Anomalous Model-Driven-Telemetry Network-Stream BGP Detection

Rostand A. K. Fezeu[+], Zhi-Li Zhang[*]
Department of Computer Science, University of Minnesota - Twin Cities
fezeu001@umn.edu[+], zhzhang@cs.umn.edu[*]

***Abstract*—There is a growing demand for real-time analysis of network data streams. In recent years, Model Driven Telemetry (MDT) has been developed – in place of conventional methods such as Simple Network Management Protocol (SNMP), Syslog and CLI commands – to provide a fine-grain holistic view of a network at the control, data and management planes. High-frequency MDT data streams generated from network devices enable new ways of designing Network Operation and Management (OAM) solutions, laying the foundation for future "self-driving" networks.**

In this paper we study anomaly detection using MDT data streams in a data center environment. In many commercial data centers, BGP is re-purposed for (policy-driven, path-based) intra-routing (as opposed to inter-domain routing that it was originally designed for) to take advantage of rich path diversity. Several vendors have developed MDT data models using YANG that allow routers/switches to express and stream various BGP features for (centralized) network OAM operations. We develop a systematic MDT data processing and feature selection framework that is portable to multiple MDT vendors. Furthermore, we advance *NetCorDenstream* that builds and improves upon *OutlierDenStream* proposed in [10] for real-time detection of streamed anomalous MDT data. We show that *NetCorDenstream* achieves a 59% reduction in alarms raised when compared with *OutlierDenStream*, thereby reducing the (attention) burden placed on network operators. In particular, it increases alarm detection precision significantly while decreasing false alarms at the expense of a slightly delayed response time.

***Index Terms*—OutlierDenStream, NetCorDenStream, MDT Data.**

I. INTRODUCTION

As networks become increasingly complex, Network Operation and Management (OAM) tasks such as capacity planning, security, network health monitoring, troubleshooting are critical to ensure normal operations of business services. Traditionally, network operators rely on mechanisms and tools such as Simple Network Management Protocol (SNMP), Syslog, and CLI which provide only slow (e.g., every 5 minutes), course-grain, incomplete and often specific information that is hard to operationalize. Their limitations restrict network automation. To these drawbacks, *Model-Driven Telemetry* (MDT) [11] has been developed in recent years and is widely adopted by network vendors. MDT leverages the power of models to implement a high frequency (in the order of seconds) push-based data approach that is more granular and provides greater scalability. For example, MDT enabled devices in a network can generate more than 2.8 billion write requests per minute, store 4.5 petabytes of time series data and receive 25,000 query request per minute [12].

978-1-7281-6992-7/20/$31.00

MDT data streams provide fine-grained visibility to network devices and enable data processing in near real-time, making it possible to achieve network automation at scale. However, processing huge amounts of device-level MDT data streams to obtain *network-wide* visibility and infer network conditions for OAM decision making, e.g., anomaly detection, is still very challenging. Take commercial (enterprise) data centers as an example. Many of them employ BGP as the intra-routing protocol [7] Instead of conventional OSPF and IS-IS to take advantage of rich path diversity in data center topology. Several vendors have developed BGP MDT data models using YANG [8] over NETCONF, RESTCONF or Google Remote Procedure Call (gRPC) protocols to express routers/switches data features used for configuration, notification, and state sharing. The YANG model is a data structure that defines hierarchical (i.e., tree) data where each node has several sensor groups (using, e.g., Cisco's MDT workflow/features), with each group possessing numerous counters. For example, Cisco IOS-XR 6.2.2 YANG hierarchy comprises roughly 378,000 lines with over 45,000 sensor groups described with approximately 6000 related to the BGP protocol alone. Each sensor group contains an average of 100 counters. The average length of stream is greater than 17000. Clearly, careful analysis of these sensor groups is paramount given a specific network task as network telemetry protocols and frameworks are standardized across several vendors. Availability of *voluminous*, *fine-grain* MDT data streams with *rich features* calls for rethinking OAM operations, especially anomaly detection.

