January 22, 2019

IDENTIFYING NETWORK FAULTS IN DATA CENTERS USING A COVARIANCE MATRIX

Raghu Rajendra Arur

Nicholas Basker

Hasmit Grover

IDENTIFYING NETWORK FAULTS IN DATA CENTERS USING A COVARIANCE MATRIX

AUTHORS:
Raghu Rajendra Arur
Nicholas Basker
Hasmit Grover

ABSTRACT

It may be difficult to identify root causes of protocol failures or degradations in application traffic performance due to errors. Embodiments presented herein identify the exact diagnostics needed to perform a root cause analysis. In particular, a covariance clustering methodology is used to generate correlated errors, which, when passed through a partial order set-based rule engine, are used to determine the root cause of a network fault due to protocol errors.

DETAILED DESCRIPTION

In a large distributed fabric, there may be thousands of switches, with each switch having tens of interfaces that may each run different Layer 2 (L2) and Layer 3 (L3) protocols, as well as application workloads. In such complex systems, zeroing in on the cause of a protocol error may take several hours. For example, an error in Layer 1 (L1) might cause protocol issues in L2 and/or L3. An error in L2 may cause L3 protocols to flap. Similarly, L1 issues, such as cyclic redundancy check (CRC) errors or buffer overruns, may increase the latency of application workloads. Complex protocols like Intermediate System to Intermediate System (ISIS), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP) maintain state with respect to the state machine. These states can also be correlated, as each state may be dependent on the other states when running. In a modern networking infrastructure, there is a multitude of raw telemetry available. For example, in switches there are counters for CRC errors, Link Layer Discovery Protocol (LLDP) flaps, and Link Aggregation Control Protocol (LACP) flaps.

Furthermore, there may be state machine error counters for L3 protocols, and counters for packet counts and latency between end-points and end-point groups. Thus, there is a need for discovering correlations on-the-fly in order to diagnose the problem. Present embodiments provide an algorithm to detect correlated clusters to find the root cause of issues using partial order based rule engine.

Correlating application latency to interface level errors is unintuitive, and may require hours of debugging to arrive at the cause of the problem. As networks grow in complexity and run different overlay protocols, correlating errors at different layers becomes very challenging. Embodiments presented herein propose a new algorithm for identifying the root cause of a problem and provide exact diagnostic details in real time using a covariance matrix and Pearson's correlation coefficients to identify a cluster of related issues. This cluster of related issues is passed through a recommendation system that has a set of rules defined as partial order sets that emit out the exact cause of the problem.

Present embodiments analyze network telemetry that is streamed out from switches, which is light-weight compared to system logs. Instead of building templates that are learned after reviewing multiple incidents, present embodiments utilize an unsupervised learning approach wherein correlated events are learned on the fly using Pearson's correlation coefficients, and then run through a simple rule engine. Unsupervised learning is combined with partial order set rules to identify the problem in near real-time. Since plain telemetry data is used, present embodiments are highly efficient in terms of processing speeds and resources required to accomplish the root cause analysis.

First, required telemetry is streamed out of switches and spines to one or more telemetry servers, where it may be stored on a data lake. A machine learning-based analytic engine runs on top of the data collected to find the faults and provide diagnostics. Figure 1 depicts an example of data pipeline architecture in accordance with present embodiments.

