TE-based Machine Learning Techniques for Link Fault Localization in Complex Networks

Srinikethan Madapuzi Srinivasan, Tram Truong-Huu, Mohan Gurusamy
Department of Electrical and Computer Engineering
National University of Singapore, Singapore
Email: elemss@nus.edu.sg, tram.truong-huu@nus.edu.sg, elegm@nus.edu.sg

***Abstract*—Communication networks such as wireless sensor networks, Internet of Things and vehicular ad-hoc networks are becoming more complex and increasing in size. This leads to high overhead (network and computation) and difficulty in determining the accurate network topology, which is an important information for traffic engineering and network management. Localization of link failures in such networks is a challenging problem and requires a novel approach to achieve the goal without any prior information about the network topology. In this paper, we present a traffic engineering (TE)-based machine learning approach to detect and localize link failures. Instead of using topology information and actively injecting additional packets to localize a failed link, the proposed machine learning model adopts a passive mechanism to learn the network traffic behavior from propagation delay, number of flows and average packet loss at every node in the network under normal working conditions and failure scenarios. We train the learning model with machine learning algorithms such as naive Bayes, logistic regression, support vector machine, multi-layer perceptron, decision tree and random forest. We implement the proposed approach and carry out extensive experiments using the Mininet platform. The performance study shows that our proposed approach localizes link failures with at least 90% accuracy using random forest algorithm while requiring less time-to-localization of a link failure compared to other existing works.**

***Index Terms*—Traffic engineering, fault localization, complex networks, machine learning, data analytics**

I. Introduction

The number of Internet-connected devices has exponentially increased and is expected to exceed 50 billion devices by 2020, heralding the era of Internet of Things and complex networks [1]. The increase in the size and complexity of such networks magnifies the difficulty in network operation and management. Particularly in fault management, large number of network devices and the connections between them implies high frequency of failures. On the other hand, failure recovery task becomes much more challenging due to the complex nature of the networks. A failure of a crucial network component, if not recovered in a timely manner, could lead to severe consequences such as service disruption to many customers and revenue loss to network operators. Further, anecdotal evidence indicates that it can take hours or days to resolve a failure by using a manual recovery approach based on combination of ping, trace-route and other functionalities for Ethernet and MPLS [2]. Thus, it is desirable that the network should be equipped with an efficient fault management system that is able to firstly diagnose failures as soon as possible, and then quickly recover the network from such failures, e.g., using a proactive failure recovery mechanism [3].

Fault diagnosis consists of fault detection and fault localization. Fault detection is the process of determining that one or more failures may have occurred based on network traffic behaviors. Fault localization is the process of determining the location of exact failures in the network based on the observed traffic behaviors. The correctness of fault diagnosis depends on the correctness of the data (i.e., network state) captured from the network, which in turn depends on the frequency of data collection (i.e., data sampling frequency) and the method by which the data is captured. Active measurements (also known as active probing using "hello" packets) have been extensively used in the literature [4]–[6]. In an active probing approach, several measurement endpoints (MEPs) or measurement intermediate points (MIPs) are deployed in the network, which inject and exchange additional control packets among themselves to detect and localize network failures. Higher the frequency of probing network components, better the accuracy of fault diagnosis. However, this leads to additional traffic overhead in the network and also increases the delay in fault detection due to the waiting time for the return of probing packets. Thus, there is a need for a passive mechanism that uses only measurements captured from normal traffic for fault detection and localization.

In order to realize failure diagnosis, network topology helps operators observe the network state and traffic behaviors. Many existing works have developed fault diagnosis algorithms based on network topology using graph theory as presented in the survey [7]. However, obtaining the network topology in complex networks is a cumbersome process and not always feasible. This implies that many existing approaches that are based on graph theory such as the works presented in [6], [8], [9] are no longer applicable. Yet, even with the availability of network topology, the graph-theoretic approaches have non-linear computation complexity. Thus, they cannot scale well with the increase in network size. Achieving fast detection and localization of failures becomes challenging when limited computing resources are available.

Addressing all the above discussed challenges, we present in this paper a traffic engineering (TE)-based machine learning model for link fault localization in complex networks. By a TE-based approach, we mean that instead of using topology information, which may not be available in practice, we capture

978-1-5386-7503-8/18/$31.00 ©2018 IEEE
DOI 10.1109/FiCloud.2018.00012

the network state, i.e., traffic information at network nodes and localize a failed link upon failure. Further, instead of actively injecting additional control packets in the network, i.e., using an active probing approach, the proposed machine learning model learns the behavior of normal traffic by passively monitoring network state at every node of the network. The learning features include the number of flows, packet loss and propagation delay of a packet from the monitored node to other nodes in the network. Upon a change in traffic behavior, a link failure could be detected and localized.

