A Machine Learning Approach for Packet Loss Prediction in Science Flows

Anna Giannakou[a], Dipankar Dwivedi[b] and Sean Peisert[a]

[a]*Data Science and Technology Department Lawrence Berkeley National Lab California, USA*
[b]*Geochemistry Department Lawrence Berkeley National Lab California, USA*

ARTICLE INFO

Keywords:
network performance
packet loss
random forest regression
science networks

ABSTRACT

Science networks and their hosted applications require large and frequent data transfers, but these transfers are subject to network performance degradation, including queuing delays and packet drops. However, well known network dynamics along with limited instrumentation access complicate the creation of an accurate method that predicts different performance aspects of data transfers. In this study, we develop a lightweight machine learning tool to predict end-to-end packet retransmission in science flows of arbitrary size. We also identify the minimum set of necessary path and host measurements needed as input features in our predictor in order to achieve high accuracy. In our evaluation process our predictor demonstrated low training times and was able to provide accurate estimates (97–99%) for packet retransmissions of data transfers of arbitrary sizes. The results also manifest that the our solution was able to predict retransmit behavior reasonably well (66%) even for previously unseen data if training and testing datasets had similar statistics.

1. Introduction

Science networks host applications that process large amounts of data derived from a diverse set of complex experiments. Oftentimes, these applications require large and frequent data transfers with explicit network performance requirements such as high-speed data delivery. Consequently, these transfers become very sensitive to performance degradation events. Even the slightest amount of packet loss can significantly increase the overall data transfer time [1], which in the context of complex experiments can be interpreted in delays in data availability. In order to ensure timely data availability, science networks feature dedicated systems (e.g., PerfSONAR [2]) that are able to detect and report on performance degradation events (e.g., latency increase, throughput degradation). However, these systems report transfer degradation events after they occur and to this day no scientific method that predicts different negative performance events is available for science networks.

Solutions that predict different aspects of TCP performance such as throughput or packet loss prediction, are largely centered around two approaches: *formula-based* and *history-based* predictions. *Formula-based* methods predict performance aspects using mathematical expressions that relate the predicted variable to path and end host properties such as Round Trip Time (RTT) or the receiver's window size. In most cases, measurements for the aforementioned properties are gathered using different active or passive network measurement tools. A common shortcoming of *formula-based* approaches is that they are greatly affected by the continuously evolving TCP implementations, making maintenance of up to date formula-based models a cumbersome process. On the other hand, *history-based* approaches produce a time series forecast of the desired attribute (e.g., packet loss) based on measurements derived from previous file transfers, collected either passively (e.g. through monitoring a link) or actively (e.g. by conducting file transfers of different size). Although for certain aspects of TCP performance, *history-based* approaches tend to be more accurate than *formula-based* predictions [3], existing solutions focus mostly on predicting network throughput.

The goal of this work is to develop an accurate light weight machine learning tool to predict end-to-end packet loss, manifested in the number of retransmitted packets in science flows of arbitrary size. We believe that understanding the nature of packet retransmissions would allow both scientists and network operators to mitigate packet loss through different host or flow reconfiguration techniques. We investigate the hypothesis that packet retransmissions are due to a combination of factors related to the selected "path" along with end host network configuration. We argue that the accuracy of formula-based solutions can be augmented by a tool that takes into account measurements of path and host attributes from previous data transfers.

Towards our goal for developing a robust analytical framework for retransmission prediction we focus our efforts on answering the following questions: 1. Which path properties or combination of path properties are needed in order to generate an accurate prediction? Do different combinations of input parameters demonstrate different levels of accuracy? 2. Do we need to take into account end host (i.e. client/server) network configuration parameters in our prediction? 3. Can we generate accurate predictions for data transfers of arbitrary sizes? and finally, 4. How robust is a history-based solution shifts in data transfer behavior? (e.g., when path properties change significantly or when the end hosts are reconfigured).

The contributions of this paper are:

- A light weight machine learning tool based on Ran-

agiannakou@lbl.gov (A. Giannakou); DDwivedi@lbl.gov (D. Dwivedi); sppeisert@lbl.gov (S. Peisert)
ORCID(s):

dom Forest Regression that is able to predict packet retransmissions in science flows of arbitrary size. Our predictor takes into account a combination of path properties (including RTT) and host parameters (such as TCP maximum congestion window) in order to predict the number of retransmitted packets in each network flow.

- A thorough evaluation of our proposed solution made with real world flow traces that represent data transfers between different scientific facilities and/or end users across the world. We evaluate our predictor using flow data that follow multiple network paths that exhibit fundamentally different behavior in terms of TCP related parameters (e.g., Round Trip Time, available throughput, etc). We measure the accuracy of our tool under different subsets of input parameters in order to make a recommendation on the most suitable combination of path and host related properties. To generate predictions for different sizes, we purposely train and test on datasets with transfers that range between a few hundred megabytes to many gigabytes.

