Configuration faults detection in IP Virtual Private Networks based on machine learning

El-Heithem Mohammedi, Emmanuel Lavinal, Guillaume Fleury

Configuration faults detection in IP Virtual Private Networks based on machine learning

El-Heithem Mohammedi[1,2], Emmanuel Lavinal[1], and Guillaume Fleury[2]

[1] IRIT, University of Toulouse, France
firstname.lastname@irit.fr
[2] IMS Networks, France
firstinitialLastname@imsnetworks.com

Abstract. Network incidents are largely due to configuration errors, particularly within network service providers who manage large complex networks. Such providers offer virtual private networks to their customers to interconnect their remote sites and provide Internet access. The growing demand for virtual private networks leads service providers to search for novel scalable approaches to locate incidents arising from configuration faults. In this paper, we propose a machine learning approach that aims to locate customer connectivity issues coming from configurations errors, in a BGP/MPLS IP virtual private network architecture. We feed the learning model with valid and faulty configuration data and train it using three algorithms: decision tree, random forest and multi-layer perceptron. Since failures can occur on several routers, we consider the learning problem as a supervised multi-label classification problem, where each customer router is represented by a unique label. We carry out our experiments on three network sizes containing different types of configuration errors. Results show that multi-layer perceptron has a better accuracy in detecting faults than the other algorithms, making it a potential candidate to validate offline network configurations before online deployment.

Keywords: Configuration Faults Detection · Machine Learning · Virtual Private Networks · BGP/MPLS Networks.

1 Introduction

Demands from companies to interconnect their remote sites and provide them with Internet access have continued to increase strongly in recent years. BGP/MPLS IP Virtual Private Networks (VPNs) [15] remain the most reliable and widely used technology for this purpose. It can also be complementary to SD-WAN solutions (Software-Defined Networking in a Wide Area Network) which can be used as an overlay technology to optimize WAN edge infrastructures.

Within network service providers, VPNs multiplication leads to an increase in the configuration's complexity, which can cause several faults that impact the reachability of the customers' sites. Traditional methods to detect network

configuration faults generally require to specify a list of formal constraints to verify the configuration's validity (e.g., checking type correctness or cross-elements dependencies). Ensuring the constraints completeness and manually updating them are well known issues in this context. In this paper, we propose another approach based on machine learning to detect and locate connectivity incidents related to configuration faults in BGP/MPLS IP VPNs. Although relying on machine learning for network management is not new [20], to the best of our knowledge, it has never been applied for configuration fault detection in the context of IP VPNs.

The approach we propose relies on a supervised learning paradigm in which a model is trained thanks to thousands of network configurations that are labeled as being either valid or not. This classification problem targets specifically the reachability state of a customer's site. After the learning process, the goal is to be able to identify if an unknown configuration provided as an input allows all the customers' sites in a VPN to connect to each other and, if it is not the case, the system should locate the sites that are not reachable. We validated our work by testing the learning model on three algorithms: decision tree, random forest and multi-layer perceptron. On a network of 100 client routers, results show that we obtain an accuracy in detecting faults between 60 and 80 percent depending on the learning algorithm.

This paper is organized as follows. In Section 2, we briefly review previous related work. In Section 3, we present BGP/MPLS IP VPNs, including their main configuration elements and possible errors. We formulate the learning problem in Section 4 and we explain our approach of data collection and feature engineering in Section 5. Section 6 provides experimental tests and evaluation results before concluding the paper in Section 7.

2 Related work

A lot of work on faults localization in the field of computer networks has been done for more than fifteen years. Various techniques have been used such as rule-based expert systems, decentralized probabilistic management or temporal correlation [6]. Other proposals address network configuration verification [3] or data-plane verification such as Header Space Analysis [9] and Veriflow [11] that verify if safety properties hold for the current network state. These approaches present scalability issues and they require that the verified configuration has been deployed in a live network.

The increase in the size and complexity of networks makes fault detection and localization a difficult task that requires new scalable approaches, which inspired researchers to apply machine learning methods for this purpose. Some of these works target specific networks such as cellular or wireless sensor networks [10,12]. Others are more general, such as [8] in which the authors proposed a supervised learning model for incident management. Their model is composed of two sub-models: a classification model to categorize network incidents and a regression model to predict the duration of incidents, both are fed by data collected from

application databases related to incident handling processes (customer relationship management, network monitoring system, etc.). However, configuration errors are not taken into account. In [19], a passive monitoring approach is used to identify and localize the link where the fault occurred. The authors proposed to capture the loss rate, the end-to-end delay and the aggregated transmission rate for each source/destination node from traffic in both normal working conditions and failure scenarios. The centralized network monitor then transmits these metrics as features to the machine learning based fault manager. Again, the data comes from network state and not configuration.

