Available online at www.sciencedirect.com

ScienceDirect

Computers & Security 00 (2023) 1–26

Procedia Computer Science

Detection of Anomalies of a Non-Deterministic Software-Defined Networking Control

DESGEORGES Loïc[a], GEORGES Jean-Philippe[a], DIVOUX Thierry[a]

[a] *Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France*

Abstract

Software Defined Networking (SDN) is a network architecture within the control is centralized through a software-based controller. Being a single point of attack makes the controller the preferred target in the SDN architecture. Multi-controller architecture has been introduced to reinforce the control plane. However it requires a communication interface between the controllers which is a security threat. In this objective, a dual controller architecture is introduced and it consists of one nominal controller in charge of the data plane computation plus a second one in charge of the detection of anomalies in the decisions taken by the main controller. In the case of non-determinist algorithm, the detection logic aims at determining a likelihood score of the decisions taken by the controller. A multi-criterion detection approach is proposed by considering both the performance of the decisions and the structure of the decisions taken by the controller. Such computations are probabilistic and attention has been paid to machine learning algorithms to determine this likelihood. More precisely, three formalisms are compared: Probabilistic Finite Automaton, Hidden Markov Model and Recurrent Neural Network. The impact of the control variance in the detection accuracy depending on the formalism used is discussed on a case study.

Keywords:
Software Defined Networking, Safety, Security, Multi-Controllers, Observability, Hidden Markov Models, Recurrent Neural Networks

1. Introduction

During the last decade, a huge activity has focused the Software-Defined Networking (SDN), presented in [1]. Fundamentally, SDN separates the control from the network infrastructure. This control is then centralized through a software-based controller, which takes all the decisions related to the network. As a consequence, the controller is a preferential target of a SDN architecture as resumed in [2] [3]. Indeed, from the controller, an attacker has access to the information of the whole network and can damage it through a Denial of Service Attack by flooding the tables of the switches or by modifying the information of the commands sent by the controller as developed in [4] or [5].

This paper focuses on the security and the safety of the network control using a multi-controller architecture. Such architecture is widespread in the literature for several reasons as presented in [6] [7]. Firstly because a failure of the controller inhibits the network service [8] and a multi-controller architecture permits to provide a redundancy of the main controller like proposed in [9]. Also, such architecture presents some benefits in terms of security and safety. For example, it permits to set up a decision-making security architecture as in [10]: each rule proposed by a controller is subject to a vote among all the others. Such process permits to limit the influence of a malicious controller. Also, more recently, Blockchain has also been considered as an option to secure the control layer and especially the communication interface between the controllers as in [11] or [12].

Regarding the related literature, all the proposed multi controller architectures are with an East-West interface. But such interface of communication is a threat in terms of security, as described by [13]. Nevertheless, some mechanisms have been proposed to secure it such as in [14] which proposed to encrypt the communication. But still, the decisions taken by a controller depend on the information given by another which might be malicious and the underlying question is: is it possible to detect an anomaly in the control without communication with the controller ? In this objective, we proposed to develop an original multi-controller architecture without communication between the controllers. There is one controller which takes decisions and one observer which observes the activity of the main controller in order to detect anomalies through its activity as presented in Fig. 1 and proposed in [15]. However, such analysis is based on the assumption that the behaviour of the controller is deterministic (such as routing using Dijkstra or Belleman-Ford algorithms). In such a case, the observer expects that the controller takes similar decisions as the ones learned previously. As a consequence, the detection method has to be adapted to detect anomalies of a non-determinist control (for instance, where the decisions are the result of an automatic learning technique). In such a case, the aim of the detection logic tends to determine a likelihood score of the decisions of the controller. To determine this score, a multi-criterion detection approach is proposed by considering both the performance of the decisions and the structure of the decisions taken by the controller. Also, such computations are probabilistic and the first intuitive approach to develop a model is based on classical automaton. However, in order to improve detection accuracy and minimize the low false alarm rate, the literature developed machine learning (ML) techniques. Moreover, nowadays the deep learning (DL) approach has also been used extensively in the tracks of anomaly detection. This paper will explore the effects of an anomaly detection algorithm depending on these algorithms. In this work, the algorithms used to determine the likelihood score are: Probabilistic Finite Automaton, Hidden Markov Model and Recurrent Neural Network. Each formalism represents a different level of abstraction in the model: first, a classical discrete approach with a stochastic automaton (Probabilistic Finite Automaton) which is then complexified for a Hidden Markov Model and finally a continuous approach is proposed through a Recurrent Neural Network. Therefore, regarding our multi-criteria approach the questions are: what are the limits of each criterion and what are the benefits and limits of these algorithms ?

(a) Classical mutli controller architecture

(b) The proposed multi controller architecture

Figure 1: Differences between a classical multi controller architecture and our proposal

The contribution of this paper is to develop an observer of the controller which detects anomalies in the control without communication with the controller. Contrary to previous work, the control considered is non-deterministic which implies that the decision of the controller is random and so the previous detection algorithm proposed cannot be used anymore. In this objective, a detection algorithm is proposed based on a multi-criteria approach and here, two criteria are proposed. The first one based on the impact of the decisions and the second on a statistics model. Regarding the second criterion, we compared the benefits of a classical automaton compared to the hidden Markov model algorithm (a machine learning approach) and to recurrent neural network (a deep learning approach).

The first part aims to present the related works, then the detection problem is introduced. An anomaly-based detection method is proposed in section 4. The used formalisms are developed in the section 5. This proposal is illustrated and discussed on a case study in section 6 and finally, a conclusion presenting the perspectives concludes the work.

2. Related Works

In a first part an overview of machine learning techniques is presented and then the interest for the security in a SDN network is described.

2.1. Overview of Machine learning Algorithms

Machine learning (MC) [16] [17] aims to learn from the data in order to achieve a task such as a classification or a regression. The aim is to make accurate predictions using data which are available and can have several forms. This can be used to classify the data into an anomaly or normality which permits to detect anomalies i the behavior and may serve Intrusion Detection System (IDS) [18] [19]. There are different types of learning strategies which depends on the dataset:

- Supervised Learning: the learner receives a set of labelled examples as training data and makes prediction for all unseen points. This learning needs external assistance.
- Unsupervised Learning: the learner receives unlabelled training data and makes predictions for all unseen points.
- Semi-supervised Learning: the learner receives a training sample consisting of both labelled and unlabelled data and makes prediction for all unseen points
- Reinforcement Learning: to collect information, the learner actively interacts with the environment and receives a reward for each action. The aim of the learner is to maximise his reward over a course of actions.

The widely used techniques to detect anomalies concerns the supervised ones as the supervised learning methods significantly outperform the unsupervised ones [20]. Regarding supervised learning. First, Decision Tree (DT) [21] [22] are those types of trees which groups attributes by sorting them based on their values. This is used mainly for classification purpose and more generally there is the Random Forest (RF) approach as used in [23]. Then, Supported Vector Machine (SVM) works on the principle of class separation [24]. Basically, the aim is to determine the separating hyperplanes between the data's classes by maximizing the margins between the class's closest points [25]. Also, Bayes' theory (BT) [26] uses conditional probability to determine the probability of an event based on the data set . The aim is to infer the state of the system from the observations. Such theory may have several applications using for example a classical stochastic automaton or Hidden Markov Model (HMM) formalism introduced in [27] [28]. HMM is a statistical Markov model in which the system being modelled as a Markov process with hidden states. However, there is an observable process which is a consequence of the hidden states. Since the Markov model evolution cannot be observed directly, the objective is to infer the hidden states from the observations using Bayes' theory. These inferences permit to detect anomalies in the cyber security applications as in [29] or [30] using Bayes' theory. This discrete approach of the formalism might be extended by considering a continuous one with a Recurrent Neural Network (RNN) [31] which is spread in the literature for intrusion detection systems as in [32] [33]. It has to be mentioned that the RNN is part of the deep learning approach also well known in the literature to detect intrusion such as in [34].

2.2. Machine learning approaches for SDN security

Machine Learning algorithms can bring intelligence to the SDN controller and address issues such as traffic classification, QoS/QoE control algorithms or even the security of the network as presented in [49] [50]. These works focus on the security of the network by developing an anomaly based Intrusion Detection System (IDS). Machine learning methods are widely used in anomaly-based IDS in order to set up a model of the normal activities of the monitored system, here the control. It mainly consists of two phases: the training part uses to learn the pattern and creates the model of the system. Then, the running phase is the decision-making part where the model predicts and acts according to the training phase.

SDN brings some unique advantages to the deployments of ML-based network security solutions [51] [52] [49]. The main advantage is the centralisation of the control which permits to have a global view of the network and eases the monitoring and collection of the data of the network [44]. Also, the programmability of SDN permits an immediate

	Features considered				
Related Works	Performance	Structure	Communications	Formalism	Traffic Monitored
[35]		√	Yes	HMM	Host traffic
[36]	√	√	Yes	SVM	Host traffic
[37]		√	Yes	HMM	Host traffic
[38]		√	Yes	Bayes - DT	Host traffic
[39]	√	√	Yes	RNN	Host traffic
[40]		√	Yes	HMM - RNN	Host traffic
[41]	√	√	Yes	RNN	Host traffic
[42]	√	√	Yes	RNN	Host traffic
[20]	√	√	Yes	SVM - NB - DT - LR	Host traffic
[43]	√	√	Yes	RNN	Host traffic
[34][23][44][45]	√	√	Yes	RF - Bayes - DT - RNN	Host traffic
[10]		√	Yes	-	Control traffic
[46]		√	Yes	Blockchain	Control traffic
[11]		√	Yes	Blockchain	Control traffic
[47]	√		Yes	RNN	Host traffic
[48]		√	Yes	SNORT	Host traffic
Our proposal	√	√	No	PFA - HMM - RNN	Control traffic

Table 1: Classification of the methods based on ML techniques for the security of a SDN architecture

reaction to attacks when they are detected. [35] proposed to use the HMM formalism to analyse the structure of the traffic through the following features: length of the packet, source port, destination port, source IP and destination IP. Similarly, [37] proposed to a HMM-R detection scheme to detect Distributed Denial of Service (DDoS) attack using the source IP and destination IP in order to compute their Renyi entropy. A Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) is proposed in [39] for anomaly detection. The GRU-RNN is a technique based on classical RNN that can represent the relationship between current and previous events and enhance the anomaly detection rate. The features considered are divided between the performance on the network (bandwidth and duration) and the structure of the packets (type of protocol, source IP and destination IP). Still based on RNN, [41] developed a propose a detection and countermeasure scheme based on continuous wavelet transform (CWT) and convolutional neural network (CNN). The work of [45] proposed a hybrid classification model based on two steps. The first aims to eliminate unrelated and inconsequential features (based on some common feature selection methods) and provides the selected features to the second layer. The second layer, then, classifies the abridged data set using some popular machine learning and deep learning algorithm. Also, [42] used this formalism to detect DDoS attacks in a SDN network. They used a large set of features derived from network traffic headers. It is also possible to propose a hybrid approach which combines two formalisms as in [40]. In this work they use a hybrid of Neural Network and Hidden Markov Models for the creation of a composite model. HMM is used to capture state information in order to identify whether a network connection is legitimate or not. Against DDoS attacks which impact the controller, [20] compared the interest of four different formalisms (SVM, NB, DT and LR). A more flexible approach is proposed in [36] which introduced an IoT-Intrusion Detection and Mitigation (IoT-IDM) framework. Features represent the most critical aspects of the network traffic and this framework gives its user the flexibility to decide which fields need to be extracted, as the entire traffic is available. Also, the detection part is not related to a particular formalism to provide flexibility to the user but they used a SVM classification in their case study. [38] discussed decision tree and Bayesian approaches to predict the potential malicious connections and vulnerable hosts based on the identification of the patterns of the attack contrary to our work which focuses on the nominal behaviour of the network. The prediction results are used by the SDN controller to define security rules in order to protect the vulnerable potential hosts. Similarly, [53] proposed a similar approach using similar algorithm while [43] also used a signature-based approach using RNN.

These works combine the benefits of the SDN centralisation and machine learning techniques to protect the network through intrusion. However, machine learning may also be used to protect the controller directly as in the work of [47] which introduces a safe-guard scheme (SGS) for protecting the control plane against DDoS attacks. The procedure is divided into two modules: anomaly traffic detection and controller dynamic defense. The anomaly detection proposed consisted into distinguish forged flows from legitimate ones at the switch level of the switches. In this objective the RNN formalism is used.

