Automatic Detection of Network Traffic Anomalies and Changes

Astha Syal
Youngstown State University
Youngstown, OH
asyal@student.ysu.edu

Alina Lazar
Youngstown State University
Youngstown, OH
alazar@ysu.edu

Jinoh Kim
Texas A&M University-Commerce
Commerce, TX
kim@tamuc.edu

Alex Sim
Lawrence Berkeley National Laboratory
asim@lbl.gov

Kesheng Wu
Lawrence Berkeley National Laboratory
kwu@lbl.gov

ABSTRACT

Accurately predicting network behavior is beneficial for TCP congestion control, and can help improve routing, allocating network resources, and optimizing network designs.This task is challenging because many factors could affect network traffic, such as the number of network sessions and synthetic reordering. There are also many ways to measure the network state, such as the number of retransmissions per flow and packet duplication. For this work, we use a set of passive TCP flow measurements collected at a major computer center on multiple data transfer nodes (DTN).

To assist the operations of the computer network, we propose to detect abnormally slow network transfers in real-time. The proposed system breaks the network monitoring logs into fixed-size chunks and employs a state of art classifier to identify the slow time windows. This method will be validated on real large datasets collected from several DTNs. The proposed method is able to generate models to quickly detect large intervals of low performing network transfers, which require attention from network engineers.

KEYWORDS

Network traffic; TCP performance; UMAP; classification; `Tstat`

ACM Reference Format:
Astha Syal, Alina Lazar, Jinoh Kim, Alex Sim, and Kesheng Wu. 2019. Automatic Detection of Network Traffic Anomalies and Changes. In *Systems and Network Telemetry and Analytics (SNTA '19), June 25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3322798.3329255

1 INTRODUCTION

As computer networks grow in size, complexity and available services, network engineers have to spend significant time and effort to tune and optimize the network performance. Network traffic monitoring can provide important insights about current state of the network. Methodologies and tools have been developed to help network engineers with their routines of traffic monitoring and problem detection. Data collection is the first step to diagnose the network performance to understand the implications and interconnection of users' behavior.

Large scientific facilities use Science DMZ, which includes several dedicated data transfer nodes and high performance data movement tools, to attain high network transfers for high performance scientific applications. Network traffic analysis [14] and prediction play a vital role in maintaining healthy operations within all varieties of complex and diverse computer networks. Online traffic monitoring information, collected over time, can be used to predict future traffic volume and unexpected events in real-time.

Reliable file transfers are essential for successful science computations and experiments that need large complex data files to be moved across long distances. Networks use protocols such as TCP and UDP to support file transfers, and their performance degrades under packet losses and duplications, thereby adversely affecting science applications. Hence, scientists and engineers often monitor network statistics to find and repair network problems that cause such degradation.

Machine learning has been employed in many network applications, such as traffic prediction, traffic classification, intrusion detection, network management, network adaptation, performance prediction, and configuration extrapolation. Specifically, machine learning methods have been proposed to identify bottlenecks and explain the status of network traffic using features from passive network measurements. However, analyzing the network traffic data using machine learning and statistical methods [3, 20] is challenging for several reasons. The network data is usually automatically collected as a high volume heavy stream of high dimensionality that needs to be analyzed in real-time to provide alerts in case of unexpected events.

Another important challenge is the lack of labeled ground truth for evaluating the machine learning methods [1]. Even if network data can be relatively easy to collect, there is no automatic way to acquire labels based of the problem to solve. High-quality ground truth datasets often need to be manually created, a process that is time consuming, requires solid domain knowledge, and may have serious privacy issues, especially in the case of analyzing real traffic traces.

Machine learning models are build on existing data with the hope that their high generalizability makes them adaptable to the high-variance future network traffic values. Given the often changes of these dynamic systems, it would be unacceptable to require a model

SNTA '19, June 25, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6761-5/19/06…$15.00
https://doi.org/10.1145/3322798.3329255

retraining for every time interval manifesting significant network traffic changes.

To build well-performing models, for most machine learning algorithms it is required that the training data follows the same distribution as the target distribution. This assumption is not always true or practical in dynamically changing networking environments, where the features are highly dependent on timing, physical equipment and transfers interactions. Although previous research [21] showed that the machine learning models built using training data from a specific network environment can, to some degree, perform well in other network environments, more research in transfer learning is necessary. The analysis and methods presented in this paper provide a first step towards identifying important features to detect the network status.

Even if state of the art machine learning algorithm can achieve good performance when trained on networking data, many of the resulting models are still black boxes, not humanly readable or easily interpretable. The accountability and interpretability properties create big obstacles in practical implementations of these models. If expert network engineers do not understand how the models behave, they cannot integrate this knowledge in the real systems to provide new ways for network adaptation and configuration extrapolation.

One performance characteristic of a network can be measures using the average network throughout. High throughput means that the network is running smoothly, while a state of stable low throughput is usually a sign of network congestion known as "congestive collapse". Network congestion takes place when a network link is transferring more data packets than it can handle and usually exhibits the reduced quality of service. This network state is usually due to interference among the server's shared physical resources involved in these transfers, such as network links, disk storage systems, and CPUs. Typical effects include packet loss, duplication, and retransmission. A typical consequence of congestion is imminent decrease in network throughput. Identifying factors [15] that contribute to the decrease of network throughput is very important in determining resource allocations to use in scheduling requests.

