# Denoising Internet Delay Measurements using Weak Supervision

Anirudh Muthukumar*
*University of Oregon*
anirudhm@cs.uoregon.edu

Ramakrishnan Durairajan
*University of Oregon*
ram@cs.uoregon.edu

***Abstract*—To understand the delay characteristics of the Internet, a myriad of measurement tools and techniques are proposed by the researchers in academia and industry. Datasets from such measurement tools are curated to facilitate analyses at a later time. Despite the benefits of these tools and datasets, the systematic interpretation of measurements in the face of *measurement noise*. Unfortunately, state-of-the-art denoising techniques are labor-intensive and ineffective. To tackle this problem, we develop NoMoNoise, an open-source framework for denoising latency measurements by leveraging the recent advancements in weak-supervised learning. NoMoNoise can generate measurement noise labels that could be integrated into the inference and control logic to remove and/or repair noisy measurements in an automated and rapid fashion. We evaluate the efficacy of NoMoNoise in a lab-based setting and a real-world setting by applying it on a CAIDA's Ark dataset and show that NoMoNoise can remove noisy measurements effectively with high accuracy.**

***Index Terms*—Internet measurements, weak supervision, measurement noise.**

## I. Introduction

Understanding the delay characteristics of the Internet is one of the key goals of Internet measurement researchers, service providers, and content delivery networks. To this end, a myriad of measurement tools and techniques were proposed by the researchers in academia and industry to measure delay-related properties such as loss, latency, etc. [1]–[10]. These tools play a critical role in traffic engineering decisions [11] and how content is delivered in today's Internet [12]. Furthermore, datasets from such measurement tools are curated to facilitate analyses at a later time [1], [3], [13]–[15].

Despite the benefits of the proposed tools to measure the delay characteristics of the Internet, what is critically lacking is a *systematic framework to interpret the measurements in the face of measurement noise*.We define *noise* as the presence of non-representative values that confounds sound interpretation of measurements as it is hard to discern from the actual behavior (*e.g.,* is noise an artifact of the network [16] or the measurement tool [17]?). Said differently, noise is a set of erroneous values that do not exhibit the true characteristics of the network. For example, the presence of a series of hop latency spikes in `ping`, or `traceroute` measurements could either be classified as the representative behavior of the measured path or could simply be marked as noise due to the presence of background traffic. The notion of noise was first introduced by Paxson in earlier studies [18], [19] and by Willinger in [20]; otherwise, it has not received much attention [21].

While noise confounds all types of measurements, in this paper, we focus only on latency measurements due to its importance to the Internet. To underscore the importance of noise, consider a scenario where erroneous values span orders of magnitude higher latency values (*e.g.,* instead of a typical 10 ms delay, we might observe close to 1 s delay—is this noise or is it buffering?). In case of a representative behavior (*i.e.,* buffering), naïvely inferring the measurements as noise might result in sub-optimal control *e.g.,* due to re-routing in a sub-optimal path [12]. On the contrary, in case of an actual noise, inferring the measurements as a representative behavior, due to buffering, might again lead to unnecessary operational decisions (*e.g.,* examining packet traces further to identify the problem). In short, noise (1) leads to potentially bad operational changes and (2) confounds the true behavior of the network.

Faced with the uncertainty described above is the fact that denoising (*i.e.,* the process of removing measurement noise) is not done systematically today: it is either done manually due to the lack of hand-labeled ground truth or in a one-off fashion with simple filters (*e.g.,* mean plus two-sigma deviation) [10], [22], [23]. These result in decreased research productivity and increased person-hours (*e.g.,* typically 60% of the time is spent on denoising and organizing the data [24]). Current denoising techniques are either too naïve and/or labor-intensive.

In this paper, our main goal is to design and develop a framework for denoising latency measurements based on recent advancements in machine learning. Our high-level objectives are to make the denoising process easy, automatic, and rapid. We start by building a novel machine learning-based framework—called NoMoNoise—that will consume latency measurements (from an existing capability *e.g.,* `ping`) as *input* and classify noise from actual behaviors of the network. The key challenge is the lack of training data to classify and discern noise from true behavior. To tackle this challenge, we leverage recent advances in weak supervision: our insight is to create low-quality training data using the data programming paradigm proposed by Ratner et al. [25] and then use generated generative models to increase the accuracy of the training data; the generative model will have noise labels with probabilities to indicate accuracies. This will result in more accurate *noise labels* (*i.e.,* training data), where the labels are generated automatically.

