring*, as for example in [56]. The suggested SDN-based framework provides monitoring services for large and dynamic cloud networks. Moreover, it automatically changes the routing paths of network flows, by transmitting them through network nodes where security devices reside. This traffic redirection approach has been used also in OpenSAFE system [4] which enables the arbitrarily direction of the traffic to security monitoring applications at line rates, by using *flow manipulation* and *mirroring* functions.

In the FlowTags architecture [17], in order to keep flow traceability, security devices add to outgoing packets Tags that provide additional contextual information with a traffic flow (e.g., source hosts or internal cache/miss state). These Tags are then used by SDN switches and (other) security devices for systematic policy enforcement. A similar approach has been also applied in SIMPLE [50], which introduces policy enforcement layer based on SDN *network topology* and *flow manipulation* features to efficiently steer network traffic. The advantage of SIMPLE solution in comparison to the

FlowTags one is that no modification to legacy security devices and existing SDN interfaces is required.

Finally, active security approaches have also been proposed, as in [24]. The framework introduces programmatic control within a novel feedback loop to control the detection of attacks, data collection for attack attribution, configuration revision, and reaction to attacks. All these operation are performed by exploiting *flow manipulation* features. Hand et al. built an initial prototype that extends the FloodLight SDN controller [18] to automatically interface with the Snort [52] intrusion detection system in order to detect anomalies.

3 ANM and Control Loops: The Ancestors of Cognitive Algorithms for SDN

Due to the increased complexity of networks, the huge amount of devices and users to handle, and the resulting challenge to efficiently control/manage and operate such kind of networks, the need of *Autonomic Network Management* (ANM) has raised up more and more in the last decade [1,34]. Based on the concept of *autonomic computing*, introduced for the first time by IMB in 2003 [25], ANM allows to set-up self-managing networks that are able to detect, (re)act and reconfigure them self, and thus, recover from undesired network status (e.g., failures). The autonomous approach has paved the way to the Software Defined Networking paradigm, whose final aim is to simplify the network management, by introducing network programmability. In fact, the ANM Autonomic Manager can be seen as an SDN controller, or can be defined through the applications interacting with the controller(s) at the level of the Northbound interface.

The self-management implies adaptation to dynamical changes, that trigger new challenges, not covered by existing prevention rules, from where the need of enabling the system to learn and react based on reasoning. Cognition can be considered as the nesting view, as it encloses amongst the various facets the learning and reasoning aspects and thus answers this challenge, as outlined for ANM cases [13].

Many different models have been proposed in literature for enabling cognition in a general network management context, but *can they be applied to software defined networks, given the analogy between SDN and ANM*, mentioned before*?* This is one of the issue we have addressed in the present work. Although not named as an SDN, but rather as a *knowledge plane for the Internet*, the ideas behind enabling cognition inside an SDN context, were already conceptually outlined [13]. The proposed cognitive skeleton relies on a distributed and decentralized perspective, different from the centralized approach adopted in the SDN architecture[1].

Following the definition provided by Clark [13] and later Mitola [40], a cognition enabled SDN represents a *cognitive network* that implies the existence of a cognitive process, aware of the overall current network status, and thus, able to react based on the existing conditions. The place of cognition in a generic centralized SDN environment can be foreseen at various planes, the most natural one being the application one,

[1] It has to be noticed that the SDN architecture can be also decentralized by using several controllers within the control plane, but this is out of the scope of our work.

