Telemetry-based stream-learning of BGP anomalies

Andrian Putina, Dario Rossi, Albert Bifet
Telecom ParisTech
name.surname@telecom-paristech.fr

Steven Barth, Drew Pletcher, Cristina Precup, Patrice Nivaggioli
Cisco Systems
name.surname@cisco.com

ABSTRACT

Recent technology evolution allows network equipments to continuously stream a wealth of "telemetry" information, which pertains to multiple protocols and layers of the stack, at a very fine spatial-grain and high-frequency. Processing this deluge of telemetry data in real-time clearly offers new opportunities for network control and troubleshooting, but also poses serious challenges.

We tackle this challenge by applying streaming machine-learning techniques to the continuous flow of control and data-plane telemetry data, with the purpose of real-time detection of BGP anomalies. In particular, we implement an anomaly detection engine that leverages DenStream, an unsupervised clustering technique, and apply it to features collected from a large-scale testbed comprising tens of routers traversed by 1 Terabit/s worth of real application traffic. In spirit with the recent trend toward reproducibility of research results, we make our code and datasets available as open source to the scientific community.

1 INTRODUCTION

Nowadays network Operations and Management (OAM) increasingly rely on the ability to stream and process, in near real-time, useful "features" from network equipment. An integral part of the OAM process is, e.g., to ascertain whether the operational conditions are normal or *anomalous*.

Simple Network Management Protocol (SNMP) has long been the de facto standard to gather fairly coarse information from the network management, control and data planes. Consequently, SNMP has been used for anomaly detection for long time [21]. In the SNMP paradigm, the server initiates the data collection from hundreds of devices, with a pull-based approach, at traditionally low frequency (i.e., in the order of minutes). More recently, Model-driven telemetry (MDT) [1–4, 22] has emerged as an interesting alternative to SNMP: instead of having to periodically poll at a low rate (as in SNMP), under MDT subscribers receive continuous stream of operating state information in a standard structured format. In addition to supporting periodic export, MDT further enables, e.g., to trigger data publication when specific conditions are met.

Rather typically, a workflow common to several vendors (such as Cisco [1], Arista [2], Juniper [3] and Huawei [4]) is to express features via YANG [10, 11] data models, encoded with the Google Protocol Buffer (GPB) format, that are then transmitted via the Google Remote Procedure Call (GRPC) protocol. While the use of standard formats and protocol for their export is very desirable, and while the abundance of information is desirable for fine-grained monitoring, however it becomes necessary to also process MDT data as they are streamed – a challenging task at the heart of our work.

While anomaly detection is surely not a green field [13], however there are three key differences from this work and others related effort. Notably, we are first to leverage a dataset of features directly exported by network software via YANG, which makes the features more easily identifiable – and possibly portable across vendors. Second, whereas there is an extensive literature on BGP anomaly detection (see [9] and references therein), however the literature has mostly focused on the BGP inter-domain context. In this work we instead consider Content Service Provider (CSP) datacenter that, following recent trends[19], are designed with BGP as the only routing protocol. To the best of our knowledge, this study is the first to perform telemetry-based BGP anomaly detection on CSP networks. Last but not least, we remark an important methodological difference: whereas there is a growing attention to data stream clustering techniques[15, 20], to the best of our knowledge DenStreaming has only seldom [17, 18] been used in network-related anomaly detection (raw headers are used as simple features in [17], whereas [18] deals with twitter spam). Summarizing our contributions:

- (Sec.2) We engineer a realistic CSP testbed, loaded with up to 1 Tbps aggregate traffic, that we use to generate telemetry datasets annotated with manual ground truth about anomalous injected events. We make these datasets available to the community at [5]
- (Sec.3) We devise an unsupervised clustering algorithm based on DenStreaming[12] that is apt at operating over streaming data, offering its open-source implementation at [6]
- (Sec.4) We perform an exhaustive evaluation of the algorithm over the released datasets, comparing it classic approaches (e.g., K-Means or DBScan), gathering several findings coming from both machine-learning (in terms of portability and accuracy) as well as domain-expert (in terms of feature selection and timeliness).