The main goal of this paper is to demonstrate that supervised machine learning approaches applied to passive measurements of network flow datasets can be potentially used to identify states of congestive collapse. Assigning labels based on the average throughput to categorize network traffic flows grouped by time intervals, can help with the analysis of these large network datasets. After the initial binary labeling of network transfer flows to a small predefined number of clusters, the features of each time window can be used to generate a classification model of the traffic. New incoming flows can then be classified on the fly and assigned to one of the two classes (low versus normal throughput time window).

Specific contributions of this paper are as follows:

- Statistical analysis methods to extract throughput threshold values to label the time intervals as 'slow' or 'normal'.
- Unsupervised dimensional reduction and visualization methods based on UMAP to provide a 2-dimensional representation of the datasets to understand the structure of the data.
- Classification experiments performed on real network data collected from eight DTNs using `Tstat`.
- Checking the results of this supervised learning approach, especially the false positives and false negatives, using throughput plots for evaluation and comparison.

This paper is organized as follows. Section 2 compares our approach with previous work, while section 3 is an overview of the proposed methodology and describes its main steps. Section 4 provides an overview of the datasets, while sections 5 thoroughly discusses the experiments performed on eight large real traffic datasets. Finally, section 6 draws conclusions and presents future developments for this work.

2 RELATED WORK

TCP logs collected by `Tstat` [17] together with other logs from DTNs at a scientific computing facility have been statistically analyzed before by Liu et. al [14]. They examined transfers performed during the year 2017 at three different levels: that of user transfer requests; that of individual file transfers; and that of TCP flows. Using these logs, insights on transfer, file, and flow characteristics, and identify areas for improvement in transfer performance and resource utilization were presented. This study shows that useful insights can be obtained by combined analysis of logs from different layers of the data movement stack and some of the findings on flow, file, and transfer characteristics are applicable to other large facilities.

Network data at the flow level collected with `Tstat` is unlabeled and therefore there is no good way for the network engineers to immediately differentiate between normal and anomalous network transfers. Using perfSONAR, Rao et al. [19] collected network data in a controlled environment to detect and identify network transfer anomalies such as packet loss, duplication, and retransmission sequencing that affect file transfer performance especially related to congestion. A combined approach of dimensionality reduction and statistical analysis was used to determine important features and highlight abnormal behaviors.

Simple experiments using unsupervised feature extraction show that the proposed method works well to extract certain characteristics from known network transfer datasets. These experiments used real network data extracted from `Tstat` logs and ignored specific file details such as file size, workflow stage and link details, to allow core network properties to be extracted as general normal features. The patterns extracted using dimensionality reduction based on Principle Component Analysis (PCA) and clustering can help build feature filters to select data for any future machine learning methods. In the end this method will allow researchers and network engineers to build relationships among sensitive parameters such as congestion and availability with transfer file type. The main goal was to understand how packet loss and congestion impact the end-to-end performance.

Instead of analyzing `Tstat` features, Dao [5] only looks at the throughput characteristic of the network transfers. This approach employs a change point detection method that works by first dividing the network flows into time windows based on their time stamp and second by applying a non-parametric model for each window to describe the congestion. The idea is to designate network transfers that take significantly longer than typical expected

time as "slow" or "abnormal transfers". When many "slow" network transfers occur in the same time window, it is worthwhile to alert the network engineers and prompt them to investigate the abnormal behavior of the system.

Another unsupervised approach, based on clustering and proposed by Kim [9–12], aims to keep track of the network state based on the aggregation of multidimensional variables. The clustered result represents the state of the network with regard to the monitored variables, which can also be compared with the observed patterns from previous time windows enabling intuitive analysis. The definition of the network state depends on what type of data is being analyzed to construct clustered patterns. They proved the proposed method with two popular use cases, one for estimating traffic breakdown and the other for identifying anomalous states. The authors also applied their method to the `Tstat` data collected from ESnet to show the applicability of their method.

It is well-known that TCP anomalies such as packet loss contributes to the variance of network throughout [15]. Therefore, it is essential to be able to correctly identify all these the anomalies. Previous research [2, 11] reported statistical correlation between multiple variables collected in the `Tstat` logs and the network traffic throughput. Recently, Hidden Markov Model and Recurrent Neural Networks have been proposed [4] to predict network traffic volume from some flow statistics, such as flow counts per time interval. These flow statistics are assumed to be easier to compute compared to network throughput.

Another unsupervised/supervised technique proposed in [13], accurately identify TCP anomalies occurring during file transfers based on passive measurements of TCP traffic collected using Tstat. This method was validated on real large datasets collected from several DTNs. The preliminary results indicate that the percentage of TCP anomalies correlate well with the average throughput in any given time window.

The big data framework SeLINA, proposed by Apiletti et al. [2] was design to extract information from raw network flows data and to meet current network data analysis requirements including scalability, auto configuration capability, human readability of results, as well as evaluation of the model quality over time. The first step involves clustering the data using an automated tuning algorithm based on DBSCAN. Second, the clustering labels are used as input for a decision tree algorithm that has the capability to rank the features.