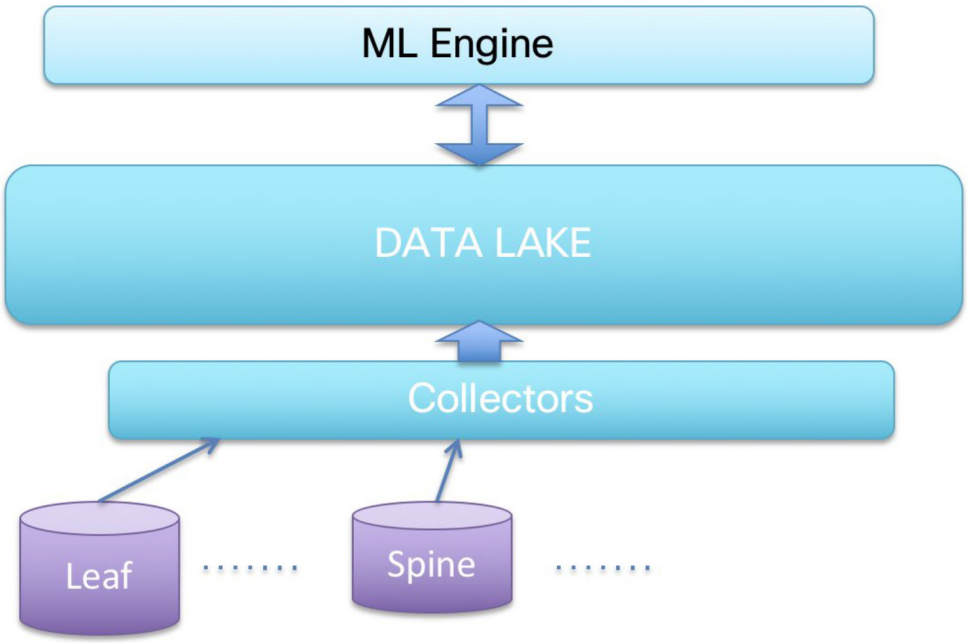

Figure 1

Error counter information is streamed to the datalake. For every error counter that is streaming data, tags are generated and kept in an error database for later user. Figure 2 depicts an example of an error database in accordance with present embodiments.

Error	Tags
Crc Error	L1 Layer, Interface
AFD	L1 Layer, drops, Interface
LLDP:Flaps	L2 Layer, lldp
Lsp:AuthError	L3 Layer, isis, Authorization, error
Csnp:MiscError	L3 Layer, isis, error
….	….

Figure 2

The streamed data is split into different overlapping windows. Data that is present in each window across all features is used to find correlations. Figure 3 depicts an example of windows of statistics that are used to find correlations in accordance with present embodiments.

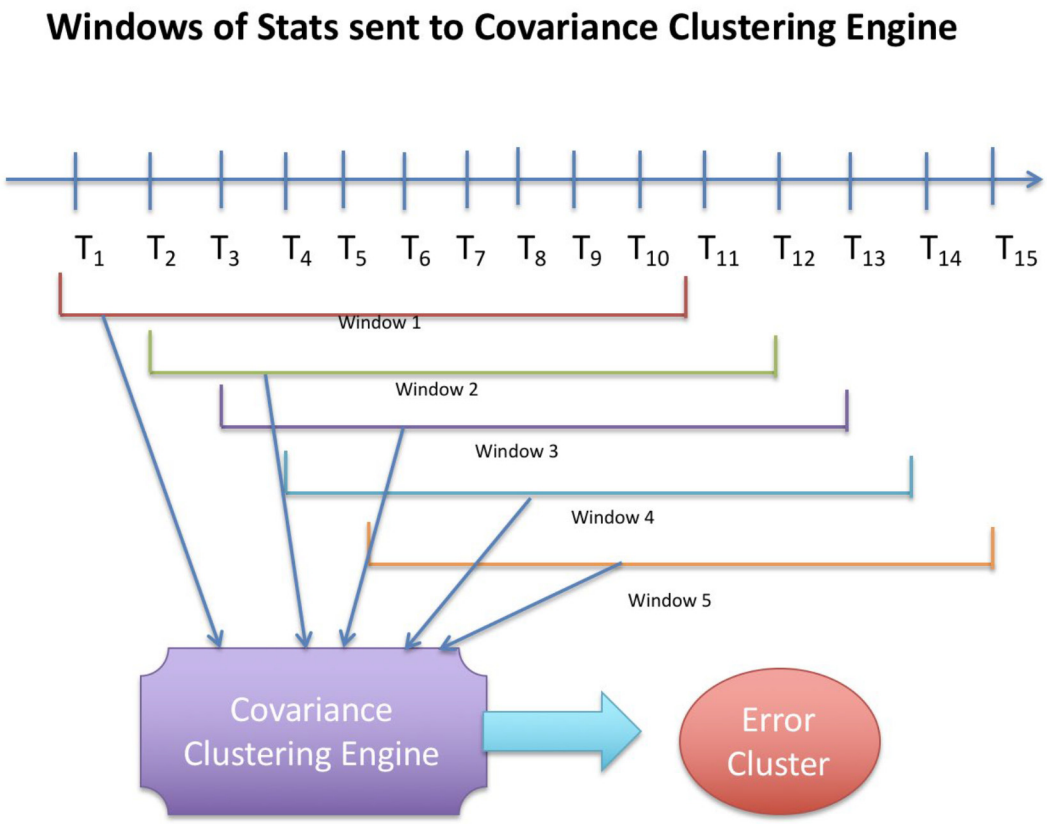

Figure 3

Figure 4 depicts an example of a covariance clustering algorithm in accordance with present embodiments.