DeepBGP [2] is an example of related work that targets network configurations. The authors rely on a Graph Neural Network (GNN) model to generate BGP configurations given the feedback from a validation unit (enhanced by an Evolution Strategies optimizer) to train the neural network. Although this paper presents a very interesting approach (such as the use of a GNN that we plan to study in future work), it does not focus on configuration faults detection, but on network configuration generation.

Compared to these related work, we propose to identify and locate incidents caused by configuration faults. We rely on a supervised learning approach, looking at the network configuration as a whole (as opposed to individual devices), which allows us to detect failures on multiple devices, even if there is a single configuration error. Another specificity is the application domain: we focus on connectivity incidents within network service providers that offer L3 virtual private networks to their customers.

3 BGP/MPLS IP Virtual Private Networks

IMS Networks is an Internet Service Provider (ISP) that offers, among other services, IP Virtual Private Networks (VPNs) to its customers [15]. This service allows the operator to interconnect a set of customer sites through its backbone while guaranteeing quality of service and isolation of flows for each customer. The operator's backbone relies on a Multiprotocol Label Switching (MPLS) architecture [16] that assigns a label for each route within a VPN. A customer data packet is tagged with the label corresponding, in the customer's VPN, to the route that is the best match to the packet's destination and is further encapsulated with another MPLS label used to tunnel the packet across the backbone. The routes for a particular VPN are exchanged thanks to the Multi-Protocol Border Gateway Protocol (MP-BGP). Fig. 1 illustrates a simple BGP/MPLS IP VPN architecture composed of two customers (A and B), each having two sites. Each VPN site must contain one Customer Edge (CE) router attached to one or more Provider Edge (PE) routers :

- A CE router is connected to the provider's network via an access service. It routes traffic between the customer's site and the backbone using a routing protocol such as eBGP (external BGP) or OSPF (Open Shortest Path First).
- A PE router is an edge backbone router to which CE devices are connected to. It contains a VPN Routing and Forwarding table (VRF) for each VPN.

The routes within a VRF are learned from the CE it is attached to, as well as from other PEs via the MP-BGP protocol. The PE router also acts as an ingress/egress label edge router for the MPLS domain.

Routers in the provider's network that are not attached to CE devices are known as "P routers". They implement primarily MPLS forwarding and control protocols.

Configuring a complete BGP/MPLS IP VPN architecture is a complex task as it involves configuring many parameters and many types of protocols. Moreover, the size of the network can also be an issue since a network service provider can have a large amount of customers deployed on a number of sites ranging from a few to several hundreds. In the next subsections, we will explain in more depth the main configuration steps that are required to deploy and run this VPN service and the main faults that can occur in this process.

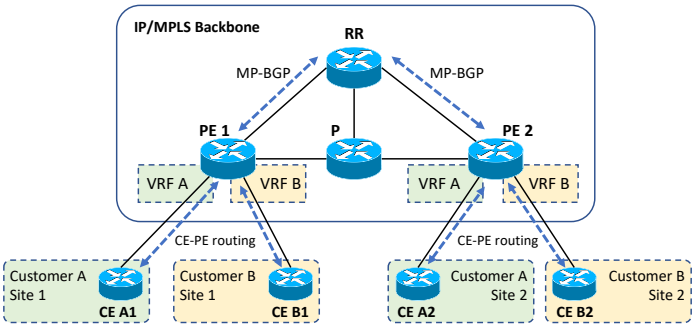

Fig. 1. BGP/MPLS IP VPN architecture overview.

3.1 BGP/MPLS IP VPNs configuration

To implement BGP/MPLS IP VPNs, it is necessary to go through two main stages: first, the configuration of the provider's backbone and second, the configuration of the customers' VPNs.

Backbone configuration Once the topology and all the routers' interfaces are configured, an Interior Gateway Protocol (IGP) must be configured between all P and PE routers (such as OSPF). Next, MPLS forwarding must be activated, as well as the Label Distribution Protocol (LDP) to distribute labels between neighbors in order to establish label switched paths. Finally, MP-BGP must be configured between all PE routers so that they can afterwards advertise customer routes. To avoid scalability issues (in which every PE router peers with every other PE router in a full mesh), an alternative is to use a route reflector (RR) to centralize MP-BGP sessions (cf. Fig. 1).

Once the operator's backbone is configured, it is generally stable and rarely changes.