From a general point of view, all these propositions are similar in the sense that they monitor the network using the centralisation offered by the SDN architecture and search for anomaly using predictions computed using a ML algorithm. The main difference resides in the selected features (which mainly depends on the monitored attack) but can be resumed in two categories: observation of the performance (such as the number of transmitted bytes ...) or the structure (i.e. the content such as IP source, destination ...) of the packets. Based on that, well-known algorithms are applied (SVM, RNN, HMM, NB, DT ...) in order to distinguish legitimate from malicious flows. Here, a multi-criteria approach is proposed in order to combine the features and provide flexibility in the method. Also, the three formalisms which are chosen and compared in this paper correspond to three levels of abstraction: Probabilistic Finite Automaton, Hidden Markov Model and Recurrent Neural Network. The first formalism is a classical discrete approach with a stochastic automaton (Probabilistic Finite Automaton). Then, this discrete approach is complexified (by adding one abstraction level through the hidden evolution) for a Hidden Markov Model [54]. Finally, discrete approach presents limitations and so an extension of these models which concerns a continuous approach is proposed through a Recurrent Neural Network [55].

These techniques are mostly used to secure the infrastructure and not the control. Indeed, in the controller, there is a detection method implemented which aims to detect any anomalies in the flow at the level of the infrastructure. Hence, not to detect the attack at this stage and not the ones focusing the control. Regarding the attack which focuses directly the control plane (for example by taking the control of the device using a Kali Linux exploit) the approaches are different. First, it is important to provide a redundancy of the controller, active or passive, and that's why the main focus is to consider multiple controllers as in [10]. Precisely, a decision-making security architecture is proposed. The aim is to prevent an attack by observing each other: to determine if the rules coming from one controller are valid, there is a vote between all the other controllers which limits the influence of a controller attack. Each controller takes part in judging which controller is infected by validating its production. In the same way, a blockchain approach is developed in order to secure the decisions of the control plane as in [11] and [46]. The blockchain developed by [11] provides a reputation mechanism: the more the controller misbehaves (i.e. the structure of the commands is different than the majority), the faster positive histories are forgotten. On the other hand, the more the controller behaves well, the faster negative histories are forgotten. The decisions of the controllers are the subject of a vote between all the other controllers in order to ensure the consistency of the taken decisions. Another solution proposed in the literature is the use of the Moving Target Defense such as in [48] with the introduction of the notion of shadow controllers. In the control architecture, they are two categories of controllers: the nominal ones in charge of the network and a second category which are the shadow ones. The aim of the shadow controller is to take the lead over the nominal ones in case of an attack. Indeed, in case of detection of probing traffic (using the tool "SNORT") then a part of the shadow controllers are selected, randomly, in order to respond to the traffic.

However, the current propositions to secure the control plane is based on a communication between the controllers. This interface is a security threat [13] as far as the information given by one controller might be malicious. More generally it has to be mentioned that each detection method using ML in the SDN context is based on the collection of the data at the controller level . This is a threat because the information given might be malicious (for example in case of a spoofing attack). Hence, our main originality is the absence of communication with the system monitored which is reflected in the absence of the interface of communication between the controllers.

3. Detection Problem

In this section, the detection problem and the proposed architecture are summarized.

3.1. Recovering a forwarding plane from the capture

The OpenFlow packets (version 1.3, [56]) exchanged through the southbound interface between the controller and the switches, are representative of the real behaviour of the controller. The whole activity (i.e. the trace Σ) of the

control is defined hence according to four types: $\mathcal{I}$ the inputs (transfer the control of a packet to the controller), $\mathcal{A}$ the commands (send the packet out of a specified port, modify the state of the switch), $\mathcal{N}$ the status (changes of port/link status of a switch) and $\mathcal{S}$ the statistics (read-state information from a switch about flow, table, queue, etc.).

A particular attention is paid on the actions since they are consecutive of the controller logic. A malicious or faulty controller will lead to missing or unexpected actions. The role of an observer will be hence to determine if the captured actions are, considering the context (the demands, the topology, the inputs, the status and the statistics) consistent and plausible with an unfaulty and non hacked control.

Such analysis depends on the control function. In this paper, we consider a routing algorithm. It means that the action consists of forwarding decisions (to which port(s) a switch need to forward a message?). Considering a demand from s to d, it means that the installed route consists of the actions defined by the controller and may be identified by the (surjective) function:

$$\mu(t, s, d) = \{\sigma \in \mathcal{A} \mid t_\sigma \in [t - \delta, t], s_\sigma = s, d_\sigma = d\}$$

where δ corresponds to the time to live of the actions.

Obviously (and as shown in the following), a valid route needs to verify specific properties (like no loop) such that a first set of faults/attacks might be detected. However, such investigation needs to be enlarged to the whole set of routes defined by the controller, i.e. the forwarding plane. Considering the list of demands $\mathcal{D}$ for which the controller decided to install active routes at time t such that:

$$\mathcal{D}(t) = \{(s, d) \mid \exists \sigma \in \mathcal{A}, t_\sigma \in [t - \delta, t],\ s_\sigma = s,\ d_\sigma = d\}$$

it is important to take into account the time to live of such actions (δ). Indeed, an action set up at $t - \delta - \epsilon$ is still valid at t.

It enables finally to define the active forwarding plane $\mathcal{P}$ (i.e. the set of active routes) at a given time t as:

$$\mathcal{P}(t) = \bigcup_{\forall (s,d) \in \mathcal{D}(t)}^{*} \mu(t, s, d)$$

where $\bigcup^{*}$ consists of a special union operator dealing with the different types of actions. If the type of the `Flow_Mod` is `add`, the (forwarding) action is directly added to the set; if the type is `delete`, the relevant previous action is removed from the set and if the type is `modify`, the relevant previous action in the set is substituted by the new rule.

In a way, this plane $\mathcal{P}$ can be seen as the internal state of the controller. The purpose of the observer consists therefore of determining online if it this state is consistency and representative of an unfaulty and unattack control.

3.2. Routes consistency

Another necessary condition about the consistency of the control deals with the connectivity of the routes. To verify this criterion there are the three (structural properties) rules introduced in [15]. We note $\mathcal{G} = (\Omega, V)$: the topology composed of the set of switches, Ω and the links between the switches, V and we introduce the function $\nu := i \rightarrow \omega$ which returns the switch identifier ω on which a node i is connected and $\xi := (\omega, \rho) \rightarrow \omega'$ the function which returns the neighbour switch ω' along a given port ρ of a given switch ω. Hence, each route from s to d included in the plane at time t will be considered consistent with the topology only if the three following rules are verified.

1. There is no loop:
   $\forall \sigma \in \mu(t, s, d), \quad \nexists \sigma' \in \mu(t, s, d),\ \sigma' \neq \sigma \mid \omega_\sigma = \omega_{\sigma'}$
2. There is no dead node:
   $\forall \sigma \in \mu(t, s, d), \quad \exists \sigma' \in \mu(t, s, d) \mid \xi(\omega_\sigma, \rho_\sigma) = \omega_{\sigma'}$
3. The destination is reached:
   $\exists \sigma \in \mu(t, s, d) \mid \omega_\sigma = \nu(d)$

By extension, a plane is consistency only if all routes are consistency. However, since an attack may lead to consistent but non desired routes, the problem of the detection of a faulty/malicious controller is hence more complex.

3.3. Routes likelihood

The problem consists now to determine if the plane implemented by the controller is plausible or not (for instance due to an attack). For example, a plane in which all routes are visiting a given common switch could be the symptom of a man-in-the-middle attack. Determining if a plane is plausible depends on the properties of the control algorithm. Regarding determinist routing algorithm, the answer might be easily obtained. For instance, [15] showed the efficiency of a method checking that the data plane is not changing while the context remains the same (i.e.same demands, same topology). However, for non-determinist routing algorithms (like in [57]), different results are possible (even for the same context).

For example, let us consider a topology of 6 switches and 6 end devices (one per switch). Each switch is connected to others. A representation of a nominal data plane is given in equation 1. Each line of the matrix represents a path in the data plane and, in the path, each value represents the next hop for the considering switch. Hence, an element $a_{i,j}$ represents the next hop for the switch i regarding the path related to the demand j. As an example, the first line consider the demand 1 in direction of 2 and the path is, from 1 to forward directly to 2. While in case of attack the representation is given in equation 2. Also regarding the first line at the demand 1 in direction of 2, here the path is $1-3-5-6-4-2$. To ease the readability, the data plane has been represented through matrix Π within $\pi_{i,j}$ represents the next hop of the switch i for the j-th demand. The routes set up by the controller in two cases satisfying the necessary conditions given above. However, the ones in case of attack are not optimal and may lead to congestion or an increase of the latency in the network due to the high number of hops.

$$\Pi_{Nominal} = \begin{pmatrix} 2 & & & & & \\ & 4 & & 6 & & \\ & & 5 & & 6 & 4 \\ & & & 6 & & \end{pmatrix} \tag{1}$$

$$\Pi_{Attack} = \begin{pmatrix} 3 & & 5 & 2 & 6 & 4 \\ 3 & 1 & 5 & & 6 & \\ 2 & 5 & 1 & & 6 & 4 \\ 3 & 1 & 5 & 2 & 6 & \end{pmatrix} \tag{2}$$

Hence, even if the necessary condition defined just above are satisfied by the data plane, it might be a malicious data plane highlighting that such conditions are not sufficient. In a first approach, the impact of the decisions of the controller might be considered. Indeed, the control is supposed to manage a set of metrics (it can be the available bandwidth, number of packet loss or latency depending on the control) and a hypothesis could be added such that the evolution of some metrics might remain limited. But still, this might not be sufficient as the delay performance can be degraded not due to a fault or an anomaly in the control but due to the appearance of a large demand for example. A second approach might also be considered by comparing the structure of the plane to a set of previous observed and known planes. These approaches are developed in the following section.

4. Anomaly-based detection

The aim of the method is to compare the running behaviour of the control to the unhacked commands observed. In fact, we propose to determine the likelihood score of a data plane $\mathscr{L}(\mathcal{P})$ according to a multi-criteria approach:

$$\mathscr{L}(\mathcal{P}) = \Sigma_{i=1}^{n} \alpha_i \times \hat{p}_i$$

with:

- $(\alpha_i)_{i\in[1,n]}$: the criterion weights such that: $\Sigma_{i=1}^{n} \alpha_i = 1$
- $(\hat{p}_i)_{i\in[1,n]}$: the normalised-likelihood of the criterion i

Therefore, a data plane $\mathcal{P}$ implemented by the controller is assumed to be consistent iff its likelihood score is upper a threshold noted TD, i.e. $\mathscr{L}(P) > TD$.

The likelihood of each criterion p_i might be obtained from several computation methods. Hence, it is necessary to normalise it and each normalised value $\hat{p}_i$ would be obtained, with P the set of values used in the normalisation, as follows:

$$\hat{p}_i = \frac{p_i - min(P)}{max(P) - min(P)} \tag{3}$$

In this work two criteria are proposed. Firstly, compare the impact of the plans by assessing the performance of each plan and secondly, compare the routes of the observed plans to the ones known as consistent. Such that it gives:

$$\mathscr{L}(\mathcal{P}) = \alpha_1 \times \hat{p}_1 + \alpha_2 \times \hat{p}_2$$

The process is represented in Figure 2. The activity of the command is observed, though the OpenFlow packets, and then this information is processed. First there is a clustering stage in order to limit the number of possible observations and then observations are analysed in order to determine the likelihood of the control.