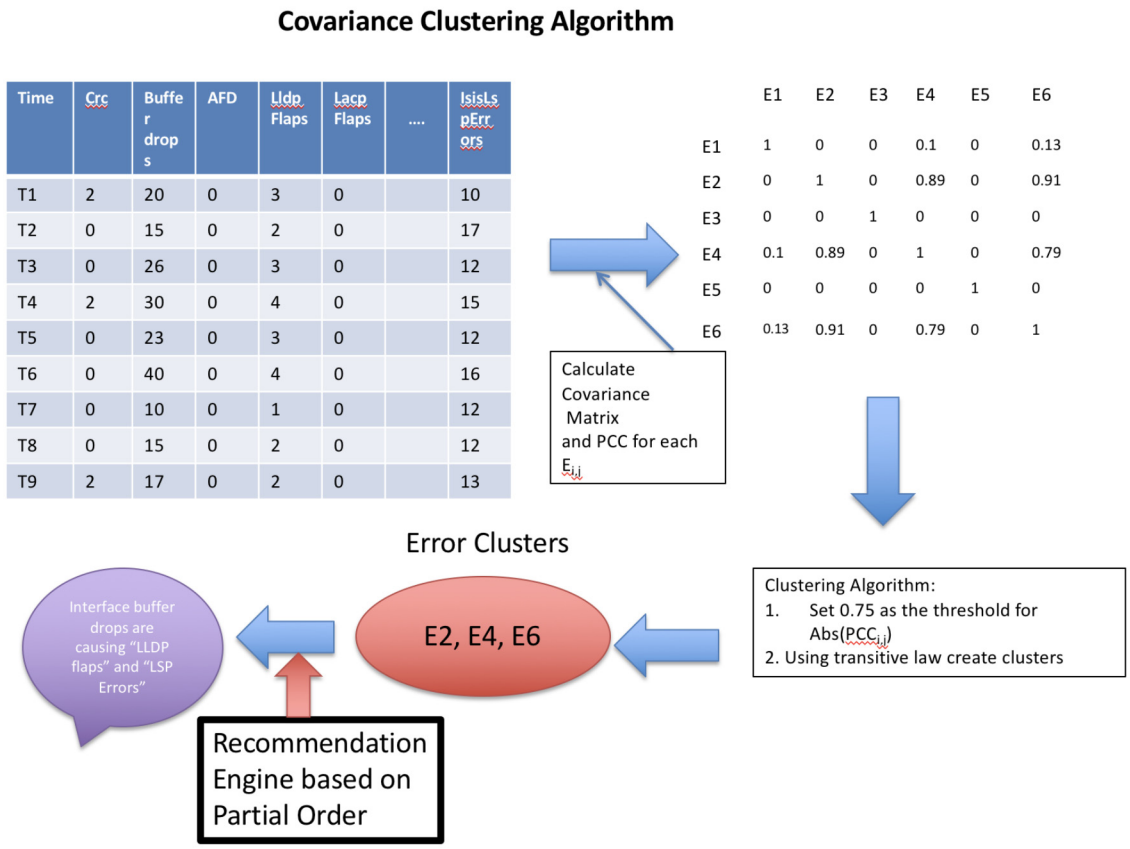

Figure 4

L1 errors may include CRC errors, buffer over flow, approximate fair dropping (AFD) drops, and high interface utilization. L2 errors may include LLDP flaps, Bidirectional Forwarding Detection (BFD) flaps, LACP flaps, and the like. L3 errors may be received from ISIS counters (including label switched path (LSP) errors, partial sequence number packet (PSNP) authorization errors, etc.). Similarly, there are many error counters for OSPF and BGP that are interface-level counters, and there are other L3 area level error counters.

The data that is streamed out includes the number of error counters that are seen within a specific period of time. These counters are indicators of a problem with respect to a protocol. The error itself may not describe a symptom, and thus does not give information about the root cause. For example, when L3 protocol errors begin to appear, the errors

might be due to some protocol state changes, or because of L2 layer protocol flap, or because of L1 errors.