For continued evaluation and comparison, SeLINA uses the average Silhouette index as the quality measurement and also a check of the degree of change that takes place in the network. When this index changes drastically, the main DBSCAN clustering model is automatically rebuilt to incorporate the new incoming traffic data. The experiments conducted highlight the system's ability to identify over-time changes in the network. One of the main contributions of this paper is the self-tuning property for the main algorithms, a step that normally requires sophisticated, expert level fine-tuning.

To prevent network congestion and overutilization of network devices, it is important to detect data transfer anomalies over the network. Flowzilla [8] is a methodology designed to achieve the same. It uses training data from `Tstat` to build the Model, which is adaptive to the changing network load. The training data is generated from the Feature Extraction Filter, which uses Random Forest Regression (RFR) to extract a subset of features from `Tstat` database. It then uses Adaptive Threshold Mechanism for detection of anomalies, which includes a threshold calculator, that calculates the detection threshold values based on previous data and the Detector, that calculates the anomalous flows based on difference in model's predicted flow size and threshold value. 92.5% accuracy is achieved using this framework.

Our approach is similar to the Flowzilla approach [8], with several differences. Here we are classifying the network transfers based on the throughput and not based on their size. The classification is performed on time intervals instead of individual transfers. Finally, most significant `Tstat` are used in our approach compared to the limited set of features used by Flowzilla.

3 METHODS

Traffic flows collected in the `Tstat` logs have no feature or variable to designate them as anomalies. However, for this paper we consider labeling the network transfers using the average throughput per time window and an adaptive threshold set as the first quartile of the dataset. Then, supervised machine learning algorithms are used to predict which flows are slow and which are normal. Specifically, a supervised approach based on the linear version [6] of the popular Support Vector Machine (SVM) approach was found suitable to build models that automatically classifies the traffic flow time windows into two separate groups with similar characteristics in terms of their throughput. All the steps of this approach are highlighted in Fig. 2.

The proposed method employs a linear SVM algorithm to identify time intervals with low averages of the throughput values over the entire dataset. SVMs are supervised learning models that usually provide high performance for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two classes, an SVM training algorithm builds a linear hyperplane with the specific property of having the largest margin. In other words, the individual data points are mapped so that the points of the two categories are divided by the optimal hyperplane that has the largest gap. The optimization step of linear SVMs can be solved efficiently using the coordinate descent algorithm, thereby reducing the convergence iteration to linear time, making this algorithm run very fast compared to other classification methods. The linear SVMs method was selected because of the low computational time required for training and high accuracy rates obtained on the testing sets.

Uniform Manifold Approximation and Projection (UMAP) [16] is a new dimension reduction technique that can be used for visualizations similar to other manifold data embedding techniques, and also for general non-linear dimension reduction. It is based on manifold theory and fuzzy topological data analysis. The algorithm builds a weighted k-neighbor graph to efficiently approximate the k-nearest-neighbor computation and calculates spectral embeddings that are later optimized using the stochastic gradient descent algorithm. The algorithm is founded on assumptions that the data is uniformly distributed on a Riemannian manifold, an assumption that does not always hold for real data.

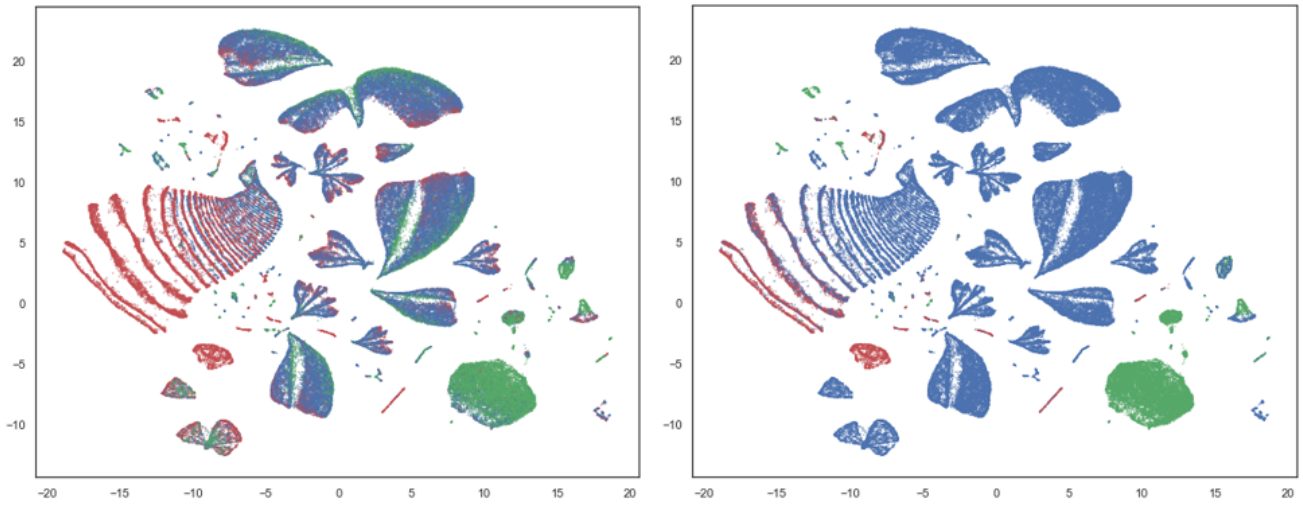

Figure 1: UMAP 2-dimensional visual representation of the network traffic flows collected from node 5 and colored based on their throughput values. Red means low, blue means normal and green means high. (a) individual flows (b) majority labels assigned based on one hour time intervals.