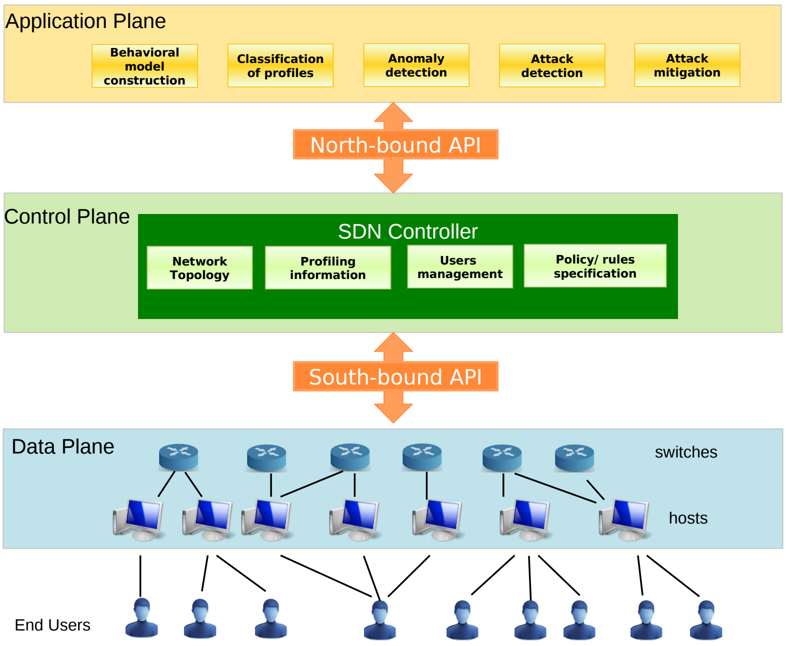

Fig. 1. A cognitive SDN architecture for security enforcement

as shown in Figure 1. The latter depicts the use of cognitive algorithms (e.g., machine learning techniques, or other probabilistic model building and profiling technique) in SDN security applications. It is worth to notice the clear separation made between the **control plane**, playing the role of knowledge keeper - gathering the information reflecting the status of the network, through the use of non-cognitive functionalities - and the **application plane**, intended to provide cognition based capabilities. As example of usage, when considering the Machine Learning (ML) as cognitive paradigm to be employed, the behavioral model construction translates in a machine learning model training.

3.1 Cognition as an Answer for Security Concerns in SDN

Following the time-line view [19] and the cognitive reasoning one (ignoring the time-line aspect) [32], we have built a self-optimization-based overview variant, depicted in Figure 2. Facing a security perspective, the cognitive reasoning, according to the level of granularity that is applied and the amount of information used, spans through three levels: *strategic*, *tactic* and *reactive*. In our case the classical time-line view approach [19] has been replaced by the reactive one, as this can be readily applied in the security context, through the controller non-cognitive functionalities (see Section 2) or for example, through the network components internal rules. It is worth noticing that by taking a traffic engineering perspective, the environment faced is intrinsically dynamic

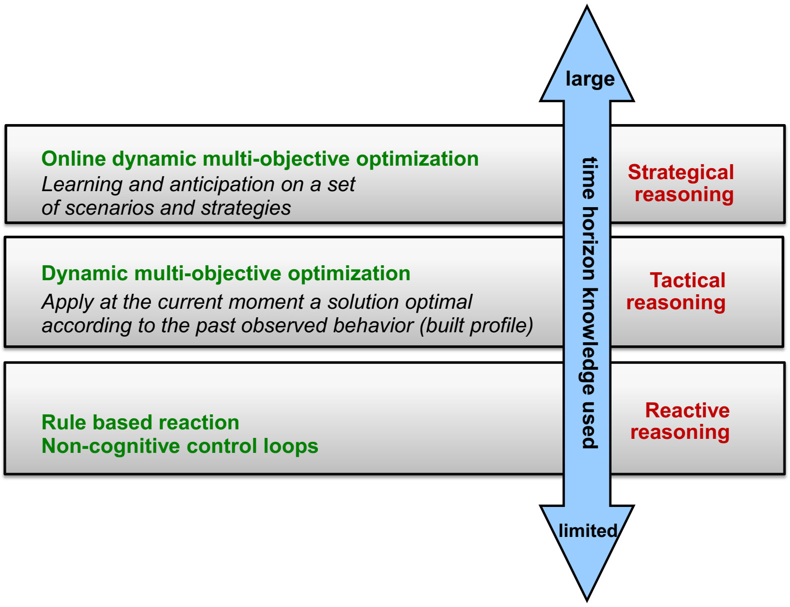
large
Online dynamic multi-objective optimization
Learning and anticipation on a set of scenarios and strategies
Strategical reasoning
Dynamic multi-objective optimization
Apply at the current moment a solution optimal according to the past observed behavior (built profile)
Tactical reasoning
Rule based reaction
Non-cognitive control loops
Reactive reasoning
time horizon knowledge used
limited

Fig. 2. Cognition approaches applicable for the SDN context from a security perspective, according to the information on the time horizon used

and as such the presented path. The left-hand side of Figure 2 outlines the possible models that apply in this context and the needed sub-components in order to provide the appropriate level of reasoning, depicted on the right-hand side.