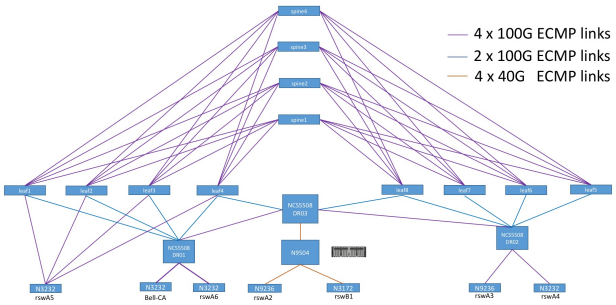

Figure 1: Topology

2 TESTBED AND DATASETS

Testbed. Our testbed is depicted in Fig.1 and replicates a traditional clos topology of a CSP datacenter. On the physical level, it comprises 8 leaf nodes interconnected via 4 spine nodes. For redundancy, each leaf is connected to each spine via 4×100Gbps fiber links, so that the average testbed node has 25 interfaces. On the operational level, the datacenter is designed with BGP as the only routing protocol, following guidelines in [19].

Real application mixtures are generated from servers in the racks (not shown in the picture) connected to the ToR switches (the Nexus 2/3/5000 and 9000 series) to generate up to 1 Tbps of aggregated traffic (a mixture of TCP, AMR-WP VoIP and G.711a calls, Skype-1050P and Blue Ray and 4K YouTube streaming). Therefore, whereas the testbed does not involve real users, it do however uses real equipment, protocols and applications used in production networks.

Data collection and labeling. We use the testbed to perform multiple experiments with different characteristics, that are listed in Tab.1. At a sampling rate of 5 seconds, each of the N nodes streams a snapshot of its F features to the collector: each experiment is a point $X \in R^{NSF}$ where N is the number of nodes, S is the number of collected samples during that experiment and F the number of features. All the available features are described by the YANG models [7] and then extracted, decoded and stored by Pipeline [8] as compressed CSV files available at [5].

In particular, we vary the traffic *load* (null, 0.5 Tbps and 1 Tbps), the *number* (from 0, to few 10s to some hundreds) and *type* (BGP port flapping, BGP leaks and BGP clear) of injected anomalies and the *duration* of the run. To assist reproducibility, the table is also annotated with the specific section of the paper in which a particular dataset is used.

We classify the working condition of our system in two categories, i.e., *normal* vs *anomalous*. The system works by default in normal mode, and each experiment starts with a normal period (lasting 40 samples), after which we start injecting controlled anomalous events at randomized node locations. We only inject a single anomaly at a time, spacing out anomalies by either 120 or 300 seconds (depending on the dataset), and keep track of the injected event in a *ground truth* database available at [5].

We point out that we do not leverage ground truth information to build our data-driven models (i.e., as one would to in case of supervised classification), but rather use ground truth only to assess the precision and recall of our unsupervised methods (see Sec.3-4).

Table 1: Experimental datasets available at [5]

#	Traffic load	No. Anomalies	Duration	Used for
0	0	0	1 h	Tuning ϵ (sect. 4.1)
1	500Gbps	0	1 h	Tuning ϵ (sect. 4.1)
2	1Tbps	11	1 h	Tuning parameters (sect. 4.1)
3	1 Tbps	8	0.55 h	Tuning parameters (sect. 4.1)
4	1 Tbps	5	0.72 h	Tuning parameters (sect. 4.1)
5	1 Tbps	12	2 h	Test parameters (sect. 4.2)
6	0	12	2 h	Valid. CP vs DP (sect. 4.3)
7	0	130	72 h	Ext. valid (sect. 4.3)
8	0	238	262 h	Execution time comparison (sect. 3.3)

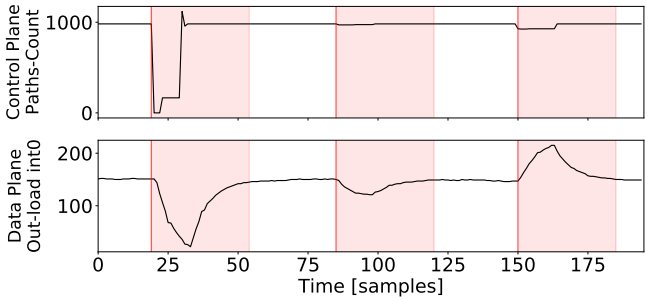

Figure 2: Example of control plane (top) vs data plane (bottom) features, and annotated ground truth

Telemetry features. The features (in machine-learning terms) available in the testbed are a subset of the state available by YANG[11] models of the Cisco routers in the testbed. In a nutshell, YANG modules define a hierarchy (i.e., tree) of data that can be used for configuration, state sharing and notifications; in the model, each node has a name, and either a *value* or a set of child nodes. All the leafs of the tree are features that could be exported in our testbed.

