ADT: AI-Driven network Telemetry processing on routers

Parisa Foroughi, Frank Brockners, Jean-Louis Rougier

ADT: AI-Driven network Telemetry processing on routers

Parisa Foroughi[†‡], Frank Brockners[‡], and Jean-Louis Rougier[†]
[†] LTCI, Télécom Paris - Network and Computer Science Department [‡] Cisco Systems
Email: parisa.foroughi,rougier@telecom-paris.fr; pforough, fbrockne@cisco.com

***Abstract*—Network monitoring is a pivotal part of network management and operations. It is responsible for monitoring the behavior of the network to assure its functionality within expectation and to guarantee a smooth-running environment for enabling of various services. Therefore, operators are interested in gaining a comprehensive assessment of their network elements and tracking operational changes to facilitate timely correction of any deviation. Commonly, this assessment is achieved by performing regular manual checks of different operational counters and defining expert rules from known root causes. The common approach requires the maintenance of a regularly updated set of rules and only goes as far as the operator's pre-gained knowledge of the system. With the growing complexity of the networks as well as the availability of more data, a more efficient monitoring approach is necessary to address the emerging network monitoring requirements.**

In this paper, a novel unsupervised approach is proposed that is capable of exploring a broader set of counters (not limited to the handpicked Key Performance Indicators (KPIs)). The goal is to leverage the dependencies between the counters in order to discover complex state changes that might have otherwise slipped the operator's view. This paper proposes ADT, an AI-driven telemetry processing solution that facilitates monitoring of a larger set of counters. The Detector block of ADT is known as DESTIN, a multivariate unsupervised change detection for high dimensional time-series data of originally low effective dimension, which provides near real-time state assessment of network devices. The efficiency of the proposed approach is demonstrated and compared with well-known methodologies on an experimental test-bed. The method's performance is also explored extensively considering different criteria such as traffic type, device and the type of events to identify its potentials and limitations. The datasets used for the evaluation are made publicly available.

***Index Terms*—Machine Learning, Principal Angles, Network Management, Change Detection, Network Monitoring.**

I. Introduction

Network monitoring facilitates the network operators by providing the insight necessary to maintain the desirable network state. Therefore, Enterprises spend a large amount of their budget on manual monitoring of their networks [1]. A common approach is for an operator to continuously monitor a few known operational counters [2], [3] also known as Key Performance Indicators (KPIs). This approach limits the operator's view to known operational state changes. A network element such as a router or switch can offer hundreds of thousands of operational counters to monitor [4] which can be retrieved by the Simple Network Management Protocol (SNMP) or via streaming telemetry [5]. The set of counters which is available but not analyzed could contain additional information, which the proposed method in this paper (ADT), is able to distill. AI-Driven network Telemetry (ADT) proposed in this paper is a step towards automation of network monitoring.

While networks are constantly growing in size and complexity, the network monitoring approaches has not undergone any significant improvements. The future networks are expected to host more services and users and carry more data. Accordingly, operators need to collect, analyze and store more data for accurately assessing the state of their network and its devices to track the operational changes. Thanks to the advancement in cloud technology and its wide-spread usage, the processing power, bandwidth overhead and storage capacity may no longer pose as major concerns for the enterprise in monitoring their network. However, the identification of important events among the vast measurements and their localization still pose a significant challenge [1].

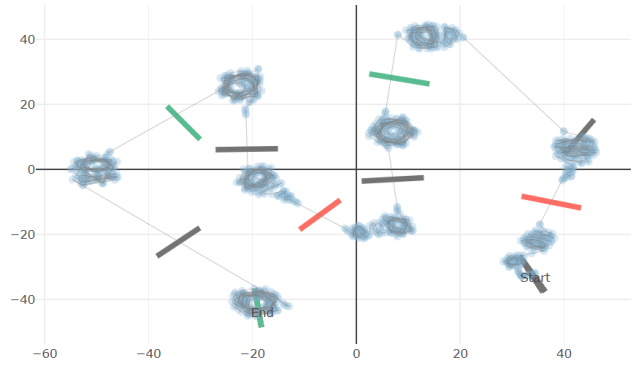


Fig. 1. t-SNE visualization of a dataset [5] of 6622 counters with perplexity=30, the bold bars are intentional inserted events- The x and y axis values are relative

Machine learning approaches have almost revolutionized various fields such as image and pattern recognition, natural language processing etc [6]. Yet, in practice it can be argued that it is still in its early stages in the networking community.

The lack of properly labeled datasets as well as the specificity of networks are some of the fundamental reasons that the application of artificial intelligence (AI) and machine learning (ML) is lagging behind in this domain. In addition, the need for an online assessment of the network state makes the application of machine learning approaches even more challenging. To confirm the potential in application of AI methods in the datasets of this paper, an offline analysis using one sample labeled dataset is performed. The dataset

consists of operational counters (interface traffic, bgp counters, bi-directional forwarding detection counters, CPU related counters, infrastructure statistics etc.), collected with a regular cadence (10 seconds) using model driven telemetry (MDT) available on Cisco routers. The dataset includes m time-steps/data-points with n columns of counters. To visualize the clusters formed from the high-dimensional data several methods such as t-stochastic neighbor embedding (t-SNE) [7], Uniform Manifold Approximation and Projection (UMAP) [8] can be utilized. Fig. 1 visualizes a dataset from a router ($n = 6622, m = 1079$), using t-SNE, revealing a set of clusters as expected. The events are inserted in real-time during the collection process. The full set of collected counters are made publicly available [9]. T-SNE is a non-linear visualizing technique for visualizing high dimensional datasets with 2 or 3 dimensions. It consists of two main steps. First, it calculates a probability distribution of data-points by calculating the pair-wise probability of the points. These probabilities are assigned in a way that the more similar points will have higher probability and the dissimilar ones will have lower probability. In the next step, it optimizes the Kullback–Leibler divergence (KL divergence) to find another probability distribution in the lower (2 or 3) dimension. The perplexity parameter in t-SNE can be considered as a parameter that balances the attention between the local and global aspects of the data. In other words, the perplexity parameter in a sense is the number of expected neighbor points for each point. In Fig. 1, one can see that the state of the network element, as represented by the n-dimensional vector either does not change much and thus, stays within a *cluster* or shows a significant change. Note that as can be seen in the figure, the changes of clusters align very well with the events inserted (shown by bold lines in the figure). Therefore, it clearly shows that the collected operational data forms clusters of points when a change happens.

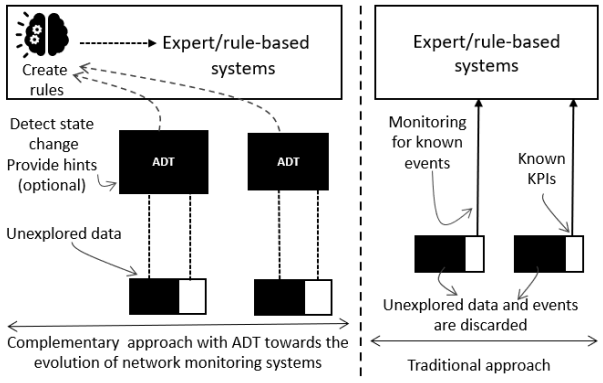


Fig. 2. The complementary scenario of DESTIN [10] in expert and rule-based systems for improving the operators insight of the network element

To improve the efficiency and the accuracy of the network monitoring, this paper introduces a novel online and unsupervised approach called ADT. Fig. 2 depicts the complementary role of ADT with legacy expert and rule-based systems in improving the operators' knowledge of the state of the network element. ADT consists of 4 components; collector, detector, explainer and exporter. In the proposed scheme shown in Fig. 2, DESTIN [10], the detection components of ADT, explores the otherwise ignored counters and analyzes their dependencies in an unsupervised way. Thus, sending a notification when a change of state is discovered. The operator can further analyze the flagged timestamp from DESTIN using data-driven methods such as the one proposed by Feltin et al. [11] to distill the most representative counters and provide a hint for operators to define new rules and KPIs. Note that though the explainer block is one of the components of ADT, it is an optional component that provides hints for understanding the detected changes. This paper is focused on the obligatory components of ADT to keep the content organized and efficient.

The contributions in this paper are summarized as follows:

- Proposing an AI-driven network telemetry data processing framework for detection and explanation of device state changes.
- Introduction of a new unsupervised online change detection approach for networking time-series data using a geometrical interpretation. The proposed method is capable of assessing the state of devices in a timely manner without requiring huge datasets for training purposes.
- Providing extensive details regarding the pre-processing considerations for streaming telemetry data and the potential approaches in usage of different types of data.
- Proposing measures to improve the dataset labeling and facilitate a more accurate and transparent evaluation of the detection methodologies.
- Open-sourcing the datasets collected via streaming telemetry used in this paper for evaluation.
- Exploring and providing insight on the quality of the datasets collected and used in this paper as well as accuracy of the proposed measure.
- Evaluating the proposed methodology in terms of recall, precision and delay to detect an event/change. The performance of the method is compared to a well-established method in networking change detection and its performance is broken down by event-type, device, and traffic type. The False positives and False negatives are then root caused manually to shed light on advantages and potential blind spots of the method.

The rest of the paper is organized as follows: a comparative overview of the state of the art is given in Section II. The practical architecture for the method application is depicted in Section III. Section IV expands on the various approaches on preparing data as well as highlighting some of the data's properties. Section V explains ADT's core methodology in terms of parameters, applicability etc. using a geometrical interpretation. Section VI includes information regarding dataset generation, terminology and an evaluation framework showcased with numerical results. Section VII expands on the results and provide insight on shortcomings and root causes of the missed or wrong detections. Section VIII concludes the paper.

II. Related Work

This section provides some of previous main works on network monitoring tools for general-purpose traffic monitoring. As the major concerns in monitoring is capability of assessing the network state and distilling information from the pool of data, the main focus of the state of the art is dedicated to comparing the existing methods of same mathematical background with the method proposed in this paper. In the last part, some examples of exploring networking data of different domains are also presented. In this section, a brief review of methodologies that can be used in change detection is introduced. Unfortunately, not all of them are applied, tested and explored in practice. Thus, though they have demonstrated the potential of the methods, they have not tested their approach in generic context to facilitate an implementation.

General purpose network monitoring [1] can be performed on different data sources such as traffic flows and streaming measurements. Based on the available data source, various monitoring approaches has been proposed. Fullmer et al. [12] and Trammel et al. [13] propose OSU flowtools package and NAF, respectively, as open-source tools for collecting, processing and monitoring of network flows and flow logs. Plonka et al [14]. propose FlowScan as a flow monitoring tool capable of aggregating flows per protocol and per port and presents the flow statistics as plots as a result. Another approach to general purpose monitoring are tools built for streaming measurements. Sellens [15] propose Thresh, a threshold based approach on SNMP data and provide a framework for visualization and data analysis. Kim et al. [16] propose NetViewer to process SNMP data and trigger alarms based on predefined thresholds of normal traffic behavior. Beverly [17] propose RTG as a tool for network monitoring that detects deviation of certain traffic from their predicted values based on the history observed as well as the methods of time-series forecasting. The mentioned tools either use rigid definitions on normal traffic for defining thresholds or attempt to perform monitoring based on one-dimensional time-series analysis. Whereas, ADT is an AI-driven network monitoring tool that can process streaming data in an online and multi-dimensional manner on the router and provide an estimation of the network device state. ADT leverages the existing advancements in technology in collection and processing power of routers and switches and detects the device's state changes in a timely fashion.

In addition to the above, there are a few works that attempt to take a step further than detection. These solutions often can recover the network from certain known events by enhancing the control plane. Koponen et al. [18] introduce Onix as a general distributed control platform for switches. The authors utilize the concept of software-defined networking (SDN) and exchange network state information files called NIBs. The concept of *state* in this paper is defined as a graph of all network devices and their connections which is different from our definition of state in this paper. Zhou et al. [19] propose, Tardis, a fault-tolerant design for network control planes. They introduce a root-causing algorithm for bugs based on their model and attempt to recover the network state using the proposed model. Curtis et al. [20] propose a modified version of OpenFlow called DevoFlow which cuts down on the controller overhead while maintaining the global visibility of the network. Yang et al. [21] propose FOCUS to allow the logically centralized SDN controllers to scale for larger networks by offloading a subset of control functions to the switches. It is noteworthy to mention that, our study does not cover the automatic recovery of the network to the original state by proposing a control plane framework and only informs the operator of the occurrence of any events. Note that although it is not the main focus of this study, our definition of state and the extracted information at the time of the event could be a potential message template for network controllers. We believe that it can potentially decrease the messaging overhead in controllers, however, more exploration and experiments are required to reach any definitive statements. The following will compare the existing change detection methods with ADT's core methodology (DESTIN).