In this paper we study anomaly detection using BGP MDT from a large data center as a case study. Our work improves and builds on [10], [9] which employs a clustering approach – OutlierDenStream [1]– for detecting anomalies and raising alarms. In OutlierDenStream, each node in the network operates individually. An alarm is raised upon reception of k consecutive outlier samples at a node and for multiple nodes during the same time slot. In this work, We first engineer a systematic MDT data processing and feature selection framework that is portable to various vendors. By leveraging fine-grain MDT data, we develop a novel unsupervised anomaly detector engine that incorporate time as well as network proximity-based heuristics. Our engine intelligently combines node level alarms which decreases false positive alarms while increases the true positive alarms. Therefore it reduces the attention burden placed on network operators for verifying anomalies

and making decisions. The outline and major contributions of our paper are summarized below.

- First, we begin by engineering a systematic MDT data processing and feature selection framework, dubbed *Bravo* portable to other MDT vendors (see Section II-B). The source code, MDT data and all prerequisite scripts of our results are available on github [5].
- Second, we formulate two unsupervised learning mechanisms befitting MDT streaming data using *Denstream* [3]. The first mechanism ingeniously archives a 59% reduction in alarms raised juxtaposed with *OutlierDenStream* [1], the comparative method (see Section III-B for an overview), thus increasing precision at the expense of delay. These two mechanisms are presented in Section IV.
- Third, the second benchmark bubbles heuristics within fine-grain MDT data fostering early bird anomaly reconnaissance diagnostic steps. The efficacy of our method is demonstrated through experimental evaluations presented in Section V and the paper is concluded with discussion in Section VI.

II. Datasets and Bravo

In this section, we first describe the datasets used and annotated ground truth labels. Next, we present *Bravo*, a systematic MDT data preprocessing and feature selection framework.

A. Datasets

To the best of our knowledge, the only publicly available MDT data is the dataset by Rossi. et. al. [10]. We use this dataset to evaluate the anomaly detection algorithms proposed in this study. We also engineer a virtual lab and generate MDT data with up to 1.2 Tbps cumulative network traffic. Table I shows all datasets used.

Network topology: Rossi. et. al. [10] dataset was collected on an engineered Cloud Service Provider (CSP) data center network. The network contains 12 physical Cisco IOS-XR 6.2.2 routers connection in a tree-like topology with eight child nodes and four parent nodes. The BGP protocol is the only routing protocol configured on the network as per [7]. Each child node is connected to the four parent nodes via 4X100GB fibre cables with an average of 25 interfaces. The Cisco routers are then connected to commodity servers to generate traffic including 4K YouTube streaming, VoIP, Skype-1050P, TCP, G.711a calls and AMR-WP.

In our virutal lab, the network architecture (not depicted here because of lack of space) meets a realistic multi-tiered data center network topology as per [13]. It is equipped with six Cisco IOS XR Release 6.2.1 virtual routers connected in a tree-like architecture. All the nodes publish telemetry data to a server collector as per pipeline [2]. Unlike the Rossi. et. al. network architecture configured with only BGP protocol MDT sensor groups, our virtual lab is richer with more MDT sensor groups configured within each router[1], thus significantly increasing the length of streams.

TABLE I: Experimental MDT data used

Duration	Stream length	No. Anomalies	Network traffic
1 h	200×10^3 Pts	11	1 Tbps
0.55 h	250×10^3 Pts	8	1 Tbps
0.72 h	150×10^3 Pts	5	1 Tbps
2 h	1.3×10^6 Pts	12	1 Tbps
24 h	1.5×10^6 Pts	None	1.2 Tbps

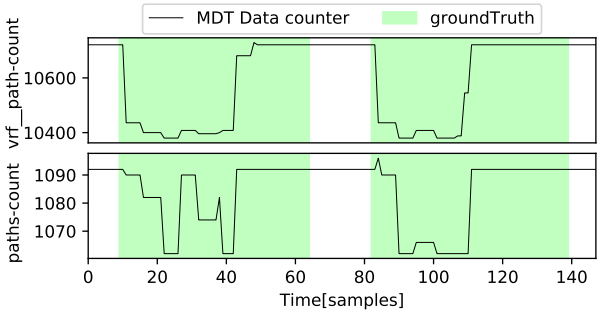

Fig. 1: Sample MDT counters and annotated ground Truth

MDT data collection and labeling: Every five seconds, all the routers push the pre-configured telemetry features to a centralized data collector. All features are described via YANG [8]. At the collector, the MDT data is decoded, and stored in compressed CSV files [4].