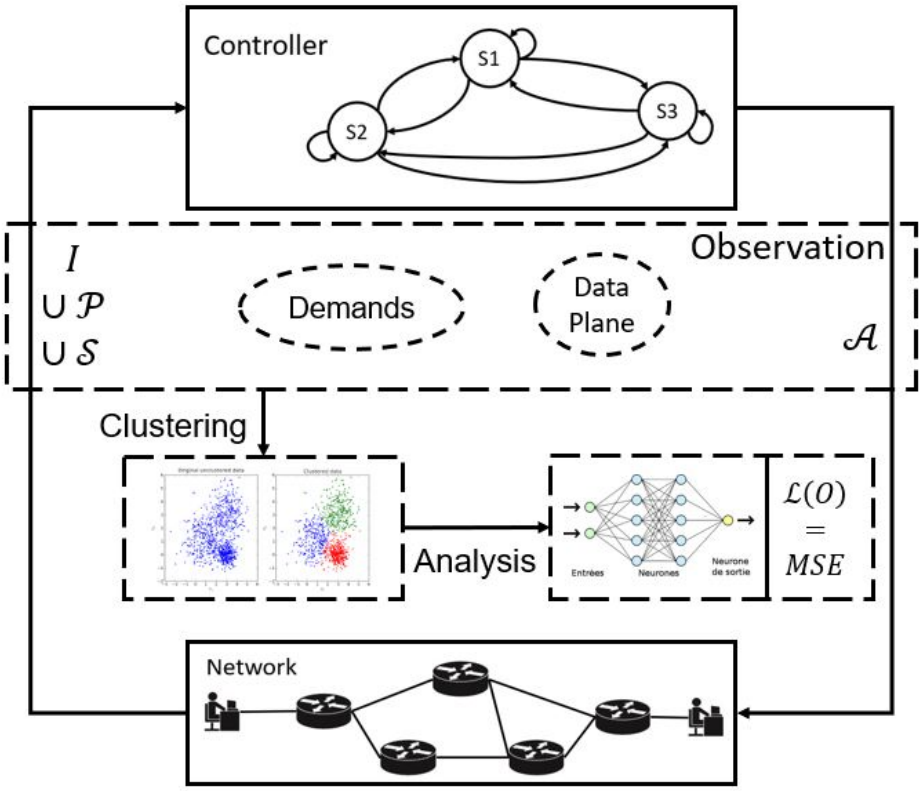


Figure 2: General Process

In what follows, p_1 and p_2 are developed.

4.1. Criterion 1: planes performance

The first considered criterion concerns the impact of the plans. The evaluation of the performance of a decision depends on considered metrics (depending on the control it can be the latency or the packet loss for example) and the objective of the control.

Here, we focus on the bandwidth offered compared to the demand throughput. For each flow (s, d), the traffic sent N_{Sent} and the received one $N_{Received}$ are compared. For a given flow (s, d), the plane efficiency is computed as follows:

$$m(\mathcal{P}, (s, d)) = \frac{N_{Sent} - N_{Received}}{N_{Sent}}$$

With such a definition, the efficiency depends on the considered demand. Since a data plane aims to gather all routes, a function φ which corresponds to the distance between two planes $\mathcal{P}$ and $\mathcal{P}'$ (whatever the demand) is introduced such that:

$$\varphi \quad : \quad (\mathcal{P}, \mathcal{P}') \quad \mapsto \quad \max_{\forall (s,d) \in D(t)} |m(\mathcal{P}, (s,d)) - m(\mathcal{P}', (s,d))|$$

This function measures the gap between the current plane $\mathcal{P}$ and a previous plane observed $\mathcal{P}'$ by considering the maximum difference of efficiency observed. To note that it is possible to consider the *min* or *average* operator instead of the *max* in the objective to be more tolerant.

Hence, the likelihood of the n-th data plane, $\mathcal{P}_n$, corresponds to the distance between it, $\mathcal{P}_n$, and the set of previous planes observed:

$$p_1 = 1 - \max_{i=0...n-1} \varphi(\mathcal{P}_n, \mathcal{P}_i)$$

Such a definition implies that a plane might be considered as faulty/attacked as long as according to it is different to at least one single previous observed plane. Similarly that for the φ function, this statement can be slightly changed by considering another function as the *min* or the *median*.

This criterion has some weaknesses. Indeed, in case of an attack that aims to degrade performance slowly, the evolution between the planes may remain low such that the threshold TD will not be reached and the fault/attack not detected. That's why the second criterion will focus on the routes diversity themselves.

4.2. Criterion 2: planes structures

The second criterion consists into analysing the structure of the routes and decisions of the controller. The objective is to determine the likelihood score of an observed sequence of decisions, i.e. data plane here, of the control noted W.

$$p_2 = \mathscr{L}(W) \tag{4}$$

Such likelihood $\mathscr{L}(W)$ might be computed through several methods and using several formalisms. However, the decisions of the controller depend on its intern variables. As there is no East-West interface between the controller and the observer, we do not have access to the evolution of these interns variables. The only source of information is the activity of the controller (i.e. the OpenFlow messages), resulting itself of the evolution of its interns variables. Hence, the aim of the formalism chosen will be to predict and establish a likelihood score to each decision of the controller based only on the previous decisions observed.

Here after, three formalisms are proposed for such computation.

5. Proposed formalisms to determine the likelihood

As a we consider non-deterministic control, a non usual event may occur without being an issue or due to a malicious controller. Therefore, we propose to classify and determine the likelihood of a sequence of n observations. Face to such problematic, the first model which can be used is finite state automaton in order to link statistically the observations between each other directly. However, the literature also developed machine learning and deep learning techniques. For each of the approach we compared several formalisms as SVM, kNN, LSTM ... and we observed similar conclusions about the sensibility of the approach. As the objective of the paper is not to be exhaustive and just only one control is considered, we decided to retain just one formalism for each approach and focus on the conclusions linked to the approach. In this objective we propose to use three formalisms: Probabilistic Finite Automaton (PFA), Hidden Markov Model (HMM) and Reccurent Neural Network (RNN) which are presented here after.

5.1. PFA

A first approach is to consider that the planes are observed and we are just considering the sequence of the planes without considering the intern states of the controller.

In this objective we proposed to use a Probabilistic Finite Automaton (PFA) defined by a 5-tuple:

$$PFA = \langle Q_A, q_0, \Sigma_A, \delta_A, P \rangle \tag{5}$$

With:

- Q_A is a non-empty finite set of states
- $q_0 \in Q_A$ is the initial state
- Σ_A is the alphabet of events
- $\delta_A = Q_A \times \Sigma_A \times Q_A$ is a set of transitions composed of a departure state, an event conditioning the firing of the transitions and an arrival state
- $P : \delta_A \rightarrow [0, 1]$ is the transition probability

It has to be mentioned that the for each state, the sum of all outgoing transitions is equal to 1:

$$\forall q \in Q_A : \Sigma_{q' \in Q_A, o \in \Sigma_A} P((q, o, q')) = 1 \tag{6}$$

An example of a PFA is given in Fig. 3. It can be observed that the observations are related directly. From one observation, for example O_1, the formalism permits to determine, statistically, what is the next observation according to the probability associated to the possible event such as e_{O_4}.

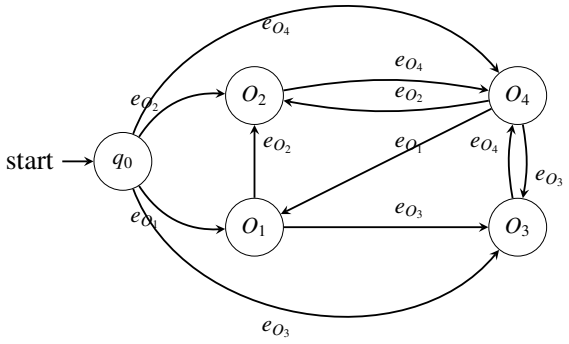


Figure 3: Example of a PFA

In this work we are considering the activity of a non deterministic control. Thus, the apparition of one unusual event is not a problem as far as it is assumed that this is possible. In this perspective we will consider the likelihood of the sequence of n events.

Hence, the observation of the sequence of n plans $\mathcal{T} = (\mathcal{P}_i)_{i \in [1,n]}$ corresponds to the firing of $(\delta_i)_{i \in [1,n]}$ in the PFA θ. We assume that:

$$p_2 = p_\theta(\mathcal{T}) = \Pi_{i=1}^{n} \frac{P(\delta_i)}{n} \tag{7}$$

However, this approach is limited and is relevant for control with a limited variance. To overcome this limitation we propose to infer, from the observations, on the interns variables of the controller. Here, as there is no East-West interface with the controller, we do not have access to the evolution of the interns variables of the command algorithm. We just have access to the observation of the activity of the command which is the event resulting of the evolution of the interns variables. Then, we propose as a second approach to infer on the intern variables. The process is represented in Fig. 4. Instead of making a direct relation between the sequence of observation $O_1 - O_2 - O_3$, here the aim is to infer and estimate the internal states $S_1 - S_2 - S_3$. Then, determine if such evolution is likely or not.

In this objective, we propose two other formalisms which permits to infer on the intern states of the controller based on observation.

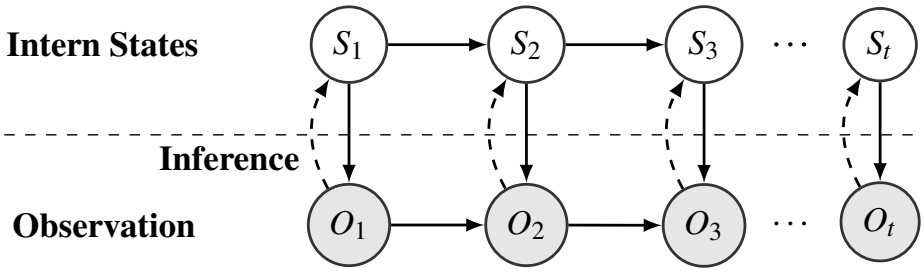


Figure 4: Inference process

5.2. HMM

Here, we assumed that the evolution follows a Markov Process. Thus, we propose to use the Hidden Markov Model (HMM) formalism. The general definition of HMM encompasses the case where the hidden variables are continuous (see [58]) but here we will consider the most common definition: discrete variables [59]. This formalism has already been used in the context of the security of the SDN controller as in [37] for example. The principle of this formalism is to infer the interns variables of the controller based on the observation. And then to determine the likelihood of an observation by inference over the internal states. In this objective, probabilistic theory such as the Bayes' theorem is used.

A Hidden Markov Models (HMM) is defined by 3-tuple $HMM = (\pi, A, B)$ defined as follows:

- N: the number of states
- $S = \{s_1, \dots s_N\}$: the set of states, st_t is the current state at the time t
- M: the number of observations
- $O = \{o_1, \dots o_N\}$: the set of observations, ot_t is the current state at the time t
- $A \in M_{N,N}$: a transition probability matrix A, $A = (a_{i,j} = p(q_t = s_j | q_{t-1} = s_i)$: represents the probability of moving from state s_i to state s_j
  $\forall i : \Sigma_{j=1}^{N} a_{i,j} = 1$
- $B \in M_{N,M}$: observation probability matrix, $B = (b_{i,j} = p(ot_t = o_i | st_t = s_j))$: each expressing the probability of an observation o_i being generated from a state s_j
  $\forall i : \Sigma_{j=1}^{N} b_{i,j} = 1$
- $\pi \in M_{1,N}$: an initial probability distribution over states.

A representation of a HMM is given in Fig. 5. As mentioned, here the aim is not to set a direct relation between the observation but to infer over a set of internal states. Hence, in the example there are two possible observations and four internal states. The transition matrix permits to determine the probability of transition between the internal states, for example $a_{1,2}$ represents the probability to reach the state s_2 from s_1. The observation probability matrix expresses the probability of observing an event depending on the internal state. For example, $b_4(o_2)$ represents the probability to observe o_2 from s_4.

HMM introduces three problems [59] which are resumed as follows:

1. Given a HMM $\lambda = (\pi, A, B)$ and an observation sequence O, determine the likelihood $p_2 = P(O|\lambda)$.
2. Given an observation sequence O and an HMM $\lambda = (\pi, A, B)$, determine the best hidden state sequence Q.
   $Q^* = argmax_Q p(Q|O, \lambda)$
3. Given an observation sequence O, the set of states S, determine the HMM parameters π, A and B.
   $\lambda^* = argmax_\lambda p(O|\lambda)$

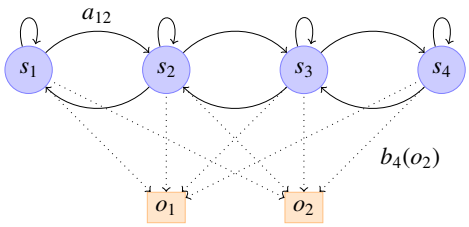


Figure 5: Example of a hidden Markov model

To solve these issues there are well-known algorithms proposed in the literature: Forward-Backward algorithm [59], Viterbi algorithm [60] [61] and Baum-Welch algorithm [62].