We integrate the machine learning model into several algorithms that use the techniques support vector machine, decision tree, random forest and neural networks, respectively. Training the learning model is a one-time cost and it can be carried out offline or online in parallel with fault localization to enhance the accuracy of the model based on new traffic observations. Evaluation of new traffic observation is a microsecond task, and it can be invoked on a periodic basis or an event-driven basis, thus enabling fast failure recovery. We note that the proposed approach has the most benefit when applying to complex networks. However, there is no limit of its application to other kinds of networks. We evaluate the performance of the proposed approach in terms of detection and localization accuracy on two networks: a ring network and a random network that is a model of complex networks [10].

The rest of the paper is organized as follows. We review the related work in Section II. We present our proposed approach in Section III. We present performance evaluation in Section IV before we conclude the paper in Section V.

II. Related Work

Fault localization plays an important role in network operation and management. In [7], [11], the authors presented a taxonomy on a variety of fault localization techniques that have been developed in the literature. These techniques are classified in several categories such as rule-based techniques [12], [13], case-based techniques [14], [15], probability-based techniques [16]–[19] and model-based techniques [20], [21]. The rule-based techniques heavily depend on a knowledge base that is a set of if-then statements (i.e., rules) pre-defined by system experts. However, the rule-based techniques cannot adaptively learn from experience and the dynamics of the networks with unseen traffic behaviors. Further, it is also difficult in updating and enriching the knowledge base. The case-based techniques perform fault diagnosis based on the expert and experience gain in the past. Thus, the solutions can only apply for the failures that are similar to the ones that have happened.

The probabilistic techniques presented in [18], [19] can perform fault diagnosis automatically and dynamically based on a probabilistic model of the network links. The probability mass function of the model indicates the location of the failed link. While these two works require a lightweight computation compared to other probabilistic techniques such as [16], [17], they need the knowledge about network topology as an input of the probabilistic model. The model-based techniques also use a knowledge base that describes the behavior of the network in the form of a mathematical model. The newly-observed traffic behaviors will be compared to the behaviors predicted by the model. Failures are detected if the observed behaviors fail to conform to the predicted ones. Thus, to achieve high accuracy of fault diagnosis, the model-based techniques require accurate information of network connectivity and operations. However, such information is hard to obtain in highly-dynamic networks such as complex networks. Differing with all the above techniques, our technique proposed in this paper enables a fast fault localization without requiring knowledge about network topology and the previously occurring failures. The proposed technique can achieve high accuracy and can adaptively adjusting the learning (fitting) model with new traffic observations, which can be caused by the network dynamics or errors during the data sampling process.

III. Data Analytics for Fault Localization

In this section, we describe our proposed TE-based machine learning approach for localizing link failures in the network. In our approach, we combine fault detection and localization together in a process, involving two phases: training and prediction phases. The training phase estimates the relationships between the features and the data points captured from the network in the past while the prediction phase identifies the link failure in the network, if any, for a newly-captured data point. To realize the proposed approach, two practical implementations can be considered. The first possible implementation is to use a central server that monitors the network nodes to periodically capture end-to-end traffic measurements such as delay and packet loss. Frequent the extraction of traffic measurement, faster the reaction if a link failure occurs. The second possible implementation is to use an event-driven approach to trigger link fault localization. When a node experiences abnormal behaviors in network traffic, it sends a request to the central server along with traffic measurements for failure detection and localization.

It is worth mentioning again that the machine learning model is trained with the end-to-end traffic observations as the features of the input data at the central server. This is an online learning approach since the training data set is gradually enriched with more data points captured from the network. The communication between the central server and network nodes can be frequent since the end-to-end traffic measurements sent to the central server from the nodes are not data-intensive.

A. Learning Model

To guarantee high accuracy of the machine learning algorithms, we need to suitably identify the features to be extracted from network traffic measurements for localizing link failures in the network. At every node in the network, we extract following traffic measurements:

- Number of flows that destine to other nodes in the network. We denote $f_{s,d}$ as the number of flows that originate from node s to node d.

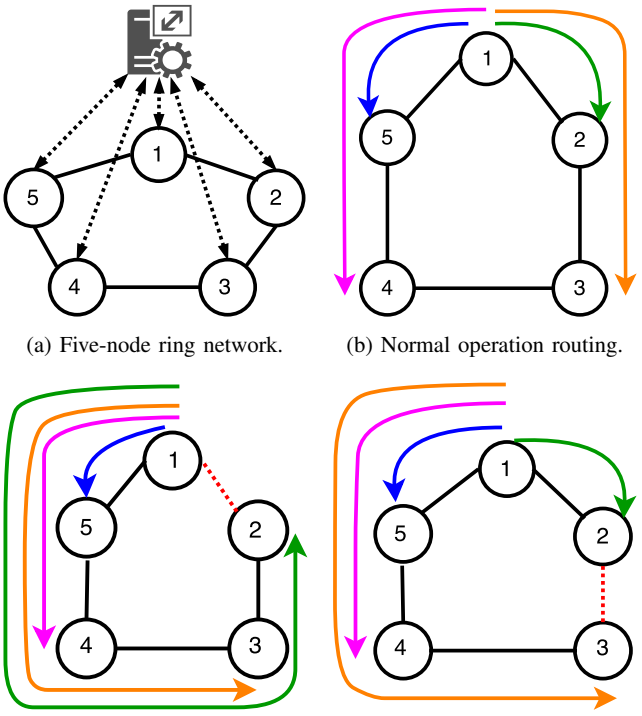
- Average end-to-end delay denoted as $d_{s,d}$ that is computed as the round-trip time (RTT) delay of a packet sent from node s to node d.
- Average packet loss denoted as $l_{s,d}$ is the number of packets lost on the path between node s and node d.