The traffic load on the network is varied from null, 0.5Tb to 1 Tbps and controlled anomalies are injected and labeled. Controlled anomalies of types BGP Clear, BGP port flapping, and BGP leaks are manually triggered and the start and end time, node on which the anomaly was triggered, and the type of anomaly is logged. Every anomaly experiment conducted on the network starts with a *normal* period lasting about four minutes. Different anomalies are injected at randomized nodes recording the ground truth labels.

To validate the ground truth labels, Fig. 1 shows the *vrf_path-count* and *vrf_paths-count* values for the BGP clear experiment of a parent node and its annotated ground truth. The lime band indicates the ground truth start and end time. As observed, the BGP clear command triggers a *vrf_path-count* and *vrf_paths-count* re-configuration on the router. Thus, this validates the correctness of the ground Truth labels logged. It is important to mention that, we leverage the ground truth labels solely to assess the performance of our models. Precisely, the ground truth is use to validate the precision, recall, false alarms and delay incurred during our experiments.

B. Bravo

Bravo is a "plug & play" system for "on-the-fly" customization of metrics of interest and for configuring various (readily

[1] Cisco-IOS-XR-cdp-oper:cdp_nodes_node_neighbors_summaries
Cisco-IOS-XR-infra-statsd-oper:infra-statistics_interface_latest_generic
Cisco-IOS-XR-infra-syslog-oper:syslog_messages_message
Cisco-IOS-XR-ip-rib-ipv4-oper:rib_rib-table-ids_summary-protos
Cisco-IOS-XR-ipv4-arp-oper:arp_nodes_node_traffic-interfaces
Cisco-IOS-XR-ipv4-bgp-oper:bgp_instances-active_default-vrf_neighbors
Cisco-IOS-XR-ipv4-bgp-oper:bgp_instances-active_default-vrf_process-info
Cisco-IOS-XR-ipv4-ospf-oper:ospf_processes_default-vrf_ospf-summary
Cisco-IOS-XR-nto-misc-oper:memory-summary_nodes_node_detail
Cisco-IOS-XR-pfi-im-cmd-oper:interfaces_interface-summary
Cisco-IOS-XR-procmem-oper:processes-memory_nodes_process-ids
Cisco-IOS-XR-shellutil-oper:system-time_uptime
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring_cpu-utilization

available) algorithms used in MDT data preprocessing to produce normalized ML-ready datasets. *Bravo* is also a systematic counter selection framework that is specifically geared towards network MDT data preprocessing via YANG. While designed using Cisco BGP YANG data model, it is portable to other MDT vendor workflows, and thus deployed on networks with different vendor routers producing telemetry data. The pipeline is modularized and each (optional) module contains a readily available toolbox with a variety of techniques suitable for different MDT ML tasks.

The counter selection pipeline begins with the *split_by_nodes* module. This is because current MDT framework deployments are designed with a central telemetry collector [6]. This module aids in node-level ML tasks. The next module is a *data_cleaning* module – a threshold-based filter – to remove faulty telemetry counter readings such as NaN and negative readings. Since not all counters per sensor group are relevant to a given specific network event of interest, we introduce a *counter_correlation* module. We apply statistical methods such as Pearson, Spearman and Kendall correlation tests and discard all counters with less than 0.1 correlation factors. We assume each counter value follows a normal distribution and introduce a *counter_distribution* module that discards features with less than 1% (user-defined threshold) standard deviation, i.e., unaffected counters given a specific network event. The last module *Normalization* characterizes and normalises the data to be ready for ML tasks.

We validate *Bravo* by performing a domain expert counter selection. The datasets obtained after *Bravo* preprocessing have at least 60% counters overlap compared to the domain expert *ControlPlane* and *DataPlane* counters selection. In Sec. V-B, we compare the precision, recall, false alarms and delay of our algorithm against both counter selection approaches.

III. DenStream and OutlierDenStream

This section starts by presenting *DenStream* [3] and next introduces *OutlierDenStream* [10] and its major drawbacks.