Each of the covered layers (strategic, tactic and reactive) brings its own specificity and the level of complexity of the applied techniques increases accordingly:

Reactive reasoning: solely based on feedback control loops, implies the use of the non-cognitive security functionalities.

Profiling (classification based): useful in filtering data and data dimensionality reduction; constructs a probabilistic model from the observed behavior history of the monitored components (applies for the strategic and tactic reasoning).

Anticipation: applies only for the *strategic reasoning*. Implies the use of heuristic approaches or more advanced learning-based approaches applied in a dynamic context. Involves the use of several test scenarios and strategies.

An illustrative example on the use of cognitive reasoning for the prevention and mitigation of anomalies in an autonomous SDN management setting is provided in Section 4.

3.2 Main Components of Cognitive Approach

As in the autonomous network management, besides the representation of the problem's knowledge, the two main components that define cognition in an SDN context are: (1) **a cognition mechanism** (defined, e.g., through a particle-like method), and (2) **a control loop component**.

As a non-cognitive alternative, control loops can be developed as stand-alone solutions (i.e., without the use of any cognition), although with a limited range of applicability and covering deterministic (already observed) security threats. The confines of security issues they address - in the intent of providing the base functionalities enabling self-management - lay from self-configuration, to self-healing, self-protection or self-optimization. On the other hand, coming from control theory, control loops can provide also the proper environment for employing cognition in the SDN context.

Control loops are implemented by Autonomic Managers (AMs) that can operate at two different layers of the ANM architecture. In detail, AMs can work as *touchpoints* that directly manage the network components through monitoring and reacting/forwarding interfaces; or as *orchestration* mechanisms that coordinate the behavior of multiple AMs. As detailed in Table 1, several control loop mechanisms have been developed during the years, having different levels of applicability, and it is possible to distinguish between: *local control loops* that are applied in a specific part of the network or on a specific group of users; and *global control loops*, applied to the whole network and all its components.

Amongst the specific functionalities offered by the aforementioned environments to the security concerns, it is worth to notice, for example, that the 4D architecture encloses detection of forwarding loops and avoiding routing anomalies (persistent forwarding loops and permanent route oscillations). On the other side, MAPE [25] allows attack mitigation, through a self-protection functionality, enabling reaction to unauthorized access and use, virus infection and proliferation or denial-of-service attacks. In MAPE the control loops are classified according to their role in four categories: self-configuring, self-healing, self-optimizing and self-protecting.

FOCALE brings as a plus an orchestration effort, in which the cognition perspective has been integrated also through a fusion model that makes use of information models, ontologies, machine learning and reasoning.

Table 1 is not intended to exhaustively cover the domain, as various other projects exist, although with no specific focus on security aspects or intended for the autonomous management case. For instance, Haggle [22] and GENI [20] developed intrinsic control loops, intended mainly for the reliability and maintenance of the system, which can also be useful to some extent in the security context.

3.3 Cognitive Mechanisms

Biological systems are by nature self-managed, and allow the organism to dynamically adapt to environmental changes. Therefore, bio-inspired autonomic networks have been developed, by applying some biological principle to network management [36]. In the artificial intelligence area several techniques coming from Bayesian probability, evolutionary computation or machine learning areas can be employed.