*Work done during internship at the University of Oregon.

Next, we use noise labels to create and train predictive models (*e.g.,* deep learning models using TensorFlow [26]). The predictive models could then be used to remove and/or repair the noise. We evaluate the efficacy of the framework in two generic settings: (a) a lab-based setting by adding synthetic noise to the outputs of `ping` tool, and (b) a real-world setting by applying it on the publicly available CAIDA's Ark [1] datasets that are created using `traceroute` tool. Our evaluations show that NoMoNoise can identify and capture noisy measurements as well as improve accuracies up to 2x in comparison with naïve unsupervised techniques.

## II. Background and Related Work

Applying machine learning to solve networking problems is of interest to the community for more than a decade. A survey of such efforts is presented here [27] and a comparative study on the performances of various supervised learning techniques is here [28]. These approaches are later improved using statistical techniques such as expectation maximization [29], [30]. Complementary to the supervised learning techniques, several efforts used unsupervised learning techniques such as clustering and Principal Component Analysis (PCA) to detect anomalies in BGP [31] and network traffic [32]–[34], network diagnosis [35], and event detection [23].

Weak supervision is the third learning technique that attempts to combine and learn noisy labels from many weak sources to build a predictive model. The most popular form of weak supervision is distant supervision [36], [37]. The next popular form of weak supervision is crowdsourcing with unreliable non-expert annotators [38], [39]. These efforts offer less accuracy and less coverage on a training set—a problem tackled by the Snorkel project [40], [41]. Snorkel combines labels from different weak supervision sources to increase the accuracy and coverage of training sets using data programming paradigm where users can programmatically create lower-quality training data [25]. Unfortunately, there is a huge scientific gap between Internet measurement and machine learning communities, underscoring the critical need for operational frameworks to denoise delay measurements.

## III. Methodology

### A. Problem Statement

Given a set of latency (*e.g.,* RTT) measurements, our goal is to design a framework to classify and label them either as *good* or *noisy* measurements with a certain amount of confidence. More formally, our goal is to create a machine-learning classifier, $f : X \rightarrow Y$. $X$ is a set of (RTT) measurements (*i.e.,* $\{x_o, x_1, ..., x_n\} \mid x_i \in [0, \infty)$). Similarly, $Y = (X, L, C)$ is the set of RTT measurements with the classified labels (*i.e.,* $\{l_o, l_1, ..., l_n\} \mid l_i \in [0, 1]$) and inferred confidence values (*i.e.,* $\{c_o, c_1, ..., c_n\} \mid c_i \in [0, 100]$).

### B. Challenges in using Machine Learning to Denoise Measurements

As described above, the natural way to discern noise from the actual behavior is to take a machine learning-based approach. This has been the research theme of many prior efforts (see §II). However, to make machine learning a viable option, one has to tackle the *problem of unavailable training data*—a key hindrance to using machine learning techniques in the networking and security domains [42], [43]. Specifically, supervised learning techniques construct predictive models by learning from a large number of training examples with labels indicating the ground truth. Most successful techniques, such as deep learning, require strong supervision in the form of ground truth labels, which is impossible to obtain for large amounts of training data in the Internet. Moreover, creating hand-labeled training data manually is a formidable task, to say the least.

The problem with unsupervised learning methods is that not all types of clustering are suitable for analyzing RTT values. This is because there is no rigid mathematical definition for outliers (measurement noise in our case) [16], [17]. Moreover, issues with PCA-based methods are pointed out by [44], [45]. Similarly, the sensitivity of PCA to calibration and its corresponding implications are discussed by Ringberg *et al.* [46]. Due to these challenges, prior efforts often follow a one-off approach to denoise measurements. For example, prior efforts use simple filters (*e.g.,* mean plus one-sigma deviation) or heuristics or thresholds, which are ineffective (as shown below), to remove the noise values or erroneous measurements [10], [22], [23].