The state of the art methodologies based on subspaces and matrices are closer in principle to DESTIN which are the Principal Component Analysis (PCA) [22] and covariance matrix-based methodologies, respectively. Lakhina et al. first proposed Principal Component Analysis (PCA) [22] as a technique that is applicable to link-traffic data obtained from SNMP that is capable of detecting network-wide anomalies. The authors refined their approach by proposing a combination of the PCA subspace with a distribution based technique using entropy [23] to detect intrusion attacks for flow data. Ringberg et al. [24] state the limitations of PCA for traffic anomaly detection. The sensitivity of the false-positive rate to the number of principal components, the contamination of normal subspace, the hardship in identifying the right flows, and the right traffic aggregation level for input are the main axis of the paper. Brauckhuff et al. [25] tackle the limitation of the PCA in detecting temporal changes. They propose to use Karhunen-Loeve transformation [26] along with the covariance matrix. Kwitt et al. [27] propose a robust PCA transforming it into an unsupervised approach by using Minimum Covariance Determinant (MCD) estimators. Cardot et al. [28] investigate and provide a comprehensive view of how to use PCA for online change detection in high dimensional data. Vaswani et al. [29] investigate the different approaches to improve the robustness of the PCA and validate their proposed approaches on images and social networks. Ding et al. [30] propose a compressed version of PCA, assuming a spiked covariance matrix for high dimensional data. Their proposed method can track temporal changes and is at the same time an online solution while suitable for high dimensional data of low intrinsic dimensions. Li et al. [31] proposed a PCA-based method for flow data and identification of the IP flows. Despite all the efforts to improve PCA, combining all the approaches is not trivial if at all possible. Moreover, some of the enhancements use rigid constraints for data that may count as over simplification considering the nature of networking data. The proposed method in this paper can detect changes in an online unsupervised manner with higher precision compared to PCA. It is also robust enough to filter out noise and symmetric spikes in traffic data. The results in Section VI-C depict the

improvement over PCA in detection of changes of states.

Another group of techniques similar to DESTIN are covariance matrix-based approaches. Yeung et al. [32] propose to detect flooding attacks by tracking the covariance matrix and correlations of data. Haung et al. [33], [34] propose a communication-efficient approach to apply PCA to a network. They define a filter based on the covariance matrix and stochastic perturbation theory to only send information from a router to a coordinator when necessary. The same authors propose a new approach to dimensionality reduction based on covariance matrix properties and a distance metric [35]. Compared to the prior work, this paper consumes the original input data rather than computing the covariance matrix. DESTIN is computationally efficient in comparison with the state of the art approaches because of its capability of working over small windows of data-points.

Principle angles which is the core of DESTIN have been originally investigated in the context of auto-regressive-moving-average (ARMA) models [36] to find the hidden relations between the linear systems and their input parameters. In addition to ARMA models, Yuan et al. [37] and Lie et al. [38] use canonical correlations which is an extension of the original principal angles method for information fusion and similarity discovery between groups of variables. Wolf et al. [39] also introduce an improved principal angles that facilitates searching for non-linear patterns by using a kernel. The principal angles technique has been previously used in information fusion and control systems for linear systems but it is not yet used for detecting changes in high dimensional multivariate data. This paper builds on the method capabilities and provides guides on how to customize the method for change detection.

In addition to the subspace methods other techniques has also been used in anomaly detection. Leung et al. [40] proposed an unsupervised density and grid-based approach for anomaly detection. The authors explore different pre-processing steps and the algorithm's time complexity. However, they do not explore the parameter tuning and the generalization of the methodology. The method's time complexity is linear with the number of features. Canbalaban et al. [41] proposed an algorithm based on cross-layer neural network for intrusion detection which performs reasonably well based on their tests. Yet, they use a totally different set of features compared to the ones used in this study and they require training with labeled datasets. Hassas et al. [42] propose Kandoo as an scalable control plane that detects elephant flows based on tuned thresholds and propagate the event to other switches. Although the authors' main goal and the data used are different from this study, they do use a threshold-based method for detecting changes which suffer from the mentioned shortcoming in the Section I. Jain et al. [43] proposed a trend-based methodology for automated load-balancing. The authors limit their set of events to deviation from an average traffic volume. They then determine the trend of deviation to take appropriate measures to balance the traffic. Compared to Jain et al. [43], the method proposed in this paper is capable of detecting more complex changes. laskov et al. [44] apply a set of known supervised (C4.5 [45], k-Nearest Neighbor [46], Multi-layer Perceptron [47], Regularized Discriminant Analysis [48], Fisher Linear Discriminant [49], Linear Programming Machine and Support Vector Machine [50]) and unsupervised algorithms (k-Means Clustering [51], Single Linkage Clustering [52], Quarter-Sphere Support Vector Machine [53]) on the well-known KDD cup dataset and conclude that the problem of test data being drawn from a different distribution cannot be solved within the purely supervised or unsupervised techniques. In the domain of cable broadband networks, Hu et al. [54] propose a statistical method to generate (feature engineer) new features from the time-series and uses the customer trouble tickets as hints to infer the abnormal thresholds. However, their feature space is tremendously different from our work. Al-Musawi et al. [55] in their survey explore 20 works on border gateway protocol (bgp) anomaly detection. They categorize the state of the art by the data source (features) and the type of anomaly itself. They claim that none of these works offers a combination of detecting in real-time for all types of bgp anomalies, differentiating between them, and identifying the source cause of the anomaly.

III. Architecture and setup

This section sheds light on practical considerations of ADT's deployment on a router. Fig. 3 depicts the building blocks of a working scenario of ADT. It also shows how DESTIN can be deployed in conjunction with the existing technology and breaks down its important building blocks. DESTIN can be deployed either on the router or on a server which receives continuous data streams from the router. The proposed scenario in Fig. 3 has three main building blocks: *collector*, *detector* and *exporter* plus an optional block of *explainer* that can help the operator in defining new rules based on the hints it provides. The collector is responsible for data retrieval from a router and for parsing the data to time-series (Fig. 3). The collection mechanism can be a simple periodic SNMP data retrieval mechanism or it can use streaming telemetry to form regularly time-spaced collections of a router statistics in forms of time-series. The detector (DESTIN) takes care of pre-processing which consists of first grouping data based on the operators' knowledge or data-driven data patterns (*e.g.*, interface level traffics) and then scaling the time-series data to make differently loaded interfaces comparable in value. The more diverse the patterns in one data group, the more caution have to be taken in pre-processing the data. The explainer/descriptor is an optional component which is triggered by the notification generated by DESTIN. It can provide hints to the operator regarding the most representative counters associated with the event. The exporting block functionality depends on the overall on-box or off-box approach. In case DESTIN is deployed on the box, the exportation is event-triggered and so there is no need for continuously exporting counters out of the router. ADT is currently available on Cisco XR routers [56] as an extension of their streaming telemetry solution [57]. In the remainder of this section, some more information regarding the implementation of ADT on Cisco routers are provided.

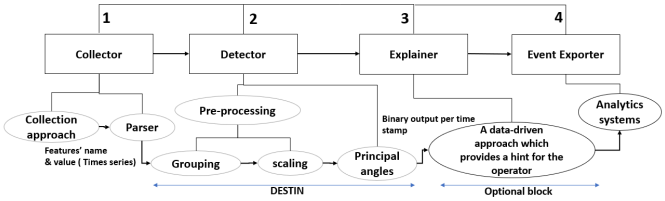


Fig. 3. Building blocks' details of ADT for state assessment of network element- optional, existing and proposed blocks

A. Collector

The counters are collected on the box through subscriptions to the streaming telemetry [58] with a regular cadence. The groups of counters to subscribe to are inferred from the configuration of the router (i.e. bgp, ipv6, icmp, etc.). The data collection cadence is automatically tuned according to the router's available resources. There also exists a parser code block which synchronizes all the counters from different subscriptions. The output of this block are equally spaced time-series ready to be fed to the detector block.

B. Detector

The detector block is responsible for data scaling and the detection of state transitions. The time-series data is first pre-processed according to its operational type. The processed data is fed to the detector (DESTIN) which generates an indicator representing its input. The indicator, in a sense, is a translation of the information about the device state from various counter into a uni-variate indicator. DESTIN consumes multivariate data and detects state transitions timely without any need for prior knowledge about the network. A detailed description on the mathematical basis of this indicator is presented in the Section V. The stability of this indicator infers the stability of the relationship in its input data. As long as the indicator does not undergo an extreme divergence from its values, the device state is assumed to remain unchanged [10]. To automate the monitoring process for this indicator generated by the detector, a sigma detector is applied on this uni-variate output as detailed in Section V-B5.

C. Explainer

This block is responsible for selecting the counters which can better explain the event occurred at the timestamp flagged by the detector as an event. This block looks into all the available counters and presents hints for the operators to justify the event. More details are available in [11]. Though it is optional block it can reduce the effort necessary to assess the device state.

D. Event exporter

The system only exports data out of the router when a change (event) is detected. To export the counters and the detector details, a customized group of YANG model [59] is defined. The defined YANG model file includes the detector internal details such as the change indicator, binary detector parameters and the selected features from the explainer. This model has to be subscribed to when running the analytics system to facilitate the exporting of data in an event-triggered manner.

In this paper, the detector block is explained and its functionality is explored extensively. The explainer is out of scope of this paper and is deserving of its own spotlight. Feltin et al. [11] provide an example of application of such methods. In addition, the explainer block is introduced as an optional that improves the operator's experience but its absence does not impact the ADT's output drastically. In the explainer's absence, all counters around the detected change could be exported. In the explainer's absence, we are still exporting data in an event-triggered manner, the only difference is the amount of the exported data. A demo scenario of ADT with technical details on collector and exporter is provided in [60].

IV. Pre-processing

In this section, without going into the method's details, the pre-processing constraints and options for DESTIN are discussed. DESTIN's mathematical and geometrical interpretation is then provided in Section V.

Pre-processing is one of the inseparable steps of working with any data. Yet, necessary considerations in this step are often neglected or restricted by the complexity of operations in online processing as well as the pre-processing's entanglement with its follow-up method. Therefore, often, general assumptions on normalization (z-normalization, etc.) and scaling (min-max, etc.) are made. These assumptions, although harmless in appearance, if not aligned with the context of the data, can mask important information or highlight trivial ones. Principal angles, the core methodology of DESTIN, on its own, is easy on its imposed restriction for pre-processing. However, for better and more precise results, going through several additional steps could improve the performance of the method. The collected data from the router can be consumed by DESTIN in one of the following formats:

A. Raw data consumption

The method can digest raw data (i.e. parsed and synced as time-series). The method itself does not impose any constraints such as wide sense stationary (WSS) properties or any specific mean or standard deviation for data consumption. Yet, it is noteworthy to mention that in this form the huge difference in the counters' amplitude is the main factor in creating bias when detecting changes. In case of considerable differences in counters' amplitude, the changes in the counters with smaller value are less likely to impact the change indicator and thus some changes related to only such counters may be missed by the detector.

B. Grouped data consumption

A better way of data consumption is the grouping data into different categories based on various concepts. Grouping, on one hand can provide a sense of importance for each group based on the groups features and their networking importance,

on the other hand, it can allow for flexibility in the pre-processing of different groups based on the data properties and/or networking context. Some examples of such cases are provided in Section IV-D. It is important to note that what is proposed here when grouping data is much simpler than feature engineering where new features are formed from the original ones. In this paper, the features are simply grouped based on the already existing YANG model by using some keywords for filtering the features. It is important to note that the YANG model data structure already integrates the operators knowledge to some extent when defining its groups and subgroups [59].

1) Grouped based on Expert knowledge: When grouping is supposed to provide networking insight on importance of features, unavoidably, it has to be based on expert knowledge. Although it can provide better results, it might be time-consuming, tiresome and in some cases unfeasible for an operator. A relaxed alternative for such cases is filtering the data based on their names using special keywords (i.e bgp, bfd, etc.). This is specially easier when the data collection format is following a specific and known structure such as Yang models [59], which is used in this study.

2) Grouped by data pattern: Principal angles' logic is based on the general pattern of the times-series. The different time-series patterns will have their own representatives in the formation of a subspace presenting the data. Therefore, if the groups are identified based on the time-series patterns the different subspaces are less noisy and can provide more accurate results compared to when a lot of them are put together. To determine the pattern of the features, we require a window of observational data, commonly known as boosting phase, before the method starts the detection process. Depending on the available resources on the device and the complexity of the potential strategies, a proper choice can be made. In this work, instead of relying on more complex methods such as DTW, 2 groups of data, based on experiments, are identified to encompass the majority of the networking features. The reason behind such grouping is simply due to the characteristics of the detector in calculating its change indicator. The two groups are as follows:

(I) Counters with most often constant values and step-like changes (number of up/down interfaces, bfd session count, some of the bgp-related counters, etc.)
(II) Increasing/decreasing counters (traffic counters, etc.).