At the same time, it is worth stressing that, e.g., routers in the testbed are equipped with Cisco IOS-XR 6.2.2, whose YANG hierarchy comprises over 378,000 lines, describing a hierarchy of over 45,000 features, with *nearly 5,000 types pertaining to the BGP protocol alone*. Since collecting and exporting features consumes CPU and bandwidth resources,

it would be impossible to collect, for all nodes and interfaces, the totality of the supported features.

We thus perform a preliminary feature selection step, using domain-expertise to configure the most relevant features in the collection process. In particular, the selected features can be ascribed to either *data plane*[1] vs *control plane*[2] categories. The full list of about 750 data-plane[3] and 25 control-plane[4] features cannot be reported here for lack of space, but is available at [5] for the interested reader. An example of data vs control plane features is depicted in Fig. 2 showing an example of a control- (`paths-count`) vs a data-plane (output load on interface 0) features during our testbed, depicting also the annotated ground truth.

We point out that, whereas experimental results in Sec.4 show very good accuracy and recall, confirming the experts' choices, we recognize the need for a more systematic feature selection (part of our ongoing work).

3 METHODOLOGY

Two families of approaches would fit our experimental datasets. Indeed, the ground truth at our disposal would allow to build very detailed *supervised* multi-class models, precisely detecting the anomaly type; at the same time, these models would probably have a narrow application, as they would hardly be portable to different topologies, traffic matrices and load levels. Therefore, we prefer to adopt an *unsupervised* machine learning approach, to gather more coarse anomaly indication, that however incur less the risk of overfitting the model to a particular dataset.

Under these premises, algorithms suitable for our problem include classic approaches such as K-Means [16] or DBScan [14] and stream-based approaches such as DenStream [12]. Briefly, K-Means [16] is a simple clustering algorithms, based on centroids and distances: the points are assigned to the nearest centroid, then the centroids are updated until all samples remain in their cluster. DBScan [14] instead computes the distances between points, and clusters altogether the points which are neighbors (i.e. whose distance is less than ϵ): by computing the ϵ neighborhood of each point, it is possible to discover clusters of arbitrary shape. Since K-Means and DBScan are fairly well known, we only provide here a terse description of DenStream, and compare the three approaches experimentally in Sec.3.3

[1]Cisco-IOS-XR `infra-statsd-oper` and `fib-common-oper` models

[2]Cisco-IOS-XR `ip-rib-ipv4-oper` and `ipv4-bgp-oper` models

[3]We select 14 (11) features of the `interface/generic-counters` (`interface/data-rate`) group per-interface, and 20 features of the `node/drops` group per-node

[4]We select 11 (14) features of the `bgp/as/information` (`vrf/process-info`) group per-node

3.1 DenStream

DenStream [12] is an algorithm designed for clustering in data streams: DenStream extends the density-based strategy introduced in DBScan making it viable for online model construction. First of all, the algorithm uses a damped window model to weight the samples, so that older ones are less important than newer ones, via a fading function $f(t) = 2^{-\lambda t}$, where $\lambda > 0$ is the aging parameter. The main idea of the algorithm is the introduction of the so called *micro-cluster (MC)*, i.e., group of close points p_{ij} with creation time stamps T_{ij} defined as $MC = (w, c, r)$ where $w = \sum_j f(t - T_{ij})$ is the weight of the micro-cluster, $c = \frac{1}{w} \sum_j f(t - T_{ij}) p_{ij}$ is its center and $r = \frac{1}{w} \sum_j f(t - T_{ij}) d(c, p_{ij})$ its radius with $d(\cdot, \cdot)$ the Euclidean distance. By breaking clusters in MCs, DenStream allows to obtain clusters of arbitrary shapes.