### C. Datasets Used

CAIDA's Ark project [1] consists of a globally distributed active measurement infrastructure serving the network research community with datasets for more than a decade. From the Ark project, we collected one day's worth of `traceroute` data for 6 different vantage points in the US, which forms the basis of the first dataset in our analysis (see §IV-A). We use sc_warts2json tool [47] to process the dataset. The dataset includes 3,214,134 traceroute measurements and 6,960,105 RTT measurements. In addition, we also use 10k `ping` measurements collected in a lab-based setting (see §IV-B).

### D. NoMoNoise Approach

Inspired by Snorkel [40], we propose that the most promising approach to denoise delay measurements is to use the labels produced by an unsupervised technique as weak supervisions to train a predictive (deep learning) model. Based on this, we develop NoMoNoise: a novel framework that will consume latency measurements as input, apply an unsupervised technique to generate weak labels (step 1), use the generated weak labels to produce more accurate training data and confidence values (step 2), and use the training data and train a predictive model to denoise the measurements (step 3). We explain each one of the steps below.

**Step 1: Establishing a baseline for noise.** In this step, we leverage the recent trend in machine learning—specifically, in weak supervision—which shows how users (*e.g.,* researchers) can programmatically create lower-quality training data using *data programming* paradigm [25], [41] to establish a baseline

| Clients | Q3 + 1.5 x IQR | KMeans | LOF | $\mu$ | $\mu$+$\sigma$ | $\mu$+2$\sigma$ | $\mu$+3$\sigma$ | EE | ORCE | IF |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 1.576 | 1.534 | 1.836 | 0.622 | 1.001 | 1.338 | 1.459 | 0.557 | 0.634 | 0.558 |
| B | 0.335 | 0.304 | 0.307 | 0.295 | 0.302 | 0.302 | 0.302 | 0.302 | 0.303 | 0.302 |
| C | 0.325 | 0.303 | 0.304 | 0.294 | 0.302 | 0.302 | 0.302 | 0.303 | 0.304 | 0.303 |
| D | 0.462 | 0.479 | 0.512 | 0.415 | 0.439 | 0.455 | 0.469 | 0.415 | 0.414 | 0.415 |
| E | 0.470 | 0.489 | 0.530 | 0.424 | 0.454 | 0.470 | 0.482 | 0.423 | 0.423 | 0.423 |
| F | 0.480 | 0.484 | 0.528 | 0.425 | 0.452 | 0.464 | 0.475 | 0.425 | 0.424 | 0.425 |
| G | 0.470 | 0.482 | 0.520 | 0.424 | 0.448 | 0.460 | 0.471 | 0.423 | 0.423 | 0.424 |
| H | 1.200 | 1.004 | 1.010 | 0.924 | 0.977 | 0.987 | 0.991 | 0.948 | 0.945 | 0.956 |
| I | 4.177 | 1.042 | 4.627 | 1.056 | 1.632 | 1.745 | 1.775 | 1.770 | 1.623 | 1.770 |

TABLE I

WEAK LABELS GENERATED BY DIFFERENT TECHNIQUES FOR RANDOMLY-SAMPLED CLIENTS FROM ARK'S EUGENE VANTAGE POINT. CLIENTS ARE ANONYMIZED WITH ALPHABETS. Naïve application of classifiers results in widely varying noise thresholds.

for what qualifies as noise. However, the availability of a wide range of techniques to establish a noise baseline poses the most important challenge: which technique to use? With this question in mind, we build a *label generator* component in NoMoNoise to generate low-quality (weak) labels using user-written labeling functions (*e.g.,* select RTT values > 100 ms between endpoints A and B, select a value if the historical RTT values are an order of magnitude lesser, etc.). The component will be able to generate weak labels (*i.e.,* threshold values to separate good vs. noisy measurements) using a host of labeling functions ranging from simple thresholds (*e.g.,* Q3 + 1.5 x IQR) to mean ($\mu$) plus one/two/three sigma deviation ($\sigma$) filters to unsupervised methods such as KMeans and hierarchical clustering techniques to anomaly detection techniques such as local outlier factor (LOF) to outlier detection techniques such as elliptic envelope (EE), overly robust covariance estimation (ORCE), and isolation forest (IF).