A. DenStream

DenStream is a streaming clustering algorithm for evolving data streams. DenStream uses a Damped window model to extend the density-based approach introduced in DBScan making it suitable for online model. The algorithm assigns a weight/importance to each data point and this weight decreases exponentially with time via a fading function $f(t) = 2^{-\lambda t}$, where $\lambda > 0$ is the decay factor and t is the time. The algorithm modifies the core-objects of DBScan and introduces the concept of core-micro cluster (*c-micro-cluster*) defined as *CMC*(w, c, r), where w, c and r are the weight, center and radius respectively. The weight of *CMC* must be greater than or equal to μ (user-defined parameter) and the radius $r \leq \epsilon$ [3]. Formally, *CMC*(w, c, r) for close points $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ with time stamps $T_{i_1}, T_{i_2} \ldots, T_{i_n}$ is defined as:

$$w = \sum_{t=0}^{t=t_c} f(t - T_{i_j}),$$

$$c = \frac{\sum_{t=0}^{t=t_c} f(t-T_{i_j}) p_{i_j}}{\omega},$$

$$r = \frac{\sum_{t=0}^{t=t_c} f(t-T_{i_j}) dist(p_{i_j}, c)}{\omega}$$

where $w \geq \mu$, $dist(p_{i_j}, c)$ denotes the $Euclidean$ distance between point p_{i_j} and the center c, and r is the radius.

Due to the dynamic nature of streaming environments, DenStream introduces the concept of potential core-micro cluster, (*p-micro-cluster*) for incremental computation. The *p-micro-cluster* weights are defined in term of the weight linear sum ($\overline{CF^1}$) and the weight squared sum($\overline{CF^2}$) of the points, with:

$$\overline{CF^1} = \sum_{t=c}^{t=t_c} f(t - T_{i_j}) p_{i_j},$$

$$\overline{CF^2} = \sum_{t=c}^{t=t_c} f(t - T_{i_j}) p_{i_j}^2$$

The center and radius are:

$$c = \frac{\overline{CF^1}}{w},$$

$$r = \sqrt{\frac{\overline{CF^1}}{w} - \left(\frac{\overline{CF^1}}{w}\right)^2}.$$

While the *p-micro-cluster* allows the model to be maintained dynamically with time, in an event of an outlier, it will generally not capture the illustrative of the data stream as more data points are streamed. Thus, DenStream also introduces the concept of an *outlier buffer* to temporally store outlier-micro cluster (*o-micro-cluster*) giving them the opportunity to be promoted to a *p-micro-cluster* as more points are streamed. The main difference between a (*p-micro-cluster*) and an (*o-micro-cluster*) are the constraints on the weight, $w \geq \beta\mu$ and $w < \beta\mu$ respectively. $0 < \beta \leq 1$ is the parameter that determines the threshold of *o-micro-cluster* relative to the *p-micro-cluster*.

Therefore, when a new sample p arrives, DenStream merges p into its nearest *p-micro-cluster*, provided the new radius after p, is less than or equal to ϵ. If the merge attempt fails, the system then tries to merge p into its nearest existing *o-micro-cluster*. If the radius is less than ϵ, it merges p. If the new weight of the *o-micro-cluster* is large enough to be promoted to a *p-micro-cluster*, the system removes it from the outlier buffer and adds it to the system as a *p-micro-cluster*. If p cannot be merged to any existing *micro-cluster*, a new *o-micro-cluster* is created in the outlier buffer.

In our context, when a new telemetry sample p is merged to a *p-micro-cluster*, the sample is classified as *normal*; *abnormal* otherwise. Notice that λ and β are the only two free decision variables with physical meaning. λ dictates the importance of historical data to the current stream cluster decision while β is the outlier-threshold which controls when *o-micro-clusters* gets promoted to *p-micro-clusters* and vice-versa.