The first group happens to have less noise and often the change of value *could* potentially indicate either an insignificant or major change in the network, while the other category includes counters which are always changing. A more precise alternative for grouping features based on shape is to utilize Dynamic Time Warping (DTW) [61]. The Equation (3) proposed in Section V-A can be used to calculate the final change indicator when several groups are identified. It is noteworthy to mention that not all step-like changes or rate changes necessarily result in a network event/change. Distinguishing the change in the state of the network (from normal state to change state or from change state to normal state) based on more data is what DESTIN is expected to do. Note that DESTIN does not recognize normal on its own but rather establishes a notion of the network state when it stabilizes based on its change indicator. Also, note that we will use the terms trend, shape and pattern interchangeably in this paper.

C. Scaling

Scaling is often referred to the process of making different data values to be comparable in range for a specific method. DESTIN, like other methodologies, needs the data to be of comparable ranges to avoid the bias in its assessment towards larger-valued counters. In some cases, this bias may be pleasant thus, the categorization may only be necessary for drastic differences (i.e counters being 1000 times the other ones). For instance, a dataset of traffic values could be divided to Gigabytes, megabytes and less categories for scaling before putting them in the same group for method application. This way of scaling will allow the larger values to have more impact while the smaller ones still maintain their effect, accordingly. In DESTIN's case, the preferred scaler is the data average over the recent past. In order to generalize DESTIN explained in Section V-A, some further considerations on scaling is proposed in the relevant sections of V-B3 and V-B4.

D. Datasets' properties

This section sheds light on the datasets' properties that allow for application of DESTIN. It also showcases the effect of different pre-processing and scaling options on different groups of data that support the information provided in Section IV.

DESTIN is a multivariate change detection approach based on subspaces. Multivariate change detection is of interest either when the counters (features) have dependencies and correlations or they are supposed to represent the parts of one system as a whole [62]. Therefore, an important factor in exploring data of multivariate nature for methods based on subspaces is to investigate the effective dimension of that data. If the effective dimension is significantly less than the original space, it is implied that the counters have strong inter-dependencies. One common practice when investigating the effective dimensions of a subspace is to observe the variance ratios on the principle components.

Fig. 4 shows the scree plot for the variance percentage of principal components for BGP and traffic related counters. It also shows that the effective dimension of the subspace is indeed low and most of the data variance is inside the first few principal components. A simple cumulative sum over the scree plots in Fig 4 shows that the number of principal elements to contain 99% of data variance in traffic data is 29 and 3 components for min-max and average scaling, respectively. The same numbers for BGP-data are 338 and 150. It also shows that the best suited pre-processing schemes for different groups of data can be different. For instance, scaling by the average for traffic data in addition to preserving most of the variance in the first few elements has the sharpest knee pattern, while for BGP-data the pre-processing schemes do not change the knee position. However, in BGP data the highest variance of the first few elements is for the min-max scaling. This property is a reminder to consider the nature of data changes/events which are expected to represent an event.

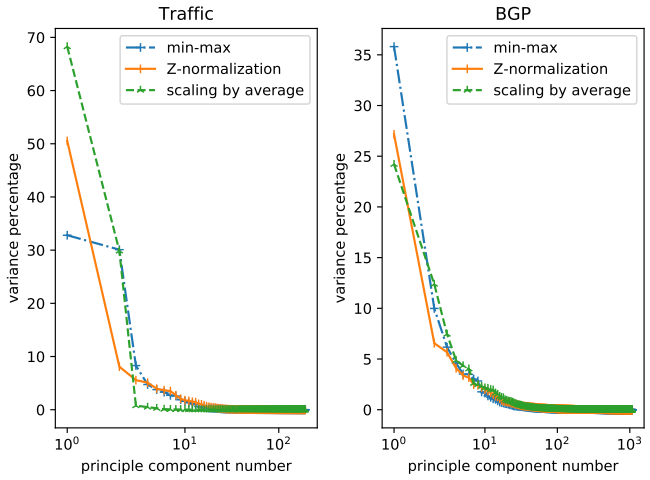


Fig. 4. Scree plots of variance for traffic and BGP counters with different scaling mechanisms- dataset available in [9]

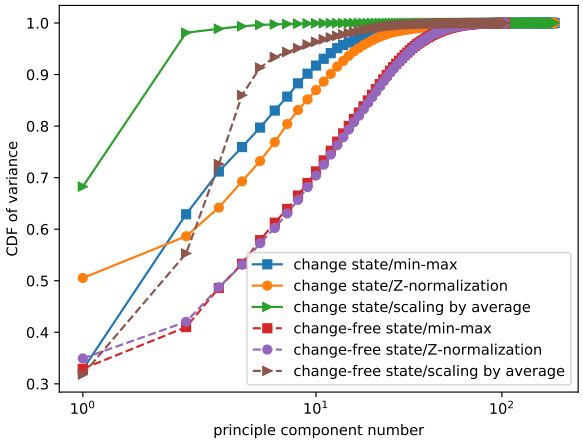


Fig. 5. CDF of variance shown for 192 principal components-comparison of change and change-free-dataset available in [9]

Fig. 5 shows that for a given number of principal components the cumulative variance in change state (data with an event included) is more than that of the change-free state (data without any events). In other words, if a certain normal state(X) is established and presented by a certain number of elements(n_v), a change(C) in that state and transition to another should be noticeable in the number of elements that were already considered for explaining the normal state. This behavior is rather expected since an occurrence of change would result in increasing the variance of the system and thus less elements are needed to represent the same level of variance in data. This property was leveraged in Section V-A in Remark 5-7. The fundamental points of this section are summarized in the following two remarks.

Remark 4-1 : Despite the high initial dimensionality of streaming telemetry, it can be explained with low dimensions. The very low number of 3, in our case, is due to the stable distribution of traffic over a node's interfaces and sufficient traffic multiplexing.

Remark 4-2 : The number of principal elements that can explain a normal state can reflect the impact of the change. It is because the number of effective dimensions is lower when a change happens compared to the normal state.

V. Methodology: Principal Angles

In this section, an overview of DESTIN's mathematical and geometrical meaning is presented. Section V-A includes the definition of a similarity measure τ composed of the cosine of the angles for two matrices A and B which are formed from the streaming telemetry. A geometric interpretation of the principal angles in DESTIN is used for reasoning about the choices made throughout this section. Section V-B proposes two variations of DESTIN, for change detection using principal angles' concept for groups of operational data. The figures in this section are produced using a sample dataset [9]. For more information see Section VI-A.

Principal angles (PA), also known as subspace angles [63], is a subspace-based method that uses angles to present the similarity of two matrices, where a column is an observation vector. The time series collected from a device can form a matrix of observations for any time interval. Let $A^{n\times j}$ and $B^{n\times k}$ be two column matrices with the real-valued column vectors a_i, $i = 1, 2, ..., j$ and b_i, $i = 1, 2, ..., k$, respectively where each column vector is of size n (number of counters/features) and, j and k are the number of observations (data-points). In practice, A and B are the reference and test matrices. Assuming that $\mathcal{A}$ and $\mathcal{B}$ are orthonormal subspaces that span the matrices A and B in $\mathbb{R}^n$ space, and that $p = dim\mathcal{A} \geq q = dim\mathcal{B} \geq 1$.. The vectors $u_1, ..., u_k$ and $v_1, ..., v_k$ in Eq. 1 are called principal vectors of the pair of spaces. Note that the principal vectors need not be uniquely defined, but the principal angles always are. Assume that the maximum is attained for $u = u_1$ and $v = v_1$. Then, θ_1 is defined as the smallest angle between the orthogonal complement of $\mathcal{A}$ with respect to u_1, and that of $\mathcal{B}$ with respect to v_1. Continuing in this way until one of the subspaces is empty, we are led to the following definition of the angles $\theta_i, i = 1, ..., q$ [64]:

$$cos(\theta_k) = \max_{u\in\mathcal{A}} \max_{v\in\mathcal{B}} u^T v = u_k^T v_k \tag{1}$$

$$s.t \ \ \|u\| = \|v\| = 1, \ u^T u_i = 0, \ v^T v_i = 0, \quad \forall i = 1, ..., k-1$$

Therefore, the principal angles $\theta_1, ..., \theta_k$ between $\mathcal{A}$ and $\mathcal{B}$ are recursively defined by Eq. 1.

Assumption 5-1 : Without loss of generality, it can be assumed that $j = k$. This assumption implies that the comparison is between the same number of data-points.

Assumption 5-2 : For the sake of near-real time detection, the time windows (number of data-points in a matrix) are considered to be less than n. Thus, it means that $n > j$.

A. Principal angles' parametrization

1) Principal angles' parameters : To determine a change indicator from principal angles, the first step is to explore the angles' properties. Principal angles between subspaces $\mathcal{A}$ and $\mathcal{B}$ are the rotations along their orthonormal axes to

make the two subspaces overlap [65]. Therefore, the cosine of the angles are the indicators of similarities between the two subspaces [63]. In algebra, a subspace is defined by its spanning vectors. Equivalently without loss of generality, the subspace can be defined by the angles between these unit vectors [66].Thus, principal angles are to represent all the information in a given subspace. In practical measurements, the values which form the subspaces can be divided into two groups of signal and noise. In this context, noise and signal terminology is meant as a notation to distinguish intentional triggered changes from the background changes related to the network dynamics. Therefore, principal angles is not applicable to change detection in its original form and requires modifications. Filtering of the noise in principal angles can be controlled by two main parameters: i) the number of orthonormal vectors (n_v), and ii) the number of angles (n_a). As shall be seen, n_v is the number of axes which contain the most important information of a device state and n_a is the number of rotations needed to overlap the subspaces while filtering out as much noise as possible. The reduced subspace from choosing n_v orthonormal vectors from the existing ones is presented as $\mathcal{S}(n_v, A, B)$. This function reduces the subspaces of $\mathcal{A}$ and $\mathcal{B}$ to n_v principal vectors.

Remark 5-1 : The orthonormal bases of a matrix of size $n \times j$ is upper bounded the rank of the matrix (i.e. $min(n, j)$). Based on Assumptions 5-1 and 5-2, the principal vectors of matrices A and B for the reduced subspaces of $\mathcal{S}(n_v, A, B)$ is upper bounded ($n_v <= j$).

Remark 5-2 : Based on the geometrical interpretation, the angles in PA are presented in an increasing order. This is because the angles are sequentially calculated from the vectors of strongest signal to the least. The more noise in the vectors, the larger the rotation needed for the overlap of subspaces. Therefore, for fixed subspace dimension of n_v, the average value of the cosine of angles is a decreasing function of n_a.

Remark 5-3 : Based on Remark 5-2, the maximum noise is seen in the last angles of a given n_v. Therefore, increasing n_v pushes the noise to higher orders of n_a. Thus, for a fixed n_a, the average value of the cosine of angles is an increasing function of the number of vectors.

The change indicator for PA in this paper is defined to be a product of cosines rather than the average to avoid smoothing the differences in test and reference subspaces. The change indicator is defined as follows:

$$\tau(\mathcal{S}(n_v, A, B), n_a) = \prod_{i=1}^{n_a} cos(\theta_i) \tag{2}$$

In case the data is already categorized into different groups based on Section IV-B the indicator's definition can be extended as follows:

$$\tau_{gen}(\mathcal{G}) = \sum_{j=1}^{j=r} \alpha_j \tau_j(\mathcal{S}({n_v}^j, A^j, B^j), {n_a}^j) \tag{3}$$

Where $\mathcal{G}$ consists of r groups of data (observations) where the features are grouped based on the data operational importance or their pre-processing requirement differences. α_j being the allocated weight to the different groups based on their importance. In case the weights are not specified in this paper, the weights are set to 1. The parameters of n_v^j and n_a^j are determined for each group for better accuracy and A^j and B^j are simply the input test and reference matrices in group j.

B. Principal angles' applicability

In this section the proposed method is first mapped to the state change detection problem and then a gap metric is introduced to test the method's applicability and measure the distance between values in *normal* states and states in which changes occur.