The MC weight w plays a key role in the model construction, as it discriminates between *outlier* ($w < \beta\mu$) vs *core* ($w > \beta\mu$) micro-clusters (where β and μ are free parameters). When a new sample is available, DenStream merges it to the nearest *core* MC provided that the radius of the merged cluster does not exceed a given threshold ϵ; otherwise, DenStream attempts at merging the point to the closest *outlier* MC, and a new outlier MC is finally created if the merge fails. In our context, when a sample is merged to a *core* MC, we consider the sample as *normal* (and *anomalous* otherwise).

Not only MCs are easy to maintain *incrementally* at each new data point, but notice that model construction is a continuous process in DenStream: an outlier MC can indeed become a core MC when its weight increases as new points are added to it, and similarly a core MC becomes an outlier MC (and ultimately vanish) if no new data points are added for long periods – which makes it suitable for dynamic environments. Whereas DenStream introduces a number of parameters (namely, λ, ϵ, β and μ) we show in Sec.4.1 that their tuning is relatively straightforward and that DenStream performance are quite robust for a large range of settings.

3.2 Telemetry-based anomaly detector

We next define two simple criteria for raising alarms based on DenStream clustering, which we which we experimentally evaluate in Sec. 4.2 using the classic key performance indicators of information retrieval.

Temporal order κ_T. : *each node operates independently and an alarm is raised only upon reception of* κ_T *consecutive outlier samples at a node.* In particular, a true positive detection (false alarm) occurs when κ_T consecutive samples are labeled as *anomalous* and the ground truth labels the system as being in anomalous (normal) state. The parameter κ_T tradeoffs precision for recall and delay: intuitively, increasing κ_T reduces the amount of false alarms, but at the same time reduces the

overall amount of raised alarms, and mechanically inflates the delay by a factor of κ_T.

Spatial order κ_S. : *all nodes operate altogether and an alarm is raised when, during the same time slot, at least* κ_S *nodes label a sample as an outlier.* As before, the ground truth labels assist in evaluating the accuracy of the method. As before, increasing κ_S tradeoffs high precision for lower recall (as anomalies that affect fewer than κ_S nodes in the same time-slot can go unnoticed), although the detection delay is minimized in this case.

3.3 Comparison at a glance

We report a brief experimental comparison of K-Means, DBScan and DenStream in Tab. 2. In particular, we employ the longest dataset #8 lasting for 262h hours, corresponding to 187,777 samples, and focus on the execution time of each algorithm. K-Means is used as a baseline: since we don't know a priori the number of clusters K, for the sake of comparison we feed K-Means using the number of clusters returned by DBScan or DenStream respectively (execution time is exponential in the worst case, but typically much lower). DBScan solves the main issues of K-Means, as it automatically finds the number of cluster and supports clusters of arbitrary shapes, but is not directly suitable for an online application: we thus apply DBScan either on an ∞-horizon window, or on a windowed approach where we breakdown the analysis by batches of 60 samples (i.e., 5-minutes horizon). Finally, we apply DenStream in incremental fashion to the whole dataset (with settings that we defer to Sec.4.1 but that however only minimally alter its execution time). Tab. 2 clearly shows that DenStream is significantly faster than K-Means and DBScan, processing the whole dataset in 23s, whereas DBscan requires 87s – 120s (depending on the horizon) and K-Means about 64s – 276s. This computational advantage, coupled to the ability to continuously evolve the model on streaming data, confirms DenStream a good candidate for our problem.

Table 2: Algorithm comparison at a glance (dataset 8, 180k samples)

Algorithm	Complexity	Execution time	Parameters
K-Means	$O(knT)$	K_{DBScan}: 64s $K_{DenStream}$: 276s	K
DBScan	$O(n \log(n))$ $O(n_w \log(n_w) \cdot (n - n_w))$	∞-horizon: 87s 5min-horizon: 120s	ϵ, $MinPTS$
DenStream	$O\left(\frac{W}{\beta\mu} + n\right)$	23s	ϵ, λ, β, μ

4 EXPERIMENTAL RESULTS

4.1 DenStream sensitivity

We start by tuning the parameters of the DenStream algorithm, using datasets #1–#5 described in Tab. 1.

Radius threshold ϵ. DenStream performance depend on the choice of the radius parameter ϵ. We advocate for a $\epsilon_{dynamic}$ selection of the threshold, that is automatically computed as the radius of the cluster obtained clustering together the first $S = 40$ samples at the beginning of the model construction. We contrast this choice with an ϵ_{fixed} choice of the threshold, where a network expert suggests to compute it as the radius of the cluster obtained during an experiment in which no anomalies are injected (datasets #0–1) and tested on the remaining datasets (#2–5).