B. OutlierDenStream

OutlierDenStream [1] is an anomaly detection engine based on *DenStream* [3]. It is governed by two criteria: 1) *Temporal order,* k_t: Each node on the network operates individually and an alarm is raised upon reception of k consecutive outlier samples at a node and 2) *Spatial order,* k_s for multiple nodes during the same time slot. Major drawbacks of *OutlierDen-*

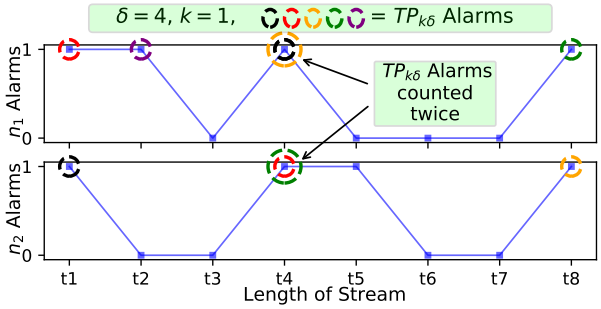

Fig. 2: $TP_{k,\delta}$ alarms counted twice

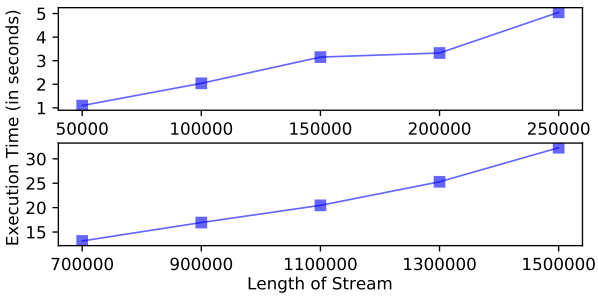

Fig. 3: Execution time vs. length of stream

Stream which hinders applicability and deployment include: 1) The total number of alarms raised is very high, which may overwhelm data center network operators. 2) The number of false alarms is also high resulting in a waste of resources.

IV. Anomaly Detection Mechanisms

To detect anomalous MDT data and raise alarms, we treat each MDT data stream pushed by routers as points to be clustered. We devise two simple unsupervised network correlation anomaly detection mechanisms for anomaly detection.

It is important to mention that we acknowledge the need for a sensitivity analysis of λ and β given different MDT datasets. However, Rossi. et. al. [10] have conducted a thorough sensitivity analysis on the publicly available MDT datasets and have concluded that $\lambda = 0.05$ and $\beta = 0.15$ are suitable values for these datasets. Thus we adopt these values for all our experiments.

A. Time Proximity Network Correlation

To reduce the total number of alarms raised, we adapt a holistic approach and intelligently correlate alarms raised by different nodes on the network. Therefore, we introduce *Time proximity NetCorDenStream*, $TP_{k,\delta}$.

DEFINITION 1 (***TimeProximity NetCorDenStream, $TP_{k,\delta}$***). *An alarm is raised only if node n_i detects an anomaly at time t, and at least k-neighbors of node n_i flags a sample as anomalous at time $t + \delta$.*

Particularly, a false positive alarm occurs when node n_i and at least k neighbors of node n_i flags a sample as anomalous but the ground truth labels the system as being in normal condition. Increasing δ reduces the number of alarms raised as $TP_{k,\delta}$ alarms for less than δ go un-noticed. Similarly, increasing k reduces alarms raised as $TP_{k,\delta}$ alarms less than k go un-flagged by the system. Intuitively, increasing δ trade offs the precision for increased in delay.

Challenge: The performance of *Time proximity NetCorDenStream* unsupervised learning criteria is heavily affected by the alarms flagged by each nodes on the network. $TP_{k,\delta}$ alarms raised as per Def. 1 are double counted. We demonstrate this challenge in Fig. 2. Consider a network with two nodes, n_1 and n_2 with n_1 and n_2 neighbors. Now assuming $k = 1$ and $\delta = 4$, i.e., $TP_{1,4}$ raises an alarm if at time t_4, node n_2 raises an alarm and node n_1 previously flagged an alarm at time t_1 or vice versa. The anomaly flagged by node n_1 at time t_1 and node n_2 at time t_2 are $TP_{1,4}$ correlated (red dotted circle). And at time t_4, the alarm raised by n_2 is $TP_{1,4}$ correlated to the anomaly flagged by n_1 at time t_8 (green dotted circle). Thus $TP_{1,4}$ alarms raised at time t_4 are double counted.

To address this challenge, we designate a *baseNode* depending on the data center network architecture. Ideally, this *baseNode* is the most centrally located, with more neighbors (the e-gress/in-gress) node on the network. In a multi-tiered or fat tree network, the *baseNode* would be at the core level. In a hybrid topology, the *baseNode* will be at level 1 [13]. In our case, the MDT dateset was collected on a multi-tiered network, thus our *baseNode* is one of the four parent nodes in the case of the open sourced datasets. In our virtual lab, it can be any node at level 1.