VPNs configuration When configuring a VPN instance for a specific customer, a VRF must be configured on every PE router connected to that customer's site. This is done, notably, by specifying a route distinguisher (RD) to ensure route uniqueness, along with route targets (RT) to determine which VRFs it should export or import routes to or from. The VRF must also be associated to the appropriate interface on which the CE is connected to. Regarding CE-PE routing, we can use eBGP, thereby configuring a direct peering between CE and PE routers (for a specific VRF). Otherwise, another routing protocol can be used (e.g., OSPF) and, in this case, route redistribution between the MP-BGP protocol and this other routing protocol must be configured for that VRF.

It should be noted that, contrary to the backbone's configuration, several customer sites (CE routers) can be added, updated or deleted each day, which requires human expertise to always check the changes before committing the new configuration. Besides, adding a single customer's site to the network requires almost one hundred lines of configuration on the CE and PE routers, which is a very error-prone task.

3.2 BGP/MPLS L3 VPNs incidents

Within the Network Operation Center (NOC) at *IMS Networks*, we have observed that the majority of network incidents are due to configuration faults, hence the interest of this work. In this article, we consider several types of faults that impact the reachability of CE routers and therefore lead to VPNs disruptions.

Faults on CE-PE routing An error on a customer's IP subnet on a CE or misconfiguring the CE-PE routing (e.g., eBGP peering or route redistribution) are examples of faults that make a CE unreachable to all other CEs of the VPN, even if they are connected to the same PE.

Faults on VRFs A network engineer can forget to configure a VRF instance on a PE router or can make a mistake on a specific parameter such as the RD or RT. Once the configuration has been committed with this type of fault, all the CE routers of the concerned customer connected to this PE router will be unreachable.

Faults on MP-BGP A peering configuration fault between a PE router and the route reflector will prevent establishing the MP-BGP session and therefore the PE will neither be able to send nor receive VPN routes. All the CEs connected to this PE will therefore be unreachable. It is also necessary to configure a BGP instance for each VRF. Forgetting this step will prevent the PE router to communicate the customer's routes to the route reflector and thus all the CE routers of that customer will be unreachable.

Table 1 summarizes these configuration faults as well as their impact on the reachability of the customers' sites.

Table 1. Main configuration faults and their impact

Faults		**Affected CEs**
a) CE-PE routing config. faults	Routing error between CE i and the PE	CE i
	CE i subnet misconfigured	CE i
b) VRFs configuration faults	VRF j not configured on PE i	All CEs of customer j connected to PE i
	RD/RT misconfigured for VRF j on PE i	All CEs of customer j connected to PE i
c) MP-BGP configuration faults	BGP instance not configured for VRF j on PE i	All CEs of customer j connected to PE i
	MP-BGP peering error between PE i and the RR	All CEs that are connected to PE i

4 Problem formulation

In this section, we formulate the learning problem and illustrate it on a simple example.

The general goal of our work is to build a system that can learn if a BGP/MPLS IP VPN configuration is valid or not. More precisely, the system should identify if the target configuration allows every customer's site to be reachable. If not, it should be able to pinpoint the sites that are not reachable due to a configuration error.

Let us consider the topology presented in Fig. 1 and two scenarios in which configuration errors cause reachability issues for one or more CE routers: *i)* if the MP-BGP session between `PE1` and `RR` is not properly configured, then `CE_A1` and `CE_B1` connected to `PE1` will be unreachable regardless of their VRF configuration (case *c.2* in Table 1); *ii)* an incorrect RD or RT configured on VRF B on `PE2` will prevent routes emanating from `CE_B2` to be associated to customer's B VPN and therefore `CE_B2` will be unreachable from the customer's other sites (case *b.2* in Table 1).

Learning if a CE is reachable or not is a *classification problem*; that is predicting whether it belongs to a particular category. In our context, there are only two categories (or classes): reachable or not reachable (in fact the reachability property does not depend only on the network's configuration but also on its state; however we use the term "reachable" for the sake of simplicity). Although there are two classes, this is not a binary classification problem since one configuration includes many CEs and a single fault can generate reachability issues for several CEs. In the above topology, the system should be able to classify if

each CE is reachable or not, hence a vector of 4 labels: [`CE_A1`, `CE_B1`, `CE_A2`, `CE_B2`]. This is known as a multi-label classification task [18] in which an input x is mapped to an output binary vector y (with a value of 0 or 1 for each label in y). Therefore each label represents a particular CE, the value 0 indicating that it should be reachable (according to the network's configuration) and 1 indicating that the CE has a reachability issue. Back to the previous example, the fault scenarios *i)* and *ii)* should generate the binary vectors [`1`, `1`, `0`, `0`] and [`0`, `0`, `0`, `1`], respectively.