While the number of flows between every node pair gives us the information about network load, the end-to-end delay and packet-loss features provides indirect information on the path taken by each flow and congestion status in the path. The end-to-end delay can be captured at the source node of a flow with the timestamps information carried in the TCP headers. With the same number of flows (the same load) between a source and destination, longer delay and higher packet-loss implicitly mean that a certain link has failed. It is to be noted that we consider the scenario where routing information (network topology) is not practically available in complex networks. However, these traffic measurements captured for different pairs of source and destination nodes help machine learning algorithms to learn the correlation between the learning features to different failure scenarios. These features are fed to machine learning algorithms as a vector $\mathcal{A}$ as defined below:

$$\mathcal{A} = \begin{bmatrix} f_{1,2}, f_{1,3}, \ldots, f_{i,j}, \ldots, f_{V-1,V}, \\ d_{1,2}, d_{1,3}, \ldots, d_{i,j}, \ldots, d_{V-1,V}, \\ l_{1,2}, l_{1,3}, \ldots, l_{i,j}, \ldots, l_{V-1,V} \end{bmatrix} \quad (1)$$

where V is the number of nodes in the network. It is to be noted that the number of features is dependent on the number of aggregate flows in the network, which in turn is dependent on the number of nodes in the network. Given a network with V nodes, the total number of aggregate flows denoted as $\mathcal{N}$ is given by $V(V-1)$. We extract three traffic measurements for each aggregate flow as discussed above, and therefore, the size of vector $\mathcal{A}$ is three times the number of aggregate flows: $L(\mathcal{A}) = 3V(V-1)$.

Since different link failures may cause different network behaviors represented by the traffic measurements, link failure localization in the network can be considered as a multi-class machine learning classification problem in which each failure represents a class. In our work, we consider a single link failure scenario such that a link failure can be detected and recovered before another failure occurs. This is a practical assumption since protection and recovery of the network from multiple simultaneous failures require large amount of resources to be reserved while the failures are not frequent. Each link failure is then defined as a separate class along with the normal network measurements for training the machine learning model. Thus, the total the number of classes $\mathcal{C}$ required for training the machine learning algorithm to detect and localize a single link failure in the network is $\mathcal{C} = |\mathcal{E}| + 1$ where $\mathcal{E}$ is the set of links in the network. It is worth mentioning that since topology information is not available, the number of classes considered in the machine learning algorithms depends on the size of the training data set. Larger the training data set, better the accuracy of classification.



(a) Five-node ring network. (b) Normal operation routing.

(c) Routing when link (1-2) fails. (d) Routing when link (2-3) fails.

Fig. 1: Routing for different failures in a 5-node ring network.

B. Illustration Example

The impact of link failures on the features can be better understood with an illustrative example as depicted in Fig. 1. Given a five-node ring network with homogeneous link capacity and propagation delay, we assume that all flows are routed by the shortest paths. Consider the flows from node 1 to other nodes in the network, Fig. 1b depicts the paths taken by the flows under normal working conditions. Fig. 1c depicts the paths taken by each flow upon failure of the link between node 1 and node 2. We can see that the flows to node 2 and node 3 are affected and they have to be rerouted through alternate paths, which are longer than the paths used under normal working conditions, leading to longer propagation delay. Further, all the flows experience longer delay since all of them traverse through the same link (1-5) and share a limited amount of bandwidth.

Similarly, packet loss is a very useful measurement to identify link failures. Under normal working conditions, packet loss of a flow occurs only due to congestion on the flow path. However, when a link fails, all the packets sent through the failed link will be dropped before an alternate path is found, thus increasing packet loss. Longer the time needed for the network to find an alternate path, higher the packet loss. In the failure scenario shown in Fig. 1c, the flows from node 1 to node 2 and node 3 experience high packet loss when link (1-2) fails. Further, when the affected flows are rerouted through alternate paths, the congestion on link (1-5) may occur, thus increasing packet loss. Fig. 1d depicts routing solutions when

link (2-3) fails. Except for the flows between node 1 and node 3, all other flows keep using the same paths as in normal working conditions. However, the change in routing path of the flows between node 1 and node 3 affects traffic measurements of the flows from node 1 to node 4 and node 5 since the flows from node 1 to node 3 share link (1-5) with other flows. It is worth mentioning that with five nodes in the network, the size of the feature vector is $L(\mathcal{A}) = 3 \times 5 \times 4 = 60$ and with five links in the network, the number of classes in the machine learning classification problem is $\mathcal{C} = 6$.