B. Signature Proximity Network Correlation

To bubble anomaly heuristics that aid during early troubleshooting steps, we introduce *Signature proximity NetCorDenStream*, $SP_{k,\delta}$.

DEFINITION 2 (***SignProximity NetCorDenStream, $SP_{k,\delta}$***). *Counters involved in $TP_{k,\delta}$ alarms at time $t + \delta$ with values beyond one standard deviation from the mean of past observed normal samples i.e., p-micro-cluster samples are sign proximity network correlated.*

Specifically, $SP_{k,\delta}$ goes beyond $TP_{k,\delta}$ alarms raised. As more $TP_{k,\delta}$ alarms are raised by decreasing both k and/or δ, more counter metric involved in the alarm are flagged and bubbled. Intuitively, $SP_{k,\delta}$ precision, recall, and false alarms tradeoffs the delay for affected counters to be surfaced. Moreover, $SP_{k,\delta}$ KPIs are transitively affected by k and δ via $TP_{k,\delta}$ alarms.

V. Experimental Evaluation

In this section, we present a thorough experimental evaluations of our proposed algorithms. We implement *OutlierDenStream* [1], the comparative method and our benchmarks. All experiments were conducted on a 2.6 GHz Intel Core i5 MacBook Pro with 16GB memory, running macOS Mojave.

A. Bravo Evaluation

The efficiency of our proposed systematic MDT data processing is measure by the execution time. In Fig. 3, we show the execution time vs. length of stream of both the

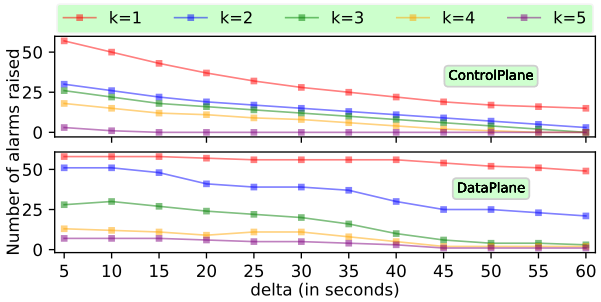

Fig. 4: Alarms raised vs. delta

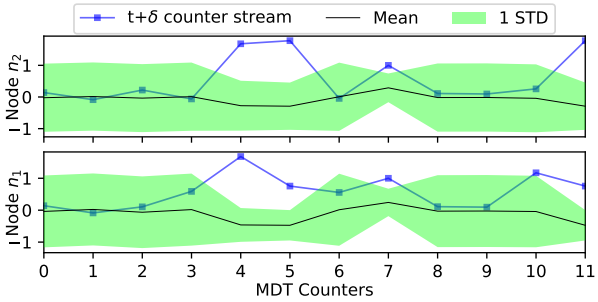

Fig. 6: $SP_{k,\delta}$ Results for $k = 1$ and $\delta = 10$.

virtual lab MDT data collected (bottom plot) and the open sourced available MDT dataset (top plot). We can see that, the execution time is directly proportional to the length of the stream. Additionally, observe that our proposed MDT data processing pipeline takes less than 3 seconds to process 100,000 point and about 33 seconds to process 1.5 million data points. Therefore, it can comfortably handle high speed data streams. In the next section, we further demonstrate the benefit of our counter selection approach while showing that it achieves better precision compared with the domain expert counter selection.

B. Time Proximity Network Correlation Evaluation

First, we demonstrate how varying k and δ significantly affect $TP_{k,\delta}$ alarms raised. In Fig. 4, when delta is altered from 5 seconds to 60 seconds, $TP_{k,\delta}$ alarms generally decreases. Reason being, $TP_{k,\delta}$ alarms less than delta go unnoticed. Additionally, tuning k from 1 ($k = 1$) to 5 ($k = 5$) neighbors, decreases $TP_{k,\delta}$ alarms raised. i.e., Monitoring more neighbors reduce the likelihood for $TP_{k,\delta}$ alarms, provided the underline traffic is not routed via all k nodes[2].