Hence, to determine the likelihood of a sequence of observation the Forward-Backward or Viterbi algorithm may be used while the Baum-Welch algorithm is used for the learning phase. However, HMM has a fundamental limitation. The Viterbi algorithm, used to determine the best hidden state sequence given an observation, is expensive, both in terms of memory and computation time. For a sequence of length n, the dynamic programming for finding the best path through a model with N states takes a time proportional to $N^2 \times n$. Other algorithms for hidden Markov models, such as the Forward-Backward algorithm, are even more expensive and take times which are proportional to N^n. That's why we propose a third formalism; RNN which is a signal approach (and so, continuous approach) of the similar problem.

5.3. RNN

Recurrent Neural Network (RNN) and HMM share similarities since they both involve latent variables, but they differ from the way those variables are built [55]. The main difference between these two formalisms here is that HMM constitute a discrete approach of the problem while RNN is a continuous one.

A RNN can be viewed as an extension to a feed-forward Neural Network (NN), where the output of a hidden state h_t depends on the previous time h_{t-1} and are sequentially and deterministically computed through a given activation function σ which depends on a set of parameter $\theta = (W, U, b_h)$ with:

- x_t: input vector
- h_t: hidden layer vector
- W: parameter matrix of connection between the hidden layer
- U: parameter matrix of connection between the input layer and the hidden ones
- b_h: parameter vector
- σ: activation function

And then:

$$h_t = \sigma(W \times x_t + U \times h_{t-1} + b_h), \text{ for } t \in [1, T] \tag{8}$$

An example of a RNN structure is given in Fig. 6.

To measure the likelihood of a sequence in RNN we propose to measure the gap between the prediction, which corresponds to the expected behaviour, $(\hat{O}_i)_{i\in[1,n]}$ and the observed $(O_i)_{i\in[1,n]}$ sequence by determining the mean squared errors MSE.

$$p_2 = 1 - MSE = 1 - \frac{1}{n} \times \Sigma_{i=1}^{n} (O_i - \hat{O}_i)^2 \tag{9}$$

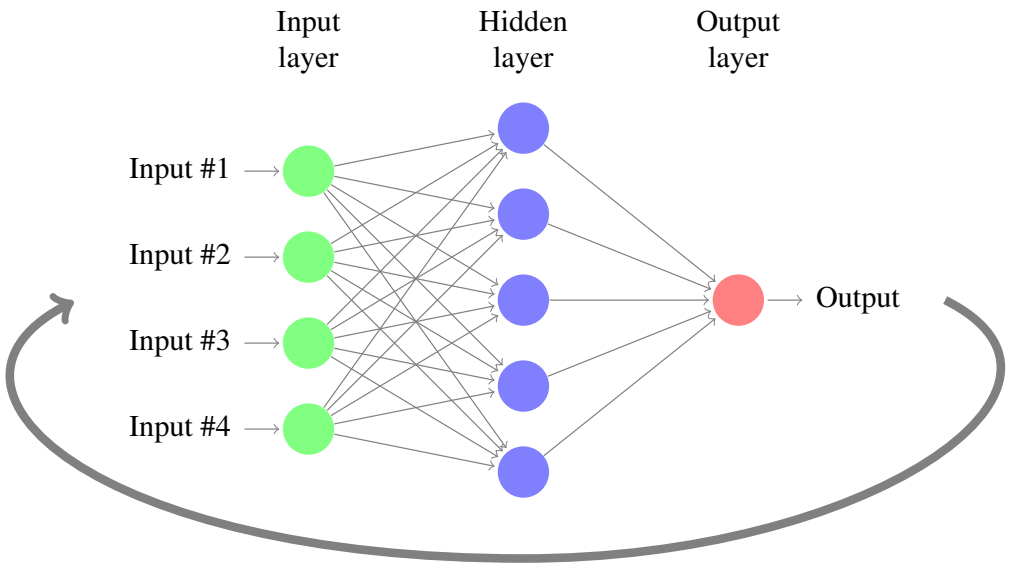


Figure 6: Representation of a RNN

5.4. Solutions in case of state explosion

The proposed two first formalisms are limited in case of large networks and large variety of possible decisions because it will lead to a state explosion. For example, if we consider the routing control of the GÉANT network with 23 nodes there is more than 2^{506} possible data planes. Depending on the variance of the control, a large number of states might be expected.

In such a case, it is possible to cluster the solutions using for example k-Means techniques as presented in [63]. The aim is to gather the decisions which can be considered as similar according to a set of predefined features. An example of features in case of a routing control might be the total number of hops in the data plane. Hence, each data plane with a similar number of hops will be considered equivalent and be gathered in one single state. Finally, each observation state corresponds to one cluster.

6. Case Study

This section aims to present the case study considered for the analysis of the proposed paradigms.

6.1. Test Environment

The case study is based on the network controller introduced in [57]. Here, the control function is a multi-objective proactive routing through reinforcement learning technique. The metrics used are the delay, the packet loss and the available link bandwidth. Since the control is proactive, a new data plane is periodically set up by the controller. The code is available on [64] and is implemented over a Ryu controller [65]. The topology consists of 23 nodes and 37 links as shown in Figure 7 and is similutaed using Mininet[1]. To tackle Mininet limitations, we scaled the 10 Gpbs, 2.5 Gpbs, and 155 Mbps link capacities of GÉANT to 100 Mbps, 25 Mbps, and 1.55 Mbps, respectively.

The considered traffic is the dataset given in [66] which relies on intra-domain traffic over the GÉANT topology.

The considered attacks corresponds to a `Flow_Mod` modification as presented in [67]. Firstly, the attacker takes control of the controller using Metasploit, a library of Kali Linux. Secondly, the data planes computed by the controller are modified in order to redirect the traffic through the link with the less capacity with the objective to cause congestion and in consequence degrade the service (without sending any traffic on the network).

[1] http://mininet.org/

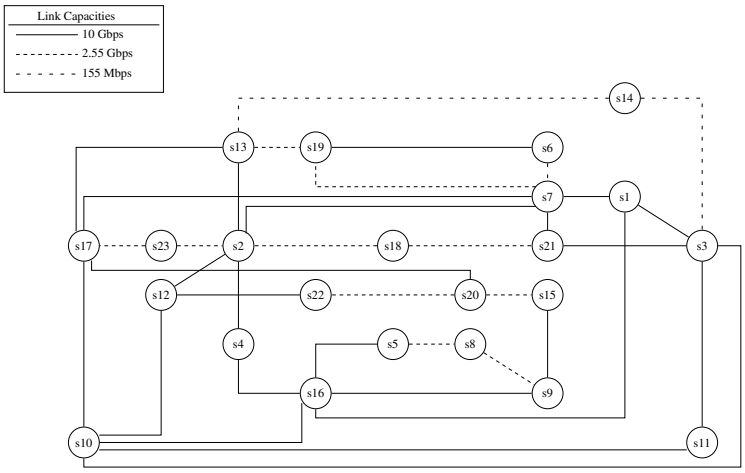


Figure 7: Topology of the network GÉANT with 23 nodes.

The benchmark infrastructure considered is represented in Fig. 8. Here after, a proof of concept and discussion over the sensitivity to the weight is made. Then, the evaluation of the criterion 1 is presented in two cases: face to a brutal and to a slow attack. Finally, the criterion 2 is evaluated and discussed.

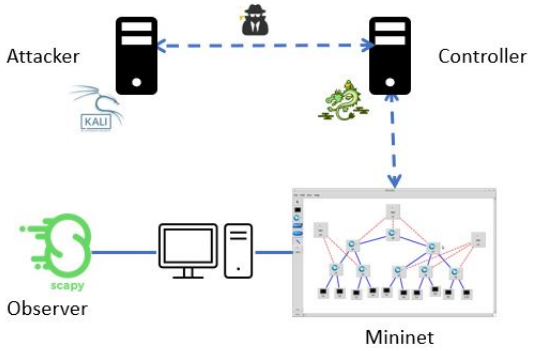


Figure 8: The physical topology

6.2. Sensitivity to the weight

To recap the attack consists into taking the control of the controller and set up malicious data planes: re-direct the traffic through the link with the minimum capacity to cause congestion (and so delay on the network). In this part we will consider $\alpha_1 = 0.5$, $\alpha_1 = 0.9$ and $\alpha_1 = 0.1$ (so respectively $\alpha_2 = 0.5$, $\alpha_2 = 0.1$ and $\alpha_2 = 0.9$). The aim of this first study is not to compare the three proposed formalisms and so only one is used, PFA here. The results are represented on Fig. 9. The graph above represents the likelihood computed for the three values of α_1, the second graph represents the normalised evolution of the criterion related to the performance of the decisions (p_1) and the third graph represents the normalised evolution of the score related to the structure of the decisions (p_2). The experience is as follows: 40 minutes of nominal behaviour (until iteration number 60 on Fig. 9) and 20 minutes of attack (from iteration number 60 and until iteration number 89 on Fig. 9). The threshold TD has been determined experimentally according to the worst case observed and fixed to 0.7.

The alarms raised during the experiments are represented by a node on Fig. 9. There are several false negative after the attacks for $\alpha_1 = 0.5$ and $\alpha_1 = 0.9$ (for example the 61st to 65th iterations), this is due to the fact that the impact on p_1 is still not significant, as in can be observed on the second graph of Fig. 9, and so the average with p_2 stays upper the threshold even if a decrease is observed on Fig. 9. Hence, a prioritisation of p_2 would have permit

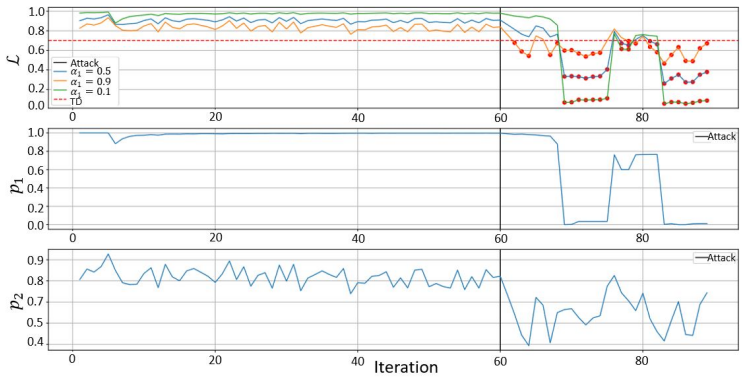


Figure 9: The results with three different values for (α_1, α_2)

to detect the attack earlier in this case as it can be observed for $\alpha_1 = 0.1$. Indeed, in order to be more flexible it is possible to modify the value of α_1 and α_2 in order to prioritise the criterion over the other. This choice depends on the attack to monitor or feared and also on the considered control (its variance) as it will be discussed. In what follows, the impact of each individual criteria will be evaluated according to the metrics defined here after.