The paper is organized as follows: Section 2 describes related work and Section 3 describes the individual datasets used for our analysis as well as the collection and aggregation tools used. Our solution and architecture is presented in detail in Section 4. Important evaluation aspects and obtained results are presented in Section 5. Finally, we conclude with important observations and suggestions for future work in Section 7.

2. Related Work

This section describes solutions that utilize machine learning techniques for predicting different performance aspects of TCP connections. We then present some empirical studies for TCP performance followed by mathematical models that are used to predict different aspects of a TCP flow.

Mirza et al. [4] propose a throughput estimation tool based on Support Vector Machines. The tool uses multiple flow-level features as input in order to predict end-to-end throughput. Although their solution's accuracy is considerably higher than standard history-based methods, it was only evaluated on artificial network traces where only specific network paths were considered. Furthermore, the tool has not been tested in a high volume and scientific traffic environment.

Nunes et al. [5] use the *Experts Framework* machine learning technique in order to provide accurate estimates of the round trip time (RTT) in TCP transfers. Their framework quickly adapts the predicted value based on average distance of previously predicted RTT from the actual value. The suggested solution achieves a reduction in the number of retransmitted packets but the evaluation process did not not include tests with large file transfers or congested network links. Hu et al. [6] aim at predicting RTT between two specific IP pairs (sender/receiver) based on their geographic distance. The authors collect their own network traces by discovering (using traceroute) the intermediate routers between source and destination pairs. Then they use the difference between two pings in order to calculate intermediate latencies. Their approach has not been tested in the context of science traffic or for source/destination pairs that belong to different scientific organizations (with large georgraphic distance between them).

Paxson et al. [7] conducted a comprehensive empirical study of TCP behavior focusing on modeling different patterns of packet loss. Although this work exposed a plethora of other issues as well (e.g., queuing delays, bottlenecks, etc) it also proved that the distribution of the packet loss duration across different TCP bulk transfers exhibits infinite variance. Barford et al. [8] expand this work by diving into the relationship between large transfer latency and individual packet loss events and server/client related delays. The authors focus only on http traffic and analyze traces obtained from only eight hosts in close proximity to each other. Furthermore, their analysis was limited to small file sizes (up to 3.2 MB). Ghasemi et al. [9] investigate the effect of sender/receiver misconfiguration (e.g., small receiver buffer or slow server) to packet loss and overall TCP performance in cloud environments. Their work identifies the effects of end host configuration on poor performance of TCP transfers. Their analysis only includes data from a single client/server pair conducting a small file transfer of 1MB. Although the solution yields satisfactory results when compared with the ground truth, it has not been evaluated in the context of large high frequency scientific data transfers.

In terms of developing mathematical expressions that correlate different aspects of TCP performance with packet loss Abouzeid et al. [10] use a stochastic model to predict network throughput based on bursty packet loss events. The evaluation process only included packet traces generated from ns simulator [11]. Parisi et al. [12] use Markov chains to correlate packet losses due to timeout and TCP flow performance. Their analysis is only theoretical without real network flow data. Altman et al. [13] also derive a stochastic model for packet loss and flow throughput incorporating the sender's window size. The authors create long lived TCP connections with large file transfers in order to test their approach. However, the dataset analyzed only includes three TCP connections.

Our work differs from the approaches described above in two core elements: First, we predict packet retransmission in the context of scientific data transfers using real network traces from data flows of arbitrary size. Second, our training and testing datasets include transfers between end hosts in different geographic locations (i.e., respectively varying RTT values) and network configuration parameters.

3. Data

This section describes the data that we used for testing our solution. We present the available features and the flow collection tool we used, as well as any modifications made in the original datasets.

Our packet loss prediction model operates on flow data.

We define a network flow as a five tuple identifier of Source IP, Destination IP, Source Port, Destination Port and Communication Protocol (TCP or UDP). Our data are collected from the 10 systems referred to as *Data Transfer Nodes (DTNs)* located at the *National Energy Research Scientific Computing Center (NERSC)* [14]. DTNs are explicitly dedicated and fine tuned for performing large data transfers between the NERSC scientific facility and the external scientific community. They tend to have low-latency, high-bandwidth network interface cards (NICs) and I/O systems designed to limit disk-related bottlenecks. A variety of tools such as Globus online [15] and GridFTP [16] are typically used in order to automate transfer of large datasets.

We collected flow-like data from each DTN using the "tstat" network monitoring tool [17]. Tstat is able to aggregate packet traces into flows and derive detailed statistics and performance metrics for each flow. Grouping packets into flows is particularly useful for optimizing efficiency in processing large amounts of network data. As opposed to other flow collection tools like NetFlow [18], tstat records non-sampled network data and also computes a wider set of performance features (the full list of the 53 metrics can be found here [17]). Furthermore, for ensuring anonymity of source and destination hosts, we drop the last octet of source and destination IPs.