Table 1. Control loops (CL) addressing security concerns and autonomous management challenges

Name, reference	Area of knowledge covered		Type of failures covered		Security aspect covered	CL specificity	Cognition
	local	global	single failure	multiple failures			
OODA[a], Boyd (1987) [7]	✓		✓		Attack prevention	*De facto* standard	-
4D architecture, (2005) [21]	✓		✓		Anomaly detection	Protocol level	Heuristics
MAPE[b], IBM (2006) [25]		✓	✓		Attack mitigation	Uses a shared knowledge base	Classification: association rule mining
FOCALE[c], Strassner *et al.* (2006) [60]	✓	✓	✓		Attack, intrusion prevention (based on OODA)	Policy driven, Network level, cognitive	Finite-state-machine
ConMAN, (2007) [3]	✓		✓		Intrinsic functionalities	Protocol abstraction enabled	Classification
GANA, (2013) [16]		✓	✓	✓	Data poisoning prevention, anomaly detection and prevention	Hierarchical	Three levels of cognition apply: strategic, tactical and classical
CogMan, Sungsu *et al.*(2013) [32]		✓	✓	✓	Failure recovery	Priority based CL	Fast Flow Setup algorithm

[a] Observe-Orient-Decide-Act (OODA).
[b] Monitor-Analyze-Plan-Execute (MAPE).
[c] Foundation-Observe-Compare-Act-Learn-rEason (FOCALE).

Machine learning as applied in the more general context of security in networking, can take several facets from *diagnosis* **(detection)**, to *maintenance* **(reaction)** or *prognosis* **(anticipation)**. As for the the detection task this can be focused on anomalies [37], attacks [11] or even intrusion detection [12], while considering cases where no information or scarce information is available. In the management of an autonomous network unsupervised learning represents the solution to be applied in a dynamic context, while supervised learning can be seen as an *a priori* task for detecting patterns that are to be implemented in the control loop.

Various highly modulable platforms are available in the community in order to support the development of such cognitive techniques. Only in the evolutionary area, one can notice various flavors based on a variety of languages as C/C++ for ParadisEO [9] or JAVA for ECJ [63], with an additionaly designed Multi-agent simulator of Networks (MASON) [35]. These are useful assets in coping with the optimization side of the problem. Also the machine learning side is well represented through data mining platforms, such as Weka [23].

4 Self-optimization in a Dynamically Changing SDN Environment

Static optimization theory has already been considered as a potential solution in inhansing the performances and management of classical networks. Nevertheless one of the main drawbacks, as compared with regret minimization, was related to the unhandled online aspect. Through the current proposal we aim at providing a possible modeling that overcomes this aspect, by simultaneously handling multiple objectives in an online dynamic manner, inside an optimization context. The multi-objective area enables inclouding a set of conflicting or different by nature objective functions to be handlded simultaneously (e.g. one following the global network-wide perspective and one maximizing the individual gain), thus answering the local to global pefromance perspective raised in [2].

As outlined in the previous section, self-optimization represents one of the main directions enabling improvements in the security of a self-managing SDN. In this section we present a preliminary study on how the so-called **strategic reasoning** (as depicted in Figure 2), and in detail, dynamic multi-objective optimization can be applied in an SDN context. Finally, the feasibility of using multi-objective formulation is also investigated in a specific SDN use case, taken under analysis.

4.1 Multi-objective Dynamic Formulation

The first step to design dynamic self-optimization consists in identifying the factors that trigger complexity. In the SDN context, there are several layers where dynamic changes can occur, triggered by

(a) External factors
- Environmental changes affecting the network functioning, such as intrusion or attacks.

(b) Internal factors
- **Network configuration changes**, leading to *anomalous behavior*, involving:

  - the physical network configuration;
  - the switches rules;
  - the applications connected to the controller;
  - the controller behavior (e.g., priority of applications).
- **Traffic changes**, leading for instance to *congestion*.
- **Users changes**: addition/removal of users, mobility of users, remodeling of user groups.