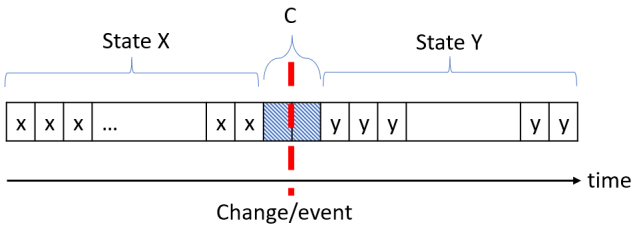


Fig. 6. illustration of a change scenario- x and y are example matrices of features in two different states, C is the interval of change which includes one or more data-points impacted by the change (event)

Fig. 6, illustrates a change scenario, the data blocks and the corresponding matrices are notated as X, Y, C and x, y, c, respectively. The proposed method requires 2 back to back in time matrices if not stated otherwise. The matrices are consisting of j and k observations/collections of a fixed set of n features. Thus, if the first column of collections is collected at time stamp t for the reference matrix, the reference matrix will include the next $t + j - 1$ samples and the test matrix will include the collections from $t + j + 1$ up to $t + j + k$. The value τ calculated for the above test and reference will be assigned to time stamp $t + j + k$. A state is a set of collections that their τ values does not change drastically. As long as the proposed indicator τ (Eq. 2, 3) does not show considerable difference in values, the state of the device has not changed and it is assumed that we are in one state. Thus, a state is defined by stability of the change indicator. The few observations where the difference between reference and test matrices (in terms of change indicator) is large is called the transition/change state C. After the change indicator values are once again stabilized either at the original value before the change or at another value necessarily above the change state values, we have entered the new state (Y). Therefore, a state merely consists of a number of collections that have similar τ indicators. To simplify the concept of states and highlight the similarity of matrices in the same state in Fig. 6, all matrices in one state (i.e. X) are noted as x as they are seen very much like from the change indicator perspective. Note that the matrices x in Fig. 6 show only a few potential reference matrices for the sake of showcasing matrix selection with respect to the time element. If one of the x matrices in Fig. 6 is selected to be the reference, the next x to its right will be the test matrix to calculate the τ value for the latest

collection. In other words, once a new column of data at time $t^{'}$ of the features is extracted/collected, the new column will be concatenated to the test matrix of $t'-1$ step, the first column of the reference matrix of $t'-1$ step is dropped and the first column of the test matrix of $t'-1$ step is concatenated to the back of the reference matrix of $t'-1$ step. The τ value calculated for this test and reference matrix at time-stamp t' will be assigned to time-stamp t' and the process will go on in time. In the remainder of this section, the difference/gap between the two matrices which is the determining factor in change detection is discussed and defined. In addition, the different manners of selecting the test and reference windows are also presented and discussed.

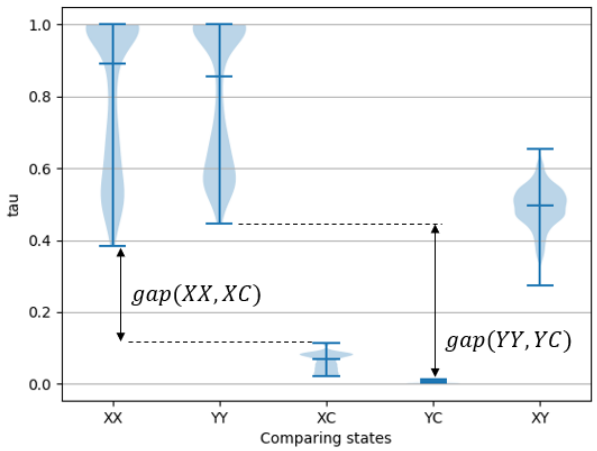


Fig. 7. Violin plot of τ ,(2), values for comparing different states of a device when $n_v = 6$ and $n_a = 3$ - Pre-processing approach is scaling by average values with WSS assumption over all the dataset when $j = k = 10$

Therefore, the few rows of data in time that resulted in a rather different τ value are change/transition state. To emphasize the difference any test matrix in this section that contains any of these observations is denoted as c. Thus, the matrix c should contain the change point. The notation XC in Fig. 7 denotes the comparison of several sets of A (reference) and B (test) matrices where A is selected from state X and B is selected from state C. In this section, τ_{XX} denotes the set of τ values calculated when the reference and test matrices are both selected from state x. As it is already known based on Fig. 6, the τ values in this set are expected to be similar. Fig. 7 illustrates the violin plot of the defined similarity measure τ (2) when comparing different *states* of a network device. The range of values in the set of τ_{XX} shows the dynamics of the network while in state X. It can be seen despite the dynamics, there is a gap between the τ values of the same states (τ_{XX}) and change states (τ_{XC}). It is also seen that the similarity measure τ is larger for similar states compared to when a change happens. It is because in similar states, less rotations are needed to overlap the subspaces than when a change happens. The possibility of detection of a change therefore depends on the existence of the defined gap where it is defined as follows:

$$gap(XX, XC) = min(\tau_{XX}) - max(\tau_{XC}) \qquad (4)$$

Based on (4), the following scenarios can exist:

1) Data with $gap(XX, XC) > 0$: a gap exists when the following two conditions are satisfied:

(I) stability: τ_{XX} values maintain a rather stable value meaning that the rotations necessary to overlap the two subspaces while in the same state should be small.

(II) τ_{XC} values should be smaller than the τ_{XX} values.

Each data-point in the data blocks X, Y and C, has an allocated time called timestamp. Therefore, when selecting matrices from data-blocks, they could be close or far away in time. As long as no event happens, the matrices which are closer in time, will require less rotation to overlap (i.e. has high τ values) compared to matrices which are far away in time when in normal states.

Remark 5-4 : Based on Remark 5-2, To avoid including the noise in τ values, not all the angles are considered in the calculation of τ values.

Remark 5-5 : Based on Remarks 5-2, 5-3 and 5-4, the proper number of angles to filter the noise and contain enough signal is distanced from the first few (Remark 5-3) and last few (Remarks 5-2, 5-4) angles. (See Fig. 8 for illustration)

Remark 5-5 decreases the search space and speeds up the search for the right n_a as opposed to the algorithms in [35].

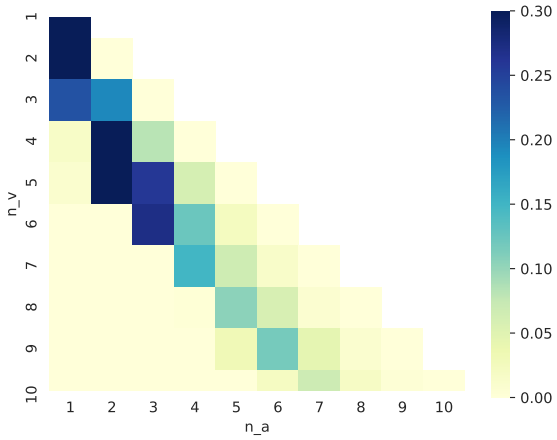


Fig. 8. Heatmap of $gap(XX, XC)$ for when A and B are not overlapping - The first even data interval in dataset available in [9] is used for illustration.

2) Data without initial $gap(XX, XC) > 0$: It is not reasonable to assume that the gap in (4) always exists. Often it is due to the violation of the first condition (stability) that the gap disappears. Overlapping the reference (A) and test (B) matrices by a few data-points can create a dummy stability for τ values. This action results in less rotations when in the same state and thus higher τ values compared to change states.

Remark 5-6 : Overlapping the matrices A and B can generate the stability necessary to satisfy the first condition of maintaining a non-zero and positive gap.

In addition to Remark 5-6, to satisfy the existence of the gap, the second condition should also hold. The next remark explains why the second condition will naturally be satisfied.

Remark 5-7: It can be shown that the normal state has a higher dimension compared to the state which includes a

change. In other words, when the two subspaces are reduced by n_v vectors, the change subspace has much more noise in terms of principal angles and thus a lower τ value. Therefore, τ_{XC} is surely less than τ_{XX}. This property was further discussed in Section IV-D and noted in Remark 4-2.

Remarks 5-6 and 5-7 ensure the existence of a gap and the number of angles can be chosen using Remark 5-5. Fig. 9 shows an example of the data behavior after removing the trend and overlapping the same data of Fig. 8. V-B3 and V-B4 propose parameters based on the logical analysis of this section to eliminate the need in most cases for a search of the right number of angles and vectors.

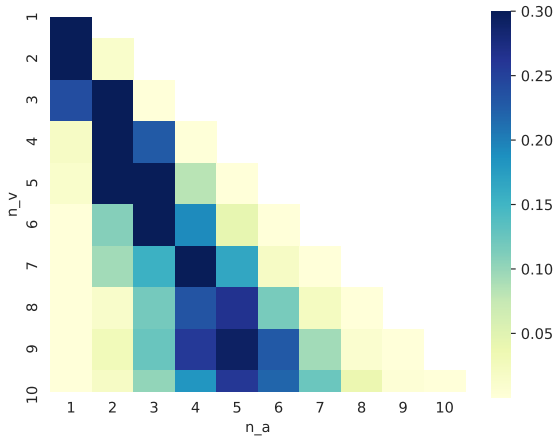


Fig. 9. Heatmap of $gap(XX, XC)$ for when A and B are overlapping by 5 data-points - dataset available in [9]

3) Specific Principal Angles (SPA) – for data with a known trend: When data has a *known trend* such as an increasing trend, leveraging this special property can introduce a fixed set of parameters when using principal angles. In such cases, if the trend is not removed, the most important signal is the result of the trend shape and the rate by which it changes. Therefore, based on Remark 5-2, only one angle which includes the strongest common signal is enough. Based on Remark 5-3, taking only the first vector is the best option since higher numbers will result in the leakage of signal on to other angles. Thus, in such cases, $n_v = n_a = 1$ can provide a trustworthy change detection. Note that the trend already ensures the stability of τ_{XX} values hence no overlap is required. Traffic counters of streaming telemetry is a good use case example of such data. Rather it should be avoided since it can dominate the only angle considered. In this paper, to establish an online detection approach, the two matrices A and B are side by side sliding matrices.

4) Generic Principal Angles (GPA): In cases where no known trend can be assumed or one prefers not to leverage the trend properties, the principal angles' parameters can be determined using the overlapping (common data-points) concept. Often half of the total data-points is a moderate and agreeable option. Though this property depends on the type of the noise and its fluctuations in general. Based on Remark 5-6, the stability of τ_{XX} is then established. In other words, in the subspace of n_v vectors, close to half of the vectors need very little rotation to overlap the two subspaces of test and reference. When overlapping the subspaces, n_v should be set to the number of data-points $n_v = j = k$ (refer to Remark 5-1), to make sure that all the important signal is contained in the subspace. This makes the number of angles(n_a) the only factor remained for filtering noise. Under no statistical assumptions for pre-processing, assuming n_a to be equal to the number of the overlapped data-points can express changes due to inevitable reflection of change when scaling common data-points. In case of Wide Sense Stationary (WSS) assumption for scaling, n_a should be more than the number of overlapped data-points. Usually one more than the overlapped data-points can reflect the data changes quite well. In cases where the input data does not produce a stable DESTIN indicator (either due to the unique state of the network or the combination set of the selected features as the input), we advise to increase the number of observations in the matrices as opposed to deviating from the above mentioned parameter configurations. The above configurations are tested extensively to ensure that any important changes of state for operational data of routers is detectable. Note that the increase in the number of observations will relatively increase the delay in detection of the event. The performance benchmark results for SPA (V-B3) and GPA (V-B4) using traffic-related counters at interface level are presented in Section VI-C.

5) Binary decision making: In this paper, a simple sigma detector is used on the univariate τ values to flag changes. A sigma detector defines a change to be when the values deviate from their average by more than a given δ times of their standard deviation. The sigma detector cools down (stand-by mode) for a certain amount of time after a detection is made to avoid multiple detections of the same event. The cooling time is advised to be set to a value equal or more than twice the matrices' time interval when applied for DESTIN (i.e $2 \times k$ where k is the number of data-points in matrices A and B). This choice stems from the fact that sliding the time windows side by side in time to form the reference and test matrices results in the impact of event to persist until the two matrices are both slide passed the change point.

It is important to note that more complicated binary online detectors such as the ones proposed in [67]–[69] likely provide improved results compared to the ones provided in this paper. However, to avoid complexity and facilitate clarity of results analysis, the simple sigma detector is used over all the methods in this paper.