In our analysis we only use flow data where the percentage of packet retransmissions is greater than zero. We opt for flows that demonstrate packet loss and discard "perfect," loss-free data transfers.

4. Proposed Solution

In this section, we describe our proposed solution along with the subset of features used as input as well as any data preprocessing made.

Our tool predicts the percentage of retransmitted packets for TCP data transfers of arbitrary size based on prior data transfers. Per transfer measurements include a combination of end host configuration metrics along with path-related metrics. Our solution aims to address two fundamental questions:

1. which path and end host properties provide the most accurate prediction of retransmits per flow? and
2. is there a confidence value that we can include in our predictions?

Predicting the number (and respectively percentage) of retransmitted packets can be formulated as a regression problem of predicting the value of a real valued number (i.e. number of retransmitted packets) based on multiple real-valued input features. Each data transfer is represented by a collection of features $x = \{x_1, x_2, ..., x_i\} \in R$. Each x_i is an observed feature e.g., size of the file being transferred, TCP congestion window, average round trip time, etc. Our goal is given x to predict the number of retransmitted packets $y \in R$. This is achieved by training the predictor using training data i.e. previous data transfers with known features and the corresponding measured number (and percentage) of retransmitted packets.

The analytical framework that we apply to this problem is *Random Forest Regression*, an established machine learning technique suitable for multivariate regression. We discuss details of RFR on the following section.

4.0.1. Random Forest (RF)

Random Forest Regressor (RFR), demonstrates several properties that make it suitable for our solution: 1. It can accept multiple features or combination of features [19] and use all of them to generate the prediction for the number of retransmitted packets. In addition, RFR has the ability to show the importance of different input features in the prediction outcome. 2. The input provided to RFR does not need to be in any specific parametric form as opposed to strict formula-based solutions. Finally, 3. RFR has low computational costs and exhibits small training times for large datasets that include hundreds of thousands of data transfers.

To best describe the RFR, we first describe a regression tree and forest. A forest is an ensemble of trees (Figure **??**); regression trees are created by partitioning the samples (i.e., the root nodes into homogeneous groups [nodes]). This process is repeated recursively until the terminal nodes are not defined. Each split is chosen according to a splitting criterion and on the values of a selected variable. The response of input variables can be predicted by simply following the path of a tree from the root node to the terminal node. The predicted response value is computed by RFR by averaging the probabilistic prediction of the ensemble of trees in that terminal node [20]. For our random forest model, we leveraged the RandomForestClassifier [21] as part of the scikit-learn package [22] with default parameters.
To avoid the overfitting, K-fold (fivefold) cross-validation was used in predicting the restransmit behavior. Furthermore, we also provide the relative importance of features to choose the best subset of inputs for predicting the packet loss with the highest accuracy.

4.1. Feature Selection

In order to select the optimal set of features for our prediction we first need to investigate the relationship between major causes of packet retransmissions and the available flow-level metrics in our datasets. Since retransmissions are mostly due to physical loss along the selected path and suboptimal end host network tuning, different combinations of end host properties (e.g., TCP max segment size) and path-related measurements (e.g., Round Trip Time) need to be taken into account in our model. We note that incorporating all 52 tstat collected measurements would provide an overly generalized solution with increased computational cost. The set of selected features is shown in Table 1.

A variety of studies [24] [25] [8] have demonstrated the relationship between TCP performance (in terms of throughput and packet loss) and different flow-level measurements such as file size (a host-controlled parameter), round trip time (a path-related parameter) and congestion window (a host-controlled parameter as well). Based on the findings

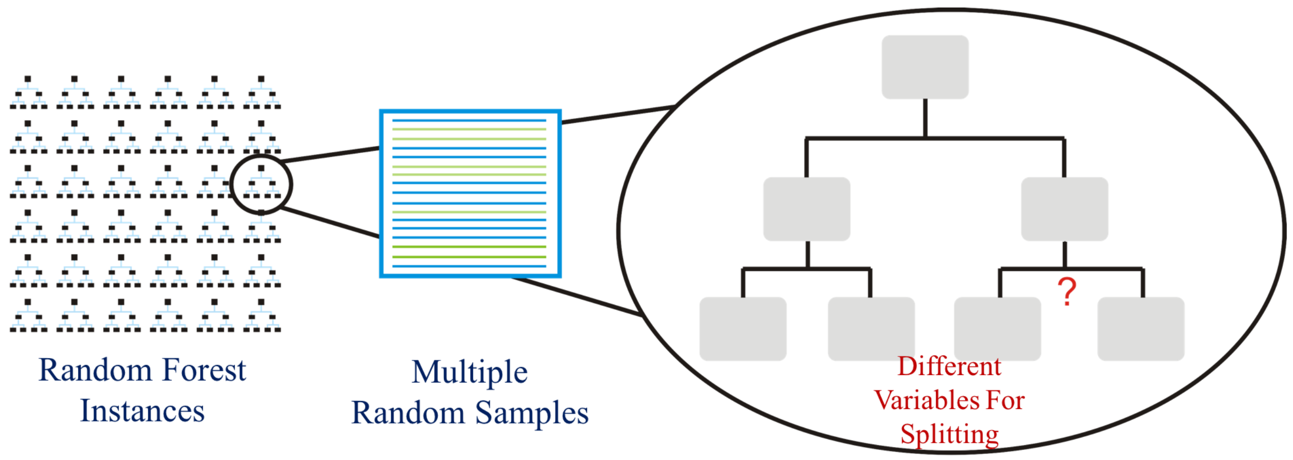

Figure 1: A Random Forest Regressor (modified from [23]).