In order to model the dynamic behavior based on the outlined dynamic factors we introduce a dynamic multi-objective problem formulation. Let t be a monotonically increasing value on a time period $[t_0, t_{end}] \in \mathbb{R}^+$ and let σ be an environment derived set of parameters, uncontrollable and external to the system (i.e. the students' anual courses timetable influencing their network usage - the volume and type of requests submited according to the various periods: exams, holidays, etc.). The behavior of a dynamic multi-objective optimization problem (system) for the SDN context can be depicted by $H(F_\sigma, D, x, t)$, where

F_σ	multi-objective support function;
D	time-dependent functionals modeling the dynamic behavior of a specific component of the system;
x	the set of variables / parameters.

The objective functions, defined as

$$F_\sigma : X \to Y, F_\sigma(x) = [f_{\sigma,1}(x), \ \dots \ , f_{\sigma,k}(x)] \tag{1}$$

evolve in time according to

$$\int_{t_0}^{t_{end}} F dt = \left(\int_{t_0}^{t_{end}} f_1 dt, \ \dots \ , \int_{t_0}^{t_{end}} f_k dt \right) \tag{2}$$

The most complex setting to be handled in this context comes from the presence of external factors (e.g., attackers) triggering complexity. In this case we are facing an online dynamic multi-objective formulation - a 4^{th} order formulation, following the classification proposed in [61] - that can be depicted as

$$H(F_\sigma, D, x, t) = F_{D(\sigma,t)}(x, t), \tag{3}$$

leading to a formulation of the type

$$\arg\min_{\mathbf{x}(t)} \int_{t_0}^{t_{end}} F_\sigma(\mathbf{x}(t), t) \text{ dt} \overset{def}{=} \min_{\mathbf{x}(t)} \left\{ \left(\int_{t_0}^{t_{end}} f_{\sigma,i}(\mathbf{x}(t), t) \text{ dt} \right)_{1 \le i \le k} \right\}$$

In this case the external parameters are modeled through the $D(\sigma, t)$ and can be represented through context dependent probability distributions, for which the parameters need to be estimated. In order to provide competing results a set of scenarios and strategies needs to be defined. The scenarios can be based on existing attack patterns, e.g., Address Resolution Protocol (ARP) poisoning.

When tackling scenarios from the traffic engineering perspective, the network can be modeled as a graph, with x enclosing the representation of the network topology (e.g., through adjacency matrix, etc.). Next, scenarios can be built by drawing samples from, for example, a power law distribution with random graphs modeling different instance types, as the case in [39]. In this mentioned work, besides the random graph (topology) sampling, each edge is modeled by a Bernoulli random variable. In order to deal with sparse graphs, the authors rely on a log-linear model for edge probabilities which, w.r.t. specific context aspects, can be seen as a generalized linear model.

Regarding the objective functions to be used, they can be related to the quality of the flow clustering, required in determining traffic patterns, useful for both internally and externally driven dynamic problems. One possible scenario making use of the flow classification for reinforcing the security in SDN can cover the anomalous switches behavior, as described in the following section.

4.2 Secure SDN Use Case: Mitigation of Anomalies Due to Misbehaving Switches

In a centralized single-controller SDN architecture, the controller has the role of mitigating anomalies by means of specific applications, connected with the controller through the Northbound interface (Figure 1). We assume one of the applications consists in a dynamic multi-objective optimization technique, using the problem definition for anomaly mitigation as depicted herein.

The investigated security concern is induced by anomalous behavior of switches. Although, there exist frameworks such as, e.g., FRESCO [57] that help in preventing conflicting rules to be applied at a switch level, in an SDN network, switches can be under the influence of malicious users/attackers for different reasons. First of all, the use of the Transport Layer Security (TLS) protocol [14], for protecting the channel between the controller and the switches, is optional according to the OpenFlow specification [46]. Beside that, some attack patterns are known *a priori*, such as ARP poisoning, DHCP snooping, broadcast/multi-cast rate limiting, MAC address limits. This allows to identify which network properties should be monitored (e.g., the number of flows per switch).

Switches in an SDN network can deviate from the controller tasks in a number of ways including: redirecting or delaying traffic, dropping or modifying packets, falsifying statistics or requests sent to the controller. According to [5], the only way to address this problem is to dump and inspect the flow tables on a per-switch basis which is a very resource consuming procedure. An alternative consists in learning from the past behavior of the switches in terms of traffic flow transiting the switch. This can be done through *profiling techniques* and *unsupervised learning techniques*. Once the anomalous switch has been identified, the controller needs to mitigate the anomaly, by applying countermeasures at the traffic level.