Table I shows the weak labels generated by different techniques based on per-hop RTT measurements from 10 sample clients (IP addresses) as seen in Ark's Eugene vantage point. If the RTT value for a client is less than the generated weak label for a particular labeling technique, then that value is labeled good according to that labeling technique; otherwise, it is classified as noise. From this table, we make two key observations. (a) Less surprisingly, for every client, the thresholds generated by these labeling techniques differ significantly from each other. (b) Naïve application of these (differing) labels to denoise measurements will lead to sub-optimal and error-prone inferences.

**Step 2: Weak supervised learning.** The above step creates training data using user programs or labeling functions that will result in low-quality training data with inaccuracies. In this step, we use generative models to learn and improve the accuracy of the training data using Snorkel[1]; similar to Ratner et al., we assume very limited availability of hand-labeled ground truth data whose creation is economical as well as feasible [25], [41]. Given a dataset with delay values $X$, let us assume that we want to predict the unknown labels $L$. In traditional machine learning where we take a predictive approach, we will model $P(l|x)$. On the contrary, in a generative approach, we will model $P(x,l) = P(x|l)P(l)$, where we are learning the accuracies of the labeling functions, *without* access to true labels. Learning happens from the (dis)agreements between different labeling functions and the limited hand-labeled ground truth data, which is the key insight in this step. This step is instantiated in NoMoNoise via the *generative modeler* component, which takes the weak labels generated in step 1 and connects to the Snorkel library to emit generative models with accuracies in the form of probabilities/confidence values. More concretely, Snorkel scales the labels from the labeling functions to assign probabilities to each RTT value.

**Step 3: Denoising RTT measurements.** The final step is to automatically denoise the erroneous values using generative models created in step 2. The user of the NoMoNoise framework (*e.g.,* measurement researcher) can either choose to remove the noise. For the option of removing noise, a user can use the generative model with probabilities from the previous step to create and train a predictive model such as Long Short Term Memory (LSTM) *e.g.,* using TensorFlow [26], PyTorch [48], etc.

## IV. EVALUATION OF NOMONOISE

We evaluate the efficacy of NoMoNoise framework in two different settings: (a) a real-world setting by applying it on the publicly available datasets that are created using `traceroute` tool and that is part of CAIDA's Ark [1] project, and (b) a controlled lab-based setting by generating datasets using `ping` tool. All the datasets used in this study are divided into three sets: a development set (100 RTTs from a random client), a testing set (100 RTTs from another random client), and a training set (RTTs from the remaining clients in a dataset). The training set is unlabeled and contains weak-labeled noisy data based on the labeling functions in Table I. We manually annotate the 200 RTT values by assigning labels (*i.e.,* with 0 (noise) or 1 (good)) and split them into a development set and a (blind) testing set. Based on the training/development data and the (dis)agreements between them, a probabilistic model is generated by Snorkel (in step 2). Once trained, the accuracy of the model is checked by allowing it to predict the labels for the held-out testing set.

In our evaluations, we use LSTM to train our predictive model (based on the probabilistic model) to ensure that (a) the

[1]One of the main challenges in using Snorkel in NoMoNoise is that Snorkel was originally meant to handle textual data which is in contrast with our case where we deal with delay measurements. We overcame this challenge by re-writing the data processing pipeline in Snorkel to accommodate numerical values.

long-term dependencies are well captured and (b) prior noisy measurements are remembered. We train the model for 20 epochs with a learning rate of 0.001.[2] Due to space constraints, we only report F1 score, which is the weighted ratio of precision and recall.[3] Reported F1 scores are based on the held-out testing set. High F1 scores indicate perfect precision and recall, which is also the rationale behind presenting this metric in the paper.

### A. Using community datasets

We use the traceroute-based RTT measurements from the CAIDA's Ark [1] project to evaluate the efficacy of NoMoNoise. We format the data into <source, destination, $rtt_1$, $rtt_2$, ..., $rtt_n$> tuples. We select RTT measurements per source-destination (SD) pair as features and divide the data into three sets (explained above). We wrote different labeling functions including KMeans (shown in Figure 1), $\mu$ plus one/two/three $\sigma$, LOF, EE, etc. to assign weak labels (either 0 (bad) or 1 (good)) to the RTT measurements to the training data. Original weak labels assigned by KMeans labeling function are shown in Figure 2-(left) for a sample SD pair. Generated probabilities (*i.e.,* noise labels) capturing inaccuracies in the assigned labels (across all labeling functions) using Snorkel library [40] are shown in Figure 2-(right). Note that the changes in output probabilities values are indicative of more accurate training data [41]. Based on the resulting generative models (with probabilities) from the corresponding labeling functions per SD pair, we then trained LSTM predictive models (per source-destination pair). We tested the trained LSTM models on the held-out test data.

```
def LF_KMeans(c):
    val = c.number1.get_attrib_tokens()
    if float(val[0])<=kmeans_threshold:
        return 1
    else:
        return 0
```

Fig. 1. A sample KMeans labeling function.