C. Model Training with Machine Learning Techniques

To train the learning model, we use different machine learning techniques such as naive Bayes, logistic regression, suport vector machine, multi-layer perceptron, decision tree and random forest. This allows us to compare the efficiency of different techniques. In the following, we briefly describe the machine learning techniques used in this paper.

1) Naive Bayes: Naive Bayes is a probabilistic classifier that is based on Bayes rule [22]. The term naive is used since the classifier assumes that the learning features are independent from each other, which may not be true in practice. The training and prediction with naive Bayes is quick as it has low complexity. However, since the correlation between features is not considered in this classifier, the accuracy of the model trained with this classifier may not be as good as other machine learning techniques.

2) Logistic Regression: Logistic regression [6], [23] assumes a linear combination between the features and tries to predict the log probability of occurence of an event given by:

$$\log_e \left(\frac{p}{1-p} \right) = \mathcal{B}_0 + \mathcal{B}_1 a_1 + \mathcal{B}_2 a_2 + ... + \mathcal{B}_n a_n \quad (2)$$

where p is the occurence probability of the event that is specified by the feature vector $(a_1, a_2, \ldots, a_n)$. The training phase estimates the regression coefficient vector $(\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_n)$. This develops a hypothesis between the input features and output variable. This hypothesis is then used for prediction. Applying to our problem of link fault localization, each class of link failures will be trained and provided with a specific regression coefficient vector. Given a traffic measurement captured from the network (i.e., a data point), the probabilities of occurence of different link failures are computed. The link failure with the highest log probability is predicted to have occurred. Logistic regression is fast and simple. However, it assumes a linear correlation between the features.

3) Support Vector Machine: Support vector machines (SVM) [24] is a supervised machine learning technique primarily used for classification problems. SVM tries to find the best hyperplane, which divides the data points into two classes. The objective of SVM algorithm is to maximize the margin, thus creating the largest possible distance between the separating hyperplane and the data points on either side of it. For a multi-class classification problem, multiple hyperplanes are constructed by SVM algorithm to correctly classify the data points. However, SVM fits model better for binary classification problems than multi-class classification problems [25]. In this work, we choose a linear kernel for SVM algorithm to train our learning model.

4) Multi-layer Perceptron: Multi-layer perceptron (MLP) is a class of artificial neural networks [26]. It has at least three layers of neurons: an input layer, an output layer and a hidden layer of neurons, each having a non-linear activation function. MLP uses a back propagation algorithm for training the neural network. An advantage of MLP is that it guarantees a finite convergence for linearly separable data. In this work, we consider a MLP with 4 hidden layers, each having 20 neurons with ReLU activation functions.

5) Decision Tree: Decision tree classifier is a simple and commonly used classifier [27]. A decision tree classifier identifies the attribute that gives the maximum information gain and asks a series of questions regarding the attributes of the records. The usefulness of a question depends on the previous answers. Each time the classifier receives an answer, a follow-up question is asked until a class label is identified. Decision tree classifiers can classify non-linear points with multiple linear surface boundaries. However, they have higher risk of overfitting and hence the parameters of decision tree classifiers need to be chosen carefully.

6) Random Forest: Random forest [28] is a learning classification method that constructs numerous decision trees at training time and outputs the mode of the individual trees as the class label. The risk of overfitting with decision trees is overcome with random forest techniques. Random forest technique also helps in identifying the important features from the training data set. Random forest suits well for multi-class classification problems such as the link fault localization problem considered in this paper. Another advantage of the random forest technique over other techniques is that it works well with both numerical and categorical data. The input data can be in various scales and we do not need to perform any additional pre-processing to normalize different features.

IV. Performance Study

A. Simulation Settings

We implement the proposed approach and carry out experiments on Mininet platform to evaluate its performance. We consider two network topologies: a five-node ring network and a twenty-node random network with 26 links. The data points used for simulations are also generated by using the Mininet platform. The traffic between the nodes in the networks is generated using *iperf3* tool at a constant data rate and data size while the number of flows generated between each node pair is randomly chosen in the range $[0, 30]$. After collecting all data points, outliers are removed since they may skew the learning model and lead to inaccurate classifications. These outliers are generated at the nodes, which fail to capture the actual statistics at certain instants and rather send a random large value to the server. Further, while random forest algorithm is invariant to feature scaling, the other algorithms require feature scaling to achieve high accuracy. Thus, for all the algorithms

except random forest, the data points are normalized based on the maximum and minimum values of each feature before training the machine learning model.

For the five-node topology, we train the machine learning model with 2000 data points for each class of link failures, i.e., we use in total of 12000 data points in the training data set. The test data set includes 3000 data points. For the twenty-node topology, we train our machine learning model with 15000 data points for each class, i.e., about 400000 data points in the training data set. The test data set has 100000 data points.