6.3. Performance indicators of the detection

To measure the performance of our detection method we propose to consider the number of true positive TP, false positive FP, true negative TN and false negative FN. These values will be used to determine some properties of the systems: the precision noted P, the recall noted R and the global accuracy noted Acc. The precision corresponds to the number of correct alarms (TP) compared to the total of alarms ($TP + FP$) while the recall is defined by the number of correct alarms (TP) relative to the number of alarms that have to be taken ($TP + FN$)

Formally, they are computed as following:

$$P = \frac{TP}{TP + FP} \tag{10}$$

$$R = \frac{TP}{TP + FN} \tag{11}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

7. Evaluation of the Criterion 1: $\alpha_1 = 1$ and $\alpha_2 = 0$

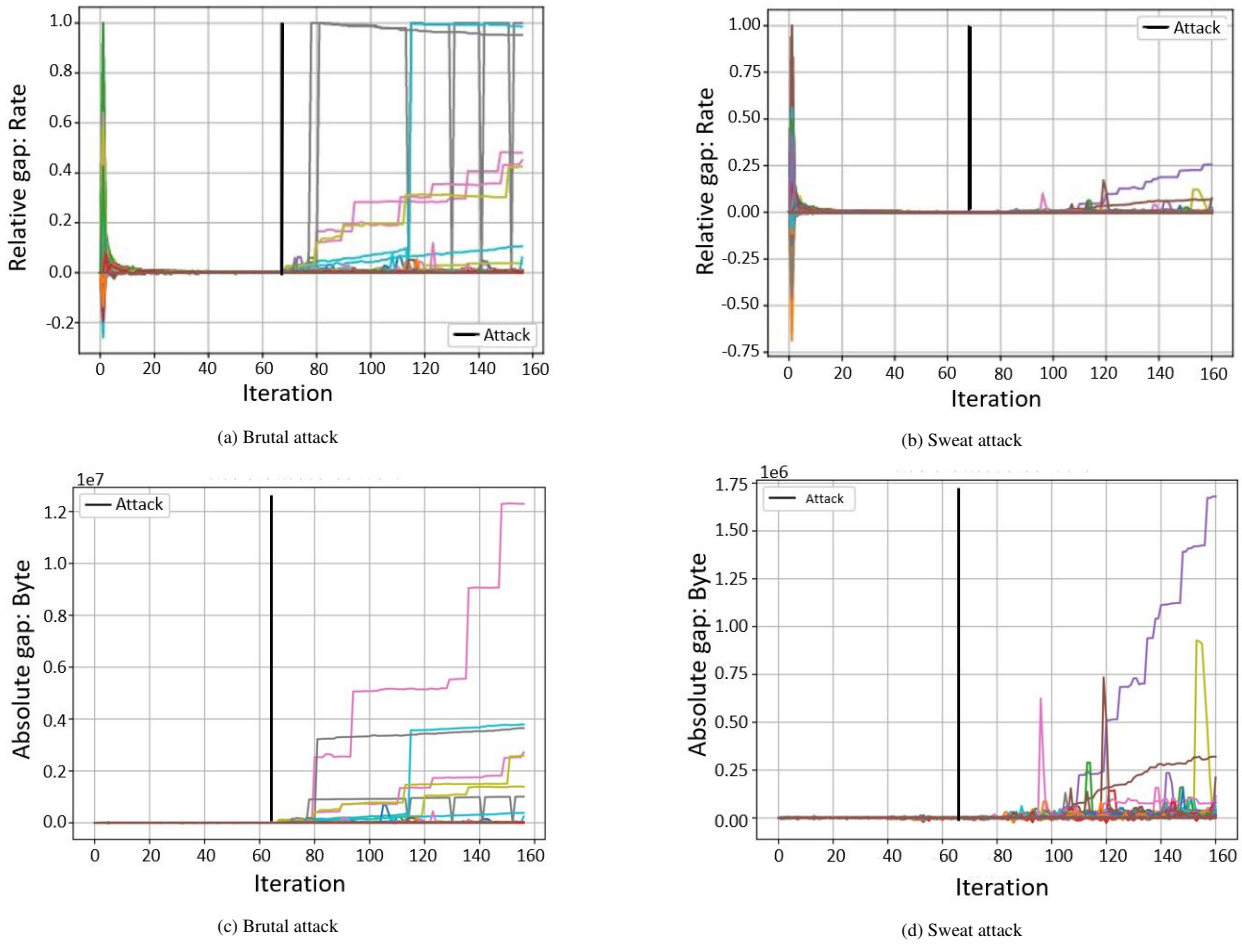
Here, we assume that $\alpha_1 = 1$ and $\alpha_2 = 0$ such that $\mathscr{L}(\mathcal{P}) = p_1$ in order to observe the impact of the first criterion p_1 on the likelihood computation.

7.1. Considered control

The considered control is still the one presented in [57]. This control is pro active, it is based on a greedy method which implies a random aspect and so in the same context the controller may set up different data plane. The architecture is named Reinforcement Learning and Software-Defined Networking Intelligent Routing (RSIR) and aims to optimise the following Quality of Service metrics: available bandwidth, delay and packet loss. They propose to add a Knowledge Plane and define a routing algorithm based on Reinforcement Learning (RL) that takes into account link-state information to explore, learn, and exploit potential paths for an intelligent routing, even during dynamic traffic changes.

7.2. Proof of concept

The considered attack aims to set up data planes leading to congestion in the network. Basically, instead of the routes computed by the RSIR algorithm, the attacker will install routes which pass by the links l with the less capacity. As the total demand over the network is bigger than the capacity of the link l, congestion will hence happen. In fact, two types of attacks are considered: the first one considers a brutal effect by changing all the routes in the same time and the second considers a smooth effect, as in low and slow DDoS attack, by changing one single route at a time. Fig. 10a and Fig. 10b deal with $m(\mathcal{P}, (s, d))$ while Fig.10c and Fig.10d show $|N_{Send} - N_{Received}|$ for each flow.



(a) Brutal attack

(b) Sweat attack

(c) Brutal attack

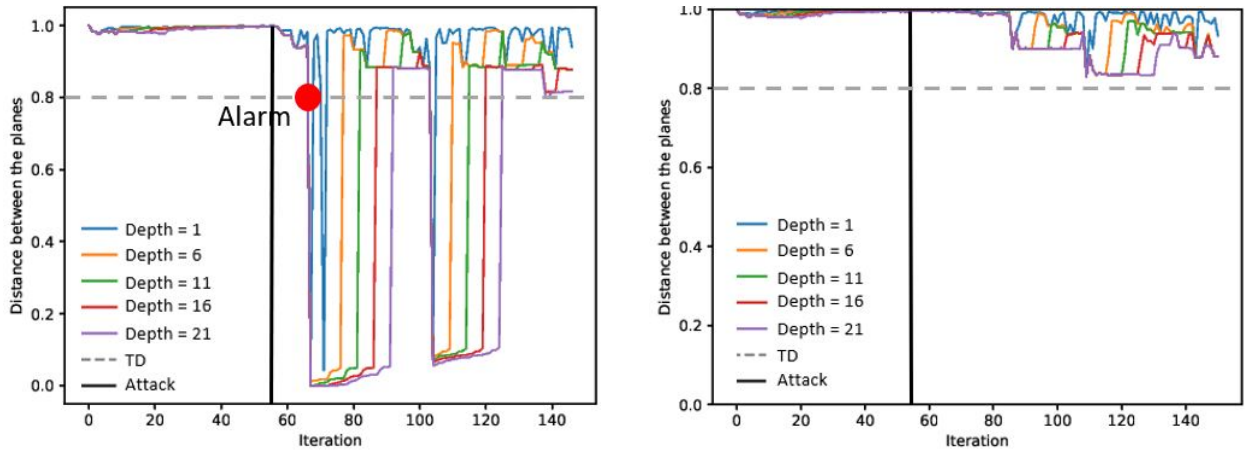
(d) Sweat attack

Figure 10: Impact of the attacks

It can be observed on Fig. 10a and Fig. 10b that there are losses at the beginning of the experiments (the 10th first iterations). Indeed, initially there are some packets which are sent but, due to the network latency, not yet arrived. It corresponds to the expected initial convergence of the network and will not be considered in the following. For the 70th first iterations, the data planes computed by the controller are not modified, the attack begins at the 71st iteration. For the brutal attack, the effects are visible around the 80th iteration while the derived of the sweat attack stays relatively constant.

The evolution of the criterion p_1 is finally given in Fig. 11. A threshold has been fixed to $TD = 0.8$. It can be observed that for a sweat attack, the criterion does not evolve significantly and stays relatively constant. Here, it stays under the threshold and no alarm is raised. However, in case of a brutal attack, an alarm is triggered. With the proposed configuration, the triggering occurs after 20 measures of the statistics ($\sim$ 400 seconds). Nevertheless, the reactivity and the sensitivity of the detection using this criterion depends on the tolerance fixed. But, whatever the

fixed tolerance, the consideration of the performance is sensitive to the case of a sweat attack.



(a) Brutal attack

(b) Sweat attack

Figure 11: Distance between the plans

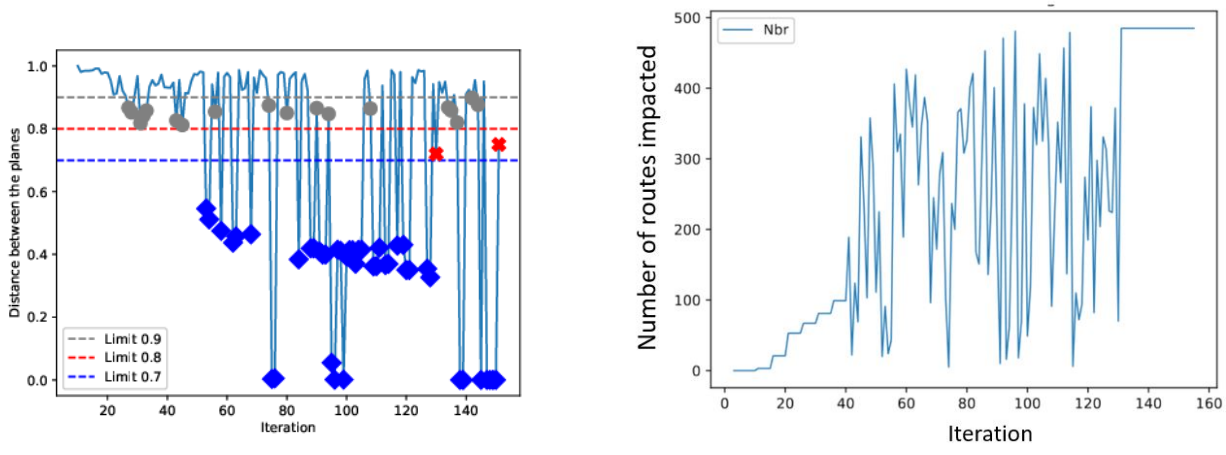
Also, to speed up the calculations it is possible to change the *depth* of the previous observed planes such as:

$$p_1 = 1 - \max_{i=n-depth...n-1} \varphi(\mathcal{P}_n, \mathcal{P}_i)$$

Fig. 11 highlights that for $i < j$ the criterion for $depth = i$ is lower than for $depth = j$. Hence, higher value of *depth* implies more alarms raised but it changes nothing in terms of reactivity.

7.3. Impact of the threshold

In this section, a uniform distribution is used to guarantee the modification of the routes from the original data plane computed by the controller. The results are given in Fig.12. Each dots corresponds to an alarm. Also, we add three horizontal lines which correspond to three examples of TD: 0.7, 0.8 and 0.9. This limit means that 10% (or respectively 20 and 30) of the data transmitted might not be arrived yet.



(a) Distance

(b) Number of evolution

Figure 12: p_1 evolution in case of a random routes attack.

The evolution of p_1 is represented on fig. 12a while the number of the route which passes by the link l for each iteration is represented on 12b. Also, the alarms are represented by the points on the fig. 12a. On fig . 12 we compared

the alarms in case of three different threshold values. Increasing the value of the threshold implies to let less flexibility to the evolution of the criterion. This is the reason why, for 90% there are more false alarms than for the two other values. However, increasing the value of the threshold leads to tolerate evolution of the criterion which may be linked to an attack. For example, it can be observed that the 82-th iteration is under attack but detected just for one value of the threshold.

It can be observed that in the first part of the experiment (until the 40-th iteration) there is less than 20% of the routes which pass by the link l which leads to no congestion but some delays in the worst case. This is the reason why the p_1 does not decrease under 80%. Then, between the 40-th and 130-th iteration the number evolves randomly. As it evolves randomly, the routes concerned also evolve. Which means that the link l is not permanently congested. Moreover, it can be observed at the iteration number 75 that there are less than 10 routes which pass by the link l which leads to a correct transmission of the flow (and the liberation of the link) which explains a decrease of p_1 in the next iterations.

Regarding the thresholds $TD = 0.9$, there are obviously more alarms than the other due to the lack of tolerance. However, increasing the value of the threshold has an impact over the reactivity. It can be observed that for $TD = 0.7$, the first alarm is raised after 55 iterations which means that during the 15th previous iterations the service is damaged. Such tolerance may lead to a worst case as observed in the proof of concept with Fig.11b, where no attack is detected. Hence, the threshold should correspond to a compromise between the reactivity and the precision of the detection.

The value of the performance indicators of the detection are given in the Table. 2.

