June 2020

METHOD AND APPARATUS FOR AUTOMATED LINK FAILURE DETECTION IN NETWORKS USING MACHINE LEARNING

Karthik Babu Harichandra Babu

Xinyuan Huang

Ajay Anand

Mankamana Mishra

METHOD AND APPARATUS FOR AUTOMATED LINK FAILURE DETECTION IN NETWORKS USING MACHINE LEARNING

AUTHORS:
Karthik Babu Harichandra Babu
Xinyuan Huang
Ajay Anand
Mankamana Mishra

ABSTRACT

Techniques are presented for automatic link failure prediction and detection with an automatic troubleshooting capability to quickly identify switches/ports involved in the failure before the failure leads to a complete outage. A framework to predict the failure and easily handoff all routing protocols to an alternate path is provided. Such prediction helps a network to react before a failure occurs. Almost zero traffic loss during a failure may be achieved by predicting the failure in advance.

DETAILED DESCRIPTION

Generally, network recovery mechanisms have an ability to first detect a network element failure, followed by remedial action and restoration of the connectivity using secondary, or backup, links. All of these network recovery approaches are reactive. That is, the failure must first occur and be detected before remedial action can be taken, e.g., routing the traffic along a secondary (backup) path.

Ethernet link Operations, Administration, and Management (OAM) Link Fault Management (LFM) is one way to detect a link failure in a network. IEEE 802.3ah OAM LFM can be configured on point-to-point Ethernet links that are connected either directly or through Ethernet repeaters. The IEEE 802.3ah standard meets the requirement for OAM capabilities even as Ethernet moves from an enterprise technology to a Wide Area Network (WAN) and access technology, because the standard remains backwards compatible with existing Ethernet technology.

A link failure is a very severe and serious network condition that can quickly lead to a network outage and also can disrupt business operations. OAM frames, called OAM Protocol Data Units (OAM PDUs) use the protocol destination MAC address. The protocol is a relatively slow protocol with a maximum transmission rate of 10 frames per second,

resulting in minor impact to normal operations. Link monitoring detects and indicates link faults under a variety of conditions and uses the event notification OAM PDU to notify a remote OAM device when problems on the link are detected. However, when link monitoring is enabled, the CPU must frequently poll error counters and the number of CPU cycles is proportional to the number of interfaces that must be polled. Thus, the number of CPU cycles may quickly increase resulting in the slow response of the protocol.

To improve the response times and prevent network outages, a system for automatic link failure detection based on a set of indicators is presented. Before the detected failure can lead to a complete outage, the system can quickly identify the switches/ports involved in the failure and, together with an automatic troubleshooting capability, circumvent the failure.

A syslog message sent to a peer node when the local node fails or dies may be, for example:

> *Jun 11 09:10:56.473: %ETHERNET_OAM-6-LINK_TIMEOUT: The client on interface Et0/0 has timed out and exited the OAM session.

The techniques presented include predicting the link failure based on the PDUs timeout, assisting in troubleshooting to find the exact device and details of the interface that is about to go down, and triggering a redundant link.

The goal is to fundamentally change the approach to network element failure by relying on a Machine Learning (ML) based architecture to allow for prediction (forecasting) of failure with high precision so as to proactively reroute traffic on a secondary (backup) path when a failure is predicted.

Enhancing existing technology with ML to find and predict a fault is proposed. That is, machine intelligence is added to an already available and widely deployed technology, e.g., Ethernet-OAM. Rather than introducing new/existing protocols specifically designed for the Ethernet-OAM, the existing technology is enhanced with ML to find faults earlier through prediction. The techniques relate to predicting an Ethernet link failure and, thus, proactively activating a backup/redundant link and rerouting along the backup/redundant link traffic subject to the predicted failure using related ML architecture.

The ML architecture includes training of ML algorithms. Based on the training, inferences from the trained ML architecture may be applied either globally (e.g., in the cloud) or on-premise (e.g., inference at the edge of the network).

Basically, logic may be implemented on one or more controllers, to monitor and examine a set of indicators, which when combined can lead the system to determine that a link failure is in-progress. More specifically, the system will proactively monitor the following events and states:

1. OAM Packet Data Units (PDUs) getting delayed
2. Interface Database Status change
3. OAM Link monitoring event state

The above indicators may be part of telemetry information that is either collected periodically or pushed asynchronously to the one or more controllers.

As mentioned earlier, a link failure result in network meltdowns and hence need intervention/remediation immediately. Additionally, error specific action may be implemented. With the advent of networking operating systems with distributed software architecture, e.g., Polaris IOS-XE, catalyst switches are able to perform device level analytics and report events.

The logic uses a Complex Event Processing (CEP) system, with a sliding window of width up to 5 minutes, to identify and correlate the above events together. Indicators are evaluated every 10 seconds as the window slides forward, i.e., events and/or states are correlated looking back 5 minutes. Once the presence of leading indicators within this window of time is established, a reasoner engine analyzes the events and/or states.