Experiments (not shown for lack of space) show that the use of a ϵ_{fixed} threshold reduces recall and precision by over a factor of 2×. The radius indeed, depends on the normal working condition of the network as traffic load, number of neighbors, on the topology, etc., which makes a fixed selection fragile and impractical. Dynamic threshold selection is easy to compute (as few samples suffices) and relevant (as it is, by construction, portable to any scenario).

Maximum weight μ^+. The weight parameter μ is used jointly with the potential factor β to decide when a given outlier MC becomes a new core MC (particularly, when $w > \mu\beta$). Given the exponential fading function $f(t) = 2^{-\lambda t}$, and considering a fixed-rate sampling as in our case, the maximum weight a micro-cluster can reach is $\mu^+ = \sum_j f(j) = \frac{1}{1-2^{-\lambda}}$ (since $|f(t)| < 1$ for $\lambda > 0$) which solely depend on λ. By setting $\mu = \mu^+$ we therefore reduce the parameter cardinality, and the rule for outlier MC promotion becomes $w > \beta/(1 - 2^{-\lambda})$.

Fading λ and Potential β factors..
The only free parameters in our setup are thus λ and β, which both have a physical interpretation. In particular, λ is a time-related parameter that tunes the timescales at which *old samples should be considered as totally independent from the current system state.* Once $lambda$ is fixed, the potential factor β has then a geometric interpretation, as *it determines the minimum number of samples needed for outlier MC promotion to core MC.*

We point out that a machine-learning expert would select λ and β as a result of a "grid selection" procedure, whereas a network domain expert could be tempted to select λ and β according to physical properties of the system. We adopt both viewpoints in what follows. For instance, a network domain expert could thus decide to set λ according to the expected convergence time of the BGP protocol (e.g.. imposing that a 5-minutes old sample has at most 1% of its initial contribution equals to selecting the largest $argmax_\lambda 2^{-\lambda \cdot 60} < 10^{-2} - \lambda \approx 0.111$ and requiring that an outlier MC should have at

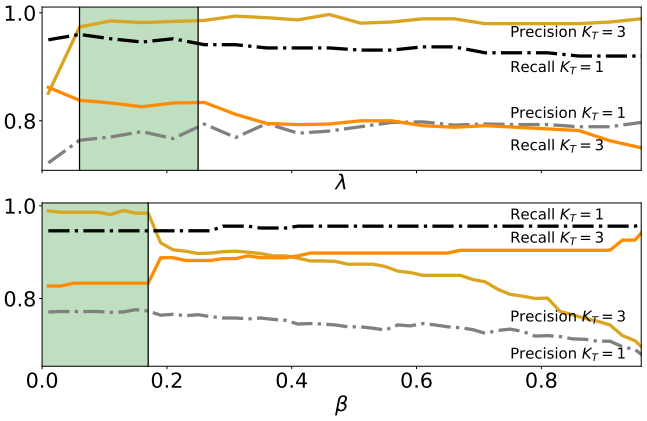

Figure 3: Sensitivity analysis of the fading λ and potential β factors.

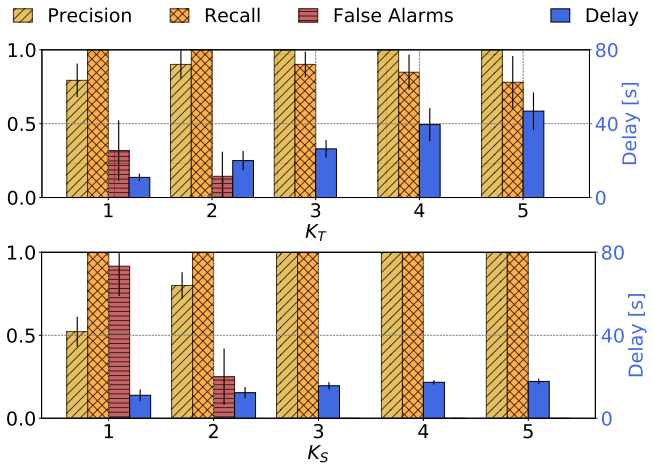

Figure 4: KPI for Temporal detection of order κ_T (top) and Spatial detection of order κ_S (bottom)

least 3 samples before becoming a core MC. We perform a sensitivity analysis and question these choices: Fig. 3 shows the precision and recall results varying $\lambda \in [0, 1]$ with fixed $\beta = 0.05$ (top) as well as for varying $\beta \in [0, 1]$ for fixed $\lambda = 0.15$ (bottom). Without loss of generality, we show results for two values of the temporal order $\kappa_T = \{1, 3\}$.