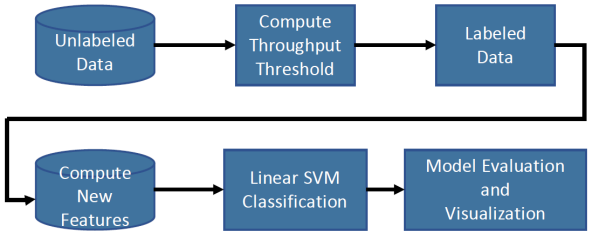

Figure 2: Proposed machine learning guided methodology for identifying and categorizing low performing network flows.

To understand the underlying structure of this particular dataset, a two-dimensional representation in an embedding space for the node 5 dataset is presented in Figure 1. Similar to principal component analysis (PCA) representations, the values of the x-axis and y-axis of the UMAP scatterplot are nothing more than a representation in the embedding two-dimensional space. Part (a) of this figure shows all the individual flow with their calculated throughput. Red represents flows with low throughput (throughput lower than the first quartile), blue means normal (throughput between the first quartile and the third quartile), while green shows the high performing flows (throughput larger than the third quartile). Figure 1 (b) shows the same network transfers, colored using the same encoding; however this time the flows are assigned with the majority class of the one-hour time windows in which they belong.

Given the time series property of the network transfers datasets, it does not make sense to split them randomly into training and testing or perform standard cross-validation shuffling, but instead a "time series cross-validator" is more appropriate. The procedure used to split time series is described next. One moment in time can to be chosen as the delimiter between the training and testing sets. Also, to perform cross-validation at each split, test indices must be higher than the indices used for testing before, and the entire training set needs to have timestamps from before the test set. Unlike standard cross-validation methods, successive training sets are supersets of those that come before them, but they can also be limited to a certain size. Cross-validation is the preferred validation method for larger datasets because it better estimates the generalization ability of the model, which is very important for the problem we are trying to solve.

Quantitative classification evaluation, which evaluates the goodness of classification results, can be done using the traditional measures such as accuracy, precision, recall, and F1 score. Precision is the ratio between the correctly classified positive instance and number of all positive instances. It gives an idea of the amount of elements from the positive class that were misclassified. Recall is the ratio between the correctly classified positive instance and the number of all instances classified as positive. The F1 score is an average between precision and recall. Precision, recall, and the F1 score are very important, especially when the training and/or testing datasets are highly imbalanced.

By showing the correctly classified time intervals versus the incorrectly classified on the time series average throughput plots, a qualitative classification evaluation is possible.

4 DATASETS

Today, network traffic statistics at the flow level can be collected using passive monitoring tools such as `Tstat` [7]. A passive probe located on the access link that connects each Data Transfer Node (DTN) located at the National Energy Research Scientific Computing Center (NERSC) to the ESnet (Department of Energy's dedicated

science network) inspects all packets flowing on the link and extracts the information to be summarized. The Tstat software rebuilds each TCP and UDP network flow by matching incoming and outgoing segments.

Table 1: Datasets Statisctics

Node	# of Flows	60min	30min	5min
1	2,447,602	4,232	8,450	47,310
2	1,119,470	2,639	4,372	13,618
3	5,131,592	4,029	7,845	36,089
4	455,244	3,286	5,716	21,006
5	135,531	421	659	2,140
6	166,116	598	1,060	4,359
7	157,247	412	659	2,439
8	169,233	626	1,096	4,539

Tstat offers output statistics at packet and flow level. The flow-level analysis provides a summary of the connection properties that is logged [18] for further analysis. It can be used to collect many different statistics for TCP, UDP, and RTP/RTCP traffic. For TCP connections, congestion window size, out-of-sequence segments, duplicated segments, number of bytes and segments retransmitted, and RTT are some of the statistics that it collects. Tstat distinguishes between completed and not completed flows, and between clients (hosts that actively open a connection) and servers (hosts that passively listen for connection requests). Tstat also records UDP messages. However, since UDP communication contributes a very low percent of the total bytes moved from/to the major computer center, we did not included UDP communications in this study.

Among the measurements collected by Tstat, some of the metrics are believed to be correlated to both system configuration and possible performance issues. For example, the measure of Round Trip Time (RTT) is usually related to both the distance from the server, but also possible to reveal congestion on the path. Similarly, both reordering and duplicate probabilities increase during periods of congestion. The duration and amount of carried data are used to compute the actual throughput and could also distinguish between the type of service the flow carries, e.g., short-lived signaling flows carrying little data rather than long lived data flows carrying a large amount of data. We included all these Tstat measurements in our experiments.