The ability to predict the failure is a core component. The use of the reasoner engine as a follow up action for troubleshooting and remediation is one additional component.

To predict a failure, the reasoner engine uses background knowledge that is captured in a formal semantic model (ontology) to perform rule-based machine reasoning. The engine starts with data that was collected by the CEP system as initial context and collects additional information regarding the physical topology from the topology-service in the controllers. Based on this analysis, the reasoner engine determine a set of ports that are suspected to go down or fail.

The next step is to verify the determination by having the reasoner engine request the most up-to-date state of the ports from the various network devices. Note that Topology Change Notifications (TCNs) are also received by the CEP system. The reasoner engine performs all the required analysis and determines the set of network devices that should be contacted before the effects of the link failure impact the network and prevent communication between controllers and those affected switches.

The reasoner engine gathers the additional data, but also will reach out to all other remote devices to inform those devices of a risk of failure based on the failure model prediction. But here, relying on controllers processing time to perform such a notification in real-time may overburden the controllers. To overcome this drawback, classifiers may run on the devices instead of over burdening the controllers.

Using high-performance machine reasoning, the reasoner engine identifies which redundant port needs to be enabled in order to avoid any loop in the network and also make sure the network outage has not yet occurred. If the user provides pre-approval for the automation, the reasoner engine can remediate the situation automatically by shutting or disabling the failing/faulty port and/or informing a network administrator about the device and failing/faulty port information.

Referring to Figure 1, in which, as an example, the maximum one-day delay allowed for Streaming Video is 5sec and a delay in a link is 6 sec. The reasoner engine may use the delay as a feature in combination with another set of features to predict a possible link failure as an outcome. Once a failure is predicted, the system will reroute the traffic to a redundant path to prevent the outage

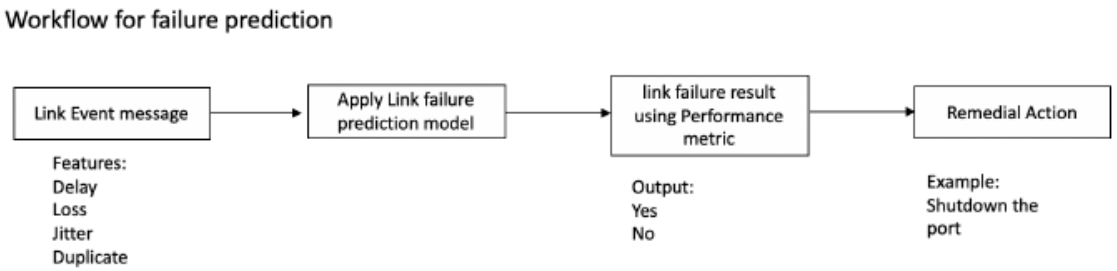

Figure 1

A machine learning model can be trained on historical data and then be deployed to the reasoner engine to detect link failures. The model may work as follows.

1. Pre-Process and Feature Extraction

Referring to Figure 2, a set of historical logs with link failure messages are taken and parsed into structured data with key-value fields. The extracted information may include, but not limited to:

- timestamp
- link name/id
- event type
- severity level
- event message keywords

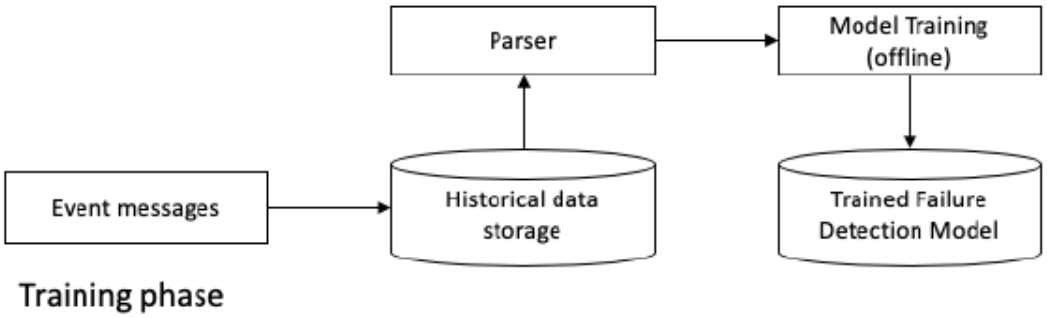

Figure 2

Referring to Figure 3, the key-value data is converted into a matrix where each row corresponds to one log message and the columns are organized in the following way: if a field of the structured log has numerical value, then its value is kept in a single column, if a field of the structured log has categorical value, then it is turned into multiple columns where each column corresponds to one possible value of the field. Thus, the value of the column can be either 1 or 0 depending on the existence of the value in the particular log message. The timestamp of the log is converted to an integer number and listed in a separate column. The rows of the matrix are further aggregated by time (at a tunable time interval), where multiple statistics of each columns can be extracted (e.g., minimum/maximum/average number of "error"-word per second. Note that if a column has 3 statistics, it becomes 3 columns after processing). The above process will result in a matrix where each row is considered as a "sample" and each column is considered as a

"feature". The link failure messages are identified from the original logs and their corresponding timestamps are noted as failure points.