Any available error counters may be streamed at a predefined frequency (e.g., every 10 seconds). The time series data received is divided into overlapping moving windows (e.g., 10 minute windows with an overlap of 1 minute offset). A Co-variance matrix is created across these features. A covariance matrix is a matrix whose element in the *i*, *j* position is the covariance between the *i*th and *j*th elements. Covariance between two features X, Y is given by:

$$COV(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

Pearson's correlation coefficients (PCCs) are calculated for each (X, Y) entry in the co-variance matrix using the formula:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where cov is the covaraiance, σ_X is the standard deviation of X, and σ_Y is the standard deviation of Y.

A correlation matrix (or PCC Matrix) is generated from the covariance matrix. PCC values always lie between +1 and -1. The stronger the association of the two variables, the closer the Pearson correlation coefficient, *r*, will be to either +1 or -1, depending on whether the relationship is positive or negative, respectively, which is useful in identifying errors are that are linearly related to each other.

In some embodiments, while creating PCC matrix, the absolute value of the PCC values are used instead of regular PCC values, as a positive linear relationship or negative linear relationship are of equal importance. In the case that the standard deviation of either X or Y is 0, then the PCC value may be set to 0 to avoid not a number (NaN) error values in the matrix. In a properly-functioning network, most of the error counters are 0. Hence, this minor modification helps in telling that there is no correlation between two independent features.

An important observation that can be noticed is that PCCs have a transitive property. For example, if X and Y are linearly correlated and Y and Z are linearly correlated, it can be inferred that X and Z are also linearly correlated. This important

observation is used in creating clusters of features or error variables that are dependent on each other. While creating clusters, only those entries having a PCC value of greater than or equal to 0.75 are considered in order to ignore any weak signals.

Next, a recommendation system is used to generate pinpointed diagnostic messages. The rule engine or rule table is pre-defined using the domain knowledge that describes the partial order set. Figure 5 depicts an example of a partial order set that provides dependency information which can be used for root cause analysis in accordance with present embodiments.

Rule Evaluation Order	Partial Order
1	Layer 1 < Layer 2 < Layer 3
2	Csnp:AuthError < Csnp:MiscError
3	Csnp:Error < LAN:Error < P2p:Error
…	…

Figure 5

The rules are evaluated in the evaluation order provided in the table. Each error cluster has an error counter or the feature name. The tags associated with each error counter are fetched from the error database, and the rules are evaluated with respect to the tags. The first rule that receives a hit in the table provides the partial order set or the dependency information. Using this partial order set, a diagnostic message may be generated.

Using this approach, it is also possible to detect application performance issues. One of the observations that can be made is that application traffic is mostly dependent on the health of L1 in order for data traffic to flow. For application performance issues, L1 counters, latency counters for end-point groups, and packet count counters for end-point groups are considered. A similar PCC matrix is calculated on the selected features, and clusters are generated. Based on these clusters, the cause of performance issue can be quickly discovered using the recommendation engine.

Thus, an online unsupervised learning system using a Pearson's correlation coefficient matrix (or covariance matrix) identifies relationships between statistical error counters that are collected across various networking layers and protocols. This algorithm assumes no domain knowledge when clusters are generated. The clustering algorithm works on a time-series data which can be visualized as wave forms. Whereas traditional clustering algorithms, such as k-nearest neighbors (kNN), mean shift clustering, expectation maximization clustering, etc., work on distance as the parameter to form clusters that are not suitable for time-series data or wave forms, present embodiments use structural relationships of time-series data to identify strong correlations to be considered as parameters for clustering.

The clustering method presented herein provides correlations between error features. As these clusters are generated without domain knowledge, it is possible to create a partial order set of rules based on tags. The partial order set of tags are a high level relationship between errors. Generally, a rule set can be definitive; for example, “if A, B, and C occur, then the recommendation is X," or "if A and B occur, the recommendation is Y". When the number of features that are being monitored grows, the rule set becomes correspondingly large. Partial order set-based rule descriptions enable minimalistic dependencies on tags, which may be few in number. The tags of the clustered errors are used to generate the appropriate recommendation.

Additionally, using a domain expert-provided order of anomalies as a rule engine to match cluster of dependent errors and thereby identify true positives enhances the authenticity of the algorithm.