Figures 3-(left) and -(right) depict the average of F1 metrics before (naïve) and after noise removal (NoMoNoise) for the test data from 6 different Ark vantage points in the US. From these figures, we note the following. First, in a majority of the cases, NoMoNoise performs well when compared to the naïve counterparts with F1 score improvements up to 2x (*e.g.,* observe the improvement for $\mu$-based classifier for eug clients before and after removing noise) with high precision (0.89 to 0.98) and high recall (0.93 to 0.99) (results not shown). This improvement is due to NoMoNoise minimizing the noise based on the disagreements between RTT measurements. Next, LOF, even though it fails to capture the RTT distribution and inherent density in certain cases, performs well compared to the remaining techniques. This is because LOF is very selective: (a) it does not predict many measurements to be good in comparison with the naïve predictions and (b) while all noisy measurements predicted are indeed noisy, some are missed in the predictions. Next, IF, ORCE and EE perform extremely well for all cases in comparison with naïve unsupervised techniques (see Figure 3-(left)).

| Labelling function | Probability |
|---|---|
| [[1]] | 0.645767073486 |
| [[1]] | 0.645767073486 |
| [[0]] | 0.5 |
| [[1]] | 0.645767073486 |
| [[0]] | 0.5 |
| [[0]] | 0.5 |
| [[0]] | 0.5 |
| [[1]] | 0.645767073486 |
| [[0]] | 0.5 |
| [[0]] | 0.5 |
| [[1]] | 0.645767073486 |
| [[0]] | 0.425 |
| [[0]] | 0.45 |
| [[0]] | 0.01 |
| [[0]] | 0.5 |
| [[0]] | 0.5 |
| [[0]] | 0.5 |
| [[1]] | 0.645767073486 |
| [[1]] | 0.645767073486 |
| [[1]] | 0.645767073486 |

Fig. 2. Original weak labels assigned by KMeans are shown on the left. Generated probabilities (*i.e.,* noise labels) by learning inaccuracies in different weak labels are shown on the right. Note the changes in output probabilities values that are adjusted by learning the inaccuracies in the training data and are indicative of better training data.

### B. Using controlled lab-based datasets

Next, we evaluate NoMoNoise using RTT measurements from ping data obtained in a controlled, lab-based setting. This experiment consists of 10,000 RTT measurements obtained by pinging an internal server from a host in the same network, following the same training-development-testing split. Figure 4 shows the F1 scores of different techniques on this dataset. From this figure, we note that the LOF, once again, is selective when compared to other techniques with high recall and high precision values. Overall, we observe a 1.4x improvement in the average F1 score.

### C. Limitations of NoMoNoise

Though the results above show that NoMoNoise performs well (*i.e.,* high F1 scores due to high precision and recall values), there are cases where NoMoNoise resulted in false positives and false negatives (*e.g.,* better F1 scores for KMeans and LOF for naïve techniques in Figure 4). We discuss these cases and describe the limitation of our tool, addressing which is our ongoing focus. We observed cases where the labels predicted by NoMoNoise led to false positives. The main reason for false positives is due to the calculated thresholds from different techniques (in step 1) that are greater than the maximum of RTT values observed in the dataset. Figure 5-(left) and -(right) depicts the CDF of RTT measurements and the corresponding threshold (Q3 + 1.5 x IQR) calculated for two such cases from the Ark dataset.[4] We note that NoMoNoise is applicable *only* if the established threshold is smaller than the maximum

[2]These values are not prescriptive. They emerged through multiple runs of NoMoNoise, the results of which are not shown here due to space constraints.