Fig. 2 illustrates the general workflow of our approach from data collection and features selection to model validation. After analyzing and selecting the features, we generate a dataset from labeled configurations and feed it to the learning model. We repeat these steps by adjusting features and model parameters until we get decent results. It is important to note that this workflow is done offline before using the verification service. Once the model has been trained, it is deployed, a network engineer submits an L3 VPN configuration and the results should be able to predict if it is correct or not before committing the changes on a production network.

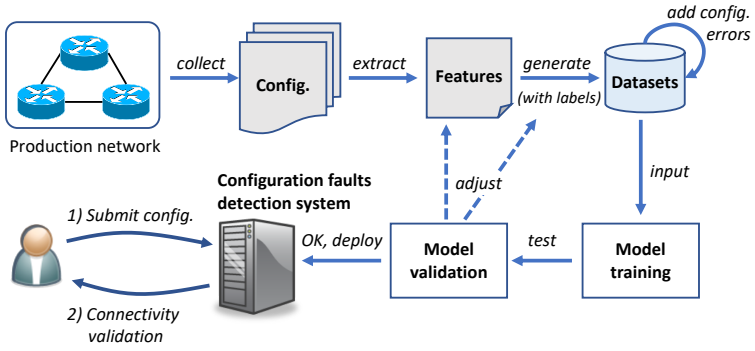

Fig. 2. General workflow from data collection to model deployment.

5 Configuration data collection and analysis

In the learning paradigm, data collection, analysis and feature extraction are very important steps since the efficiency of the learning process is directly correlated to the amount and quality of data. Finding proper features is the key to fully release the potential of data [20].

5.1 Data collection and feature engineering

A complete BGP/MPLS IP VPN configuration is composed of many parts related to physical topology, backbone IGP, MPLS/LDP protocols, VRF tables, MP-BGP sessions and CE-PE routing.

As explained in the previous section, in this work we focus on configuration faults that impact VPN connectivity between the customers' sites. *IMS Networks*' experience in troubleshooting connectivity issues shows that most of the incidents come from routing configuration errors on CE and PE devices, and very few come from the backbone's configuration (IGP, MPLS/LDP). In addition, the backbone's configuration rarely changes while the VPNs' configuration can be updated on a daily basis. Therefore, after having collected and analyzed a complete BGP/MPLS IP VPN configuration, we decided to focus on configuration parameters covering CE-PE routing, VRF tables and MP-BGP. We formalized these parameters as features having either a binary value indicating if the configuration parameter is present or not (e.g., MP-BGP active for VRF v on PE p), or a numerical value with a specific range (e.g., IP prefix with a range from 1 to 32 and RD/RT with a range from 1 to 100 as we assume that the maximum number of VRFs is 100). As an example, nine features in our data model are necessary to add a new VRF with a single CE, which is equivalent to approximately one hundred lines of configuration in the PE router. The selected features have a direct impact on the connectivity issues described in section 3.2. This is an important aspect of feature engineering as the extracted data must be relevant with respect to the problem that is being addressed [4, 20].

Configuration collection and analysis, as well as features extraction constitute the first steps of the overall workflow illustrated in Fig. 2.

5.2 Dataset generation

In communication networks, a large amount of data is available: traffic traces, performance metrics, security alerts, logs, etc. However, network configurations is the exception since once a configuration is in production, there is no need for the service provider to maintain hundreds of other configurations. Additionally, it is very difficult to obtain a large number of configuration errors in a production network and it is unrealistic to inject faults in the network just to have training data. We thus decided to generate configuration data based on existing configurations specified by and deployed at *IMS Networks*. This allows us to have a dataset large enough to be used by the learning model and diverse enough to contain both correct and incorrect configurations.

To be able to generate realistic configurations, we defined three network architectures of different sizes which we used to create three different datasets: *a)* a small network with 5 PEs, 20 CEs and 3 customers; *b)* a medium-sized network with 8 PEs, 50 CEs and 5 customers; and *c)* a larger network with 10 PEs, 100 CEs and 10 customers. In the three networks, CEs are connected to PEs and assigned to customers randomly. According to this random allocation, the features are initialized with either, a value derived from the device or customer

id (e.g., IP address, route distinguisher), or a binary value indicating if the configuration parameter is present or not (e.g., VRF v not configured on PE p). The result is one valid global configuration for each network architecture. Since we rely on a supervised learning paradigm, we have to assign labels to the datasets to establish ground truth. In our context, a label exists for each CE in the network to identify if it is reachable or not (from a configuration point of view). When generating the global configuration, all the labels are initialized to 0 since there is no configuration error.