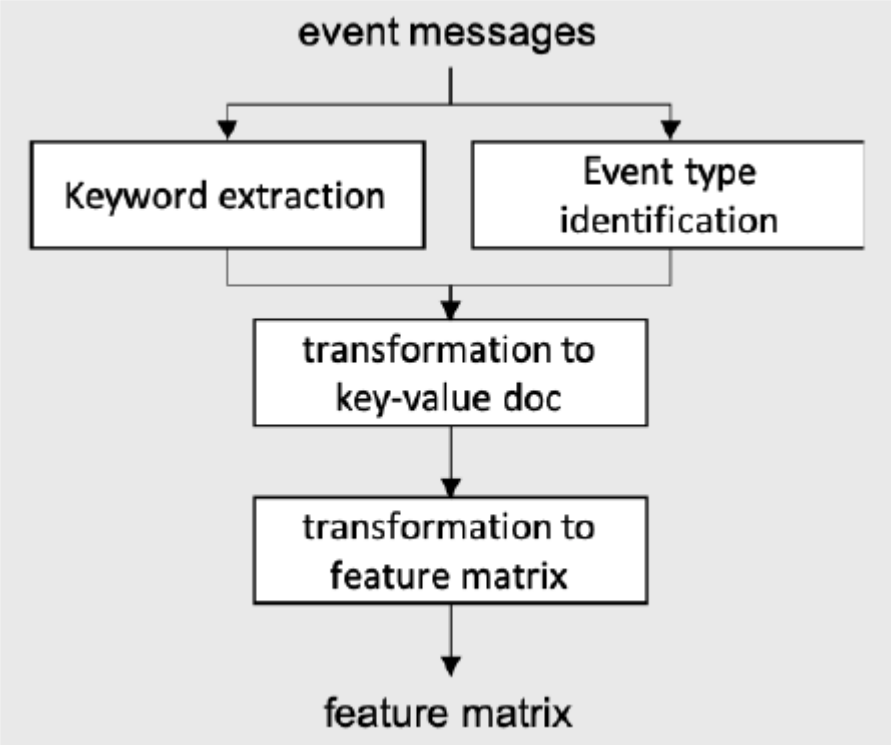

Figure 3

2. Model Training and Tuning

For each failure point, a portion of samples preceding the failure point are filtered out and labelled as "pre-failure" samples. There can be multiple sets of pre-failure samples depending on the tunable length of time preceding the failure, e.g., 5 sec, 1 min, 10 mins, etc. All other log messages are labelled as "normal" samples. A classification model can then be fit that can separate the normal samples and the pre-failure samples based on their probability distribution.

The sample aggregation interval and the pre-failure length mentioned above are both tunable. Different combinations of the possible values are tested in experiments and the best combination resulting in highest model accuracy may be chosen.

Referring to Figure 4, once the model is trained, it can be deployed to run inference in real-time. The inference phase will perform similar a pre-process and feature extraction

for new incoming logs but in a stream manner. The trained model is then applied to the processed data to return the possible category (e.g., "normal" or "pre-failure") of the logs from a given time period.

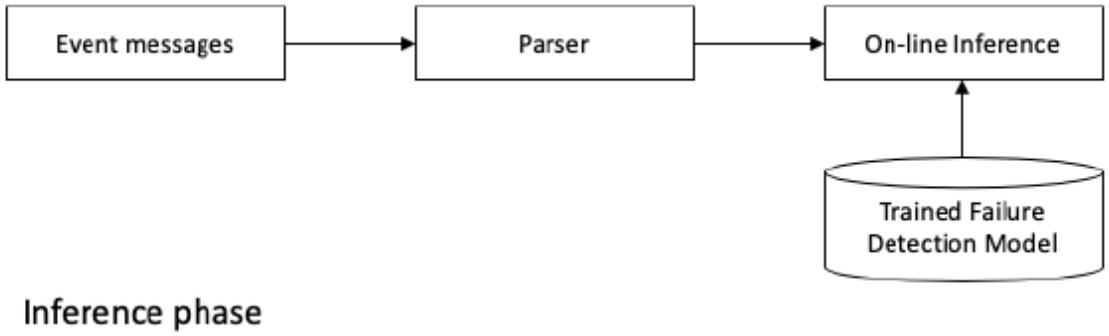

Figure 4

Using techniques described herein the following may be achieved:

- Transfer of relevant indicator information from network device to controllers Network Data Platform using batch or streaming data;
- Continuous monitoring of possible indicators for link failures;
- Prediction of failure detection;
- Prediction of impacted port; and
- Taking remedial actions for network link failure to avoid a network outage.

In summary, techniques are described herein for automatic link failure prediction and detection with an automatic troubleshooting capability to quickly identify switches/ports involved in the failure before the failure leads to a complete outage. A framework to predict the failure and easily handoff all routing protocols to an alternate path is provided. Such prediction helps a network to react before a failure occurs. Almost zero traffic loss during a failure may be achieved by predicting the failure in advance.