Threshold	Precision	Recall	Accuracy
0.9	0.89	0.55	0.63
0.8	1.0	0.44	0.60
0.7	1.0	0.42	0.58

Table 2: Performance of the paradigm 1

Let us focus on the precision. It can be observed that the precision increases with the tolerance due to the number of false positive as it can be observed on Fig. 12a. But, it can be observed that p_1, after the alarms raised for the tolerance of 0.9, decreased significantly so it may be too early to raise an alarm.

Now, let us consider the recall. Such properties corresponds to comparison of the number of attacks detected over the number of attacks that should have been detected. The rate of false negative increases while the tolerance increases but the question is how to interpret such false negative. The aim of the method is to verify the consistency of the control and so if the performance of the decisions of the controller is within the tolerance, this is not an issue. This is the reason why the value of the recall is under 0.6: there is an attack but the impact on the network is tolerated which means that's regarding the criterion 1 there is no anomaly. It has to be mentioned that the value of the recall is relatively low. This is linked to the condition of the experiment: the "abnormal" path of the data plane is randomly selected, as represented in Fig. 12b, which means that the congestion takes more time to appear compared to the brutal attack. Hence, the attack has a sweat effect on the network which explains that the recall is low. However, a consequence of considering a high tolerance is to be sensitive to a slow attack as presented in fig. 11b.

8. Evaluation of the Criteria 2: $\alpha_1 = 0$ and $\alpha_2 = 1$

In this part we assume that $\alpha_1 = 0$ and $\alpha_2 = 1$ such that $\mathscr{L}(\mathcal{P}) = p_2$. The goal is to observe the impact of the second criteria on the likelihood computation, i.e. the likelihood of the sequence observed.

8.1. Considered control

There is the deterministic algorithm such as Dijkstra or Belleman-Ford. In such a case, the controller returns the same solution for each similar request. However, regarding a Non-Deterministic control the controller may return several different solutions in response to a similar request. Moreover, the number of different solutions depends on the variance of the control and regarding our case study, in the worst case we may have 2^{506} possibilities. Basically, it corresponds to the algorithms developed based on learning techniques. Indeed, nowadays the use of ML techniques is

increasing. It is considered that these techniques are better as compared to traditional (and deterministic) algorithms, particularly for the processing and analysis of large volumes of data.

As presented in [68] they are several ways of applying machine learning for routing optimization. Regarding supervised machine learning techniques, NeuRoute introduced by [69] is a framework of dynamic routing for SDN that leverages ML and solves the Maximum Throughput Minimum Cost Dynamic Routing Problem, achieving the same result as other dynamic routing algorithms, but requiring less execution time. Also, [70] presents AIER, an ANN to predict the minimum congestion probability among all path configuration. The network is trained to predict the congestion given the loads for all data flows and all the available path configuration.

The supervised approach is quite common but it is also possible to use unsupervised learning as in the work of [71] which explores the applicability of ML algorithms for selecting the least congested route for routing traffic in SDN. The authors propose two ML methods: a K-means clustering algorithm and the Vector Space Model.

Finally, the reinforcement learning (RL) is also a spread approach to develop routing algorithm. Such as in the work of [57] as already presented in the paper. Moreover, [72] emphasizes the need to define a reliable Quality of Servise (QoS) routing mechanism for large-scale SDN-based networks. To solve this issue, they propose QoS-aware adaptive routing in multi-layer SDN using RL.

Each of these approaches returns a random route in response to a solicitation and hence can be model as random variable. The difference between the approaches corresponds to the variance of the random variable.

Therefore, the aim of these sections is to study the accuracy of the proposed formalism depending on the control and its variance. Hence, we took a set of data planes set up by [57] and the new control aims to choose the one to set up randomly after a random draw according to a Poisson-distribution with fixed parameters. The choice of the kind of distribution does not matter, the main parameter corresponds to the variance. This parameter defined the variability of the control and we will observe the impact of this variability relatively to the formalism. So we consider a control C as follows:

$$C \sim \mathcal{P}(\lambda) \tag{13}$$

8.2. Proof of concept

Parameters of the models

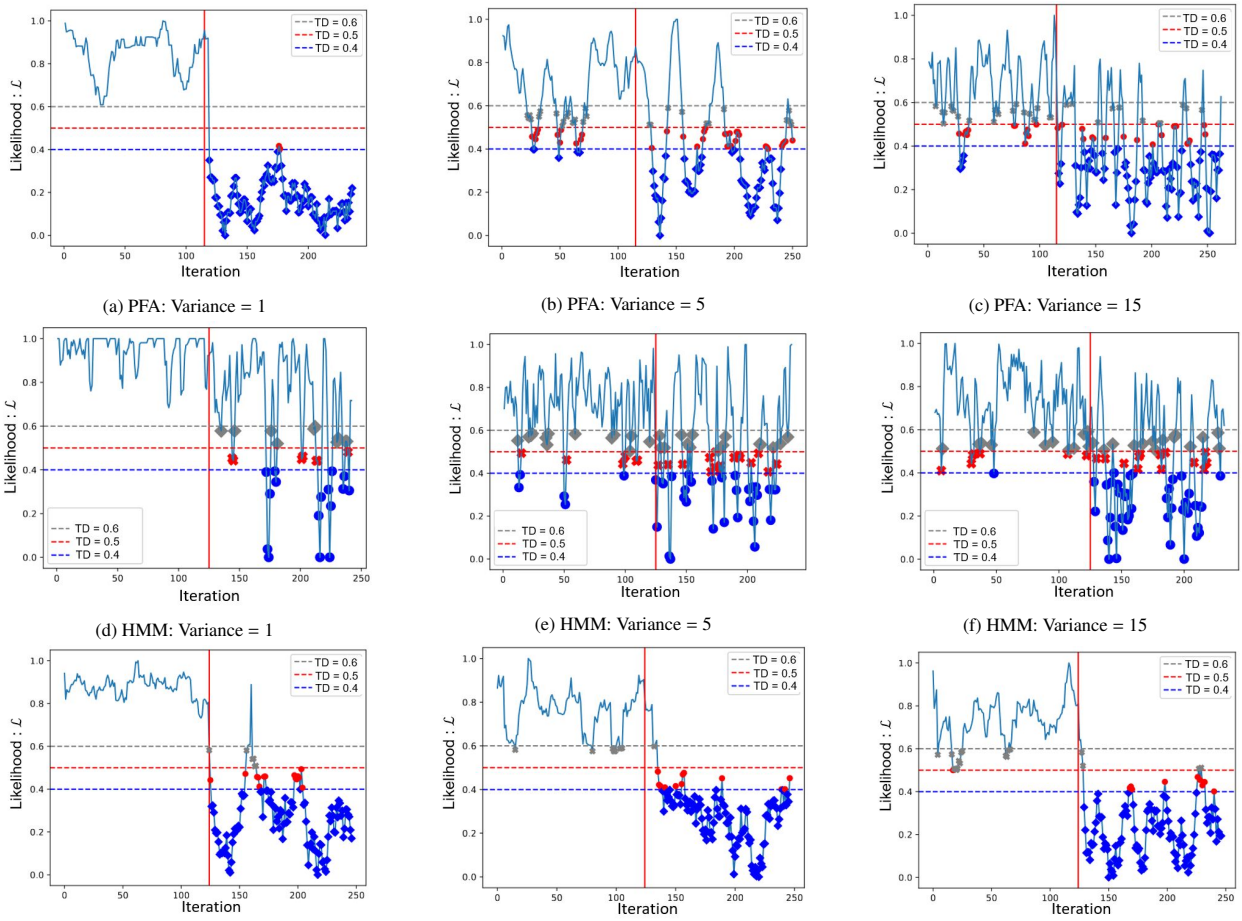
Regarding PFA, there is no specific parameter. For the HMM we will consider $N = 3$ intern states. A resume of the configuration chosen for the RNN is given in Table. 3. The observation, which corresponds to input, are data planes and we propose to associate to each data plane a score corresponding to its frequency apparition expected in this context. Also, for the RNN the value x of the dataset D has to be normalised in the range [0, 1] by min-max scaling as follows:

$$x_{Norm} = \frac{x - min(D)}{max(D) - min(D)} \tag{14}$$

Variable	Parameters
Input Layer	5
Hidden Layer	4, 3
Output Layer	2
Activation function	Tanh
Loss Function	Mean Squared Error
Batch Size	1
Epoch	20

Table 3: Reccurent Neural Network (RNN) Model Structure.

The process of the method is resumed in Fig. 14. First, there is a learning phase which aims to build a model of the system based on a set of observations. Obviously, the way of building the model depends on the formalism given. Then, given an observation and the model chosen, the likelihood of this observation is given.



(a) PFA: Variance = 1 (b) PFA: Variance = 5 (c) PFA: Variance = 15

(d) HMM: Variance = 1 (e) HMM: Variance = 5 (f) HMM: Variance = 15

(g) RNN: Variance = 1 (h) RNN: Variance = 5 (i) RNN: Variance = 15

Figure 13: Comparison of the normalised-likelihood of a nominal sequence compared to an attacking one using the three formalisms proposed.

Learning Phase

For the learning phase, we considered a 5000 draws according to the distribution described just above. The structure of the PFA is first determined and the transition probability is computed as in [73]. Here, we assumed that our PFA is deterministic and so to determine the probability, $P(\delta)$, of each transition we will count the number, $N(\delta)$, of times the edge is fired.

$$P(\delta) = \frac{N(\delta)}{\Sigma_{\delta' \in \Sigma_A | \delta'[1]=\delta[1]} N(\delta')} \tag{15}$$

Regarding the initial location, we assume that there is an equal probability for each event. The Baum-Welch algorithm is used to determine the parameter of the HMM. Regarding RNN, the model is trained according to the parameter given in Table. 3

Running Phase

Now, we draw a sequence of 125 planes following the same principle to constitute the nominal sequence. For the attack sequence, a draw of 125 planes is done using a uniform distribution. The objective is to set up different distribution during the nominal and the attack phase. Thus, the process consists into the observation of a plan and the analysis of its likelihood in order to determine whether it is consistent or not.

The experiment has been run for three different variances: $C \sim \mathcal{P}(1)$, $C \sim \mathcal{P}(5)$ and $C \sim \mathcal{P}(15)$. The result of the experiments are given in Fig. 13. Three thresholds have been applied: $TD = 0.4$, 0.5 and 0.6. The curves are divided

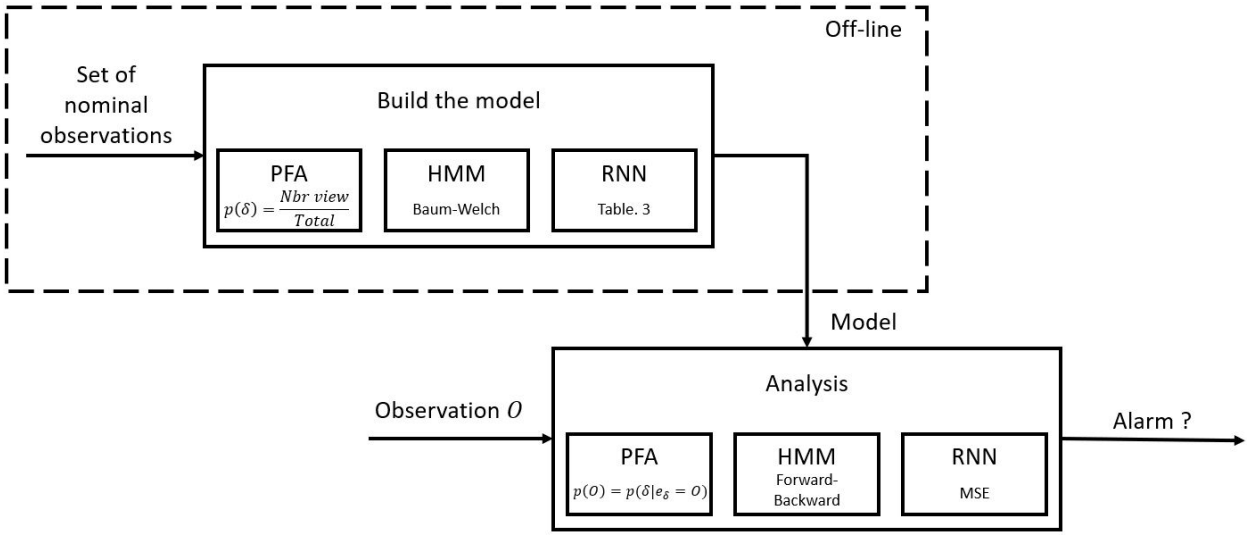


Figure 14: Resume of the process to map the case study.

in two parts: at the left of the vertical line there is the nominal case, no attack while at the right it the abnormal case. The alarms are represented by a particular point (depending on which threshold raise the alarm) on Fig. 13

Firstly, it can be observed for PFA and HMM that the distinction between the attack and the nominal case is less clear as soon as the variance increased. Indeed, the aim of the method is to detect any inconsistency in the control. However, when the variance increases, the difference between the two distributions is not clear. Hence, the PFA and HMM tend to confuse the two distributions. Regarding PFA, the offset between the likelihood in an attack case and in a nominal case decreases with the increase of the variance: the offest is clear for Fig. 13a, around 0.8 in the nominal case instead of 0.3 during the attack. But, whatever the case, the curves tend to 0.5 which means that the observer tends to the indecision and is not able to dissociate the attack to the nominal behavior as it can be observed on Fig. 13c.