For each network architecture, in order to train the model with sufficient data, we reproduced the configuration 150000 times. We then generated faults on these configurations (cf. section 3.2). We considered 10 types of faults distributed in 3 categories: CE-PE routing faults (customer's IP subnet address or mask error, eBGP peering error), VRF faults (VRF not configured, bad RD or RT) and MP-BGP faults (BGP not configured, peering error with RR, VPN address family extension not activated, BGP instance not configured for VRF). First, we combined all the faults (CE-PE routing, VRF and MP-BGP faults) within the same dataset and second, we considered the different fault categories independently from each other. Therefore, for each network architecture, we generated several datasets with different categories of faults. For each dataset, the faults are generated by randomly selecting the type of fault and randomly selecting the device on which to apply the fault. During this process we made sure to uniformly distribute the errors over the configurations and that, at the end, each CE would be unreachable in 50% of the data points. Finally, for each fault added to a configuration, we updated the output labels accordingly (i.e., setting the labels of the unreachable CEs to 1).

6 Learning to detect reachability issues

This section covers the two last steps of the workflow illustrated in Fig. 2: model training and validation. We start by explaining the algorithms we relied on to train our model and then we discuss the test results that vary according to the algorithm, network size and fault type.

6.1 ML algorithms and evaluation metrics

In order to implement our learning problem, we have tested three different ML algorithms: decision trees, random forest and multi-layer perceptron. We chose these three algorithms as they are commonly used in the machine learning community when dealing with supervised classification problems.

- Decision Tree (DT) is one of the simplest approaches of supervised learning. During the training, it builds a decision tree which represents a function that takes as input a vector of features values and returns a "decision" (or "class") as output value [17].

- Random Forest (RF) is an extension of the decision tree classifier, it contains a set of individual decision trees that operate as an ensemble. The random forest output class is the most predicted class chosen by each individual decision tree [5].
- Multi-Layer Perceptron (MLP), or feedforward neural network, is a machine learning approach that is based on artificial neural networks. It contains at least three layers of neurons: an input layer, one or more hidden layers and an output layer. Neurons in layer n are connected to neurons in layer $n+1$ with a certain weight. These weights are adjusted during the training process using back propagation [4, 14].

We tested DT and RF using Scikit-learn [13], a Python module that offers implementations of various machine learning algorithms. For MLP, we used Keras [7], a high-level neural networks API built on top of TensorFlow [1].

After training the models, to evaluate their performance, we relied on conventional metrics used in classification problems: accuracy, precision, recall and F1-score. To understand these metrics in our context, we need to define True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) classes. TP and TN represent correctly predicted outcomes for, respectively, positive instances (a CE with an error, i.e. not reachable) and negative instances (a CE with no error, i.e. reachable). On the contrary, FP and FN describe incorrect predictions: predicting a reachability problem that actually does not exist, and predicting a reachable CE that in fact is not.

In the following definitions, we will refer to a *positive CE* as a CE containing an error, i.e. not reachable (labeled 1 in the dataset); and a *negative CE* as a CE with no error, i.e. reachable (labeled 0 in the dataset):

- *Accuracy* represents the proportion of true predictions (i.e. the number of CE whose reachability state is correctly predicted) among the total number of predictions (i.e. the total number of CEs).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision* is the ratio of the number of positive CEs correctly predicted over the total number of CEs predicted as positives.

$$Precision = \frac{TP}{TP + FP}$$

- *Recall* is the ratio of the number of positive CEs correctly predicted over the total number of actual positive CEs.

$$Recall = \frac{TP}{TP + FN}$$

- *F1-score* is the harmonic average of Precision and Recall.

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

6.2 Results and discussion

Our experiments were performed on the datasets described in section 5.2: three network sizes (20, 50 and 100 client routers) and for each one, different types of configuration faults. Each dataset contains 150000 configurations and is divided in two: one for training (containing 70% of the data points), one for testing (containing 30% of the data points).

Training time We first start by measuring the training time of each algorithm for each network size and for different levels of configuration faults (first, VRF tables only; second, adding MP-BGP; third, adding CE-PE routing). The results are shown in Table 2 (experiments were performed in a VM with 4 vCPU and 48 GB of RAM). We can see that the number and type of faults does not have an influence on the training time. This time is fairly constant for the three algorithms whether faults belong to one, two or three categories. Also, the results show that MLP training takes more time than DT and RF (approximately by factors three and two compared to RF for the medium and large networks respectively). However, the total training time for the larger network with the three categories of faults is less than twenty two minutes which is very reasonable for offline training. Indeed, as shown in Fig. 2, the training is done offline before deploying the system, that is before validating any new configuration update on the operational network. Therefore the approach does not have any real time constraints.