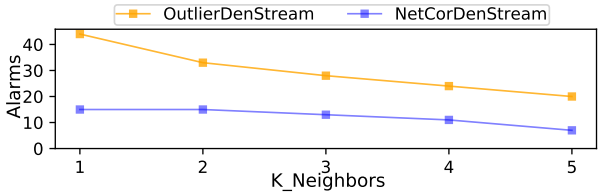

Fig. 5: Alarms vs. k_neighbors.

Second, we compare the number of alarms raised by *OutlierDenStream* and our proposed detection criteria, *NetCorDenStream,* $TP_{k,\delta}$. As shown in Fig. 5, *NetCorDenStream* with $\delta = 5$ intelligently correlates different alarms raised across several neighbor nodes, decreasing the overall number of alarms. Thus, we increase the precision at the expense of delay[3]. Intuitively, a higher δ increases the precision even further[4].

Next, we seek to demonstrate the reliability of our proposed algorithm by contrasting the domain expert counter selection with our systematic counter selection, *Bravo*. While it is possible for single node alarms to go un-flagged, we assume zero false negatives in this case. In Table II, we keep k constant ($k = 2$) and vary *delta* (δ = 5, 30, and 55 seconds) showing the precision, recall, false alarms and delay. Notice that, all performance metrics exhibit the same behavior per dataset regardless of the counter selection approach. However, our systematic counter selection yields a higher precision, reducing the number of false alarms raised at the expense of delay. On the contrary, Table III shows the KPIs obtain for fixing δ ($\delta = 15$) and changing k (1, 3, and 5 neighbors). Again observe that, our counter selection approach achieves higher precision and lower false alarms which tradeoffs the delay.

C. Signature Proximity Network Correlation Evaluation

In Fig. 6, we show the system state at time $t + \delta$ for two neighbor nodes. The lime colored band shows one standard deviation from the mean of the MDT data counters, and the blue line represents MDT data streamed at time $t + \delta$. Notice that similar counters metrics are involved in anomaly for both nodes, i.e., counters 4, 5, 7 and 11 are both involved in the $TP_{k,\delta}$ alarm raised, thus a $SP_{k,\delta}$ alarm. By identifying the counters involved in an alarm (i.e., $SP_{k,\delta}$), an operator can have a first insight into what might have cause the alarm. Domain knowledge can be used to quickly attend to the alarm.

VI. Discussion and Conclusion

In this work, we have proposed *Bravo*, a systematic MDT data processing and feature selection framework that is portable to other MDT vendors. We have also developed *NetCorDenStream*, a novel unsupervised anomaly detector engine equipped with two anomaly mechanisms: (i) *TimeProximity NetCorDenStream*, $TP_{k,\delta}$, Which intelligently combines node level alarms raised increasing true positive alarms and precision at the expense of delay. (ii) *SigProximity NetCorDenStream*, $SP_{k,\delta}$, which incorporates counter heuristics from within $TP_{k,\delta}$ to aid network operators in early-bird troubleshooting steps. Evaluation demonstrates that our system outperforms *OutlierDenStream*: it reduces the number of alarms raised by 59% while increasing precision; the decrease in false alarms rate is achieved with a slight delay in alarms.

We conclude with several takeaways: 1) k and δ are tuneable parameters set by network operators. On one hand, operators who seek to increase precision at the expense of delay can set a higher k ($k = 5$) and delta low ($\delta = 20$ seconds).

[2] In this experiment, the dataset used is *bgpclear_apptraffic_2hourRun*. Bottom plot shows Domain Expert Data plane counter selection approach. Top plot shows the Control plane counter selection approach

[3] As expected, *NetCorDenStream*, incurs more delay as delta increases. This is because an alarm is delayed to be flagged only after delta seconds. Hence, generally, the delay of each alarm will be at least delta seconds

[4] As delta increases to 60 seconds, $TP_{k,\delta}$ decreases the number of alarms raised by about 83% compared with *OutlierDenStream*.

TABLE II: $TP_{k,\delta}$ Results for $k = 2$ delta changing.