C. Complexity analysis

In this section, the time complexity of the PCA and DESTIN is compared. The time complexity is quantified in this paper by number of floating point operations (flop). The practical implementation of principal angles cosists of the following steps:

(I) Take truncated singular value decomposition (SVD) of A and B and denote the first n_v vectors of each by $trnc(A, n_v)$ and $trnc(B, n_v)$. Note that taking SVD is one way of calculating the $\mathcal{A}$ and B mentioned in Section

V. The time complexity of truncated SVD based on [70] needs $\mathcal{O}(2nj^2+j^3+j+nj)$ flops for calculating all the vectors. Note that only the first n_v vectors are needed and if we were to optimize the code the time complexity is $j^3 + nj^2 + nn_v + n_vnj$ flops which is less than equal of the previous value since $n_v \leq j$. Since $j = k$ the same complexity analysis applies for calculation of $trnc(B, n_v)$.

(II) Calculate $trnc(A, n_v)^T \times trnc(B, n_v)$. The time complexity of the multiplication can be written as $\mathcal{O}(2nn_v^2 - n_v^2) = \mathcal{O}(nn_v^2)$ flops [70]. Take the truncated SVD of the the multiplication from the previous step and the sigma values are the ordered cosines of the principal angles from the smallest to the biggest angle. Similar to the first step, the time complexity can be calculated as $\mathcal{O}(n_v^3 + n_v^3 + n_vn_a + n_an_v^2) = \mathcal{O}(2n_v^3 + n_v^2n_a + n_vn_a)$ flops.

(III) Taking the first n_a values and their multiplication based on Eq. 2 will generate the change indicator. The multiplication has $\mathcal{O}(n_a)$ flops complexity.

Note that in this paper $n >> j \geq n_v \geq n_a$. Thus, in matrices that this conditions holds, as applies to all data processed and analyzed in this paper, the complexity of DESTIN is linear in time. PCA is composed of a covariance matrix calculation with time complexity of $\mathcal{O}(n^2j)$ and then an eignevalue decomposition with time complexity of $\mathcal{O}(n^3)$. Therefore the time complexity of PCA is $\mathcal{O}(n^3 + n^2j)$ [28] which is more than DESTIN. Cardot et al. [28] dedicate a whole paper on exploring the complexity of PCA and its different robust versions that reduces its complexity in detail.

In terms of practical resource consumption, the routers running ADT in a published demo subscribe to $316,000$ counters. The proposed ADT system consumes almost 1000 counters that their value often change by time. The whole system (exporter+detector+explainer(greedy optimization and entropy calculation [11])) consumes a total of 601 MB memory and 8% of CPU usage on the router (For more details please see [60]).

VI. Evaluation

This section includes detailed information on datasets, terminology and the evaluation of DESTIN's performance in terms of recall, precision and average delay to detection for SPA, GPA and PCA methods when considering data-driven filters as well as the expert-defined manual ground truth. The data-driven filters provides a confirmation means if not an alternate approach to the manual ground truth generation and dataset labeling. Thus, it introduces a potential framework for evaluation of events with similar impact explained in this paper on the time-series.

A. Lab network and dataset generation

The datasets used in this paper for benchmarking are retrieved from the routers in a lab and the devices are connected in a Clos-topology, with 4 spines and 8 leaves [71]. The datasets are made publicly available [72]. The links connecting the routers use ECMP to distribute the traffic. All leaves connect to multiple ToR (top of rack) switches, each of which aggregates the traffic from multiple traffic generator nodes. The lab uses BGP as routing protocol. All links have BFD (bi-directional forwarding detection) configured for rapid failure detection. The time series are created by retrieving several model driven telemetry (MDT) collections with a interval of 10s over spines 1-3 and leaves 3,4,7 and 8. All datasets are made publicly available. The types of changes inserted in the lab network include:

(I) Enabling/disabling of interfaces using configuration commands;

(II) Breaking and restoring of BFD sessions (by filtering-out BFD control traffic between devices);

(III) Creation of routing loops (using static routes to make traffic for a particular prefix loop between dr02, dr03 and leaf8);

(IV) Creation of traffic blackholes (removal of Forwarding Information Base (FIB) entries for a prefix using configuration commands to cause the router to silently drop traffic destined to that prefix);

(V) Change of hashing behavior across equal-cost multi path (ECMP) using configuration commands.

The traffic in the network is generated using Ixia traffic generator [73] that is capable of generating different traffic profiles. The traffic generated for datasets in this paper is either AppMix or FlowMix. FlowMix traffic includes different TCP profiles such as *TCP low*, *TCP baseline* and *TCP small packets* as well as *Youtube 4k* traffic. In other words, FlowMix mostly consists of simple TCP connections where each connection sends fixed-size TCP packets with fixed delays and settings. For example, the typical *small TCP* generates 64 byte payload packets with the largest TCP being 64Kb packets including random middle values. AppMix on other hand, doesn't focus on packet sizes and delays but rather emulates different applications. This facilitates the request/response operations with different payload sizes. AppMix, in this paper, includes multiple application profiles such as Facebook, Netflix service provider, HTTP flash video service, YouTube LTE mix, HTTPS simulated, bitTorrent Cisco EMIX, Webex, etc.. AppMix allows for indication of the number of users and thus changing the number of connections in the network.

In most of the generated datasets only one intentional event is inserted. However, there are cases where unintentional changes of state happens. At the time of the insertion of events, a general *event file* is generated that specifies the timestamp of the event. The unintentional changes are not included in the *event file*. The data collections from all devices will share the same event file whether the event is expected to propagate on all of them or not. Therefore, in addition to benchmarking the behavior on an expert-crafted ground-truth, a data-driven scheme is proposed to evaluate the methods' capabilities. It is noteworthy that the results presented in Sections V and IV-D use a different dataset compared to evaluation section: The sample dataset consists of 6622 counters while test datasets are composed of different number of features ranging from 300 to 20000 features. The simulated events in this paper are not particularly hard to detect for a known network when

using already existing rules and heuristic packages, however it is not trivial when no assumptions are made. The results in Section VI-C compares the performance of DESTIN which considers no assumptions with the expert systems.

B. Terminology and setup

1) Event versus change definition: Events inserted on a single network device can be reflected on neighboring devices as well. Examples include insertion of a wrong static route that causes traffic to loop, or the failure of an interface which will be reflected in the interface state of the connected router, but likely also on several other devices due to the associated change in routing and forwarding behavior. In this paper, to evaluate the propagation of an event inserted on a single router, in addition to leveraging the concept of event from expert's perspective (already defined rules), some data-driven measures of change are introduced to assess the event's propagation referred in this paper as changes. The data-driven measures introduced in this section also provide an upper-bound on the potentially important time-stamps for an operator.

2) Data-driven measures of change: An event can incur changes on one or more devices with different impact levels on counters (features). The most generic way to approach the evaluation of an event's detectability is to assume the most generic time series change scenarios which are mean and variance changes. Therefore, two metrics are defined in this paper to track the ratio of shift level (mean-change) to its noise (standard deviation (std) level) in a univariate and multivariate way.

3) Univariate change to noise ratio (UCNR): Assuming a perfect step-change (mean value change) in time-series, one can calculate the ratio of the step to the standard deviation of that time-series. Using the notations defined in Section V, x and y are the matrices of shape $\mathbb{R}^{n\times k}$ that belong to the larger data blocks X and Y right before and after the event, respectively. The matrix c is selected from the transient state C. Assuming that the step change value for feature i is noted as l_i, the measure can be expanded to the n available features as bellow:

$$\Delta_{uni} = \sum_{i=1}^{n} \frac{|l_i|}{std(x_i)} \tag{5}$$

Where x_i denotes the row vector of observations for feature i and *std* operator takes the empirical standard deviation of the vector given as input. Equation (5) measures the change on each time-series data in a univariate way and accumulates the effects by summing over the values. This implies that every feature has its own independent impact on the metric. However, this measure is impacted by the number of features in the dataset and cannot be compared between datasets of different sizes. To address this issue, a relativized instance of the values by the average over a fixed number of past values is used. Therefore, in addition to the paled effect of dimensionality, the difference of a change compared to the background noise is better highlighted. The relativized values are denoted as $\widehat{\Delta_{uni}}$ in this paper.

4) Multivariate change to noise ratio (MCNR): A different way to measure the impact of a change is to consider the shift change in the feature space of R^n in a multivariate manner and calculate the metric using vectors as follows:

$$\Delta_{mult} = \frac{||\overline{y} - \overline{x}||}{std((\overline{y} - \overline{x})^T \times [(x - \overline{x}) \times 1_k^T])} \tag{6}$$

Where$\overline{x}$ and 1_k denote the empirical mean of matrix x and a column matrix of all ones of size k, respectively. For the same reasons mentioned in VI-B3 the values are divided by their average and denoted as $\widehat{\Delta_{mult}}$.

5) Performance assessment parameters: To facilitate a fair comparison of methods, three different approaches are used to define what is considered *ground truth* for detecting a change on a particular network device based on the logic and metrics defined in this section so far.

- **manual -** devices which are expected to be impacted by a change, as defined by an expert based on topology and type of event. These devices are referred to, in this paper, as the *local devices* of the particular event. The rest of the devices are referred to as *remote devices*. The following terms are then defined for performance evaluation:
  - Recall: The ratio of the number of correct detection notifications on at least one of the local devices to the total number of intentional events inserted on the devices.
  - detection relevance: The ratio of number of the correct detections on all devices, either remote or local, to the number of total notifications made by the detector.

  Manual ground-truth represents the lower bound in the number of events on the very least number of devices expected to reflect the event.
- **data-driven using UCNR / MCNR -** all devices exceeding a certain threshold of UCNR / MCNR around the time of the change insertion will be selected as a valuable change point to be benchmarked. To assess the performance in these cases the following terms are used:
  - Recall: The number of correct detections on all devices (either local or remote) divided by the number of total devices expected from the data-driven measure that are expected to detect change(s).
  - Precision: The ratio of correct detections to the total number of detections made.
  - f1 score: The harmonic mean of recall and precision.

C. Numerical results

This section presents results when the matrices A and B have $j = k = 7$. The n_v and n_a parameters are chosen based on Sections V-B3 and V-B4 for SPA and GPA, respectively. The overlapped data-points and n_a values for GPA are considered both to be 3 and no assumptions such as WSS property are made. The evaluation is based on 2 ground-truths. One is determined by the operator (network expert) based on the time of insertion of events and another is ground-truth generated from both the event insertion time as well as data-driven information as discussed in Section VI-B2. The

TABLE I
RECALL, PRECISION AND TIME TO DETECTION

methods	**Ground-truth defined**									
	manual		**data-driven UCNR**				**data-driven MCNR**			
	Recall	Detection relevance	Recall	Precision	f1 score	Delay	Recall	Precision	f1 score	Delay
PCA	0.73	0.40	0.60	0.50	0.54	**9.65**	0.56	0.49	0.52	**11.36**
SPA	**0.79**	**0.78**	0.76	**0.87**	**0.81**	24.5	0.69	**0.84**	**0.75**	30.31
GPA	**0.79**	0.64	**0.78**	0.77	0.77	22.29	**0.72**	0.76	0.74	28.85
Total events	**82**		**242**				**357**			

methods are compared based on their recall, precision and f1-score as a means to compare their accuracy. For the sake of a fair comparison with PCA, traffic counters are used for performance evaluation. The parameters used can be seen in Tab. II.

TABLE II
GENERAL PARAMETERS

Parameter	**Method**		**Description**
	SPA	GPA	
j	7	7	reference window size (of data-points in reference matrix)
k	7	7	test window size (of data-points in test matrix)
n_v	1	7	number of vectors taken form $\mathcal{A}$ and $\mathcal{B}$
n_a	1	3	number of angles
overlap	0	3	number of common data-points in A and B

Tab. IV, compares different methods in terms of recall, precision, and the average delay for the detection of a change. PCA is a method that compares a number of data-points to one data-point while principal angles compare multiple data-points to each other. It is expected that principal angles (PA) have more delay in detection compared to PCA due to its input format. Tab. IV shows that the delay of PA is almost twice that of PCA subspace which in terms of data-points amounts to only 1 or 2 more data-points (i.e. 10-20 seconds).

The benchmarking results from the manual ground-truth show that in the total of 82 manually inserted events, 73%, 79% and 79% of the events were detected on at least one of the local devices for PCA subspace, SPA and GPA, respectively. Evaluation of detection relevance show that the false alarms are lower in SPA by 38% and 24% compared to PCA subspace. This shows that leveraging a known trend for data while using PA can significantly improve the noise filtering. However, PA method, even in its most generic sense as GPA, improves the precision by 24%.