Table 2. Comparison of models training time

	Small network			Medium network			Large network		
Faults on:	**DT**	**RF**	**MLP**	**DT**	**RF**	**MLP**	**DT**	**RF**	**MLP**
VRFs	0m 8s	1m 42s	8m 49s	0m 46s	4m 47s	15m 32s	2m 31s	10m 2s	21m 20s
+ MP-BGP	0m 10s	1m 53s	8m 45s	0m 42s	4m 52s	15m 57s	1m 56s	9m 3s	21m 37s
+ CE-PE rt.	0m 16s	2m 31s	8m 29	0m 68s	6m 18s	14m 40s	3m 13s	11m 39s	21m 34s

Overall performance of each algorithm In order to evaluate the general performance of each algorithm, in Fig. 3 and Fig. 4, we plotted the precision, recall, F1-score and accuracy values calculated using the test datasets for the small and large networks, respectively. These results were obtained with the datasets containing all the configuration errors described in section 5.2 (i.e., 10 types of faults).

Overall, we observe that the DT performance is lower than that of RF and MLP, particularly as the network gets larger (F1-score of 60% for the DT on the large network). Besides, MLP results are more efficient than RF results on both graphs, with a bit more on the second one (between RF and MLP, the F1-score increases by 6% on the small network, and by 17% on the larger network).

Looking at precision and recall, we can see that the F1-score on the large network (Fig. 4) cannot reach 80% due to a low recall value (for both RF and

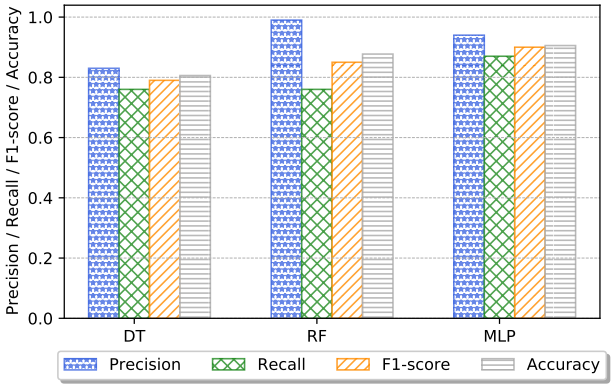

Fig. 3. Precision/Recall/F1-score/Accuracy for the small network dataset.

MLP). A low recall value indicates a high number of false negatives, that is configuration errors that are not detected (i.e., predicting a CE router as being reachable while it is actually not). This issue with the recall value will be further discussed in the subsection related to the impact of configuration error types. On the contrary, the precision is very high (almost 1 for the small network and close to 0.9 for the large network) which means that there are very little false positives, that is a CE router predicted as unreachable is almost always indeed unreachable.

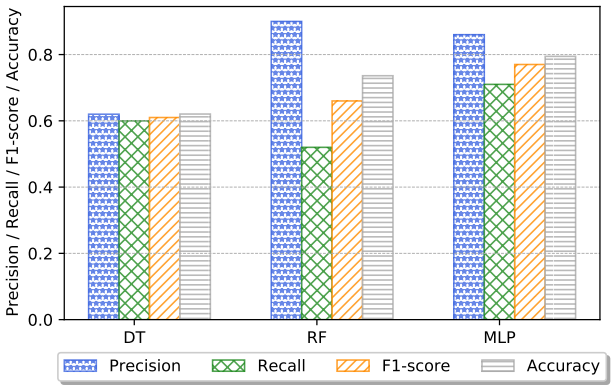

Fig. 4. Precision/Recall/F1-score/Accuracy for the large network dataset.

Impact of network size Fig. 5 shows the F1-score value for the small, medium and large networks with 10 types of configuration faults. It is clear that the performance of all three algorithms (DT, RF and MLP) is decreasing as the network gets larger. Increasing the size of the network means adding more PE and CE devices, more VRFs and thus, in general, more features and labels for each data point. The learning problem gets therefore more difficult, hence a decrease in the algorithms accuracy. However, the problem here is that the number of features and labels increases but the size of the training dataset remains the same. One way to handle this issue if to increase the size of the dataset (i.e., the number of correct and incorrect configurations) as we increase the size of the network. In addition, to improve the MLP's F1-score (which is close to 80% for the larger network), we plan on updating some model parameters such as the number of neural network hidden layers and the number of training epochs; and we can also target specific error types as we will see in the next subsection.