We use the following performance metrics to evaluate the performance of different machine learning algorithms:

- Precision: The ratio of the number of data points associated with link failures correctly detected and localized over the total number of data points predicted as failures. The precision value is computed as follows:

$$\mathcal{P} = \frac{T_P}{T_P + F_P} \tag{3}$$

  where $\mathcal{P}$ is the precision value, T_P is the number of "true positives" and F_P is the number of "false positives".
- Recall: The ratio of the number of data points associated with link failures correctly detected and localized over the total number of data points associated with link failures that have occurred. The recall value is given by:

$$\mathcal{R} = \frac{T_P}{T_P + F_N} \tag{4}$$

  where F_N is the number of "false negatives".
- F_1-Score: The F_1-Score is the harmonic average of the precision and recall values. It takes a value in the range of $[0, 1]$. Higher the value of F_1-Score, better the performance of the machine learning technique, i.e., we obtain perfect precision and recall values when F_1-Score reaches 1. It is computed as follows:

$$F_1\text{-Score} = \frac{2\mathcal{P}\mathcal{R}}{\mathcal{P} + \mathcal{R}}. \tag{5}$$

- Failure detection rate: The ratio of the number of data points associated with link failures correctly detected as failures in the network to the total number of data points associated with link failures that have occurred.
- Failure localization time: The time taken by the machine learning algorithms to localize the link upon its failure.

B. Analysis of Results

1) Performance with five-node ring network: In this section, we evaluate the performance of the six machine learning algorithms with a five-node ring network. In Fig. 2 and Fig. 3, we plot the average of the precision values and recall values for training and test data sets, respectively. It is essential to compare the metrics for training data sets and test data sets to understand how well the machine learning algorithms have trained the model. Both the average precision and recall values for the training data set are close to 100% for all the algorithms. This demonstrates that the learning model has been well trained. When testing with the test data set, it can be

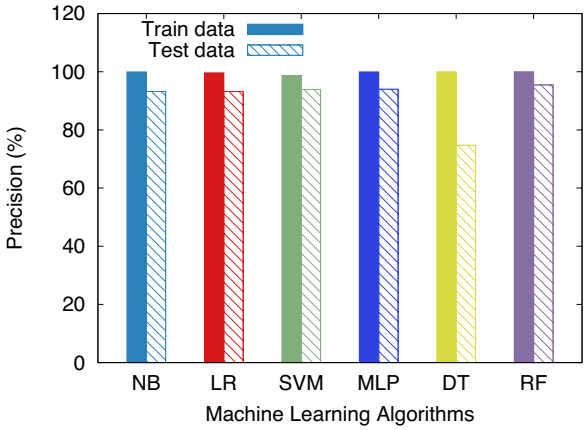


Fig. 2: Average precision with training and test data sets for a five-node ring network.

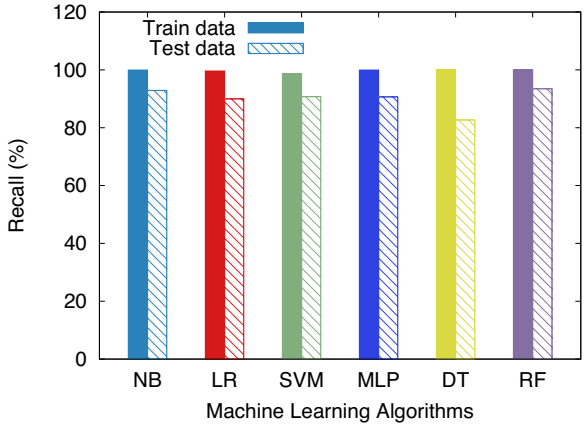


Fig. 3: Average recall with training and test data sets for a five-node ring network.

seen in Fig. 2 that the random forest algorithm outperforms the other algorithms with an average precision of about 96%, closely followed by multi-layer perceptron with an average precision of 94%. This indicates that the model trained with random forest algorithm identifies link failure with minimal misclassification or noise. The decision tree classifier performs poorly with an average precision of 74%. This is due to the overfitting problem, which is common for decision tree classifiers. Similar behaviors of the algorithms are observed when we analyze their performance in terms of the recall metric. As shown in Fig. 3, random forest algorithm performs the best with a recall value of 93%. This means that random forest algorithm correctly classifies 93% of data points.

In Fig. 4, we compare the precision and recall of all the algorithms for the test data set. The results show that most of the algorithms have higher precision value than recall value, except for decision tree algorithm. It is evident that random forest algorithm performs the best in terms of both precision and recall for the test data set.

