Anomaly Detection in Wireless Networks using Controller Logs

Taha Hajar*, George Christoudoulou†, Davide Cuda†, Domenico Ficara†, Lorenzo Granai†

* EPFL, Lausanne, Switzerland, taha.hajar@epfl.ch
† Cisco System, Switzerland, {georgchr, dcuda, dficara, lgranai}@cisco.com

***Abstract*—With the growing demand for wireless connectivity, enterprises are investing more in supporting wireless access in their large scale networks. However, the deployment of wireless access introduces a new level of complexity and opens the door for new problems in the network. While manual troubleshooting is time consuming and requires much experience, an indispensable need for smart automated troubleshooting is emerging . In this paper, a machine learning solution is proposed to automate anomaly detection in wireless networks. The solution uses the network logs to derive sequences of network events in real-time. Two sequence learning techniques are then applied and combined to detect abnormal sequences of network events. These methods use a novel combination of features and distance metric compatible with the unstructured nature of the sequences. Furthermore, F-scores and ROC curves are used to compare the performances of different models. Finally, this paper presents real case scenarios where this innovative approach improves network management by automatically gaining real-time useful information on wireless networks status.**

I. Introduction

With the explosion of the number of smartphone and wireless device connected to the Internet, WiFi network becomes one of the most critical network segment. According to reports published by Dimension Data, network access ports were 80% wired and 20% wireless in 2013. With the wireless access overtaking the wired access, the same source also predicted that the access ports will eventually become 20%wired and 80% wireless in a couple of years [1]. Nowadays, large enterprises such as universities and big companies, can have up to thousands of access points (AP) managed under the same network. Consequently, maintaining and assuring a good health of the network become crucial, critical and challenging. In this context, timely identifying issues in the wireless context is of primary importance to ensure the required quality of experience for users. However, the large number of 802.11 protocols, their complex state machine and the variance of the wireless medium make this task even harder than usual.

Numerous networking issues are reported to networks administrators daily. Although there exist tens of useful troubleshooting tools, diagnosing network problems has always been a challenging task. Furthermore, as the network size, its complexity, and the number of features it supports increases, the number of dimensions a network engineer has to consider while solving a problem drastically increases. This puts more pressure on the task of monitoring and solving network issues. Few examples of the features needed to be considered are physical connectivity, network configuration, router configurations and status (queue status, CPU load, memory usage, etc), network protocols state machines and the interaction between the application layer and the network layer. Therefore, network troubleshooting usually requires skilled network engineers that leverage years of experience.

In a typical scenario When network users experience a network problem, they usually contact the network administrator who starts investigating the network logs in order to understand what happened. However, the large number of logs and the concurrency feature of wireless networks make this task quite complex.

In this context, machine learning techniques can help in automating the log analysis task. Nowadays, the trend is to make the machine learn by itself how to assess the situation in the system by using available data. This technique is very efficient, especially when big data is available since the machine processes the data much faster than a human. The network logs, being a rich source of data, is then a potential source of useful data that can be used in automating network troubleshooting.

Therefore, in this paper we study the possibility of building a machine learning solution that can identify network anomalies automatically without human intervention. In the proposed solution, the machine processes the network logs in real-time and monitors the behavior of the network. Whenever an anomaly is detected, it informs the administrator and points her to the logs reflecting the anomaly. With this solution, the administrator can proactively take some action to limit the window of time in which users perceive some connectivity problems instead of inspecting the problem.

The paper is organized as follows. In Section **??**, we provide a concise summary of the most pertinent related work and we highlight how this approach is different from the proposed solutions. In Section **??** we formalize the problem and...

II. Related works

Automated troubleshooting for wired networks is not a new topic. Many studies tackled this issue using myriad analytic and learning techniques. However, the introduction of wireless was a game changer. Traditional tools designed for wired networks are not able to cover the problems of wireless features. Thus, recent studies started attacking the automation of wireless troubleshooting from different angles.

Nevertheless, to the best of our knowledge, none of these studies addresses the possibility of using machine learning on network logs. The closest works are machine learning tools that use measurement data to predict the source causes of a bad performance or detect anomalies in the wireless network.

A research in [8] studies the effect of media access dynamics, mobility management and 802.11 protocols on the network performance from the perspective of the physical layer to the transport layer. The authors built an automated tool that collects measurements of delays using a pre-installed infrastructure of APs for testing. The tool uses these measurements along with several trained models to infer hidden delays and detect their sources.

Two famous solutions for wireless enterprise network monitoring and troubleshooting are WifiProfiler [9] and DAIR [10]. Both solutions detect wireless problems and predict the network performance using data collected from the clients perspective without assuming any special capabilities in the infrastructure. However, the main disadvantage of these solutions is that they are instrumented to test for specific anomalies. They do not use any artificial intelligence or machine learning technique.

While log analysis is not widely used in network troubleshooting, it has been immensely used to understand the behaviors of other computer systems. In some studies such as [13,14], the common feature extraction techniques, used in the natural language processing field, were applied to classify log files.

In other log analysis studies, log files are used to study certain event correlation in order to discover new patterns and causalities between events.For example, two studies [15,16] follow a data-driven framework to acquire all needed data from a large set of log files. This approach first assimilates the semantics of uniform events and categorizes the events into common base event (CBE) format. After recognizing similar messages as the same CBE, interesting patterns are discovered by exploiting temporal dependencies among events.

In this project, we combine both log analysis approaches, the pattern discovery and the feature extraction methods. The discovered patterns are used as features to learn the different behaviors in the network and detect anomalies.