The multi-objective formulation allows among others to apply solutions that address several security challenges simultaneously, that are either conflicting or different by nature and leaves place also for the possible integration of other criteria, such as performance related objectives.

In order to mitigate security concerns various probabilistic techniques have been successfully applied on simplified model formulation, or addressing clustering issues, see for instance, the Ant Colony Optimization approach proposed in [10] or the Symbolic Dynamic Filtering applied in [29]. These techniques can be seen as potential candidates to be exploited in our future work for the experimental evaluation of the proposed use case.

5 Conclusions

In this paper we have described non-cognitive SDN features which are potential important building blocks for SDN security applications. We have also showed how such non-cognitive functions provide support against different security attacks. As a follow up, we detailed on the intersection between control loops, autnomous network management and cognitive networking. Finally, we have suggested the use of cognition for enhancing the security level that can be achieved with the non-cognitive function, through dynamic multi-objective optimization. In our future work we aim at applying the outlined cognitive module to real use cases. Moreover, we will investigate how the multi-objective formulations can be used for providing not only security, but also network reliability and better QoS performance.

Acknowledgment. This publication is based in parts on work performed in the framework of the CoSDN project, INTER/POLLUX/12/4434480, funded by the Fonds National de la Recherche, Luxembourg. The authors would like to thank Gabriela Gheorghe for useful discussions on SDN usecase.

References

[1] Agoulmine, N., Balasubramaniam, S., Botvitch, D., Strassner, J., Lehtihet, E., Donnelly, W.: Challenges for autonomic network management. In: Proc. of 1st IEEE International Workshop on Modeling Autonomic Communications Environments, MACE 2006 (2006)

[2] Avramopoulos, I.C., Rexford, J., Schapire, R.E.: From optimization to regret minimization and back again. In: Fox, A., Basu, S. (eds.) SysML. USENIX Association (2008)

[3] Ballani, H., Francis, P.: CONMan: A step towards network manageability. SIGCOMM Comput. Commun. Rev. 37(4), 205–216 (2007)

[4] Ballard, J.R., Rae, I., Akella, A.: Extensible and scalable network monitoring using OpenSAFE. In: Proc. of the 2010 Internet Network Management Conf. on Research on Enterprise Networking, INM/WREN 2010, p. 8. USENIX Association, Berkeley (2010)

[5] Benton, K., Camp, L.J., Small, C.: Openflow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 151–152. ACM, New York (2013)

[6] Bouet, M., Leguay, J., Conan, V.: Cost-based placement of virtualized Deep Packet Inspection functions in SDN. In: Military Comm. Conf., IEEE MILCOM 2013, pp. 992–997 (November 2013)

[7] Boyd, J.: Destruction and Creation. Operational level of war, U.S. Army Command and General Staff College. Center for Army Tactics (1987)

[8] Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: 2010 IEEE 35th Conference on Local Computer Networks (LCN), pp. 408–415 (October 2010)

[9] Cahon, S., Melab, N., Talbi, E.G.: Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. Journal of Heuristics 10(3), 357–380 (2004)

[10] Carvalho, L., Rodrigues, J., Barbon, S., Lemes Proenca, M.: Using ant colony optimization metaheuristic and dynamic time warping for anomaly detection. In: 2013 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–5 (September 2013)

[11] Casas, P., Mazel, J., Owezarski, P.: Steps towards autonomous network security: Unsupervised detection of network attacks. In: NTMS, pp. 1–5. IEEE (2011)

[12] Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. Comp. Comm. 35(7), 772–783 (2012)

[13] Clark, D.D., Partridge, C., Ramming, J.C., Wroclawski, J.: A knowledge plane for the Internet. In: Feldmann, A., Zitterbart, M., Crowcroft, J., Wetherall, D. (eds.) SIGCOMM, pp. 3–10. ACM (2003)