Table 1
Input features and their tstat representation

Input Variables	Tstat field
File size	*file_size_MB*
Flow duration	*duration*
Throughput	*throughput_Mbps*
Source IP	*src_geoip*
Destination IP	*dst_geoip*
TCP initial congestion window	*tcp_cwin*
Average RTT	*tcp_rtt_avg*

presented in the aforementioned studies, the seven selected features provide an accurate representation of both host and path related factors that influence packet loss in scientific network transfers. In order to make our tool robust to level shifts (i.e. when path properties change significantly) we opt for incorporating the **average** round trip time for each flow. Average values smooth out any rtt variations related to increased latency due to a congested path.

In order to increase the quality and applicability of our solution we train our predictor on a wide range of file sizes, as opposed to solutions that target only bulk transfers or small files [4].

5. Evaluation

In this section we describe our methodology for evaluating our solution, and the datasets used. We conclude our analysis with a discussion on obtained results.

5.1. Methodology

Our evaluation approach focuses on two directions:

1. How well can our solution predict packet retranmissions in scientific data transfers or arbitrary size?

2. Does feature distribution variability in different datasets affect the ability of our solution to provide reasonably good accuracy?

To answer the first question we train our RFR model with flows from one dataset (see subsection 5.2 for detailed description) and record its accuracy when tested on different datasets. Furthermore, we examine whether noise reducing techniques (e.g., data smoothing) would improve the accuracy of our predictions. For answering the second question we compute our solution's accuracy when tested on datasets that are a year apart. Depending on the obtained value, we conduct a correlation analysis that allows us to identify whether different input features maintain the same importance across datasets.

5.2. Datasets

The datasets used for training and testing our solution are described in Table 2.

Table 2
Dataset composition

Dataset	Duration	Year
Datasets for testing RFR		
Dataset1	Jan 1 – Feb 28	2017
Dataset2	July 1 – Nov 30	2017
Dataset3	Jan 1 – Feb 28	2018
Datasets for testing seasonality		
Dataset4	Feb 1 – Feb 28	2017
Dataset5	Feb 1 – Feb 28	2018
Dataset6	Jan 1 – Jan 31	2017
Dataset7	Jan 1 – Jan 31	2018

We evaluated the RFR with the data collected at three different times: January to February 2017 (dataset1); July to November 2017 (dataset2); and January to February 2018 (dataset3) (Table 2). To understand different performance aspects of data transfers, we tested the RFR's ability to predict the retransmit behavior for different combinations of datasets: (a) individually dataset1, dataset2, and dataset3 (b) combining all datasets (dataset1, dataset2, dataset3), and (3) across datasets (e.g., training on dataset1 and testing on

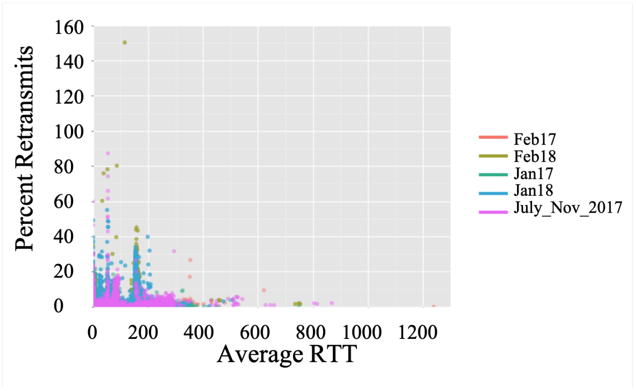

Figure 2: Retransmit data showing significant variability across times.

dataset2; training on dataset2 and testing on dataset3; training on dataset1 and testing on dataset3). All these evaluations were performed on the raw data as well smoothed data using the Automatic Smoothing for Attention Prioritization (ASAP) algorithm as described in (Section 5.3).

As opposed to other predictors [4], where both training and test traffic was artificial, our solution utilizes flow logs from real scientific data transfers of arbitrary size. Furthermore, in artificial lab-generated file transfers host configuration parameters and path properties are strictly set at explicit values (see evaluation section in [4]). However, our predictor's input features distribution demonstrate higher variability which increases the robustness of our approach under different network conditions. Finally, our solution is trained on a wide range of transfer sizes that range between a few hundred megabytes to many gigabytes.