At the large scientific facility 90K of TCP flows are collected per node daily and an approximate total of 10GB of compressed data logs are collected yearly on ten DTNs. The Tstat data used for this study was collected and provided by the NERSC computing facility at LBNL. The Tstat data contains source and destination IP addresses, and so is not publicly available for privacy reasons. To simplify our analysis, for this study we eliminated all flows that carry less than 10MB of data both ways. Table 1 shows the number of network transfer in each dataset and also the number of time intervals. The datasets for nodes 1 to 4 contain six months worth of transfers collected between 01/01/2017 - 06/28/2017, while the datasets for nodes 5 to 8 are smaller and have approximately one month worth of data collected between 06/01/2017 - 06/28/2017.

For each dataset all the features with constant values are eliminated. We also eliminate features involved in the calculation of the throughput because it is used to assign the output labels. Table 2 shows the summary statistics for the calculated throughput for all eight nodes. The first quartile values in this table are used as threshold values for our experiments. Network transfers that take place closer in time are highly correlated compared with transfers that are far apart. To account for this important characteristic of the datasets we add two additional features which where assigned based on the previous and two time intervals preceding the current time window.

All the features are then normalized using the MinMax Scaling procedure. The datasets are divided into time intervals based on three time frequencies: 5, 30 and 60 minutes. Averages for all the features including throughput are calculated and saved for further input into the classification algorithm.

5 EXPERIMENTS AND RESULTS

To detect time intervals of slow network transfer we adopt a supervised classification method based on linear SVMs. Let $X = \{(x_1, y_1), ...(x_N, y_N)\}$ be the labeled anomaly detection dataset with N total instances, where x_i represents the input feature vector that can be defined in a d dimensional space as $x_i = \{x_i^1, x_i^2, ...x_i^d\}$. This set of feature values (client/server IP Address, client/server protocol, RTT values, maximum segment size) is extracted from the raw Tstat data. The corresponding binary class label $y_i \in \{y_1, y_2, ...y_N\}$ for each input vector x_i represent normal speed time intervals of abnormally slow intervals. This class label is assigned based on the adaptive threshold defined using the first quartile of the average throughput of the training dataset. Therefore, the training dataset will always contain 25% of slow time intervals transfers. In the end, the classification models are designed to predict whether the average throughput for the network transfers flows in a given time window that is below the throughput threshold for that node.

In the detection or testing phase, based on a classifier trained with set X, every instance in a test set is assigned to the class of either normal or slow type of transfer. It is important to note that because of the dynamic nature of the network, the network traffic data would change with time; thus, the adaptive threshold needs to be periodically updated and the classification model for anomaly detection must be learned with new training data, for the purpose of keeping high accuracy for online detection.

We build several models, a different one for each combination of node dataset and time window. The experiments are done for 5, 30 and 60 minutes time intervals. For the first set of experiments, we used the entire datasets for all the eight nodes and divided each of them based on the time stamp in training and testing sets. The last week is used for testing and the rest to build the model. Results of these experiments are presented in Table 3 in terms of accuracy, precision, recall and F1 score and percent of time windows under the threshold.

Overall, the results presented in Table 3 show accuracy greater than 83%, precision more than 68% for all the datasets, while recall is less than 50% in only two cases for node 8. Best results are observed for node 7 and node 6 and are shown in Figure 3 and Figure 4. These two figures clearly show that our classification models have no

Table 2: Summary Statistics

	node1	node2	node3	node4	node5	node6	node7	node8
count	2,447,602	1,119,470	5,131,592	455,244	135,531	166,116	157,247	169,233
mean	129.46	98.836	881.61	174.46	105.23	262.35	91.439	268.05
std	317.78	115.29	978.5	375.75	69.401	335.53	81.206	631.1
min	0.00003	0.00009	0.00005	0.00007	0.0008	0.0007	0.005779	0.008
5%	13.941	20.175	43.448	7.66	22.825	22.329	21.242	26.714
25%	31.636	56.643	174.85	42.672	57.376	64.642	47.877	60.551
50%	57.901	82.353	377.81	87.981	92.339	107.12	77.219	93.842
75%	111.15	103.64	1521.1	162.7	135.83	198.24	117.02	149.5
max	9853.994	9883.041	9889.773	9725.279	3003.8	5920	3048	3271.2

Table 3: Classification Evaluation

	min	Acc.	Prec.	Recall	F1	%CI
node1	5Min	0.9013	0.8632	0.7597	0.8081	27.34
	30Min	0.8574	0.6878	0.5842	0.6318	20.94
	1H	0.8664	0.6875	0.5878	0.6337	19.66
node2	5Min	0.9312	0.9401	0.7853	0.8558	25.97
	30Min	0.9055	0.9189	0.7133	0.8031	27.03
	1H	0.8985	0.9008	0.6987	0.7870	26.85
node3	5Min	0.9181	0.8765	0.8090	0.8414	26.86
	30Min	0.9288	0.8261	0.8132	0.8196	19.87
	1H	0.9024	0.8480	0.6974	0.7653	22.82
node4	5Min	0.9396	0.9058	0.6468	0.7547	14.37
	30Min	0.9121	0.8917	0.5350	0.6687	16.58
	1H	0.9221	0.9107	0.5312	0.6711	14.95
node5	5Min	0.9413	0.9758	0.9213	0.9478	57.77
	30Min	0.8349	0.9891	0.7222	0.8349	57.79
	1H	0.8862	0.8732	0.9254	0.8986	54.47
node6	5Min	0.9311	0.8274	0.7596	0.7920	17.26
	30Min	0.9279	0.8281	0.8281	0.8281	20.98
	1H	0.9176	0.8621	0.7143	0.7813	20.58
node7	5Min	0.9703	0.9782	0.9515	0.9647	42.63
	30Min	0.9167	0.9545	0.8660	0.9081	47.54
	1H	0.8780	0.9216	0.8103	0.8624	47.15
node8	5Min	0.8824	0.8878	0.4307	0.5800	18.86
	30Min	0.8562	0.7742	0.4000	0.5275	20.06
	1H	0.8916	0.8000	0.5333	0.6400	18.07