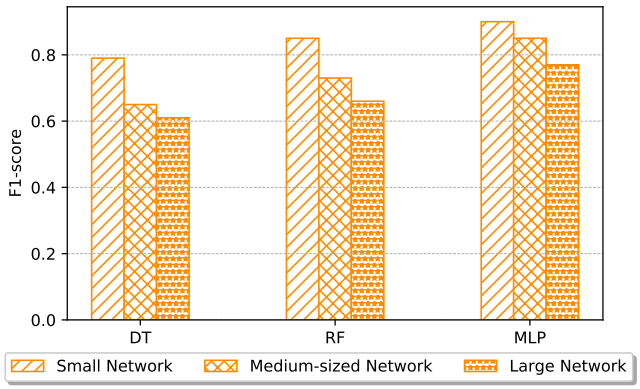

Fig. 5. F1-score for the small, medium and larger network.

Impact of configuration errors To understand more precisely the impact of the configuration error types on the learning algorithms, we calculated the F1-score for each algorithm according to the fault category: CE-PE routing, VRF and MP-BGP (cf. sections 3.2). Results are shown in Fig. 6. One can easily observe that the performance on a dataset containing only MP-BGP errors is better than the performance on a dataset containing only VRF errors which is again better than the performance on a dataset containing only CE-PE routing errors, and this for all three algorithms. This means that models learn better

about MP-BGP and VRF faults than CE-PE routing faults. This is confirmed by Fig. 7 that shows the detailed evaluation metrics for the MLP model.

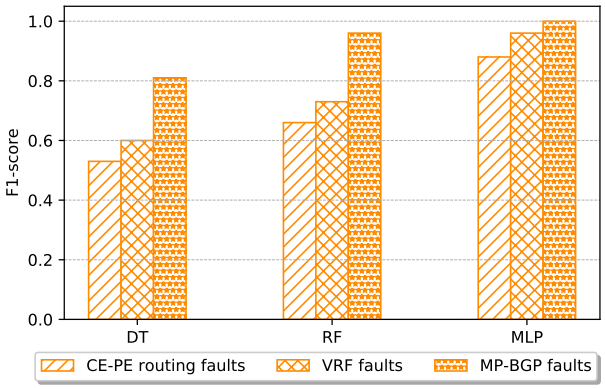

Fig. 6. F1-score for the larger network according to the fault type.

The first reason that can explain this fact is related to the impact of the configuration error on the rest of the network. As presented in Table 1, a fault on MP-BGP peering between a PE and a RR generates reachability issues on all the CEs connected to this PE (i.e., multiple labels), while a fault on a VRF does not impact all the CEs connected to the PE, but only the CEs belonging to the same customer. Finally, a fault on CE-PE routing affects only the concerned CE (i.e., one label). It is therefore easier for the learning algorithms to spot MP-BGP configuration faults and, to a lesser extent, VRF faults.

Other reasons that can explain this difference are the relationships between the features in the dataset and their data types. In CE-PE routing, the IP address and the mask of a customer's subnet are only configured once on the CE router and there is no link between this information and the rest of the configuration. Moreover, these features are numerical data, with a high cardinality (the IP address space) and no numerical relationship between the values. Therefore the learning algorithm cannot identify if these two features (IP address and mask) are correct or not. Regarding the RD/RT configuration parameters in a VRF, although these features are also numerical, they should be identical on the different sites of the same customer. This logical relationship is learned in the training process and therefore an error on those parameters can be detected by the ML algorithm. The recall value plotted in Fig. 7 confirms these assumptions: false negatives are more present in the case of CE-PE routing (i.e., unable to detect an error or the customer's IP subnet) than in the VRF configuration (i.e., better performance in detecting RD/RT errors). MP-BGP configuration

elements are, on the other hand, binary features and have an impact on the reachability of multiple CE routers, hence an accuracy of almost 1 in Fig. 7.

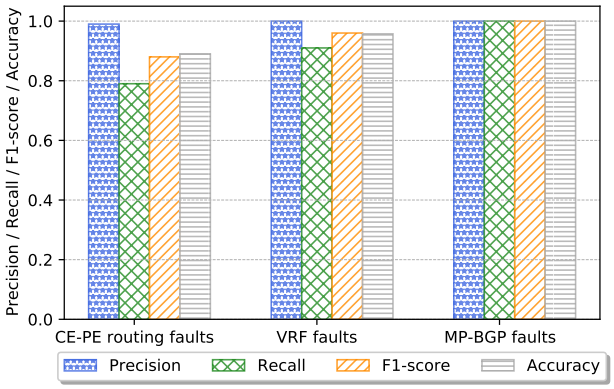

Fig. 7. Precision/Recall/F1-score/Accuracy using MLP for the larger network according to the fault type.