5.3. Data smoothing

We also evaluated the RFR by smoothing the training and testing data. The rationale behind smoothing was that small-scale noise often obscures large-scale trends. Therefore, machine learning algorithms (e.g., RFR) do not perform reasonably well in the presence of noise in the data. With smoothing, it is possible to retain the large-scale structure of the data while removing as much noise as possible. To smooth the data, we used Automatic Smoothing for Attention Prioritization (ASAP) in the Time Series algorithm developed by Stanford InfoLab [26]. ASAP makes use of the sliding window aggregation model and performs hyperparameter tuning for automatically selecting a window so that the data retain the long-term trends. More details about the ASAP algorithm can be explored in [26].

6. Results and Discussion

6.1. Accuracy

We tested RFR's prediction accuracy for the percentage of retransmissions (percent retransmits from here after) using different combination of input variables. The accuracy of a RFR's prediction was estimated using R^2 values. The results using throughput [Mbps], duration and RTT are shown in Table 3. As our results demonstrate prediction accuracy was moderate. We tested the RFR's accuracy after smoothing the data and found that smoothing significantly improved the prediction accuracy (Table 3). Figure 3, which shows the percent retransmits as a time series of the raw data as well smoothed data, demonstrates that the ASAP algorithm removed small-scale noise while keeping the large-scale trends. Hence, the RFR performed better after smoothing due to the cleaner data. In order to investigate moderate accuracy results, we examine the behavior of retransmissions on different datasets. Figure 2 shows the percentage of retransmitted (percent retransmits) versus average RTT for different times. Percent retransmits show significant variability. Although percent retransmits are consistent in the bulk part for specific parts of some datasets, there are subtle differences in percent retransmits across datasets (e.g., January 18 with July–November 2017). The same behavior was observed in percent retransmits with other input variables, such as throughput (Mbps) and duration (not shown).

Table 3
Prediction accuracy of the retransmit behavior of RFRs for different datasets with and without smoothing.

	Accuracy Without Smoothing	Accuracy With Smoothing
Dataset1, Dataset2, Dataset3, & all Datasets	60%	97-99%
Training on Feb17, and testing on Jan17	<2%	66%
Training on Feb18, and testing on Jan18; Training on Feb17, and testing on Jan18	<2%	<2%

Table 4
Correlation of the retransmits with input features. Correlation values vary significantly across datasets, for example, tcp_win_max ranges from 0.08 (shown in blue) to 0.15.

Variable	Correlation (Entire Dataset)	Correlation (Jan17 and Jan18)	Correlation (Feb17 and Feb18)
tcp_rtt_avg	0.29	0.26	0.25
tcp_initial_cwin	0.19	0.18	0.16
tcp_win_max	0.15	0.08	0.10

6.2. Feature Variability

Although the RFR predicted the retransmit behavior with moderate (without smoothing) to high accuracy (with smoothing), it did not perform well with scenarios in which there is unseen data, as shown in Table 3 (except for training on Feb17 and testing on Jan17 scenarios). To examine this anomalous behaviorwe critically analyzed data and conducted a correlation analysis. Table 3 presents the correlation analysis of retransmits with different input features and Figure 6 shows various features at different times. It is clear from the correlation values that the relative importance of each input variable varies over time. This can be attributed to the fact that packet retransmission of bulk network flows can also be