In both data-driven cases, the threshold is set to 3 for UCNR and MCNR metrics. The justification behind is that the metrics are calculated only around the event over small windows of data, it can be assumed that only in that small interval the data is expected to exhibit WSS properties. It is known that for a WSS process, a deviation of 3 times the noise value is the indicator of surpassing the normal behavior. Setting this threshold results in a total of 242 and 357 datasets, for UCNR and MCNR, respectively. The difference in the sets supports the fact that, ideally, one expects to detect more events in a multivariate approach rather than in a univariate one. In the data-driven approach in terms of precision, SPA and GPA are 37% and 27% better than PCA subspace when using UCNR as the filter. It can be said that using SPA and GPA with UCNR filter will have an average improvement of $\frac{87+77}{2} - 50 = 32\%$ in precision of PCA, whereas, the same calculation for MCNR filter will result in 28% improvement. The average recall improvement of PA using UCNR filter is $\frac{78+76}{2} - 60 = 17\%$ and the same when using MCNR filter is 14.5%.Comparing SPA with GPA shows that leveraging existing assumptions, such as known trends, can improve the accuracy in PA. Comparing the f1-scores, a measure of their relative recall and precision performance, using MCNR filter for the SPA and GPA, it can be seen that they perform at 74% and 75%, respectively whereas PCA performs at 52%. This means that PA improves the f1-score of PCA by at least $\frac{74+75}{2} - 52 = 22.5\%$ when the same calculation using UCNR filter results in 35% improvement. Tab. III depicts the results

TABLE III
RESULTS BREAK-DOWN BY EVENT TYPE FOR UCNR FILTER

network event	**Detected**			**Total**
	PCA	SPA	GPA	
Shutdown interface	53	62	**63**	73
ECMP hash changes	36	51	**52**	57
Enable interface	25	**28**	**28**	50
Blackhole	26	36	**37**	44
BFD restore/break	3	**5**	**5**	13
Routing loop	3	3	**4**	5

per event type. It is seen that all methods have almost the same potential in detecting enable interface, BFD break/restore and routing loop events. However, there is a noticeable difference in detecting changes of ECMP hashing between PA and PCA. The reason behind the fair level of improvement, is the impact of ECMP-hashing related changes on the inter-dependencies of the counters when the event happens. For changes resulting in a drop of traffic, such as shutting down an interface or traffic blackholing, PA also shows the best performance due to the method's robustness with respect to the assumptions on pre-possessing.

VII. DISCUSSION

In this section, further analysis on the quality of the datasets and the generated events based on the detectability measure, UCNR, is provided (Section VII-C). The performance of DESTIN is reported by division based on the device and the dataset's traffic type. An analysis on propagation of the events through the network is provided in Section VII-D. Finally, the datasets are checked case by case (Section VII-E) to root cause

the reasons behind the False positive (FP) and False negatives (FN).

C. Datasets quality and UCNR's accuracy

The numerical results in Section VI-C attested to the superiority of DESTIN compared to the well-established method of PCA. Yet, the results are not perfect. To provide a better understanding of the results, the quality of the datasets as well as the UCNR measure are explored in this section. As such, the reasons for the FPs and FNs are brought to light in the following sections.

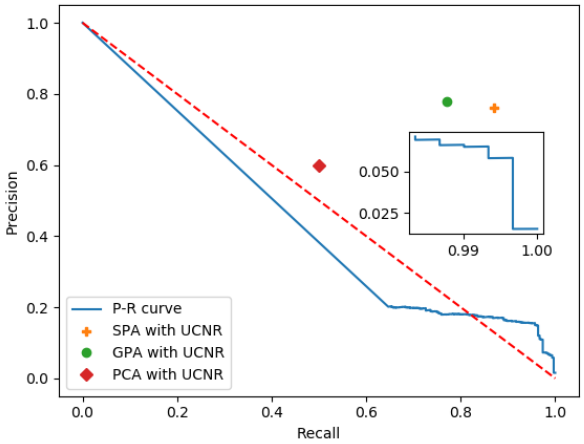


Fig. 10. Percision-Recall curve for UCNR ($\widehat{\Delta_{uni}}$) values when $\widehat{\Delta_{uni}}(th) = 5$. The dashed line is used as a reference for comparison

To determine the quality of the datasets and accuracy of the UCNR filter, the Precision-Recall curve of the $\widehat{\Delta_{uni}}$ values are plotted in Fig. 10. Precision-Recall curves are often used to showcase the quality of a classification. In this paper's context, it is used to highlight the level of distinction that $\widehat{\Delta_{uni}}$ can draw on its own in the classification of change-points from the other normal points.

To plot precision-recall curve in this paper, the probability of each point in the datatset being a change-point is calculated. To calculate this probability, a detectability threshold ($\widehat{\Delta^{th}_{uni}}$) is required . The value for $\widehat{\Delta^{th}_{uni}}$ is selected based on empirical results on various sets of datasets by filtering datasets based on different UCNR values. Based on the tests, for detectability values of 5 and more, the overall precision and recall appear to be similar. This implies that the datasets filtered with events detactability value of 5 or more remain almost the same and the most controversial events have less than 5 for their detectability value which calls for assignment of probability of points being an event-point. This means that the points having the UCNR value of 5 or more are expected to be events (points within the change area C - see Fig. 6) with probability 1. Thus, the detectability threshold $\widehat{\Delta^{th}_{uni}}$ is set to 5. And thus the probability is calculated using Equation 7. The $\widehat{\Delta_{uni}}$ values are calculated by sliding a window over the dataset in time to get the corresponding x and c matrices at each point in the dataset. In this section, the objective is to calculate $\widehat{\Delta_{uni}}$ for all the time-stamps to evaluate the data-point's probability of being a change. Thus, at each time-stamp t the matrices x, c and y are simpily determined by back to back matrices with fixed number of observations for each of them and then $\widehat{\Delta^{t}_{uni}}$ is calculated. The probability for each time-stamp(data-point) is calculated as follows:

$$\mu^t = \frac{\widehat{\Delta^{(t)}_{uni}}}{\widehat{\Delta^{th}_{uni}}} \tag{7}$$

$\forall\ t$ where t corresponds to matrices x and c at point t.

Considering the number of normal points in datasets far exceeds the number of event points, the two categories (a data-point *being* or *not being* a change) assessed based on $\mu(p_i)$ are imbalanced in size. Therefore, instead of the more well-known Receiver operating characteristic (ROC) curve, the Precision-Recall curve, which is more appropriate in such cases, is used for evaluation [74]. The μ values are calculated for all points in the datasets. Based on a set probability threshold μ_{th}, each point is then labeled as event or non-event. Every point with a probability above μ_{th} is expected to be an event while points with less probability are considered a non-change-point. For each μ_{th}, the labels (event and non-event) are assigned and cross checked with the ground truth. Thus, generating one precision and one recall data-point per μ_{th} in Fig. 10. The process is repeated for different probability thresholds over all datasets to plot Fig. 10. The following bulletins could be inferred from Fig. 10:

- The low values of precision for all thresholds (based on the jump in precision values of 0.27 to 1) indicate that the datasets include points which have high UCNR values yet they are not marked as events. This is due to two potential reasons: i)the existence of unintentional events in the datasets, ii) the limitations of the UCNR measure. The former is because of lack of full control over all the network while the later is simply due to the measure being calculated point by point. A point by point calculation introduces the possibility of sudden increase or decrease in the UCNR value due to collection issues (i.e. missing a collection, collection configuration issues, etc.) and the bursty characteristic of traffic.
- The UCNR itself is not an appropriate measure for detecting events due to the poor recall and precision values observed in Fig. 10. The poor performance is due to utilization of a fixed threshold for separating change-points from normal data-point over the entire dataset. The precision and recall will greatly improve if it was possible to assign different thresholds of detectability for different datasets. Note that the stated fact does not at all convey that UCNR cannot be used for validating the detectability of events as a filter. Section VII-D further highlights the importance of UCNR filter in the evaluation.
- The jump in the recall values from 1 to 0.997 with the threshold 0.133 indicates that there exists points that their UCNRs are less than 0.665 where at the same time they are labeled as events in the ground truth. This confirms

TABLE IV
RECALL (R), PRECISION (P) PER DEVICE AND TRAFFIC TYPE

A. Non-leaf devices

$\widehat{\Delta_{uni}}$	Traffic Type	Device									
		dr02		dr03		spine1		spine2		spine3	
		R	P	R	P	R	P	R	P	R	P
0	AppMix	0.49	0.83	0.02	0.50	0.51	0.82	0.34	0.63	0.81	0.92
	FlowMix	0.44	0.67	0.11	0.40	0.11	0.05	0.0	0.0	**0.44**	**1.0**
	Total	0.48	0.80	0.05	0.50	**0.38**	0.31	**0.24**	0.38	0.74	0.93
3	Total	0.55	0.85	**0.06**	1.0	**0.91**	0.83	**1.0**	0.83	**0.97**	**0.95**

B. leaf devices

$\widehat{\Delta_{uni}}$	Traffic Type	Device							
		leaf4		leaf5		leaf7		leaf8	
		R	P	R	P	R	P	R	P
0	AppMix	0.66	0.80	0.51	0.93	**0.95**	0.92	0.49	0.64
	FlowMix	-	-	-	-	0.0	0.0	0.44	0.67
	Total	0.66	0.80	0.51	0.93	0.73	0.75	0.47	0.65
3	Total	0.88	0.84	0.90	**0.95**	**1.0**	**0.95**	0.59	0.72

our concern regarding the manifestation of events on only a portion network devices. This is because the same *event file* created at the time of event is used for all devices and no expert knowledge is used at this stage to separate datasets of local and remote devices(see Section VI-B5 for terminology references).

It can be seen in Fig. 10, that SPA, GPA and PCA perform better than the UCNR as a detector, since they are closer to the $(1, 1)$ point. Yet, note that UCNR is an acceptable filter for verifying the manifestation of events, when an estimated time for event is given. Section VII-D demonstrates the positive impact of utilizing UCNR when evaluating a methodology and sheds light on DESTIN's performance.

D. Network analysis

Section VII-C explored some of the reasons for an imperfect recall and precision based on the dataset's quality and the filter's limitation. Some of the reasons highlighted were data collection issues, imperfect event labeling and lack of full control over dataset generation (i.e. the existence of unintentional events), to name but a few. The above mentioned are mainly rooted in the shortcomings and difficulties of collecting telemetry data on the routers and the specific test-bed used in this paper. This section will provide a detailed walk-through of the results and extracts the information about the network topology, as well as root causing the detector's performance per device.

The events evaluated in Tab. I all had UCNR value of 3 or more. Tab. VII-B depicts the precision and recall categorized by traffic profile per network device with ($\widehat{\Delta_{uni}} = 3$) and without ($\widehat{\Delta_{uni}} = 0$) the UCNR filter. It can be seen in Tab. VII-B that the performance of the detector on FlowMix datasets is not as good as AppMix datasets. Note that all the FlowMix datasets have events with UCNR values of less than 1. This attests to the bursty nature of TCP traffic and its constant fluctuations, which results in complicating the detection process. Based on results for this network, it appears that the performance of ADT is only acceptable on dr02 and spine3 for such traffic. It can also be said that it is not impossible for DESTIN to detect events with UCNR values of less than 1 with an acceptable precision. It is noteworthy to mention that some of the generated blackhole events with UCNR value of less than 1 are only detected on spine3 and not other devices. In other words, those events would have been missed if DESTIN was not running on the device. The reasons behind this observation is discussed in the next paragraph.

Based on Tab. VII-B, the performance over all datasets in the absence of detectability measure (UCNR) for different devices is rather different. Tab. VII-B depicts that some devices (leaf7, spine3 and leaf4) have better detection performance than others (spine1, spine2, dr03). There are also devices where although not all events could be detected, the detections are accurate (high precision) and the false alarm rate is low and thus, the performance is acceptable. The devices in this category are dr02 and leaf5. At first glance, the performance differences might appear random. However, it is closely connected to the event injection location and its propagation. Leaf7, leaf4 and spine3 are devices where different events were originated from, therefore, they are expected to have better recall compared to other devices whereas, spine1 and spine2 are not connected to any event insertions and dr03 is only concerned in routing loop events. The rest of the devices having high precision and lower recall indicate that the method performs well since only a portion of events are expected to be seen from these devices. The imperfect filtering of the datasets demonstrated in Section VII-C, highlights the recall values as less critical compared to precision. The results show that the propagation of an event is indeed local but may as well exceed the expectation of the operator. This is based on the fact that the number of correct detections of the event often exceeds the operators expectation of the event's propagation.

Comparing the results of all datasets with filtered datasets using the UCNR filter of more than 3 shows great improvement in precision and recall. This improvement is thanks to the correct filtering of the datasets. The filtering excludes the datasets from devices that have received no impact from the change which is due to the device's location in relation to event injection and propagation. The lowest precision in all devices is 0.72 on leaf8 and the highest is 1.0 on dr03. However, dr03 has a very low total recall even for UCNR of 3 and more.