[3]We define precision as the ratio of correctly predicted good measurements (true positive) to the total number of predicted good measurements (true and false positives). The recall is the ratio of correctly predicted good measurements (true positive) to all the *actually* good measurements (true positives and false negatives). In the results, high precision means low false-positive rate (*i.e.,* correct classification) and high recall means getting the noisy/good measurements rather than getting *all* of them correct. Finally, F1 Score = 2*(Recall*Precision)/(Recall+Precision).

[4]Other thresholds (e.g., $\mu + \sigma$) exhibited similar characteristics and are not shown here due to space constraints.

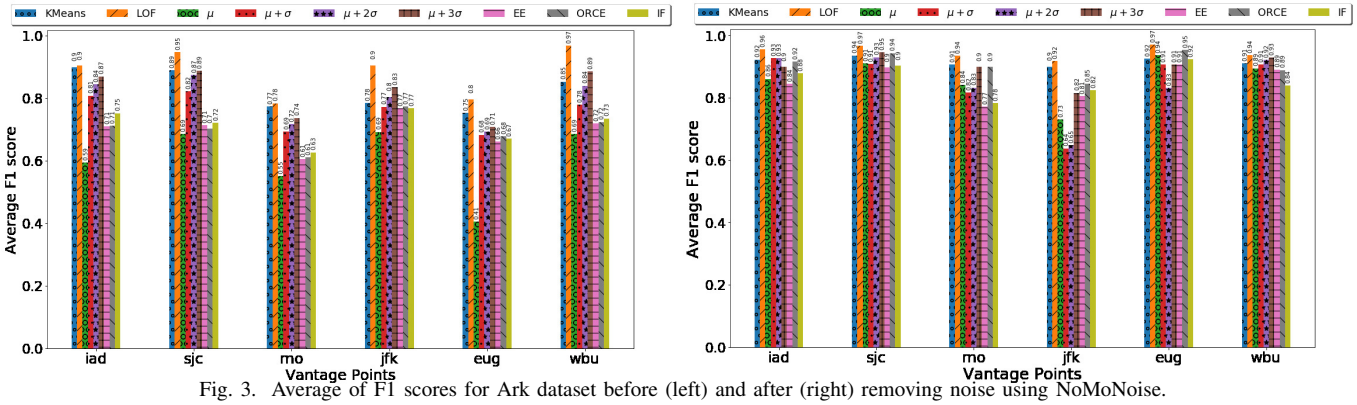

Fig. 3. Average of F1 scores for Ark dataset before (left) and after (right) removing noise using NoMoNoise.

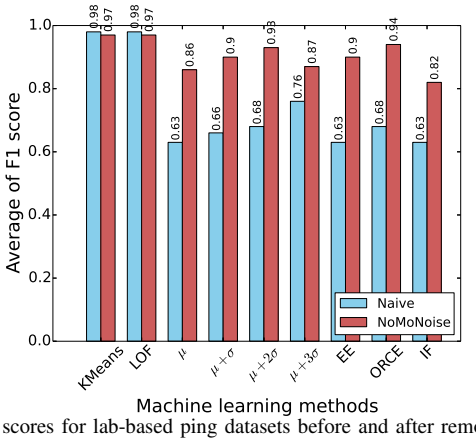

Fig. 4. F1 scores for lab-based ping datasets before and after removing noise using NoMoNoise framework.

RTT value observed. This is evident from two other sample clients whose CDF of RTT values and thresholds are depicted in Figure 6-(left) and -(right).

One way to address this limitation is to leverage mean-shift clustering [49], which is a centroid-based clustering approach for locating the maxima of a density function. Specifically, given a dataset, it partitions the dataset into $n$ different clusters (where we do not have to define $n$), and each of these clusters is formed by the density distribution. That is, the first cluster will have the highest number of points, the second cluster will have the second-highest number of points, and so on. This way one can ensure that the threshold is always maintained within the maximum observed RTT value.

## V. Summary and Future Work

Internet measurements community is critically lacking a systematic framework to interpret measurements. The key hindrance to creating such a framework is measurement noise, which we define as the presence of non-representative and erroneous values in the delay measurements. In this paper, we develop a systematic weak supervision-based framework and culminate in NoMoNoise, an open-source implementation of the framework, for denoising delay measurements in an automated and rapid fashion. We evaluate the efficacy of NoMoNoise in a lab-based setting and in a real-world setting by applying it to a community dataset (*i.e.,* CAIDA's Ark) and show that NoMoNoise can remove noisy measurements effectively with high F1 scores.