Several important takeaway can be gathered from the picture: first, performance are smoothly varying on λ and β, which can be chosen from large plateau. Second, performance are less impacted by λ and β than they are from the detection order κ_T. The fact that performance do not vary abruptly as a function of the inner DenStream parameters is a very desirable finding, since the detection order is a more intuitive knob to tune for the operator using the system (see Sec.4.2). Third, as far as the BGP timescale is concerned, the domain expert choice for λ falls in the acceptable range that the machine-learning expert would suggest (highlighted in green in the picture). Fourth, it is preferable to let DenStream produce small micro-cluster (low β).

A last remark is worth making. While we have seen that performance are smoothly varying on β and λ, we point out that their selection is still primarily correlated with the telemetry sampling rate: as such, in case of device reconfiguration and especially for very different sampling timescales, a sensitivity analysis would be recommended.

4.2 Temporal κ_T vs spatial κ_S orders

We now assess the portability of our settings, applying the parameters on a different dataset (#5). In particular, Fig. 4 illustrates the KPIs obtained for κ_T and κ_S. With the exception of the delay, precision, recall and false alarm exhibit a similar trend for both κ_T and κ_S. As expected, requiring multiple consecutive outliers from the same node ($\kappa_T > 1$) induces a sizeable delay, as $\kappa_T \approx 3$ samples are needed to avoid false alarms. Conversely, $\kappa_S \approx 3$ allow perfect recall and precision with no additional delay (while we do not observe false negative in our dataset, it is however possible for anomalies local to a single node to go unnoticed). As such, depending on the operator preferences on the above metrics, the choice will fall into $\kappa_{(S|T)} \in \{2, 3\}$.

4.3 Feature selection

Finally, we perform a "domain expert" feature selection by using control plane (CP) only, data plane (DP) only or all DP+CP features. Fig. 5 shows the KPIs considering $\kappa_{(S|T)} \in \{2, 3\}$, where we use shading and pattern to differenciate the features set. An interesting observation can be learned from the experiment: using only CP-related information, *no false alarms are raised already for* $\kappa_S = 2$ *and* $\kappa_T = 2$. Conversely, using DP-related features only generally yields to poor[5] performance, and DP+CP to intermediate performance.

As such, these experiments suggest the following. First, the algorithm is extremely reliable in detecting anomalies even with a few (20 out of nearly 5,000) features related to the protocol under investigation (CP-only case), testifying its precision. Second, the very same algorithm is still able to detect BGP anomalies even without having any information on the BGP protocol itself (DP-only case), testifying its robustness.

5 DISCUSSION

Anomaly detection is surely not a green field. Yet, the recent emergence of model-driven telemetry opens new challenges, and particularly makes the use of stream-based unsupervised

[5]Notice that DP features also bring valuable information beyond the noise: e.g., combining DP an CP features the recall increases for $\kappa_T = 3$.

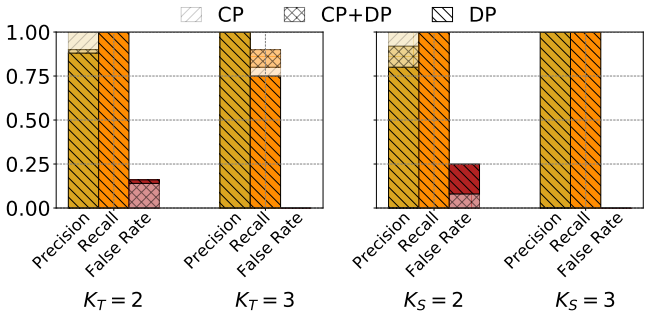

Figure 5: Domain expert feature selection: Data Plane (DP) only vs Control Plane (CP) only vs all DP+CP features

machine learning tools very appealing. In this paper: (i) we engineer a full blown testbed representative of a BGP-only datacenter network of a Content Service Provider, that we load with up to 1 Tbps aggregated traffic; (ii) we use the testbed to produce datasets with annotated ground truth, collecting several tens of thousands samples reporting hundreds of features from tens of devices, that we make available to the scientific community[5]; (iii) we develop, implement, open-source and thoroughly analyze an anomaly detection engine based on the DenStreaming[12] clustering algorithm, that we also compare with state of the art techniques such as DBScan.