According to these results, we argue that MLP has a better accuracy in detecting configuration faults than DT and RF. Furthermore, we find that the MLP approach can have a much higher performance if it is not used to detect configuration errors on numerical data that have no direct or indirect relationship with other parts of the network configuration (such as customers' IP addresses). This type of configuration data should rather be verified by other means, such as rule-based systems.

7 Conclusion and future work

In this paper, we proposed to use supervised machine learning approaches in order to detect and locate reachability incidents between customers' edge routers interconnected by virtual private networks. We focused specifically on incidents caused by configuration errors in BGP/MPLS IP VPNs that remain complex networks for service providers. In contrast to existing rule-based approaches to verify configurations, which involve manual requirements engineering to specify and maintain the set of rules, machine learning approaches rely mainly on correct and incorrect configurations to learn and detect configuration errors. The learning system takes as input, before deployment, a new configuration and predicts connectivity issues before they occur on the production network. Experimental results have shown better performance and better scalability with the approach based on neural networks. We are therefore currently working on enhancing this

model by integrating more configuration features, more fault types and larger networks.

In future work, in addition to configuration data, we plan to include state data such as network monitoring events and performance measures. With such data, we will have to integrate online training in the workflow. Furthermore, in line with the neural network approach (MLP), considering a Graph Neural Networks (GNN) model seems a natural direction for future work to represent the network topology and the routing control plane in order to detect other network incidents and identify their root cause.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th USENIX Conference on Operating Systems Design and Implementation. OSDI'16 (2016)
2. Bahnasy, M., Li, F., Xiao, S., Cheng, X.: DeepBGP: A Machine Learning Approach for BGP Configuration Synthesis. In: Proceedings of the Workshop on Network Meets AI & ML. NetAI '20, Association for Computing Machinery (2020)
3. Beckett, R., Gupta, A., Mahajan, R., Walker, D.: A general approach to network configuration verification. In: ACM Special Interest Group on Data Communication. SIGCOMM '17 (2017)
4. Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications **9**(16) (Jun 2018)
5. Cutler, A., Cutler, D., Stevens, J.: Random forests. In: Zhang, C., Ma, Y. (eds.) Ensemble Machine Learning: Methods and Applications, pp. 157–175. Springer US, Boston, MA (2012)
6. Dusia, A., Sethi, A.S.: Recent advances in fault localization in computer networks. IEEE Communications Surveys Tutorials **18**(4), 3030–3051 (2016)
7. Gulli, A., Pal, S.: Deep learning with Keras. Packt Publishing Ltd (2017)
8. Isa, M., Albarda: Designing Supervised Learning-Based Incident Management Model. Case Study: Broadband Network Service Provider. In: International Conference on ICT for Smart Society (ICISS) (2019)
9. Kazemian, P., Varghese, G., McKeown, N.: Header space analysis: Static checking for networks. In: 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12 (2012)
10. Khanafer, R.M., Solana, B., Triola, J., Barco, R., Moltsen, L., Altman, Z., Lazaro, P.: Automated diagnosis for UMTS networks using bayesian network approach. IEEE Transactions on Vehicular Technology **57**(4), 2451–2461 (2008)
11. Khurshid, A., Zhou, W., Caesar, M., Godfrey, P.B.: Veriflow: Verifying network-wide invariants in real time. In: First Workshop on Hot Topics in Software Defined Networks. HotSDN '12 (2012)
12. Moustapha, A.I., Selmic, R.R.: Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. In: IEEE International Conference on Networking, Sensing and Control. pp. 313–318 (2007)

13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
14. Ramchoun, H., Amine, M., Idrissi, J., Ghanou, Y., Ettaouil, M.: Multilayer perceptron: Architecture optimization and training. IJIMAI **4**(1), 26–30 (2016)
15. Rosen, E., Rekhter, Y.: BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, IETF (Feb 2006)
16. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031, IETF (Jan 2001)
17. Russell, S., Norvig, P., Davis, E.: Artificial Intelligence: A Modern Approach. Prentice Hall (2010)
18. Sharma, S., Mehrotra, D.: Comparative analysis of multi-label classification algorithms. In: First International Conference on Secure Cyber Computing and Communication (ICSCCC) (2018)
19. Srinivasan, S.M., Truong-Huu, T., Gurusamy, M.: Machine learning-based link fault identification and localization in complex networks. IEEE Internet of Things Journal **6**(4), 6556–6566 (2019)
20. Wang, M., Cui, Y., Wang, X., Xiao, S., Jiang, J.: Machine learning for networking: Workflow, advances and opportunities. IEEE Network **32**(2), 92–99 (Mar 2018)