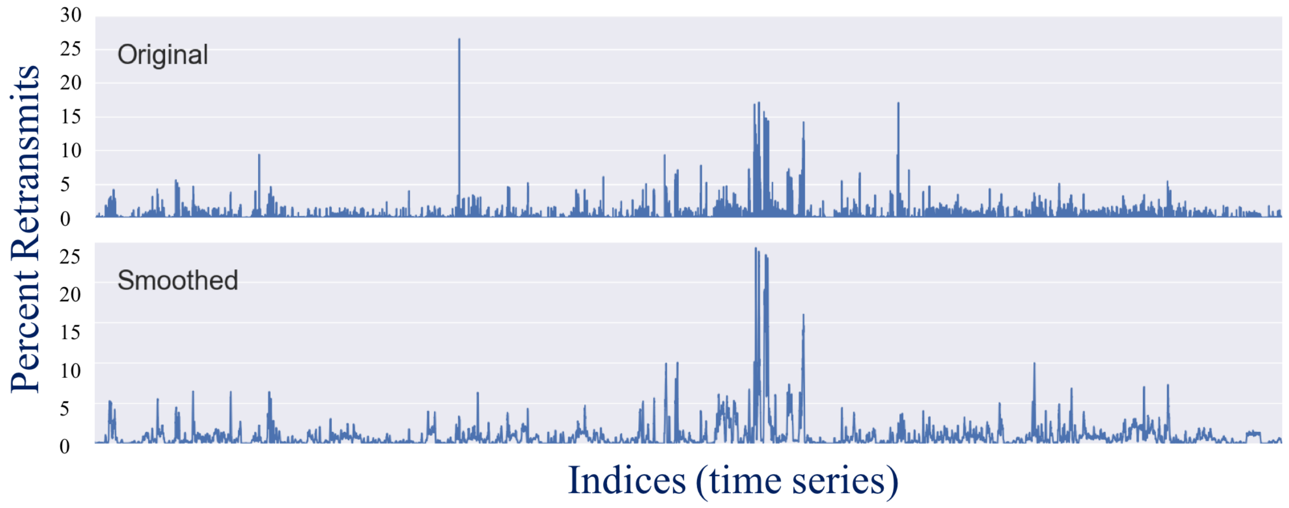

Figure 3: Reducing noise (e.g., smoothing) improves the prediction accuracy.

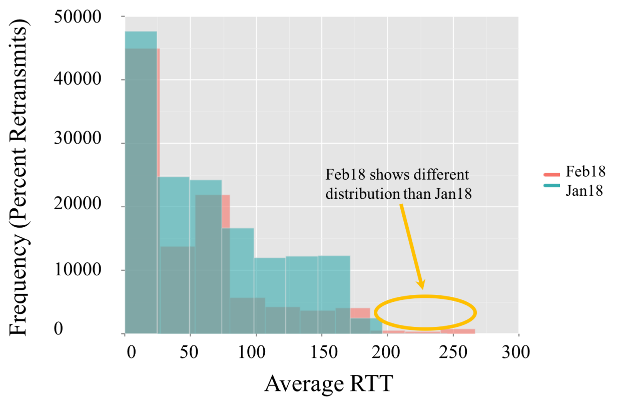

Figure 4: Data show different distributions for different times. The x-axis shows the bins that correspond to specific ranges of the average RTT, whereas the y-axis is the frequency of percent retransmits (the number of samples) for each bin.

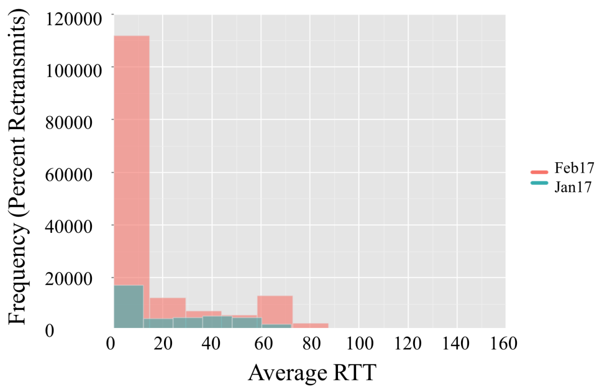

Figure 5: Data show consistent distributions for different times. The x-axis shows the bins that correspond to specific ranges of the average RTT, whereas the y-axis is the frequency of percent retransmits (the number of samples) for each bin.

affected by competing concurrent flows of different traffic type.

It is evident from Figure 6 that variability across factors is not consistent over time. For example, percent retransmits are small whereas throughput values are high with intermediate average RTT values in January 2017. In comparison, percent retransmits, throughput, and average RTT values are all high in January 2018. These results suggest that the relative importance of each feature changes over time.

Figures 4 and 5 show histograms of percent retransmits with average RTT. The x-axis shows the number of percent retransmits samples present for a range of the average RTT. The bin width on the Y-axis corresponds to a specific range of the average RTT. Smoothing did not change the distribution in general (not shown); however, the RFR's performance depended on the consistency of ranges of training and testing datasets. For example, the RFR performed reasonably well (66% accuracy) when trained on Feb17 and tested on Jan17. The moderate performance on the Jan17 dataset can be attributed to a variety of factors including the different distribution of the input variables. We noticed that only the average RTT has comparable ranges in Feb17 and Jan 17 for similar retransmit values (see Figure 5).

Taken together, we can conclude that the RFR was able to predict retransmit behavior reasonably well even for data not in the training dataset if training and testing datasets have similar statistics. Although the problem of different statistics can be avoided if the training set is made large, the total amount of data is limited in our case. However, training sets can be made large to include similar statistics as future datasets.