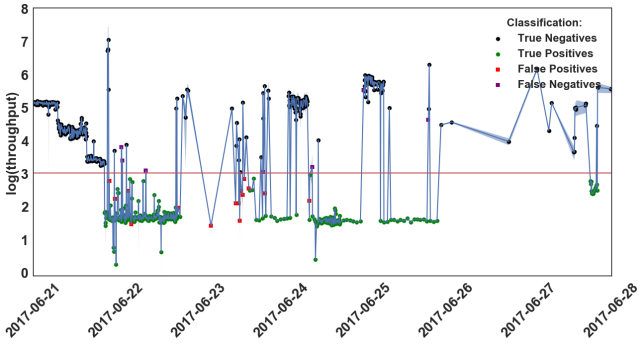

Figure 3: Average throughput for node 7, 5min time windows. The red line is throughput threshold. Recall is 95%

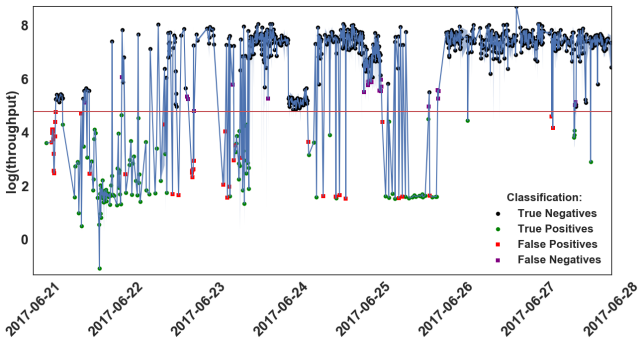

Figure 4: Average throughput for node 6, 5min time windows. The red line is throughput threshold. Recall is 75%

problem correctly identifying contiguous intervals of low network performance. The number of false negatives is insignificant, a fact denoted by the high rates of the precision measure. The number of false positives is higher, as reflected by the recall rate. This is observed especially during times when time intervals with high throughput alternate with time intervals of low throughput.

Finally, we take a look at the results for node 8 (Figure 5), where recall is the lowest. The test set of this dataset, has a very low ratio (between 18% and 20%) of transfers with low throughput, fact that makes this test set harder to classify. In this case there are four false negatives instances, two of them very close to the throughput threshold. A third of the false positives are very close to the average throughput and the others alternate with time intervals with high throughput.

The second set of experiments are done for node 1 through 4, where the datasets contain a larger number of network flows collected over 6 months. For these datasets we run cross validation using a Time Series Split with k=6 sets. The results are presented in Table 4. Accuracy is greater than 86%, precision does not fall less than 79% and the lowest recall is only 75%. Overall, the best results are obtained for node 3 and time interval of 30 minutes (Figure 6),

Table 4: Cross Validation Evaluation

		Accuracy	Precision	Recall	F1 Score
node1	5Min	0.8920 ±0.0076	0.8587 ±0.0109	0.8132 ±0.0373	0.8272 ±0.0301
	30Min	0.8884 ±0.0127	0.8381 ±0.0339	0.7924 ±0.0519	0.8045 ±0.0501
	1H	0.8753 ±0.0118	0.8181 ±0.0427	0.7638 ±0.0504	0.7794 ±0.0502
node2	5Min	0.9157 ±0.0105	0.8666 ±0.0346	0.8237 ±0.0363	0.8369 ±0.0332
	30Min	0.8758 ±0.0195	0.8175 ±0.0359	0.8099 ±0.0270	0.8068 ±0.0327
	1H	0.8643 ±0.0132	0.7972 ±0.0246	0.8052 ±0.0234	0.7987 ±0.0229
node3	5Min	0.9128 ±0.0144	0.8702 ±0.0260	0.8244 ±0.0413	0.8375 ±0.0324
	30Min	0.9176 ±0.0150	0.8858 ±0.0146	0.8702 ±0.0225	0.8727 ±0.0148
	1H	0.8980 ±0.0139	0.8638 ±0.0099	0.8349 ±0.0444	0.8341 ±0.0348
node4	5Min	0.8785 ±0.0259	0.8474 ±0.0270	0.7887 ±0.0206	0.8106 ±0.0216
	30Min	0.8676 ±0.0245	0.8354 ±0.0235	0.7716 ±0.0244	0.7899 ±0.0231
	1H	0.8614 ±0.0257	0.8221 ±0.0282	0.7586 ±0.0286	0.7719 ±0.0274