In Fig. 5, we plot F_1-Score of different classes, i.e., different link failures in the network with random forest algorithm. This helps to further understand how random forest algorithm

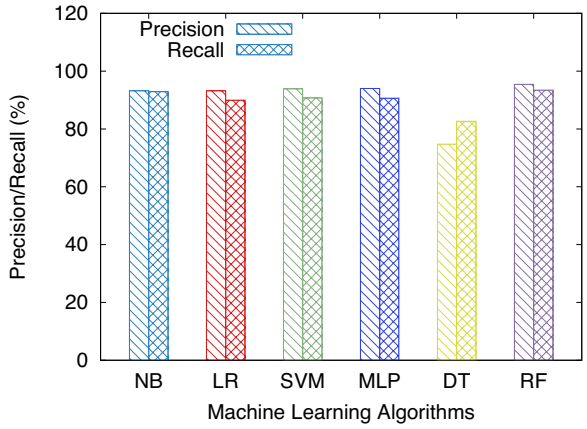


Fig. 4: Precision and recall for five-node ring network.

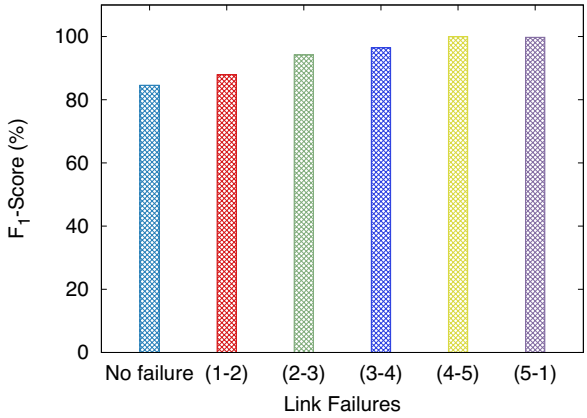


Fig. 5: F_1-Score of different link failures with random forest algorithm for five-node ring network.

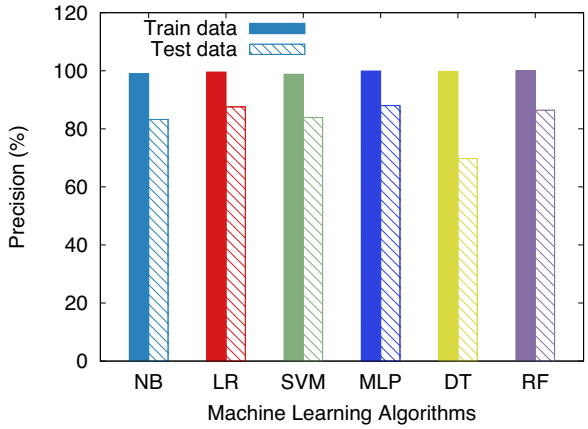


Fig. 6: Average precision with training and test data sets for a twenty-node random network.

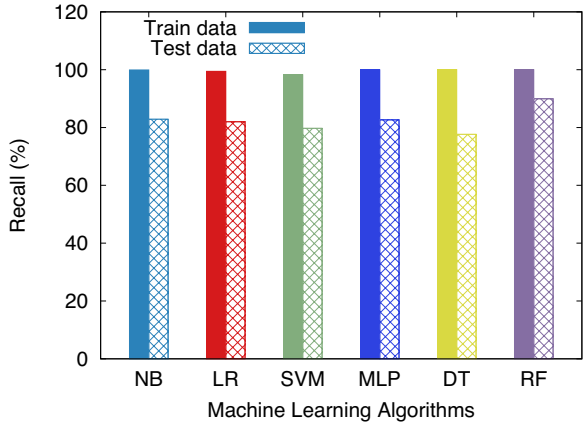


Fig. 7: Average recall with training and test data sets for a twenty-node random network.

performs in each link failure scenario. We can see that the minimum F_1-Score is 85% for the scenario with no link failure and the maximum F_1-Score is 99.9% for the scenario when link (4-5) fails (shown in Fig. 1). This demonstrates that random forest algorithm has good performance with accuracy of at least 85% in all link failure scenarios.

2) Performance with twenty-node random network: In this section, we evaluate the performance of the six machine learning algorithms with a twenty-node random network with 26 links. Similar simulations as described in the previous section are conducted again for this topology. In Fig. 6, we plot the average precision for training and test data sets of the machine learning algorithms. We obtain the same behavior as in the previous simulation. Random forest algorithm outperforms other algorithms with the highest precision of 99% and 92% for the training data set and test data set, respectively. In Fig. 7, we plot the average recall with training and test data sets for all the algorithms. Random forest algorithm also has the best performance with 90% of recall, followed by multi-layer perceptron with 85% of recall.

In Fig. 8, we compare the precision and recall values of all the algorithms for the test data set. It can be seen in Fig. 8 that random forest performs well with similar precision and recall values while the other algorithms have large differences between precision and recall values.