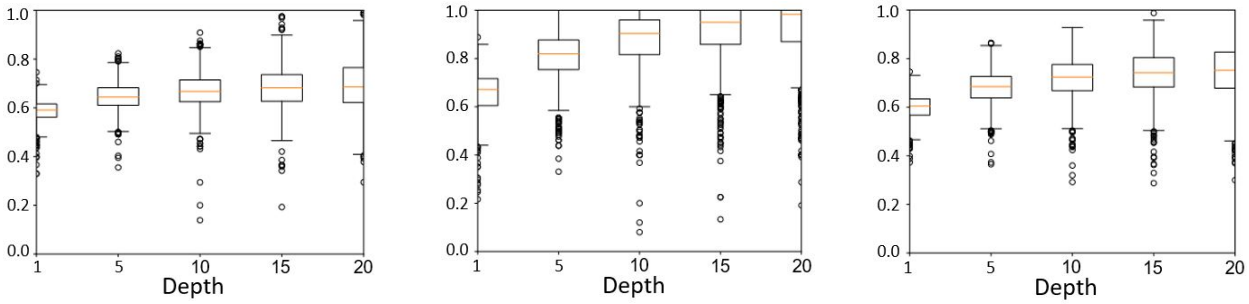
This is similar for the likelihood score determined using HMM as seen in Fig. 13d, around 0.9 during the nominal case while 0.4 during the attack. But, on Fig. 13f, the offset is smaller, around 0.7 during the nominal case while 0.4 during the attack.

However, this phenomenon is not observed with the RNN approach. This is due to the computation power of RNN compared to the two others, RNN is less sensitive to the increase of the variance. Also, as non-determinist algorithm is considered, it is possible to observe sequences which are unusual but perfectly consistent and this is the reason why we consider sequences of events and not just one. The longer the sequence, the more accurate our likelihood will be. But, regarding HMM the length of the sequence is limited due to the time complexity (which also has an impact on the reactivity of the observer). It explains the difference between HMM and RNN.

8.3. Impact of the depth of the sequence

It is clear that the *depth* of the sequence W used to determine the likelihood score has an impact on the results. For example, in the case of $depth = 1$, each plane is plausible as far as it has already been observed. Indeed, even if it is rare, it is still possible in a non-deterministic context. This means that this is not an issue to observe it again and therefore it is important to consider deeper sequences. To evaluate the impact of the *depth* of the sequence on the results we will consider just one formalism, PFA, and one variance for the distribution of the control, $V = 5$. The metrics has been evaluated for $depth = 1, 5, 10, 15, 20$ and the results of the metrics depending on the *depth* of the sequences are given in Fig. 15. The threshold used is $TD = 0.5$.

Considering greater depth makes the analysis more accurate. Even if the observed plan is consistent and one of the most frequently seen, the sequence does not correspond to the expected distribution. Increasing the *depth* of the sequence permits to reduce the proportion of the plausible sequences in the set of observable sequences. As a result, the distinction between the nominal and abnormal sequence is clearer as far as the *depth* increases which leads to



(a) Precision (b) Recall (c) Accuracy

Figure 15: Results depending on the *depth* of the sequence.

reduce the number of false positive and false negative. Hence, the greater the depth of the sequence considered by the observer is, the more complicated it will be for an attacker to set up a sequence of data plane that could damage the network without being considered as inconsistent. Finally, deeper sequence permits to distinguish the abnormal sequence from the nominal ones. However, considering high values of the *depth* has an impact on the tolerance on unusual decisions of the controller. It explains the evolution of the disparity of the precision on Fig. 15a. Also, it has to be mentioned that considering deeper sequence has an impact on the time computation as soon as the calculation are multiplied as soon as the *depth* increases. Regarding PFA and RNN the complexity of the time computation is $O(depth)$ while for HMM this is $O(N^{depth})$. Hence, HMM is sensitive to an increase of the *depth* in terms of reactivity while it is not the case for PFA and RNN.

Nevertheless, it has to be mentioned that the increase of the *depth* is limited to the consideration of the context of the network. Indeed considering $depth = 20$ may lead to include decisions which were not taken in the same context.

8.4. Comparison of the three formalisms

The experiments have been run 1000 times and the moustache diagram of the precision, recall and accuracy are given respectively in Fig. 16, Fig. 17 and Fig. 18. The threshold used is $TD = 0.5$.

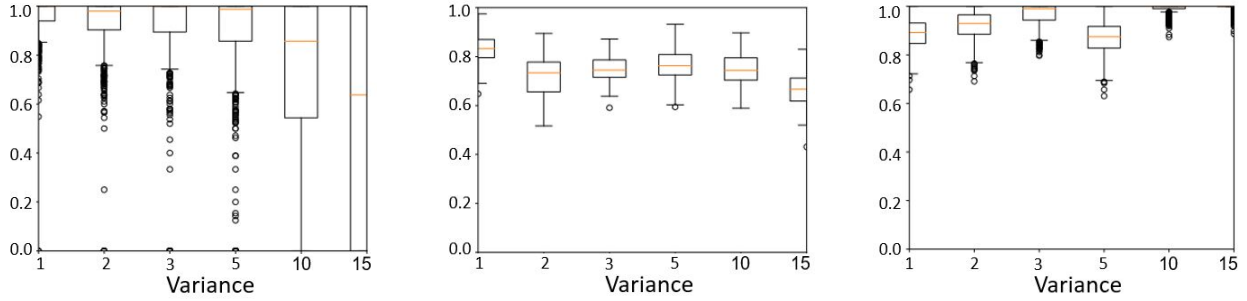
The depth used is 3 for HMM (due to time computation limitation as presented in the previous section) and 10 for PFA and RNN.

Regarding the precision, it is clear that the variance has an impact on the PFA formalism while not for HMM and RNN. Indeed, increasing the variance has an impact on the model by harmonising all the transition probability and so the PFA cannot distinguish an unusual sequence to a usual one. This is due to the fact that the gap between the usual event and the unusual ones decreased significantly as soon as the variance of the distribution increases. Then, the evolution of the normalised likelihood decreases as it can be seen by comparison of Fig. 13a and Fig. 13c. Still, this impact can be limited by increasing the *depth* of the considered sequence as presented in the previous section.

HMM and RNN have a constant precision over the variance even if the RNN has better performance. This difference is due to the approaches of the formalism which has an impact on the power of computation for RNN and HMM. Due to the discrete approach proposed by HMM, the number of intern states is limited due to the computation complexity of the algorithms used (the complexity of the forward used algorithm to determine the likelihood, is $O(N^T)$ with N the number of states and T the depth of the sequence of observation) while the continuous approach of RNN permits to deal with more intern states. The risk then with RNN is the over-learning.

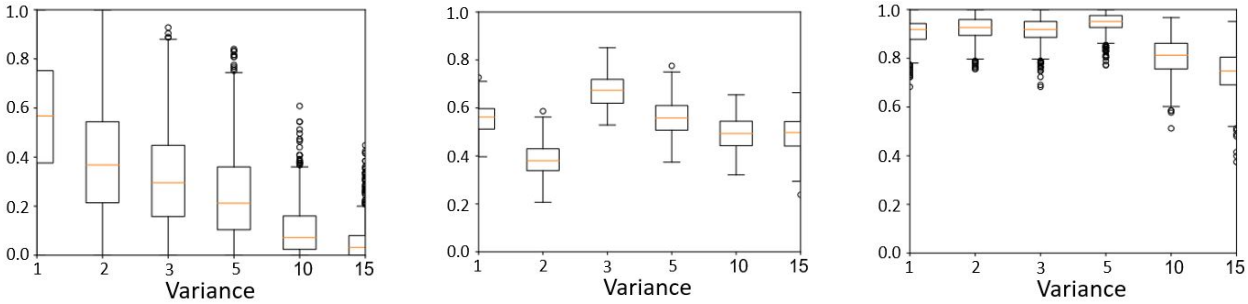
However, regarding the recall, there is a significant difference between HMM and RNN. The difficulty for HMM is that the *depth* of the sequence is limited to $depth = 3$. This means that the model considers just the likelihood of the last 3 decisions. But a random choice (in case of attack) might be ”close” to a Poisson ones (in the nominal case), especially when the variance increases. Hence, the difference between an attacked and a nominal sequence is thinner by considering just $depth = 3$ instead of $depth = 10$ for RNN. This explains the low performance for HMM.

To conclude each formalism has its benefits and its weakness but the choice depends on the considered control. We proposed to classify the non-deterministic algorithm according to the variance of their distribution. In case of a controller which does not vary a lot in its decisions (variance of the distribution < 5), PFA is well designated to detect anomalies in the control. However, this formalism is sensitive to the highest values of the variance. HMM and RNN



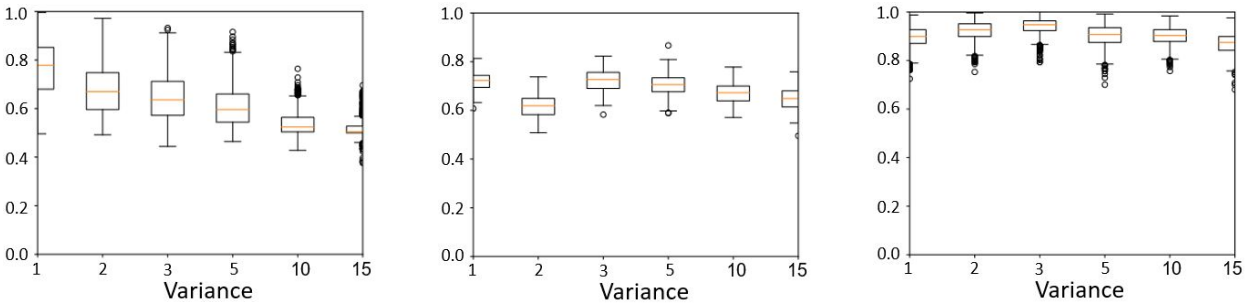
(a) PFA (b) HMM (c) RNN

Figure 16: Precision.



(a) PFA (b) HMM (c) RNN

Figure 17: Recall.



(a) PFA (b) HMM (c) RNN

Figure 18: Accuracy.

are stable regarding the variance even if RNN is more precise and accurate. The main difference between these two formalisms concerns the approaches: discrete or continuous. The RNN permits to multiply the intern states while the power of computation for HMM is limited. However, this multiplication permits to be more accurate but there is the risk of over-learning as described above. Thus, the final model has to be a compromise sequence between the accuracy needed depending on the variance of the controller.

9. Conclusion

This paper focuses on a detection method of the threats on the control plane in a SDN architecture. In this objective, a multi controller architecture without East-West interface is considered. It relies on an observer checking the activity

of the controller. The absence of East-West interface permits to avoid the threats related to the communication between the observer and the main controller.

This work extends existing logic for simple algorithms. It tackles non-deterministic control such as the ones based on machine-learning algorithms. Hence, a consistent data plane might be the result of a malicious controller. Hence, we proposed two paradigms to evaluate the likelihood of the decisions: focusing on the performance of the decisions or on the decisions itself. It has been demonstrated on a case study that the detection is working and the two criteria presented have some limitations which have been discussed. Mainly regarding the criterion 2 and the three formalisms introduced: PFA, HMM and RNN. PFA is the simplest one but is limited face to control with a variance $V > 5$ while HMM not but is limited in terms of performance due to the time complexity of the forward algorithm. Finally, RNN has the best performance (compared to PFA and HMM) but is more difficult to set up and find the correct configuration.