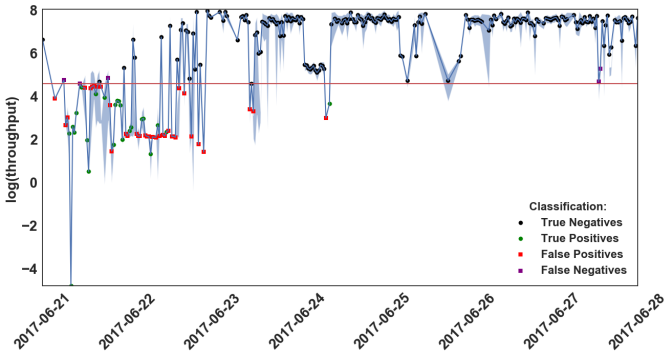

Figure 5: Average throughput for node 8, 30min time windows. The red line is throughput threshold. Recall is 40%

and the worst are for node 4 and 30 minutes time intervals (Figure 7). Node 3 has more 30-minute time intervals than node 4, however the testing results for node 4 applied on the last month's dataset shows a much lower recall. Node 3 has 44 false negatives and 48 false positives (Figure 6) where node 4 has 13 false negatives and 93 false positives (Figure 7).

As shown in Table 4 the number of time intervals in our experiments vary from 412 to 47,310. It is well known [6] that linear SVMs are capable of training datasets with over 500,000 instances in seconds. It takes less than 4 seconds to train and .02 to test the dataset for node 1 with 5 minutes time intervals which is the largest dataset used in our experiments. Data preprocessing and training can be done offline and so it doesn't slow down the detection. Testing works fast, making this method suitable for quick detection.

6 CONCLUSION

Reliable network transfers are essential for successful operations at large scientific facilities where petabytes are transferred daily. To identify possible problems such as low throughput, we propose to classify the traffic flows captured by `Tstat` with linear SVM classification algorithms. Our system splits the `Tstat` log streams into chunks so as to make predictions in near real-time. The classification model only needs to be updated and not rebuilt from the ground up. Tests show that this new method is able to accurately detect abnormally low throughput time intervals.

This paper presents a supervised data analytics system that effectively mines network traffic data. The proposed methodology is based on a two-phase approach that 1) assigns binary classification labels to network transfers using an adaptive threshold based on the throughput mean; and 2) builds a classification model to predict new data labels in real-time to identify traffic with low throughput.

This methodology features a linear SVM classification algorithm that can easily handle one year's worth of network traffic data. It is a general purpose approach, which can be easily exploited to analyze network traffic data under different network conditions. The approach has been tested using datasets from eight out of ten DTNs at the major computer center.

The tests on the proposed method showed its ability to accurately identify large windows of low throughput, but also showed problems in the case of isolated or alternating intervals. To address this problem, we plan to extend the current system with (i) the inclusion of more time related features, (ii) the evaluation of preprocessing feature selection techniques for eliminating more of the correlated features, and (iii) the design and integration of different analysis techniques, more appropriate for outlier detection.

In future, we plan to find better ways to label the data or the 'slow' time intervals and also to investigate the generalization capabilities of the presented method; larger datasets will be used for training and testing. To compare the validation results presented in this paper with results based on other existing classification approaches, we will consider methods such as random forest, SVM, Vowpal_Wabbit and convolutional neural networks (CNN).

ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and by Office of Workforce Development for Teachers and Scientists (WDTS) under the Visiting Faculty Program (VFP), Office of Science, the U.S. Department of Energy. This research used resources of the National Energy Research Scientific Computing Center.

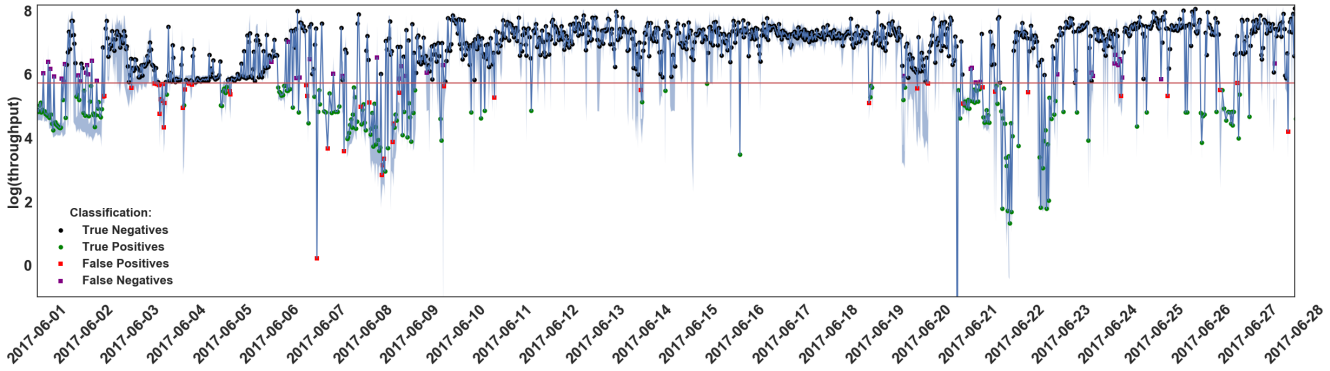

Figure 6: Average throughput for node 3, 30min time windows. The red line is throughput threshold. Recall is 81%