3) Failure Detection Rate: As discussed in the previous sections, by using machine learning techniques, we combine the failure detection and localization phases together. The metrics we have presented above evaluates how well the machine learning algorithms are able to correctly detect and localize link failures in the network. However, in some cases, the proposed approach can detect the occurence of a failure in the network but it fails or wrongly localizes the failed link. This nevertheless can help the network administrator to quickly react when a link has failed. In Fig. 9, we present the failure detection rate for the two examined networks. We can see that naive Bayes and random forest algorithms have better performance than the other algorithms for both topologies with failure detection rate of about 90% and 87% for the five-node ring network and twenty-node random network, respectively. This high detection rate leads to high performance of random forest algorithm in all the metrics presented above. It is worth mentioning again that the failure detection rate does not correspond to the accuracy of the model, which is characterized by the precision and the recall metrics defined earlier.

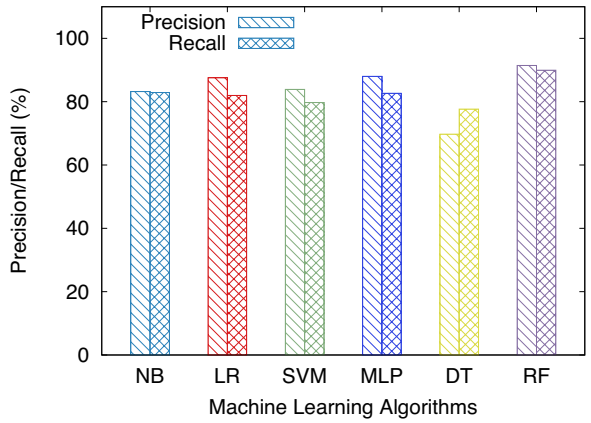


Fig. 8: Precision and recall for twenty-node random network.

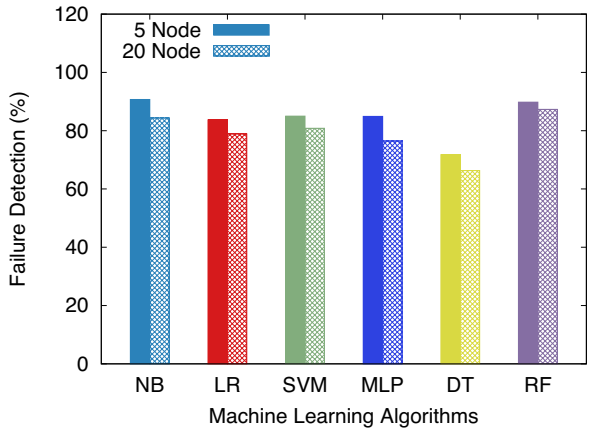


Fig. 9: Failure detection rate of machine learning algorithms for five-node and twenty-node networks.

4) Failure Localization Time: Another important metric to be considered is that of failure localization time, i.e., time taken to localize a failed link upon its failure as this affects the time to recover from the failure. Faster the link fault localization, less the impact of the failure on the network. In Table I, we present the time taken by different machine learning algorithms to localize link failures in the network. We also measure the time taken by an active-probing approach (denoted as ping-based approach) presented in [6]. The results show that all the machine learning algorithms spend in the order of microseconds to localize a failed link. The worst case time in our approach is SVM algorithm with 52μs, whereas the ping-based approach requires significantly longer time to localize a failed link. It is worth mentioning that while localization time of our approach using machine learning algorithms does not depend on the size of networks, the existing approach requires exponentially increased localization time from 249 ms with the five-node ring network to 2764 ms with the twenty-node network. This demonstrates that our proposed approach detects and localizes link failures accurately and quickly.

TABLE I: Comparison of time-to-localization of failures

Localization Methods	**Time (in μs)**
Naive Bayes (NB)	4.40
Logistic regression (LR)	0.55
Support machine vector (SVM)	52.00
Multi-layer perceptron (MLP)	1.50
Decision tree (DT)	0.24
Random forest (RF)	1.70
Ping-based approach (five-node ring network)	248857.00
Ping-based approach (twenty-node network)	2764420.00

V. Conclusion

In this paper, we addressed the problem of link fault localization in complex networks. We proposed a TE-based machine learning approach that learns from traffic measurements captured from the network in normal working conditions and failure scenarios. Without using network topology information, the learning model takes into account the number of flows, propagation delay and average packet loss captured at every node in the network. We trained the learning model with six machine learning algorithms and carried out comprehensive simulations in the Mininet platform with two network topologies to study the performance of our proposed approach. The results show that among the examined machine learning algorithms, random forest algorithm outperforms other algorithms for both topologies with a minimum of 90% accuracy in localizing link failures. We also evaluated the time-to-localization of the proposed approach and compared with an existing approach that uses active probing. The results show that while our approach requires a few microseconds to detect and localize a link failure, the existing approach based on active probing needs longer time in the order of milliseconds to do so. For future work, we will further extend this work to consider a multi-path routing problem and carry out performance study on larger networks.

Acknowledgment

This work was supported by Singapore MoE AcRF Tier 1 Grant, NUS WBS No. R-263-000-C04-112.