In future works, we plan to extend the detection approach in order to consider observability issues and so finding what are the minimum information needed to consider a plausible decision. Moreover, another perspective is the extension of the detection method in the context of an encrypted communication between the controller and the switches.

Acknowledgment

This work was supported in part by the French PIA project “Lorraine Université d’Excellence”, reference ANR-15-IDEX-04-LUE.

References

[1] H. Farhady, H. Lee, A. Nakao, Software-defined networking: A survey, Computer Networks 81 (2015) 79–95.
[2] S. Scott-Hayward, G. O’Callaghan, S. Sezer, Sdn security: A survey, in: 2013 IEEE SDN For Future Networks and Services (SDN4FNS), IEEE, 2013, pp. 1–7.
[3] I. Alsmadi, D. Xu, Security of software defined networks: A survey, Computers & security 53 (2015) 79–108.
[4] S. Lee, J. Kim, S. Woo, C. Yoon, S. Scott-Hayward, V. Yegneswaran, P. Porras, S. Shin, A comprehensive security assessment framework for software-defined networks, Computers & Security 91 (2020) 101720.
[5] T. Alharbi, S. Layeghy, M. Portmann, Experimental evaluation of the impact of dos attacks in sdn, in: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2017, pp. 1–6.
[6] D. Li, S. Wang, K. Zhu, S. Xia, A survey of network update in sdn, Frontiers of Computer Science 11 (1) (2017) 4–12.
[7] Y. Zhang, L. Cui, W. Wang, Y. Zhang, A survey on software defined networking with multiple controllers, Journal of Network and Computer Applications 103 (2018) 101–118.
[8] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, C. M. Machuca, Characterization of failure dynamics in sdn controllers, in: 2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM), IEEE, 2017, pp. 1–7.
[9] P. Fonseca, R. Bennesby, E. Mota, A. Passito, A replication component for resilient openflow-based networking, in: 2012 IEEE Network operations and management symposium, IEEE, 2012, pp. 933–939.
[10] C. Qi, J. Wu, H. Hu, G. Cheng, W. Liu, J. Ai, C. Yang, An intensive security architecture with multi-controller for sdn, in: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2016, pp. 401–402.
[11] A. Derhab, M. Guerroumi, M. Belaoued, O. Cheikhrouhou, Bmc-sdn: blockchain-based multicontroller architecture for secure software-defined networks, Wireless Communications and Mobile Computing 2021 (2021).
[12] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking, Computers & Security 88 (2020) 101629.
[13] D. Kreutz, F. M. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, 2013, pp. 55–60.
[14] F. Shang, Y. Li, Q. Fu, W. Wang, J. Feng, L. He, Distributed controllers multi-granularity security communication mechanism for software-defined networking, Computers & Electrical Engineering 66 (2018) 388–406.
[15] L. Desgeorges, J.-P. Georges, T. Divoux, A technique to monitor threats in sdn data plane computation, in: 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), IEEE, 2021, pp. 1–6.
[16] T. O. Ayodele, Types of machine learning algorithms, New advances in machine learning 3 (2010) 19–48.
[17] B. Mahesh, Machine learning algorithms-a review, International Journal of Science and Research (IJSR).[Internet] 9 (2020) 381–386.
[18] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, H. Han, A systematic literature review of methods and datasets for anomaly-based network intrusion detection, Computers & Security (2022) 102675.
[19] A. L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Communications surveys & tutorials 18 (2) (2015) 1153–1176.
[20] A. Ahmad, E. Harjula, M. Ylianttila, I. Ahmad, Evaluation of machine learning techniques for security in sdn, in: 2020 IEEE Globecom Workshops (GC Wkshps, IEEE, 2020, pp. 1–6.
[21] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.

[22] B. Ingre, A. Yadav, A. K. Soni, Decision tree based intrusion detection system for nsl-kdd dataset, in: International conference on information and communication technology for intelligent systems, Springer, 2017, pp. 207–218.
[23] S. K. Dey, M. M. Rahman, M. R. Uddin, Detection of flow based anomaly in openflow controller: Machine learning approach in software defined networking, in: 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), IEEE, 2018, pp. 416–421.
[24] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.
[25] P. Hadem, D. K. Saikia, S. Moulik, An sdn-based intrusion detection system using svm with selective logging for ip traceback, Computer Networks 191 (2021) 108015.
[26] P. Domingos, M. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Machine learning 29 (2) (1997) 103–130.
[27] L. E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state markov chains, The annals of mathematical statistics 37 (6) (1966) 1554–1563.
[28] N. Devarakonda, S. Pamidi, V. V. Kumari, A. Govardhan, Intrusion detection system using bayesian network and hidden markov model, Procedia Technology 4 (2012) 506–514.
[29] C.-M. Chen, D.-J. Guan, Y.-Z. Huang, Y.-H. Ou, Anomaly network intrusion detection using hidden markov model, Int. J. Innov. Comput. Inform. Control 12 (2016) 569–580.
[30] P. Holgado, V. A. Villagrá, L. Vazquez, Real-time multistep attack prediction based on hidden markov models, IEEE Transactions on Dependable and Secure Computing 17 (1) (2017) 134–147.
[31] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
[32] S. Nayyar, S. Arora, M. Singh, Recurrent neural network based intrusion detection system, in: 2020 International Conference on Communication and Signal Processing (ICCSP), IEEE, 2020, pp. 0136–0140.
[33] Y. Zhu, L. Cui, Z. Ding, L. Li, Y. Liu, Z. Hao, Black box attack and network intrusion detection using machine learning for malicious traffic, Computers & Security 123 (2022) 102922.
[34] S. K. Dey, M. M. Rahman, Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method, in: 2018 4th international conference on electrical engineering and information & communication technology (iCEEiCT), IEEE, 2018, pp. 630–635.
[35] T. Hurley, J. E. Perdomo, A. Perez-Pons, Hmm-based intrusion detection system for software defined networking, in: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2016, pp. 617–621.
[36] M. Nobakht, V. Sivaraman, R. Boreli, A host-based intrusion detection and mitigation framework for smart home iot using openflow, in: 2016 11th International conference on availability, reliability and security (ARES), IEEE, 2016, pp. 147–156.
[37] W. Wang, X. Ke, L. Wang, A hmm-r approach to detect l-ddos attack adaptively on sdn controller, Future Internet 10 (9) (2018) 83.
[38] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, B. Yang, Predicting network attack patterns in sdn using machine learning approach, in: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2016, pp. 167–172.
[39] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, Deep recurrent neural network for intrusion detection in sdn-based networks, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 202–206.
[40] Z. Din, J. de Oliveira, Anomaly free on demand stateful software defined firewalling, in: 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2017, pp. 1–9.
[41] R. F. Fouladi, O. Ermiş, E. Anarim, A novel approach for distributed denial of service defense using continuous wavelet transform and convolutional neural network for software-defined network, Computers & Security 112 (2022) 102524.
[42] Q. Niyaz, W. Sun, A. Y. Javaid, A deep learning based ddos detection system in software-defined networking (sdn), arXiv preprint arXiv:1611.07400 (2016).
[43] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, L. Gong, Detection and defense of ddos attack–based on deep learning in openflow-based sdn, International Journal of Communication Systems 31 (5) (2018) e3497.
[44] S. K. Dey, R. Uddin, M. Rahman, et al., Performance analysis of sdn-based intrusion detection model with feature selection approach, in: Proceedings of international joint conference on computational intelligence, Springer, 2020, pp. 483–494.
[45] S. K. Dey, M. M. Rahman, Effects of machine learning approach in flow-based anomaly detection on software-defined networking, Symmetry 12 (1) (2019) 7.
[46] H. Yang, Y. Liang, J. Yuan, Q. Yao, A. Yu, J. Zhang, Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5g and beyond, IEEE Transactions on Industrial Informatics 16 (11) (2020) 7094–7104.
[47] Y. Wang, T. Hu, G. Tang, J. Xie, J. Lu, Sgs: Safe-guard scheme for protecting control plane against ddos attacks in software-defined networking, IEEE Access 7 (2019) 34699–34710.
[48] M. F. Hyder, M. A. Ismail, Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches, IEEE Access 9 (2021) 21881–21894.
[49] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, Y. Liu, A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges, IEEE Communications Surveys & Tutorials 21 (1) (2018) 393–430.
[50] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O. M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, Journal of Internet Services and Applications 9 (1) (2018) 1–99.
[51] A. Abubakar, B. Pranggono, Machine learning based intrusion detection system for software defined networks, in: 2017 seventh international conference on emerging security technologies (EST), IEEE, 2017, pp. 138–143.
[52] J. A. Herrera, J. E. Camargo, A survey on machine learning applications for software defined network security, in: International Conference on Applied Cryptography and Network Security, Springer, 2019, pp. 70–93.
[53] S. Prabakaran, R. Ramar, I. Hussain, B. P. Kavin, S. S. Alshamrani, A. S. AlGhamdi, A. Alshehri, Predicting attack pattern via machine learning by exploiting stateful firewall as virtual network function in an sdn network, Sensors 22 (3) (2022) 709.
[54] D. Breuker, M. Matzner, P. Delfmann, J. Becker, Comprehensible predictive models for business processes, Mis Quarterly 40 (4) (2016) 1009–1034.
[55] A. Salaün, Y. Petetin, F. Desbouvries, Comparing the modeling powers of rnn and hmm, in: 2019 18th IEEE International Conference On

Machine Learning And Applications (ICMLA), IEEE, 2019, pp. 1496–1499.
[56] ONF, OpenFlow Specification v1.3 https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf, last visited the 14/09/2021 (June, 2012).
[57] D. M. Casas-Velasco, O. M. C. Rendon, N. L. da Fonseca, Intelligent routing based on reinforcement learning for software-defined networking, IEEE Transactions on Network and Service Management 18 (1) (2020) 870–881.
[58] O. Cappé, E. Moulines, T. Rydén, Inference in hidden markov models, in: Proceedings of EUSFLAT conference, 2009, pp. 14–16.
[59] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
[60] G. D. Forney, The viterbi algorithm, Proceedings of the IEEE 61 (3) (1973) 268–278.
[61] A. Viterbi, Convolutional codes and their performance in communication systems, IEEE Transactions on Communication Technology 19 (5) (1971) 751–772.
[62] L. E. Baum, T. Petrie, G. Soules, N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains, The annals of mathematical statistics 41 (1) (1970) 164–171.
[63] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern recognition 36 (2) (2003) 451–461.
[64] D. M. Casas-Velasco, https://github.com/danielaCasasv/RSIR-Reinforcement-Learning-and-SDN-Intelligent-Routing.git , last visited the 10/01/2022 (2020).
[65] R. P. Team, "Ryu application API," Available: https://ryu.readthedocs.io/en/latest/ryu_app_api.html , last visited the 10/01/2022 (2012).
[66] TOTEM, https://totem.info.ucl.ac.be/index.html , last visited the 10/01/2022 (January, 2006).
[67] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, P. A. Porras, Delta: A security assessment framework for software-defined networks., in: NDSS, 2017.
[68] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, J. M. Arco, A survey on machine learning techniques for routing optimization in sdn, IEEE Access (2021).
[69] A. Azzouni, R. Boutaba, G. Pujolle, Neuroute: Predictive dynamic routing for software-defined networks, in: 2017 13th International Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–6.
[70] Y.-J. Wu, P.-C. Hwang, W.-S. Hwang, M.-H. Cheng, Artificial intelligence enabled routing in software defined networking, Applied Sciences 10 (18) (2020) 6564.
[71] S. Kumar, G. Bansal, V. S. Shekhawat, A machine learning approach for traffic flow provisioning in software defined networks, in: 2020 International Conference on Information Networking (ICOIN), IEEE, 2020, pp. 602–607.
[72] S.-C. Lin, I. F. Akyildiz, P. Wang, M. Luo, Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach, in: 2016 IEEE International Conference on Services Computing (SCC), IEEE, 2016, pp. 25–33.
[73] K. Fouquet, G. Faraut, J.-J. Lesage, Life habits modeling with stochastic timed automata in ambient assisted living, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2020, pp. 2740–2745.