Dataset	Features	Precision	Recall	False Alarms	Delay [in seconds]	Delta [in seconds]
bgpclear_second	Control Plane	0.800	1	36	12	5
		0.763	1	36	34	30
		0.700	1	36	59	55
	Date Plane	0.771	1	77	12	5
		0.729	1	75	37	30
		0.670	1	72	62	55
	Bravo	0.835	1	16	54	5
		0.801	1	15	68	30
		0.764	1	14	95	55
portflap_first	Control Plane	0.619	1	48	19	5
		0.566	1	48	75	15
		0.558	1	47	64	40
	Date Plane	1.000	1	3	82	5
		0.690	1	3	99	15
		0.591	1	0	115	40
	Bravo	0.806	1	13	80	5
		0.783	1	13	90	15
		0.707	1	0	115	40

TABLE III: $TP_{k,\delta}$ Results for $\delta = 15$ and k changing.

Dataset	Features	Precision	Recall	False Alarms	Delay [in seconds]	K_neighbors
bgpclear_first	Control Plane	0.799	1	42	35	1
		0.818	1	32	40	3
		0.828	1	27	44	5
	Date Plane	0.675	1	128	38	1
		0.750	1	88	38	3
		0.785	1	56	39	5
	Bravo	0.957	1	3	93	1
		1.000	1	0	112	3
		1.000	1	0	107	5
bgpclear_apptraffic	Control Plane	0.733	1	56	41	1
		0.742	1	42	44	3
		0.773	1	22	49	5
	Date Plane	0.563	1	154	49	1
		0.663	1	93	50	3
		0.802	1	23	58	5
	Bravo	0.940	1	5	76	1
		0.967	1	1	82	3
		1.000	1	0	84	5

On the other hand, operators can set k small ($k = 1$) and delta high ($\delta = 55$ seconds) to reduce false alarms. 2) $SP_{k,\delta}$ benchmark quickly brings forth specific counters which trigger alarms, allowing network operators to quickly handle alarms. Thus, *NetCorDenStream* is very practical and applicable across several vendor networks.

VII. Acknowledgment

This research was supported in part by NSF grants CNS-1618339, CNS 1814322, CNS-1836772 and CNS-1901103.

References

[1] A. Anrputina, D. Rossi, A. Bifet, S. Barth, D. Pletcher, C. Precup, and P. Nivaggioli. outlierdenstream, Aug 2018.

[2] S. Cadora, S. Cadora, R. Fichmann, Leah, and Lavina. Introducing pipeline: A model-driven telemetry collection service, Apr 2017.

[3] F. Cao, M. Estert, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[4] Cisco-Ie. Network anomaly telemetry dataset. https://github.com/cisco-ie/telemetry, Sep 2019.

[5] R. A. K. Fezeu. Netcordenstream. https://github.umn.edu/fezeu001/NetCorDenStream, 08 2020.

[6] M. Korshunov and T. B. Claise. Advanced topics in cisco ios telemetry. https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2019/pdf/BRKSPG-2503.pdf, 2019.

[7] P. Lapukhov, A. Premji, and J. Mitchell. Use of bgp for routing in large-scale data centers. *Internet Requests for Comments RFC Editor RFC*, 7938, 2016.

[8] T.-f. S. I. E. T. F. I. M. Bjorklund, Ed. Yang - a data modeling language for the network configuration protocol (netconf). https://tools.ietf.org/html/rfc6020, October 2010.

[9] A. Putina, S. Barth, A. Bifet, D. Pletcher, C. Precup, P. Nivaggioli, and D. Rossi. Unsupervised real-time detection of bgp anomalies leveraging high-rate and fine-grained telemetry data. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE, 2018.

[10] A. Putina, D. Rossi, A. Bifet, S. Barth, D. Pletcher, C. Precup, and P. Nivaggioli. Telemetry-based stream-learning of bgp anomalies. In *ACM SIGCOMM Workshop on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA'18)*, Aug. 2018.

[11] H. Song. Network telemetry framework. https://tools.ietf.org/id/draft-song-opsawg-ntf-02.html, 12 2018.

[12] R. Stanic. Model driven telemetry - foundation for big data analytics. https://www.ausnog.net/sites/default/files/ausnog-2017/presentations/2.8_Rada_Stanic_AusNOG2017.pdf, 08 2017.

[13] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam. A comparative analysis of data center network architectures. In *2014 IEEE International Conference on Communications (ICC)*, pages 3106–3111. IEEE, 2014.