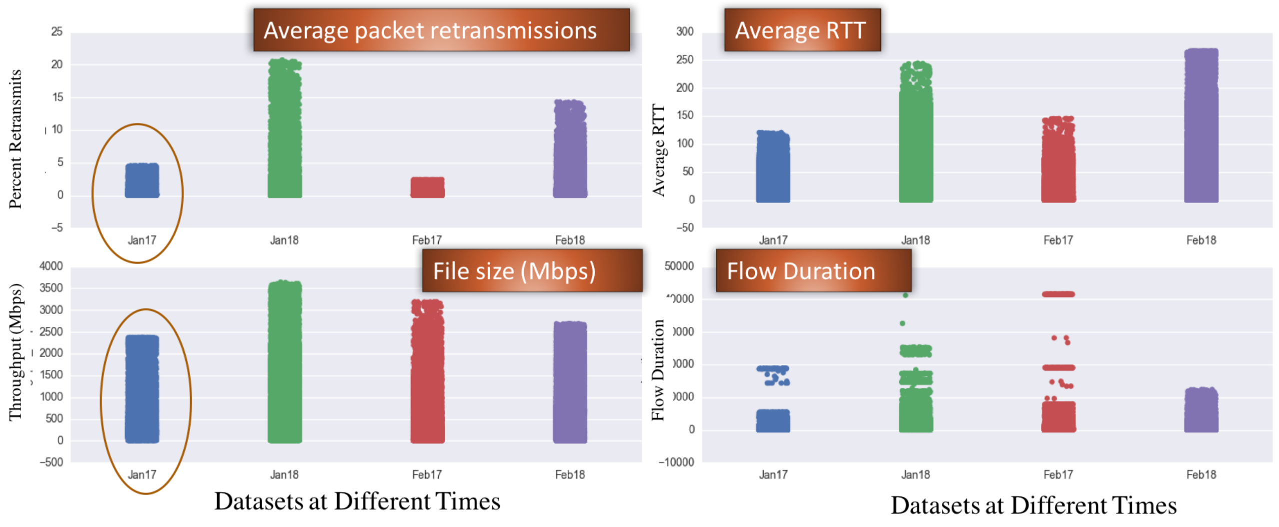

Figure 6: Different factors show variability across times.

7. Conclusion and Future Work

In this paper we investigated the issue of packet loss manifested through packet retransmissions in scientific data transfers. We presented a multi-variate machine learning framework that predicts packet retransmissions for file transfers of arbitrary size. We were able to identify the minimum set of path and host measurements that should be used as input features for generating accurate and robust predictions. Our framework, based on Random Forest Regression, demonstrates short training times and is able to provide accurate estimates for packet retransmissions occuring on data transfers of arbitrary sizes.

We have evaluated our framework on datasets that contain different number of flows exhibit significantly different distributions of the input features. In our analysis we were able to correlate different factors with the retransmission behavior. Our RFR models performed reasonably well in all datasets. We also found that smoothing reduced noise in the data by removing outlier events and significantly improved predictions. In addition, input variables showed different distributions for different times (e.g., January 17 vs. February 17); however, smoothing did not change the distributions. In order to account for outlier packet loss events, we plan to evaluate our framework's accuracy without smoothing techniques.

Although this work is a first step towards improving the performance and quality of scientific data transfers, it has some limitations. Our framework's prediction model was build using only Random Forest Regression (due to its flexibility in input parameters, see section 4.0.1 for detailed explanation). In order to compare with other model training techniques, we plan to include additional regression algorithms in our framework's prediction process. Although predicting retransmissions is one factor affecting the quality of scientific data transfers, we realize that in order to have a complete view of performance degradation events one needs to be able to also predict network throughput. We plan to add throughput predictions in the next version of our framework.We purposefully excluded flow data from data transfers that did not demonstrate any packet retransmission. Hence our solution cannot predict retransmit-free data transfers. Currently our predictor's accuracy is significantly affected by changes in the distribution of the input features (e.g., if the RTT distribution changes between two datasets then our predictions will render poor accuracy). In order to address this issue in our future work, we plan to include a weighted approach as part of data preproccessing. Furthermore, as a medium term goal, we would to include periodic passive measurements such as perfSONAR collected data in our training datasets. Improving the accuracy of our solution on previously unseen scientific data transfers can lead to mitigation of packet loss through different reconfiguration strategies.

Acknowledgments

The authors would like to thank numberous people for making this work possible. Dan Gunter for his valuable feedback; Josh Boverhof and Ravi Cheema for assisting with assisting with data access at NERSC; Jon Dugan, Mariam Kiran, Eric Pouyoul, and Brian Tierney from ESnet, Alberto Gonzales, Jason Leigh, and Alan Whinery from the University of Hawaii, and Ed Balas, Dan Doyle, CJ Kloote, Jennifer Schopf, and Alan Whinery for their helpful suggestions on the approach, data analysis, and interpretation of results.

This manuscript was funded by National Science Foundation under award ACI-1540933, and has been authored by authors at Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors

of this work.

A. Artifact Description

A Machine Learning Approach for Packet Loss Prediction in Science Flows

A.1. Abstract

Science networks hosting applications require large and frequent data transfers, but these transfers are subject to network performance degradation, including queuing delays and packet drops. Several factors can be ascribed to the network performance degradation; however, no accurate, known existing method is available for predicting different performance aspects of data transfers. In this study, we developed an accurate lightweight machine learning tool to predict end-to-end packet retransmission in science flows of arbitrary size. We also identified the minimum set of necessary path and host measurements needed as input features in our predictor in order to achieve high accuracy. The Random Forest Regression demonstrated low training times and was able to provide accurate estimates (97–99%) for packet retransmissions of data transfers of arbitrary sizes. The results also demonstrated that the Random Forest Regressor was able to predict retransmit behavior reasonably well (66%) even for previously undata if training and testing datasets had similar statistics.