III. Problem overview

In order to identify anomaly, we focus on a specific network configuration. For this network topology we collect logs, that we analyze in order to idetify anomalies.

A. Network architecture

We consider a typical enterprise or campus wireless access network. Usually a single Cisco Wireless Controller as the Cisco 8540 Wireless Controller or the Cisco Catalyst 3850 can support thousand of Access Points (AP) and ten of thousands of clients [**?**]. To simplify configurations and management, We consider lightweight APs, i.e., APs cannot act independently of a wireless controller (RFC 5412). In order to monitor and identify issues with this network while minimizing the amount of data processed, we focus on the data collected by the WLC. Collecting data on the WLC also void synchronization issues as well as ....

We focus mainly on parisng and analysing the network logs. This is because global show commands usually show a summarized information, in precise time instant

B. Problem statement

We focus mainly on parsing and analyzing the network logs. This is because global show commands usually show a specific summarized information, in precise time instant. On the other hand, the log files cover the whole sequence of events occurring in the network. Due to their complexity and lengths, the log files are hard to be processed by a human being. However, our hypothesis is that machine learning can help us find the interesting information efficiently from these log files. In our case, we are interested in detecting abnormal sequences of network events.

In order to formulate the problem, we introduce some notations and definitions. First, we consider that every log entry represents a network event. A log entry $\boldsymbol{x}$ consists of a timestamp t_x, a list of mac addresses d_x of the devices involved in the event and a message m_x describing the action that took place. In the preprocessing phase, we assign a unique ID ***ID*** for every possible action. Then, we map every message to the corresponding ***ID***. An event e_x is thus defined as the tuple containing the timestamp, the mac address list and the event ID corresponding to the log entry x:

$$e_x = (t_x, d_x, ID_x)$$

We also denote the ordered sequence s of n consecutive events by:

$$s =< e_1, e_2, ...e_n >$$

In this project, the sequences to be labeled are derived based on the mac address involved in the event. In other words, for every mac address that appears in the logs, we construct a sequence of events in which it is involved. Thus, for a dataset where m mac addresses are seen, m sequences are constructed and processed.

IV. Model and Methodology

A. Pattern discovery

In the context of this work, pattern discovery basically aims at finding interesting relations among the variables in the large log data. To the best of our knowledge, none of the previous works described in sec.**??** combine the results of the pattern discovery solutions with any learning algorithm to train a machine to understand the data. The approach taken here is to adopt discovered patterns instead of traditional features in the learning phase. Patterns have to be of frequently correlated and mutually dependent events, such that, in the presence of issues, a clear cause-effect determinism can be underlined. This decision allows for dimensionality reduction and human readability. Indeed, one of the wanted outcome of this work

is highlighting sequences of interesting (i.e. problematic) logs to humans in order to help better debugging.

The following subsections describe the problem and two algorithms adopted in this work to address it.

1) Finding patterns in wireless controller logs: The training data is a dataset S of event sequences. Each sequence belongs to a device ID and a specific time interval, while each event consists of a timestamp and a message ID. The pattern definition in this work is a subsequence of message IDs that appear in the sequences of S.The mentioned concurrency and noise specifications of the log sequences hinder the pattern discovery task since the events of the same pattern might not be contiguous: e.g. noisy or concurrent events might overlap with the pattern to be discovered. This means that all the events of a pattern will not always appear in the same relative positions to each other.

2) Sequential-Pattern Discovery: The first algorithm adopted to find patterns is based on the sequential rule mining (SRM) algorithm [1]. However, since this algorithm discovers the frequent correlated patterns only, it has been modified in order to cover the infrequent mutually dependent patterns as well. Originally, the original algorithm discovers patterns having a given minimum support and confidence $minSup$ and $minConf$ respectively. The new algorithm, called Sequential Pattern discovery or SPD is described in alg.1. It uses several instances of the original algorithm to discover frequent patterns, store them, remove them from the data, decrease the minSup threshold and then go back to the first step. The process keeps iterating until a $minSup == 0$ is reached.

Algorithm 1 Sequential Pattern Discovery Algorithm

```
1:  procedure SPD(D, minConf)
2:      S ← ∅
3:      minSup ← size(D)
4:      while minSup ≥ 0 do
5:          N ← SRM (D, minSup, minConf)
6:          S ← S ∪ N
7:          for p in N do
8:              Remove all occurrences of p in D
9:          minSup ← minSup 10%× size(D)
10:     return S
```

3) Interesting Pattern Discovery: Another approach adopted in this paper is based on both sequential rule and interesting rule mining. We use the SRM algorithm to extract frequent correlated patterns and then we use the apriori algorithm with the all confidence metric, which we call original interesting pattern discovery algorithm (OIPD), to extract mutually dependent patterns.

Both algorithms combined produce the Interesting Pattern Discovery (IPD) algorithm whose pseudocode is listed in alg.2.

Algorithm 2 Interesting Pattern Discovery Algorithm

```
1: procedure IPD(D, minConf, minSup, minAllConf)
2:     S ← SRM (D, minSup, minConf)
3:     N ← OIPD (D, minallConf)
4:     S ← S ∪ N
5:     return S
```

B. Sequence learning

C. Learning algorithm

V. Testing and Results

VI. Conclusions

Stress main idea. Usage and impact. Alternatives and future work.
Usage: Network monitor, debugging tool, assess log system
Future work: noise resilient, anomaly classification

References

[1] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mininga general survey and comparison. *ACM sigkdd explorations newsletter*, 2(1):58–64, 2000.