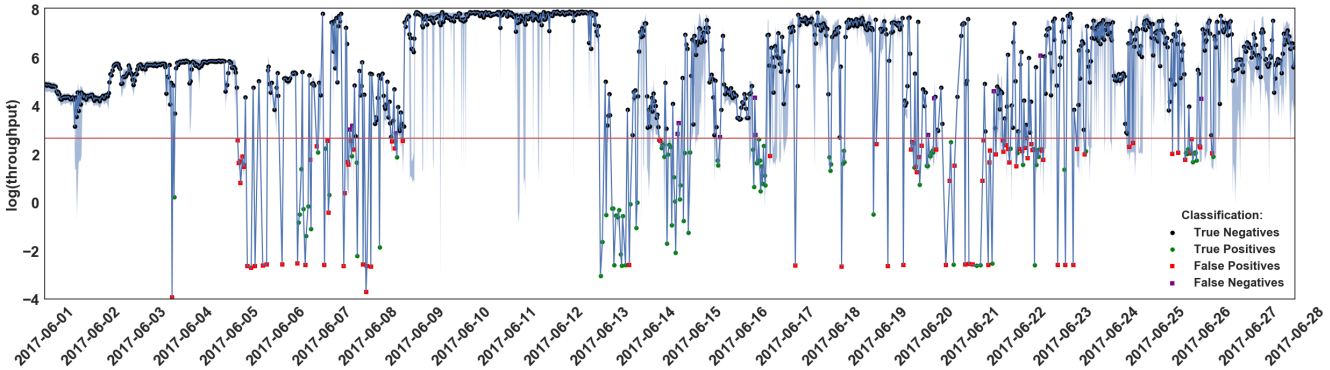

Figure 7: Average throughput node 4, 30min time windows. The red line is throughput threshold. Recall is 53%

REFERENCES

[1] Sebastian Abt and Harald Baier. 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In *2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, 40–55.

[2] Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Paolo Garza, Danilo Giordano, Marco Mellia, and Luca Venturini. 2016. SeLINA: a self-learning insightful network analyzer. *IEEE Transactions on Network and Service Management* 13, 3 (2016), 696–710.

[3] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. 2018. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications* 9, 1 (June 2018), 16.

[4] Zhitang Chen, Jiayao Wen, and Yanhui Geng. 2016. Predicting future traffic using hidden markov models. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 1–6.

[5] Cecilia Dao, Xinyu Liu, Alex Sim, Craig Tull, and Kesheng Wu. 2018. Modeling data transfers: change point and anomaly detection. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1589–1594.

[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* 9, Aug (2008), 1871–1874.

[7] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M Munafo, Politecnico Di Torino, and Dario Rossi. 2011. Experiences of internet traffic monitoring with tstat. *IEEE Network* 25, 3 (2011), 8–14.

[8] Anna Giannakou, Daniel Gunter, and Sean Peisert. 2018. Flowzilla: A Methodology for Detecting Data Transfer Anomalies in Research Networks. In *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 1–9.

[9] Jinoh Kim and Alex Sim. 2017. A New Approach to Online, Multivariate Network Traffic Analysis. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 1–6.

[10] Jinoh Kim, Alex Sim, Sang C Suh, and Ikkyun Kim. 2017. An approach to online network monitoring using clustered patterns. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 656–661.

[11] Jinoh Kim, Alex Sim, Brian Tierney, Sang Suh, and Ikkyun Kim. 2018. Multivariate network traffic analysis using clustered patterns. *Computing* (2018), 1–23.

[12] Jinoh Kim, Wucherl Yoo, Alex Sim, Sang C Suh, and Ikkyun Kim. 2017. A lightweight network anomaly detection technique. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 896–900.

[13] Alina Lazar, Kesheng Wu, and Alex Sim. 2018. Predicting Network Traffic Using TCP Anomalies. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 5369–5371.

[14] Zhengchun Liu, Rajkumar Kettimuthu, Ian Foster, and Yuanlai Liu. 2018. A comprehensive study of wide area data movement at a scientific computing facility. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1604–1611.

[15] Zhengyang Liu, Malathi Veeraraghavan, Jianhui Zhou, Jason Hick, and Yee-Ting Li. 2013. On causes of GridFTP transfer throughput variance. In *Proceedings of the Third International Workshop on Network-Aware Data Management*. ACM, 5.

[16] Leland McInnes and John Healy. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[17] Marco Mellia, Michela Meo, Luca Muscariello, and Dario Rossi. 2008. Passive analysis of TCP anomalies. *Computer Networks* 52, 14 (2008), 2663–2676.

[18] Marco Mellia, Michela Meo, Luca Muscariello, and Dario Rossi. 2008. Passive analysis of TCP anomalies. *Computer Networks* 52, 14 (2008), 2663–2676.

[19] Nageswara S Rao, Mariam Kiran, Cong Wang, and Anirban Mandal. 2018. *Detecting Outliers in Network Transfers with Feature Extraction*. Technical Report. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

[20] M Wang, Y Cui, X Wang, S Xiao, and J Jiang. 2018. Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Netw.* 32, 2 (March 2018), 92–99.

[21] Keith Winstein and Hari Balakrishnan. 2013. TCP Ex Machina: Computer-generated Congestion Control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 123–134.