A manual exploration of the datasets of dr03 revealed that this device suffered from collection issues at the time of data collection and thus, the low quality of data made it hard to detect the events. All the other devices have precision of more that 0.83 with relatively high recall (on average 0.76).

The comparison of all datasets with different UCNR filters shows that the UCNR filter can indeed help with performing a fair evaluation of datasets with the least manual efforts possible. Considering that all FlowMix datasets have UCNR of less than 1, comparing the total results with $\widehat{\Delta_{uni}} \geq 3$ and the AppMix row in Tab VII-B is the same as comparing the results on AppMix datasets with and without the UCNR filter. Based on Tab. VII-B, the difference of performance in recall are 0.76, 0.41, 0.40 and 0.22 between the two sets of results and is specially evident on spine2, leaf5, spine1, and leaf4, respectively. Leaf7 and spine3 experience slight increase in their recall compared to their stellar results and devices dr02 and leaf8 undergo an increase of 0.06 and .10 in their recall and slight improvements in precision. If ADT is enabled on all devices of the network, it can also provide a graph of devices that has detected an event with the time-stamp of detection. It can help in narrowing the event localization to a few devices and provide a priority list for manual exploration using the hints the explainer provides.

E. Case by case root causing

In this section, the problematic datasets are visited one by one and the causes of False positives (FP) and False Negatives (FN) are identified and summarized.

The reasons identified behind FPs and FNs are summarized as follows:

(I) **Short datasets:** This refers to not having enough data-points after the event insertion to determine whether the abnormality observed was due to normal traffic fluctuation or the manifestation of an event. This issue often results in FNs. Sometimes, it is not due to DESTIN's limitation but rather the binary detector's need for more data-points to make a binary (event or not) decision.

(II) **Unintentional changes:** This matter has already been mentioned in Sections VI-A and VII-D. A closer look at such datasets revealed that the most recurring unintentional changes were due to the flapping of interface *HundredGigE0/0/0/18* on leaf8. This also clarifies why the precision was lower on this device. It is noteworthy that FPs could themselves result in FNs which is due to the limitations of the binary detector. Examples of interface *HundredGigE0/0/0/18* flapping can be seen in Fig. 11, Fig. 12-b as well as Fig. 13-b.

(III) **Filtering of interfaces with low traffic load:** In the application of the method, to avoid fluctuations of the control traffic, very low volume interfaces were filtered out. This is while some of the events were actually inserted on such interfaces. This decision resulted in missing those events and creating FNs. To avoid this issue, the datasets in this paper could be filtered using regular expressions on their names to exclude management traffic.

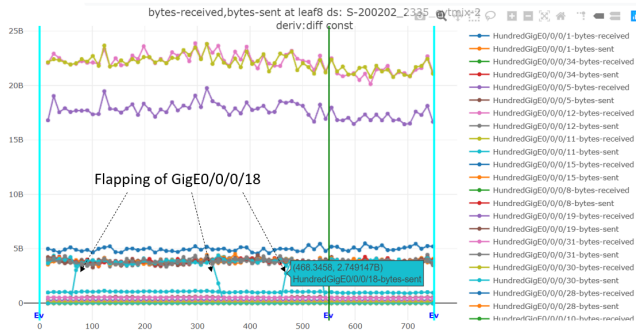


Fig. 11. Unintentional changes in the network can cause FNs and FPs

(IV) **Automation:** The events generated for this test were all created using written scripts that facilitate the automatic generation of events. However, a closer evaluation of the datasets showed that though not occasional, some of the events were not manifested while their *event file* was still generated and marked that time-stamp as an event. Such errors resulted in a few FNs in datasets specially the ones on dr02. Most of the events in this category where enable interface and break and restore BDF events. This explains why the detection of such events is less successful than some of the other events in Tab. III.

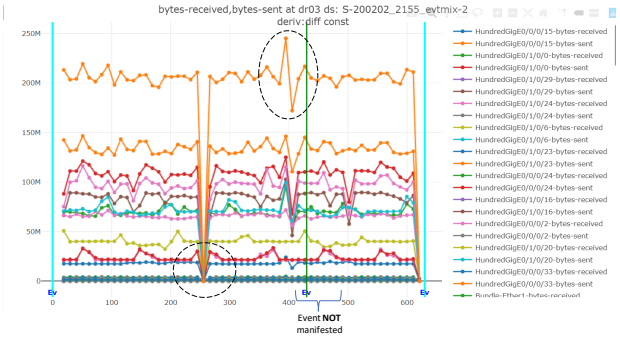
(V) **Collection issues:** As previously mentioned, missing a collection call of one or more time-stamps and wrong collection configurations (i.e. improper cadence, etc.) can create unexpected fluctuations in data and thus create FNs and FPs. The screen-shots in Fig. 12 showcase some of the collection problems faced in this study. As can be seen the 2 problems i) symmetrical spike (increase and decrease of traffic in same amounts in consecutive calls) in feature and ii) missing of collection polls can simultaneously occur in one dataset. Missing collections as can be seen in 12-c can be taken as a legit event from method's perspective whereas it is just a collection issue.

(VI) **Limitations of the binary detector:** This group of FPs and FNs are connected to one or more of the following reasons:

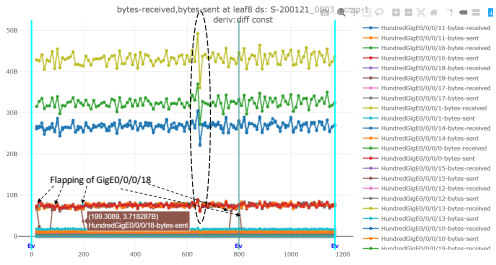
a) Delta value: The fixed delta in the sigma detector (see Section V-B5) is sometimes not robust enough for determining events as the severity in an event's depiction in DESTIN's indicator varies with different events. This may cause FNs and FPs. If DESTIN's indicator is used as a KPI on its own, the change in the indicator is recognizable by the operator.

b) The recovery/cool-off (see Section V-B5) period of sigma detector after detection of an event: This may not be an exclusive issue to sigma detector. Due to the time required for the sigma detector to establish the parameters of the new state after the change, it might miss a close event to the already detected event and thus result in FNs.

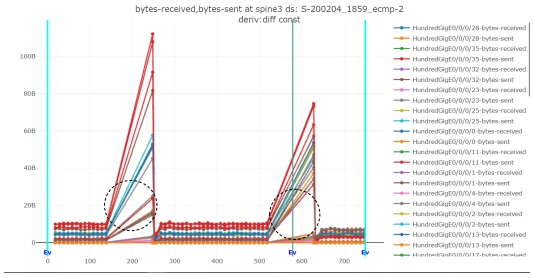
c) Polluted history of the sigma detector: The fluctuations in data that do not result in triggering an



(a) Missing one call for all counters and Symmetrical spikes



(b) Symmetrical spike on several counters, which demonstrates a collection of bug pattern and flapping of an interface.



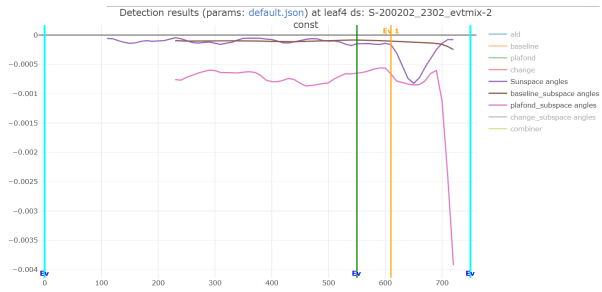
(c) Periodic missing of collection polls for 100s, resulting in an aggregated value after the issue is resolved — which creates an illusion of event (where in reality it is just a collection issue)

Fig. 12. Data collection issues: The dataset names and the devices from which the data is collected, is mentioned in each figure. The deriv-diff operation in all figures only removes the increasing trend by applying a derivative operation on each counter.

alarm will eventually pollute the history and can prevent a nearby event to be missed (FN) or a normal point to be mistaken as an event (FP).

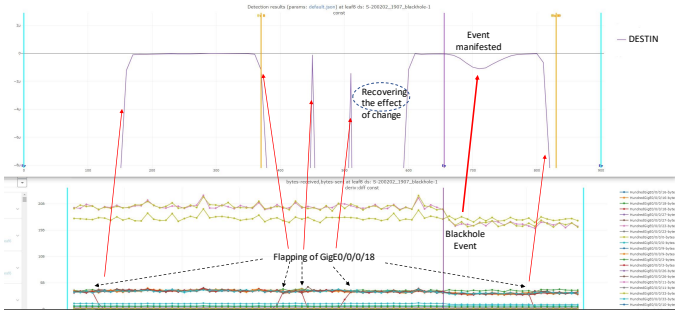
Fig. 13 showcase some of the datasets where the limitation of the sigma detector resulted in FNs or FPs.

A good example for illustrating the polluting of the history of the binary detected can be seen in Fig. 14. It is noteworthy to mention that based on DESTIN's numerical results,



(a)

FN due to fixed delta in sigma detector



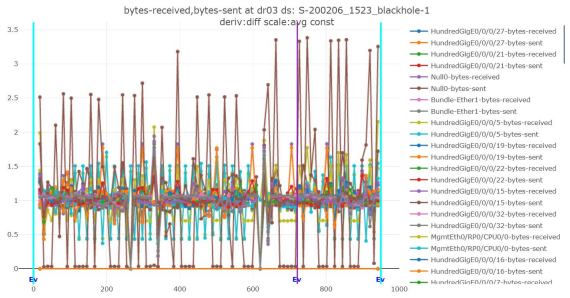
(b) Above: DESTIN indicator. Below: visualized dataset.

Fig. 13. Sigma detector issues:a)Missing of an event due to the fixed delta in sigma detector causing a FN, as well as polluting the boundary history of the sigma detector b)The flapping of interface HundredGig0/0/0/18 causing FPs as well as an FN. The corruption of the sigma detector history is due to close events. Due to the recovery period of the sigma detector, none of the immediate changes of the flapping interface is detected

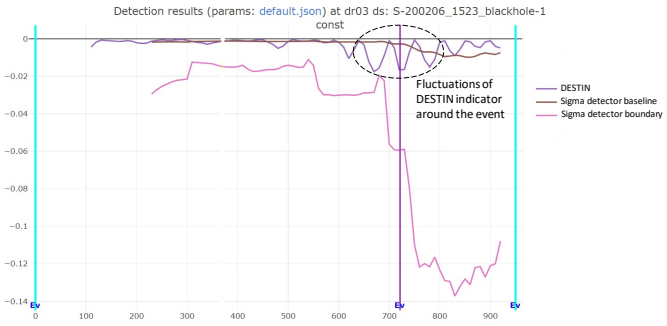
DESTIN is less efficient in detecting slight increase of traffic, compared to traffic decrease (with the same value changes). It is often due to the fact that rising of the traffic occurs for various reasons and thus, is more often seen than decreasing of traffic from the baseline. The existence of such points in the two matrices (test and reference) of DESTIN will result in a subspace which is more similar to when traffic has sudden rises in the traffic. Therefore, the angles when the traffic rises are less pronounced compared to the falling of traffic. In conclusion, both type of events can be seen in the indicator, yet using the same delta value in the binary detector might result in missing some of the enable interface and enable BFD events compared to shutdown interface and break BFD events. To address this problem, two different delta values could be considered. The second delta value can raise warnings which could be potential events. These warnings will make it possible to detect the mentioned events yet, it is noteworthy to mention that it comes at a cost of a few unwanted warnings which can be dismissed after observing only the DESTIN indicator by an operator. Note that, even if the changes may appear hard to recognize for machines, the meaningful changes in DESTIN indicator are rather evident for humans.

VIII. Conclusion

This paper proposes ADT as a novel AI-driven monitoring tool of network devices which is light-weight enough to be run on routers. DESTIN as the detector block of ADT is



(a) FlowMix data with a blackhole event



(b) DESTIN indicator for the same dataset

Fig. 14. The difficulty of detection of events in FlowMix due to the bursty nature. Though the event cannot be interpreted from the traffic at first glance, DESTIN's indicator shows the changes in the network state yet the binary detector is not capable of distilling the information. The step by step pollution of the binary detector history is very well depicted here.

an online multivariate change detection methodology based on principal angles to assess the state changes of network elements for a more efficient and accurate network monitoring. The results show that the method has a potential to timely detect state changes with 28%, 14.5% and 22.5% average worst case improvement, separately, in precision, recall and f1-score compared to PCA for traffic data. The paper provides extensive discussion on the proposed measures, the performance of the detector block per event-type, traffic and device. It also performs root causing to reveal the reasons behind the missing and wrong detections. The datasets used in this evaluation are also made publicly available. It is seen that DESTIN performs better for almost all types of changes but specially well with events such as ECMP hash changes, since these events incorporate a modification in the inter-dependencies of counters. In addition, due to the robustness in pre-processing assumptions, it performs better for traffic blackholing and interface shutdown changes. The results as presented in benchmarking framework also show that a change in a network can propagate beyond what is naturally expected by an expert.