In our ongoing work, we are extending NoMoNoise to repair and replace noise with representative value(s). To this end, we intend to design and build a matrix completion-based repair framework that builds upon prior work on latency estimation using iterative hard-threshold singular value decomposition [10].

## Acknowledgment

We thank the anonymous reviewers for their insightful feedback. This work is supported by NSF CNS 1850297. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF.

## References

[1] "CAIDA Ark Datasets," http://www.caida.org/projects/ark/topo_datasets.xml.
[2] "PingER: Ping End-to-end Reporting," http://www-iepm.slac.stanford.edu/pinger/, 2015.
[3] "RIPE Atlas," https://atlas.ripe.net, 2018.
[4] A. Hernandez and E. Magana, "One-way Delay Measurement and Characterization," in *IEEE ICNS*, 2007.
[5] L. D. Vito, S. Rapuano, and L. Tomaciello, "One-way Delay Measurement: State of the Art," *IEEE TIM*, 2008.
[6] M. Shin, M. Park, D. Oh, B. Kim, and J. Lee, "Clock Synchronization for One-way Delay Measurement: A Survey," in *Advanced Communication and Networking*, 2011.
[7] A. Pasztor and D. Veitch, "PC-based Precision Timing Without GPS," in *ACM SIGMETRICS*, 2002.
[8] O. Gurewitz, I. Cidon, and M. Sidi, "One-way Delay Estimation using Network-wide Measurements," *IEEE/ACM TON*, vol. 14, no. SI, 2006.
[9] A. Vakili and J.-C. Gregoire, "Accurate One-way Delay Estimation: Limitations and Improvements," *IEEE TIM*, 2012.
[10] R. Durairajan, S. K. Mani, P. Barford, R. Nowak, and J. Sommers, "TimeWeaver: Opportunistic One Way Delay Measurement via NTP," in *ITC*, 2018.
[11] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "Netscope: Traffic engineering for ip networks," *IEEE Network*, vol. 14, no. 2, pp. 11–19, 2000.
[12] "Akamai SureRoute," https://developer.akamai.com/learn/Optimization/SureRoute.html.
[13] "ISI Ant Datasets," https://ant.isi.edu/datasets/index.html.
[14] "CRAWDAD Datasets," https://crawdad.org/.
[15] "M-lab datasets," https://www.measurementlab.net/data/.
[16] R. Padmanabhan, P. Owen, A. Schulman, and N. Spring, "Timeouts: Beware surprisingly high delay," in *ACM IMC*, 2015.

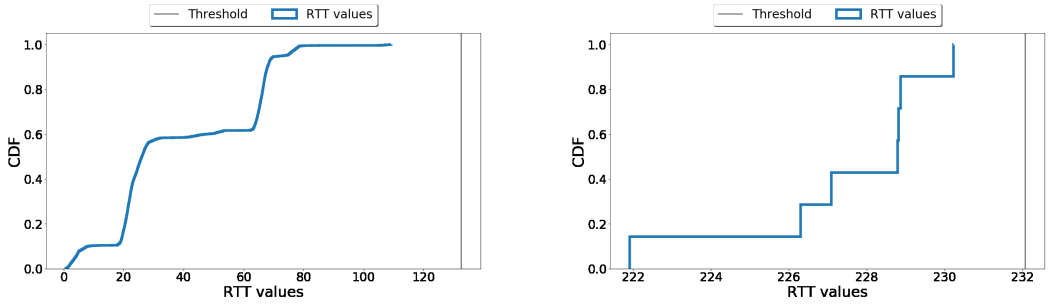


Fig. 5. CDF of RTT values for two sample clients demonstrating the limitations of NoMoNoise.