References

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," White Paper, Cisco, Apr. 2011.
[2] A. Banerjee, "Assuarance of Real-Time Cloud Services Requires Insights from Correlated Content, Sessions and IP Topology Planes," White Paper, Heavy Reading, Aug. 2012.
[3] P. Murali Mohan, T. Truong-Huu, and M. Gurusamy, "Fault tolerance in TCAM-limited software defined networks," *Computer Networks*, vol. 116, pp. 47–62, 2017.
[4] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software Defined Networking: Meeting Carrier Grade Requirements," in *IEEE LANMAN 2011*, Chapel Hill, USA, Oct. 2011, pp. 1–6.
[5] N. L. M. V. Adrichem, B. J. V. Asten, and F. A. Kuipers, "Fast Recovery in Software-Defined Networks," in *EWSDN 2014*, London, Sep. 2014.
[6] M. X. Cheng and W. B. Wu, "Data Analytics for Fault Localization in Complex Networks," *IEEE Internet Things J.*, vol. 3, no. 5, Oct. 2016.
[7] A. Dusia and A. S. Sethi, "Recent Advances in Fault Localization in Computer Networks," *Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 3030–3051, May 2016.
[8] X. Xiang-Hua, Z. Biao, and W. Jian, "Tree Topology Based Fault Diagnosis in Wireless Sensor Networks," in *IEEE WNIS 2009*, Shanghai, China, Dec. 2009, pp. 65–69.

[9] P. P. C. Lee, V. Misra, and D. Rubenstein, "Toward Optimal Network Fault Correction in Externally Managed Overlay Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 3, pp. 354–366, Mar. 2010.
[10] K. Batool and M. A. Niazi, "Modeling the internet of things: a hybrid modeling approach using complex networks and agent-based models," *Complex Adaptive Systems Modeling*, vol. 5, no. 1, Mar. 2017.
[11] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Sci. Comput. Program.*, vol. 53, no. 2, Nov. 2004.
[12] G. Liu, A. K. Mok, and E. J. Yang, "Composite Events for Network Event Correlation," in *IFIP/IEEE IM 1999*, Boston, USA, May 1999.
[13] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule Discovery in Telecommunication Alarm Data," *Journal of Network and Systems Management*, vol. 7, no. 4, pp. 395–423, Dec. 1999.
[14] L. Lewis, "A Case-based Reasoning Approach to the Management of Faults in Communications Networks," in *IEEE INFOCOM 1993*, San Francisco, USA, Mar. 1993.
[15] L. Bennacer, L. Ciavaglia, A. Chibani, Y. Amirat, and A. Mellouk, "Optimization of Fault Diagnosis based on the Combination of Bayesian Networks and Case-Based Reasoning," in *IEEE NOMS 2012*, Maui, USA, Apr. 2012, pp. 619–622.
[16] M. Steinder and A. S. Sethi, "Probabilistic Fault Localization in Communication Systems using Belief Networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 809–822, Oct 2004.
[17] A. G. Prieto, D. Gillblad, R. Steinert, and A. Miron, "Toward Decentralized Probabilistic Management," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 80–86, Jul. 2011.
[18] A. Johnsson and C. Meirosu, "Towards Automatic Network Fault Localization in Real Time using Probabilistic Inference," in *IFIP/IEEE IM 2013*, Ghent, Belgium, May 2013, pp. 1393–1398.
[19] A. Johnsson, C. Meirosu, and C. Flinta, "Online Network Performance Degradation Localization using Probabilistic Inference and Change Detection," in *IEEE NOMS 2014*, Krakow, Poland, May 2014.
[20] K. Appleby, G. Goldszmidt, and M. Steinder, "Yemanja—A Layered Fault Localization System for Multi-Domain Computing Utilities," *J. Netw. Syst. Manag.*, vol. 10, no. 2, pp. 171–194, Jun. 2002.
[21] R. Steinert and D. Gillblad, "Long-Term Adaptation and Distributed Detection of Local Network Changes," in *IEEE Globecom 2010*, Miami, USA, Dec. 2010.
[22] L. Jiang, D. Wang, Z. Cai, and X. Yan, "Survey of Improving Naive Bayes for Classification," in *ADMA 2007*, Harbin, China, Aug. 2007.
[23] A. Agresti, *Logistic regression*. Wiley Online Library, 2002.
[24] J. A. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, 1999.
[25] M. Aly, "Survey on Multiclass Classification Methods," *Neural Netw.*, vol. 19, pp. 1–9, 2005.
[26] M. W. Gardner and S. Dorling, "Artificial Neural Networks (the Multilayer Perceptron) - A Review of Applications in the Atmospheric Sciences," *Atmospheric Environment*, vol. 32, no. 14-15, 1998.
[27] S. R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, 1991.
[28] V. Y. Kulkarni and P. K. Sinha, "Random Forest Classifiers: A Survey and Future Research Directions," *Int. J. Adv. Comput.*, vol. 36, 2013.