[14] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (August 2008)

[15] Erickson, D.: The beacon OpenFlow controller. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 13–18. ACM, New York (2013)

[16] ETSI: Generic autonomic network architecture (an architectural reference model for autonomic networking, cognitive networking and self-management). ETSI GS AFI 002 V1.1.1 (April 2013)

[17] Fayazbakhsh, S.K., Sekar, V., Yu, M., Mogul, J.C.: FlowTags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 19–24. ACM, New York (2013)

[18] Floodlight project, `http://www.projectfloodlight.org`

[19] Gat, E.: On three-layer architectures. In: Kortenkamp, D., Bonnasso, P.R., Murphy, R. (eds.) Artificial Intelligence and Mobile Robots, pp. 195–210 (1998)

[20] Geni design principles. Computer 39(9), 102–105 (2006)

[21] Greenberg, A., Hjalmtysson, G., Maltz, D.A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., Zhang, H.: A clean slate 4D approach to network control and management. SIGCOMM Comput. Commun. Rev. 35(5), 41–54 (2005)

[22] IST FP6 Haggle project, `http://www.haggleproject.org`

[23] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)

[24] Hand, R., Ton, M., Keller, E.: Active security. In: Proc. of the 12th ACM Workshop on Hot Topics in Software Defined Networking, HotNets-XII, pp. 17:1–17:7. ACM, New York (2013)

[25] IBM: An architectural blueprint for autonomic computing. ONF White paper (2006)

[26] IEEE SDN/NFV subcommittee, `http://cms.comsoc.org/eprise/main/SiteGen/TC_SDN_NFV/Content/Home.html`

[27] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A.: B4: Experience with a globally-deployed software defined WAN. SIGCOMM Comput. Commun. Rev. 43(4), 3–14 (2013)

[28] Jin, R., Wang, B.: Malware detection for mobile devices using software-defined networking. In: Proc. of the 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013, pp. 81–88. IEEE Computer Society, Washington, DC (2013)

[29] Jin, X., Guo, Y., Sarkar, S., Ray, A., Edwards, R.: Anomaly detection in nuclear power plants via symbolic dynamic filtering. IEEE Transactions on Nuclear Science 58(1), 277–288 (2011)

[30] Katz, D., Ward, D.: Bidirectional Forwarding Detection. IETF Internet-Draft, draft-ietf-bfd-base-11.txt (January 2010)
[31] Khurshid, A., Zhou, W., Caesar, M., Godfrey, P.: VeriFlow: Verifying network-wide invariants in real time. ACM SIGCOMM Comp. Commun. Rev. 42(4), 467–472 (2012)
[32] Kim, S., Kang, J.M., Seo, S.S., Hong, J.W.K.: A cognitive model-based approach for autonomic fault management in OpenFlow networks. Int. Journal of Network Management 23(6), 383–401 (2013)
[33] Kreutz, D., Ramos, F.M., Verissimo, P.: Towards secure and dependable software-defined networks. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 55–60. ACM, New York (2013)
[34] Louca, A., Mauthe, A., Hutchison, D.: Autonomic network management for Next Generation Networks. In: Proc. of 11th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, PGNet 2010, pp. 1–6 (2010)
[35] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: A multiagent simulation environment. Simulation 81(7), 517–527 (2005)
[36] Manzalini, A., Minerva, R., Moiso, C.: Bio-inspired autonomic structures: a middleware for telecommunications ecosystems. In: Vasilakos, A.V., Parashar, M., Karnouskos, S., Pedrycz, W. (eds.) Autonomic Communication, pp. 3–30. Springer US (2009)
[37] Mazel, J., Casas, P., Labit, Y., Owezarski, P.: Sub-space clustering, inter-clustering results association & anomaly correlation for unsupervised network anomaly detection. In: CNSM, pp. 1–8. IEEE (2011)
[38] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. Proc. of the ACM SIGCOMM 2008 Conference. 38(2), 69–74 (2008)
[39] Miller, B., Arcolano, N., Bliss, N.: Efficient anomaly detection in dynamic, attributed graphs: Emerging phenomena and big data. In: 2013 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 179–184 (June 2013)
[40] Mitola, J.: Cognitive Radio Architecture. In: Cognitive Networks - Towards Self-Aware Networks. Wiley London, London (2007)
[41] Nadeau, T.D., Gray, K.: SDN: Software Defined Networks, 1st edn. O'Reilly Media (September 2013)
[42] Nayak, A.K., Reimers, A., Feamster, N., Clark, R.: Resonance: Dynamic access control for enterprise networks. In: Proc. of the 1st ACM Workshop on Research on Enterprise Networking, WREN 2009, pp. 11–18. ACM, New York (2009)
[43] Open Networking Foundation: Migration use cases and methods. ONF White paper (December 2013)
[44] Open Networking Foundation: SDN architecture overview. ONF White paper (December 2013)
[45] OpenDaylight project, `http://www.opendaylight.org`
[46] OpenFlow Switch Specification (version 1.1.0), `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf` (February 2011)
[47] OpenFlow Switch Specification (version 1.4.0), `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf` (October 2013)
[48] Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G.: A security enforcement kernel for OpenFlow networks. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networking, pp. 121–126. ACM (2012)