A.2. Description

A.2.1. Check-list (artifact meta information)

- **Algorithm:** Packet loss prediction leveraging the Random Forest Regression algorithm from scikit-learn
- **Program:** Python
- **Compilation:** None needed
- **Data set:** Dataset description in Section 5.2
- **Experiment customization:** None
- **Publicly available?:** Code can be made available upon request by contacting the authors of this paper. The tstat data used 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. However, NERSC periodically makes data available to qualified researchers. Inquiries should be directed to *security@nersc.gov*.

A.2.2. How software can be obtained (if available)

Code can be made available upon request

A.2.3. Hardware dependencies

None

A.2.4. Software dependencies

Required python packages: requests, sockets, elasticsearch, json, sys, os, re, datetime, ipaddress, numpy, sklearn.metrics, pandas, sklearn.ensemble. sklearn,model_selection, pandas

A.2.5. Datasets

See section 5.2 for detailed description

A.3. Installation

No installation for python script

A.4. Experiment workflow

See section 5.1

A.5. Evaluation and expected result

See Section 6

A.6. Experiment customization

None needed

A.7. Notes

n/a

References

[1] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ: A Network Design Pattern for Data-intensive Science," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. ACM, 2013, pp. 85:1–85:10.

[2] B. Tierney, J. Metzger, J. Boote, E. Boyd, A. Brown, R. Carlson, M. Zekauskas, J. Zurawski, M. Swany, and M. Grigoriev, "perfSONAR: Instantiating a Global Network Measurement Framework."

[3] Q. He, C. Dovrolis, and M. Ammar, "On the Predictability of Large Transfer TCP Throughput," in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '05. ACM, 2005, pp. 145–156.

[4] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '07. ACM, 2007, pp. 97–108.

[5] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A Machine Learning Approach to End-to-End RTT Estimation and its Application to TCP," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, July 2011, pp. 1–6.

[6] W. Hu, Z. Wang, and L. Sun, "Guyot: a hybrid learning- and model-based RTT predictive approach," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 5884–5889.

[7] V. Paxson, "End-to-end Internet Packet Dynamics," in *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '97. ACM, 1997, pp. 139–152.

[8] P. Barford and M. Crovella, "Critical Path Analysis of TCP Transactions," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2 supplement, pp. 80–102, Apr. 2001.

[9] M. Ghasemi, T. Benson, and J. Rexford, "Dapper: Data Plane Performance Diagnosis of TCP," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. ACM, 2017, pp. 61–74.

[10] A. A. Abouzeid, S. Roy, and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 3, March 2000, pp. 1724–1733 vol.3.

[11] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[12] S. Fortin-Parisi and B. Sericola, "A Markov model of TCP throughput, goodput and slow start," *Performance Evaluation*, vol. 58, no. 2, pp. 89 – 108, 2004, distributed Systems Performance.

[13] E. Altman, K. Avrachenkov, and C. Barakat, "A stochastic model of TCP/IP with stationary random losses," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 356–369, April 2005.

[14] "National Energy Research Scientific Computing Center," http://www.nersc.gov, accessed: 2018.

[15] I. Foster, "Globus Online: Accelerating and Democratizing Science Through Cloud-Based Services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, May 2011. [Online]. Available: http://dx.doi.org/10.1109/MIC.2011.64

[16] W. Allcock, "Gridftp: Protocol extensions to ftp for the grid," 07 2018.

[17] "Tstat- A statistic and analysis tool," http://tstat.polito.it/measure.shtml, accessed: 2018.

[18] "Cisco NetFlow," https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, accessed: 2018.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] U. Grömping, "Variable importance assessment in regression: linear regression versus random forest," *The American Statistician*, vol. 63, no. 4, pp. 308–319, 2009.

[21] "scikit-learn - Forests of randomized trees," http://scikit-learn.org/stable/modules/ensemble.

[22] "scikit-learn - Machine Learning in Python," http://scikit-learn.org/stable/.

[23] M. Stockman, D. Dwivedi, R. Gentz, and S. Peisert, "Detecting Control System Misbehavior by Fingerprinting Programmable Logic Controller Functionality," 2018 (in review).

[24] Y. Zhang and N. Duffield, "On the constancy of internet path properties," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, ser. IMW '01. New York, NY, USA: ACM, 2001, pp. 197–211. [Online]. Available: http://doi.acm.org/10.1145/505202.505228

[25] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz, "Analyzing stability in wide-area network performance," in *Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '97. New York, NY, USA: ACM, 1997, pp. 2–12. [Online]. Available: http://doi.acm.org/10.1145/258612.258631

[26] K. Rong and P. Bailis, "Asap: prioritizing attention via time series smoothing," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1358–1369, 2017.