In the future, the scope of application of DESTIN can be explored. More events of different nature, such as network attacks, can be simulated and tested. In addition, the boundary of application in terms of time frame can be further explored, by testing the method on micro-burst datasets.

Acknowledgment

This paper was funded by Cisco systems, their funding and support is deeply appreciated. Special thanks to Anton Kiselev, Nikolay Chunosov and Drew Pletcher for their efforts in data collection, tool building and maintaining the lab network. Also special regards to Wenqin Shao and Enzo Fenoglio for providing guidance in the theoretical aspect along the way.

References

[1] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Computer Networks*, vol. 65, pp. 84–98, 2014.

[2] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*. IEEE, 1999, pp. 120–132.

[3] G. Wang, L. Zhang, and W. Xu, "What can we learn from four years of data center hardware failures?" in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2017, pp. 25–36.

[4] J. Cordova-Garcia, "Sparse control and data plane telemetry features for bgp anomaly detection," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 240–245.

[5] Cisco Innovation Edge, "Network anomaly telemetry datasets," https://github.com/cisco-ie/telemetry/tree/master/11, 2019.

[6] O. Lézoray, C. Charrier, H. Cardot, and S. Lefèvre, "Machine learning in image processing," pp. 1–2, 2008.

[7] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[8] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[9] Cisco, "Telemetry dataset," 2019. [Online]. Available: https://github.com/cisco-ie/telemetry/tree/master/11

[10] P. Foroughi, W. Shao, F. Brockners, and J.-L. Rougier, "Destin: Detecting state transitions in network elements," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 161–169.

[11] T. Feltin, P. Foroughi, W. Shao, F. Brockners, and T. H. Clausen, "Semantic feature selection for network telemetry event description," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[12] M. Fullmer and S. Romig, "The osu flowtools package and cisco netflow logs," in *Proceedings of the 2000 USENIX LISA Conference*, 2000.

[13] B. Trammell and C. Gates, "Naf: The netsa aggregated flow tool suite." in *LISA*, 2006, pp. 221–231.

[14] D. Plonka, "Flowscan: A network traffic flow reporting and visualization tool." in *LISA*, 2000, pp. 305–317.

[15] J. Sellens, "{Thresh–A}{Data-Directed}{SNMP} threshold poller," in *14th Systems Administration Conference (LISA 2000)*, 2000.

[16] S. S. Kim and A. N. Reddy, "Netviewer: A network traffic visualization and analysis tool." in *LISA*, vol. 5, 2005, pp. 18–18.

[17] R. Beverly, "Rtg: A scalable snmp statistics architecture for service providers." in *LISA*, 2002, pp. 167–174.

[18] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks," in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[19] Z. Zhou, T. A. Benson, M. Canini, and B. Chandrasekaran, "Tardis: A fault-tolerant design for network control planes," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, 2021, pp. 108–121.

[20] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011, pp. 254–265.

[21] J. Yang, X. Yang, Z. Zhou, X. Wu, T. Benson, and C. Hu, "Focus: Function offloading from a controller to utilize switch power," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2016, pp. 199–205.

[22] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *ACM SIGCOMM computer communication review*, vol. 34, no. 4, pp. 219–230, 2004.

[23] ——, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM computer communication review*, vol. 35, no. 4, pp. 217–228, 2005.

[24] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2007, pp. 109–120.

[25] D. Brauckhoff, K. Salamatian, and M. May, "Applying pca for traffic anomaly detection: Problems and solutions," in *IEEE INFOCOM 2009*, 2009, pp. 2866–2870.

[26] Y. Hua and W. Liu, "Generalized karhunen-loeve transform," *IEEE Signal Processing Letters*, vol. 5, no. 6, pp. 141–142, 1998.

[27] R. Kwitt and U. Hofmann, "Unsupervised anomaly detection in network traffic by means of robust pca," in *2007 International Multi-Conference on Computing in the Global Information Technology (ICCGI'07)*, 2007, pp. 37–37.

[28] H. Cardot and D. Degras, "Online principal component analysis in high dimension: Which algorithm to choose?" *International Statistical Review*, vol. 86, no. 1, pp. 29–50, 2018.

[29] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust subspace learning: Robust pca, robust subspace tracking, and robust subspace recovery," *IEEE signal processing magazine*, vol. 35, no. 4, pp. 32–55, 2018.

[30] Q. Ding and E. D. Kolaczyk, "A compressed pca subspace method for anomaly detection in high-dimensional data," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7419–7433, 2013.

[31] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 147–152.

[32] D. S. Yeung, S. Jin, and X. Wang, "Covariance-matrix modeling and detecting various flooding attacks," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 2, pp. 157–169, 2007.

[33] L. Huang, X. Nguyen, M. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft, "Communication-efficient online detection of network-wide anomalies," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 134–142.

[34] L. Huang, X. Nguyen, M. Garofalakis, M. I. Jordan, A. Joseph, and N. Taft, "In-network pca and anomaly detection," in *Advances in Neural Information Processing Systems*, 2007, pp. 617–624.

[35] T. Huang, H. Sethu, and N. Kandasamy, "A new approach to dimensionality reduction for anomaly detection in data traffic," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 651–665, 2016.

[36] K. De Cock and B. De Moor, "Subspace angles between arma models," *Systems & Control Letters*, vol. 46, no. 4, pp. 265–270, 2002.

[37] Y.-H. Yuan, Q.-S. Sun, Q. Zhou, and D.-S. Xia, "A novel multiset integrated canonical correlation analysis framework and its application in feature fusion," *Pattern Recognition*, vol. 44, no. 5, pp. 1031–1040, 2011.

[38] W. Liu, X. Yang, D. Tao, J. Cheng, and Y. Tang, "Multiview dimension reduction via hessian multiset canonical correlations," *Information Fusion*, vol. 41, pp. 119–128, 2018.

[39] L. Wolf and A. Shashua, "Learning over sets using kernel principal angles," *Journal of Machine Learning Research*, vol. 4, no. Oct, pp. 913–931, 2003.

[40] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, 2005, pp. 333–342.

[41] E. Canbalaban and S. Sen, "A cross-layer intrusion detection system for rpl-based internet of things," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2020, pp. 214–227.

[42] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19–24.

[43] A. Jain, A. Gupta, A. Gupta, D. Gedia, L. Pérez, L. Perigo, R. Gandotra, and S. Murthy, "Trend-based networking driven by big data telemetry for sdn and traditional networks," *arXiv preprint arXiv:1904.10449*, 2019.

[44] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *International Conference on Image Analysis and Processing*. Springer, 2005, pp. 50–57.

[45] S. L. Salzberg, "C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," 1994.

[46] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[47] R. Rojas, "Neural networks: A systematic approach springer-verlag," *Berlin, Deutschland*, 1996.

[48] J. H. Friedman, "Regularized discriminant analysis," *Journal of the American statistical association*, vol. 84, no. 405, pp. 165–175, 1989.

[49] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image processing*, vol. 11, no. 4, pp. 467–476, 2002.

[50] B. Schölkopf, A. J. Smola, F. Bach *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[51] A. R. Webb, *Statistical pattern recognition*. John Wiley & Sons, 2003.

[52] L. Portnoy, "Intrusion detection with unlabeled data using clustering," Ph.D. dissertation, Columbia University, 2000.

[53] P. Laskov, C. Schäfer, I. Kotenko, and K.-R. Müller, "Intrusion detection in unlabeled data with quarter-sphere support vector machines," 2004.

[54] J. Hu, Z. Zhou, X. Yang, J. Malone, and J. W. Williams, "{CableMon}: Improving the reliability of cable broadband networks via proactive network maintenance," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 619–632.

[55] B. Al-Musawi, P. Branch, and G. Armitage, "Bgp anomaly detection techniques: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 377–396, 2016.

[56] "Telemetry configuration guide for cisco 8000 series routers, ios xr release 7.3.x - build intelligence on the router using ai-driven telemetry [cisco 8000 series routers]." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/iosxr/cisco8000/telemetry/73x/b-telemetry-cg-8000-73x/AI-driven-telemetry.html

[57] S. Jalilian, "Monitoring network devices by model-driven telemetry," Sep 2020. [Online]. Available: https://medium.com/@ohumeagle/monitoring-network-devices-by-model-driven-telemetry-87cfe14bb434

[58] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network telemetry framework," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-opsawg-ntf-05, October 2020.

[59] E. M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," Internet Requests for Comments, RFC Editor, RFC 6020, October 2010.

[60] P. Foroughi, W. Shao, F. Brockners, A. Kuriakose, and J.-L. Rougier, "Detection of state transitions in network elements: On-box demo," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 724–725.

[61] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

[62] M. Lavielle and G. Teyssiere, "Detection of multiple change-points in multivariate time series," *Lithuanian Mathematical Journal*, vol. 46, no. 3, pp. 287–306, 2006.

[63] A. V. Knyazev and P. Zhu, "Principal angles between subspaces and their tangents," *arXiv preprint arXiv:1209.0523*, 2012.

[64] Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of computation*, vol. 27, no. 123, pp. 579–594, 1973.

[65] C. Jordan, "Essai sur la géométrie à n dimensions," *Bulletin de la Société mathématique de France*, vol. 3, pp. 103–174, 1875.

[66] J. Hefferon, "Linear algebra third edition," 2018.

[67] X. Wang, J. Lin, N. Patel, and M. Braun, "Exact variable-length anomaly detection algorithm for univariate and multivariate time series," *Data Mining and Knowledge Discovery*, vol. 32, no. 6, pp. 1806–1844, 2018.

[68] L. I. Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 5, pp. 1175–1180, 2011.

[69] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[70] X. Li, S. Wang, and Y. Cai, "Tutorial: Complexity analysis of singular value decomposition and its variants," *arXiv preprint arXiv:1906.12085*, 2019.

[71] "telemetry/telemetry_topology_maps.pdf at master · cisco-ie/telemetry · github," https://github.com/cisco-ie/telemetry/blob/master/topology-description-docs/telemetry-topology-maps.pdf, (Accessed on 09/17/2020).

[72] Cisco Innovation Edge, "Network anomaly telemetry datasets," https://github.com/cisco-ie/telemetry/tree/master/12, 2019.

[73] Keysight, "Network visibility and network test products." [Online]. Available: https://www.keysight.com/us/en/cmp/2020/network-visibility-network-test.html
[74] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves." New York, NY, USA: Association for Computing Machinery, 2006.

Parisa Foroughi is a Ph.D candidate in computer science and networks at Institute Polytechnique de Paris, Télécom Paris, in France. She obtained her B.Sc (2017) and M.Sc (2019) in electrical engineering communication systems from Amirkabir University of Technology in Tehran. She got a Master2 in advance communication networks from university of Paris saclay in France. Her research is focused on the automation and management of the next generation of networks (IP networks and wireless networks) using artificial intelligence and statistical analysis.

Frank Brockners is Distinguished Engineer in Cisco's Emerging Technologies and Incubation group, driving software and architecture development for Edge platforms, solutions, associated services and applications. He is involved in several open source projects and is a Linux Foundation Networking (LFN) board member. Frank is an active IETF member in the standardization of observability/OAM solutions (IOAM). Frank is a frequent speaker at conferences and events, including CiscoLive, where he is recognized as a "CiscoLive Hall-of-Fame Elite", highest speaker rank. Frank holds a diploma degree in Electrical Engineering (Aachen University) and a PhD/Dr degree in Information Science (University of Cologne). When looking to de-stress, Frank grabs his crampons and goes ice climbing in the Alps, Peru, or Argentina which for most people would be the opposite of de-stressing.

Jean-Louis Rougier is a Professor at the Network and Computer Science Department of Télécom Paris, Paris, France. He has been conducting research on various themes related to networking, in particular traffic engineering and routing. His current research interests are network virtualization and automation, network telemetry, and service mesh. Jean-Louis Rougier graduated from Télécom Paris (M.Sc. 1996 and Ph.D. in 1999). He was a visiting Researcher at UCLA in 2011 and Cisco PIRL in 2019.