[49] POX controller, http://www.noxrepo.org/pox/about-pox
[50] Qazi, Z.A., Tu, C.C., Chiang, L., Miao, R., Sekar, V., Yu, M.: SIMPLE-fying middlebox policy enforcement using SDN. SIGCOMM Comput. Comm. Rev. 43(4), 27–38 (2013)
[51] Radware: Defense4All, User Guide, https://wiki.opendaylight.org/view/Defense4All:User_Guide (2014)
[52] Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proc. of the 13th USENIX Conference on System Administration, LISA 1999, pp. 229–238. USENIX Association, Berkeley (1999)
[53] Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core, RFC 6120. The Internet Engineering Task Force (March 2011)
[54] Scott-Hayward, S., O'Callaghan, G., Sezer, S.: SDN security: A survey. In: Proc. of IEEE SDN Conf. for Future Networks and Services (SDN4FNS), pp. 1–7 (November 2013)
[55] Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N.: Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Comm. Magazine 51(7), 36–43 (2013)
[56] Shin, S., Gu, G.: CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: Proc. of the 2012 20th IEEE Int. Conf. on Network Protocols, ICNP 2012, pp. 1–6. IEEE Computer Society, Washington, DC (2012)
[57] Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G., Tyson, M.: FRESCO: Modular composable security services for software-defined networks. In: 20th Annual Network and Distributed System Security Symposium (NDSS 2013). The Internet Society (February 2013)
[58] Srisuresh, P., Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational) (August 1999), http://www.ietf.org/rfc/rfc2663.txt
[59] Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical automated detection of stealthy portscans. J. Comput. Secur. 10(1-2), 105–136 (2002)
[60] Strassner, J., Agoulmine, N., Lehtihet, E.: FOCALE: A Novel Autonomic Networking Architecture. Int. Trans. on Systems Science and Applications 3(1), 64–79 (2007)
[61] Tantar, E., Tantar, A.A., Bouvry, P.: On dynamic multi-objective optimization, classification and performance measures. In: IEEE Congress on Evolutionary Computation, pp. 2759–2766. IEEE (2011)
[62] Vasseur, J.P., Le Roux, J.L.: Path Computation Ement Communication Protocol, RFC 5440. The Internet Engineering Task Force (March 2009)
[63] White, D.: Software review: the ecj toolkit. Genetic Programming and Evolvable Machines 13(1), 65–67 (2012)
[64] Xie, H., Tsou, T., Lopez, D.R., Yin, H.: Use Cases for ALTO with Software Defined Networks. IETF Internet-Draft, draft-xie-alto-sdn-extension-use-cases-01 (January 2013)
[65] Yang, L., Dantu, R., Anderson, T.A., Gopal, R.: Forwarding and Control Element Separation (ForCES) Framework, RFC 3746. The Internet Engineering Task Force (April 2004)