Our results show that: (i) despite DenStreaming is apparently plagued with several parameters, their selection is quite straightforward, and performance are robust to inner parameter selection; (ii) low spatial and temporal orders are sufficient to jointly attain high recall, high precision and low delay; (iii) detection of BGP-related anomalies is more effectively based on control-plane only features, which is due to the implicit separation of timescale between data plane and control planes.

These results opens up future work, along the following directions: (i) given the low computational complexity of DenStreaming and the better performance, it would be advisable to run models in parallel, each of which would detect anomalies from different protocols/layers/planes; (ii) individual learners could be complemented with a macroscopic model, that would learn from alarms of individual learners, as opposite to as from the raw telemetry features; (iii) finally, a more systematic experimental study of the maximum numbers of features that can be exported at the same time is a clear necessity.

ACKNOWLEDGEMENTS

This work has been carried out at LINCS (http://www.lincs.fr) and benefited from support of NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech (https://newnet.telecom-paristech.fr).

REFERENCES

[1] 2018. (2018). https://www.cisco.com/c/en/us/solutions/service-provider/cloud-scale-networking-solutions/model-driven-telemetry.html
[2] 2018. (2018). https://www.arista.com/en/solutions/telemetry-analytics
[3] 2018. (2018). https://www.juniper.net/documentation/en_US/junos/topics/concept/junos-telemetry-interface-oveview.html
[4] 2018. (2018). http://support.huawei.com/enterprise/en/doc/EDOC1000173015?section=j006
[5] 2018. (2018). https://github.com/cisco-ie/telemetry
[6] 2018. https://github.com/anrputina/OutlierDenStream. (2018).
[7] 2018. https://github.com/YangModels/yang. (2018).
[8] 2018. (2018). https://blogs.cisco.com/sp/introducing-pipeline-a-model-driven-telemetry-collection-service
[9] B. Al-Musawi, P. Branch, and G. Armitage. 2017. BGP Anomaly Detection Techniques: A Survey. *IEEE Communications Surveys Tutorials* 19, 1 (Firstquarter 2017), 377–396. https://doi.org/10.1109/COMST.2016.2622240
[10] M. Bjorklund. 2010. YANG - A data modeling language for NETCONF. *RFC 6020* (Oct. 2010).
[11] M. Bjorklund. 2016. The YANG 1.1 Data Modeling Language. *RFC 7950* (Aug. 2016).
[12] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*.
[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.
[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *ACM KDD*.
[15] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (Sept 2014), 2250–2267. https://doi.org/10.1109/TKDE.2013.184
[16] J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. Berkeley Symposium on Mathematical Statistics and Probability*.
[17] Zachary Miller, William Deitrick, and Wei Hu. 2011. Anomalous Network Packet Detection Using Data Stream Mining. *J. Information Security* 2, 4 (2011), 158–168.
[18] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. 2014. Twitter spammer detection using data stream clustering. *Information Sciences* 260 (2014), 64 – 73. https://doi.org/10.1016/j.ins.2013.11.016
[19] P. Lapukhov, A. Premji, J. Mitchell. 2016. Use of BGP for Routing in Large-Scale Data Centers. (Aug. 2016).
[20] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. 2013. Data Stream Clustering: A Survey. *ACM Comput. Surv.* 46, 1, Article 13 (July 2013), 31 pages. https://doi.org/10.1145/2522968.2522981
[21] Marina Thottan and Chuanyi Ji. 2003. Anomaly detection in IP networks. *IEEE Transactions on signal processing* 51, 8 (2003), 2191–2204.
[22] Q. Wu, J. Strassner, A. Farrel, and L. Zhang. 2016. Network Telemetry and Big Data Analysis. *IETF draft-wu-t2trg-network-telemetry-00* (Mar 2016).