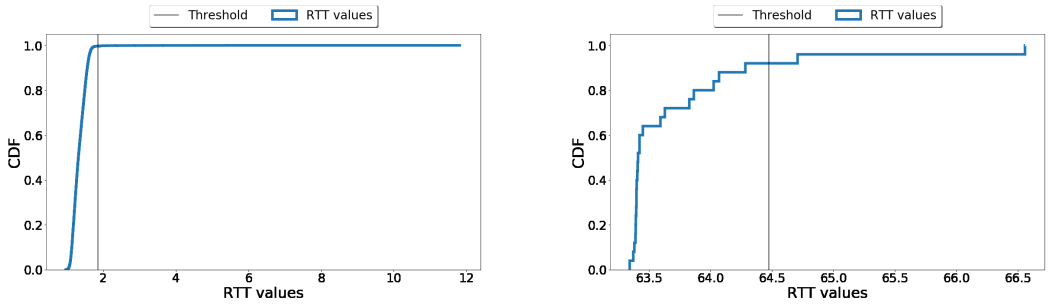


Fig. 6. CDF of RTT values for two sample clients where NoMoNoise is an effective denoising framework.


[17] T. Holterbach, C. Pelsser, R. Bush, and L. Vanbever, "Quantifying interference between measurements on the ripe atlas platform," in *ACM IMC*, 2015.

[18] V. Paxson, "End-to-end internet packet dynamics," in *ACM SIGCOMM CCR*, 1997.

[19] ——, "Strategies for Sound Internet Measurement," in *proceedings of ACM IMC*, 2004.

[20] W. Willinger, D. Alderson, and L. Li, "A pragmatic approach to dealing with high-variability in network measurements," in *ACM IMC*, 2004.

[21] J. Sommers, R. Durairajan, and P. Barford, "Automatic metadata generation for active measurement," in *ACM IMC*, 2017.

[22] R. Durairajan, S. K. Mani, J. Sommers, and P. Barford, "Time's Forgotten: Using NTP to understand Internet Latency," in *proceedings of ACM HotNets*, 2015.

[23] M. Syamkumar, S. K. Mani, R. Durairajan, P. Barford, and J. Sommers, "Wrinkles in Time: Detecting Internet-wide Events via NTP," in *proceedings of IFIP Networking*, 2018.

[24] "Cleaning big data: Most time-consuming, least enjoyable data science task, survey says," https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#791c6ed76f63.

[25] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," in *Advances in neural information processing systems*, 2016, pp. 3567–3575.

[26] "Tensorflow," https://www.tensorflow.org/.

[27] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76.

[28] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM CCR*, 2006.

[29] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *PAM*. Springer, 2004, pp. 205–214.

[30] G. Comarela, R. Durairajan, P. Barford, D. Christenson, and M. Crovella, "Assessing Candidate Preference through Web Browsing History," in *proceedings of ACM SIGKDD*, 2018.

[31] K. Xu and J. Chandrashekar and Z.L. Zhang, "A First Step toward Understanding Inter-domain Routing Dynamics," in *ACM SIGCOMM workshop on Mining network data*, 2005.

[32] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-wide Traffic Anomalies," in *ACM SIGCOMM*, 2004.

[33] ——, "Mining Anomalies Using Traffic Feature Distributions," *ACM SIGCOMM*, 2005.

[34] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández, "PCA-based Multivariate Statistical Network Monitoring for Anomaly Detection," *Computers & Security*, 2016.

[35] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and G. Iannaccone, "MIND: A Distributed Multi-Dimensional Indexing System for Network Diagnosis," in *IEEE INFOCOM*, 2006.

[36] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS*, 2005.

[37] R. Bunescu and R. Mooney, "Learning to extract relations from the web using minimal supervision," in *ACL*, 2007.

[38] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," in *IEEE SocialCom*, 2011.

[39] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Holoclean: Holistic data repairs with probabilistic inference," *VLDB Endowment*, 2017.

[40] "Snorkel: A System for Fast Training Data Creation," https://hazyresearch.github.io/snorkel/.

[41] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *VLDB Endowment*, 2017.

[42] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," *arXiv preprint arXiv:1710.11583*, 2017.

[43] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Security and Privacy*, 2010.

[44] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural computation*, 1998.

[45] M.E. Tipping and C.M. Bishop, "Mixtures of Probabilistic Principal Component Analyzers," *Neural computation*, 1999.

[46] H. Ringberg and A. Soule and J. Rexford and C. Diot, "Sensitivity of PCA for Traffic Anomaly Detection," *SIGMETRICS*, 2007.

[47] "Scamper warts2json tool," https://www.caida.org/tools/measurement/scamper/man/sc_warts2json.1.pdf.

[48] "Pytorch," https://pytorch.org.

[49] "Mean shift clustering," http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html.