Root Cause Analysis of Data Integrity Errors in Networked Systems with Incomplete Information

Yufeng Xin, Shih-Wen Fu, Anirban Mandal, Ilya Baldin
RENCI, UNC - Chapel Hill
Chapel Hill, NC, USA

Ryan Tanaka, Mats Rynge, Karan Vahi, Ewa Deelman
ISI, USC
Marina Del Rey, CA, USA

Ishan Abhinit, Von Welch
CACR, Indiana University
Bloomington, IN, USA

2021 International Conference on Information and Communication Technology Convergence (ICTC) | 978-1-6654-2383-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICTC52510.2021.9621124

***Abstract*—There are renewed interests recently in data integrity error detection and localization driven by exponentially growing data volumes over large-scale networked systems. Most existing RCA (Root Cause Analysis) systems take an infrastructure operator's view and rely on dedicated and expensive monitoring capabilities to instrument and facilitate the analysis. Unfortunately, in our targeted wide area network environment, complete network information and monitoring capability are normally lacking.**

In this paper, we present a RCA system that leverages the end-to-end flow monitoring information from the application layer, augmented by limited network information. We demonstrated that root cause localization with high accuracy can be obtained using multi-class classification models. We specifically studied the impacts of different realistic combinations of features based on the available yet incomplete information from both application and network layers.

I. Introduction

Root cause analysis (RCA) is a critical function in operating and managing complex networked systems, be it physical, software, or hybrid [1]. It aims to identify the component(s) and process(es) responsible for the fault manifested by the wrong results or system failures in a timely fashion. Traditional RCA relies on accurate system models to deduce the potential component failures from the system symptoms and behaviors. However, it proves impossible to build such models for efficient fault identification and localization in complex networked systems. As a result, RCA for those systems largely remains a guessing art that requires intensive manual debugging using traditional tools like *ping* and *traceroute* and daunting amount of communication between operators from different organizations that often takes days or even weeks.

In this paper, we focus on the root cause analysis of one important failure mode, data integrity errors, in large-scale networked systems. These systems typically run over the Internet or dedicated wide area networks that are managed by multiple middleware and infrastructure service providers in a distributed fashion. Due to the insufficient capability of existing reliable protocols (TCP, encrypted transfer and RAID, *et al.*), incomplete end-to-end integrity data coverage, and more possible component bugs, data integrity errors can stay stealth or be very difficult to be localized for long periods of time. Driven by the exponentially growing data volumes and the sheer scale of the network, detecting and localizing the data integrity errors have garnered lots of research and development interests in recent years [2], [3].

Data integrity errors carry the attributes of the so-called *gray failures* and *silent failure* in the networked system. This is in contrast to the so-called *hard failures* that cause data loss or corruption permanently. Recent studies showed that *gray failures* of probabilistic nature causing performance degradation in terms of packet losses and latency could be efficiently localized using machine learning (ML) approaches [4], [5]. The data integrity errors, on the other hand, may randomly corrupt bits in a block of data or packets over network transfer, which can evade the existing checksum mechanisms implemented in TCP and the storage system. Due to its low error probability and silent nature in which it may result in both performance degradation and just wrong results, detection and localization of data integrity error become very challenging. Modern middleware systems have just started to add end-to-end integrity check mechanisms, for example, Pegasus [6] and Globus [7], which efficiently addressed the detection problem.

As it is not possible to build a complete system model, machine learning has become the leading candidate to develop RCA systems for complex networks [8]. In [9], the authors attempted to identify the hard failures of network links via the popular multi-class ML models using end-to-end passive traffic engineering measurements (throughput, latency, and packet loss). The authors in [5] took an active probe approach to localize the fault in a virtual disk system to the finest granularity up to the network switches. In [10], a necessary condition was derived on the minimal set of paths that active probes need to be sent over the targeted network. Another line of work including [11], [12], [13] adopted statistical learning approach to infer the probabilistic relationship between the path failure and the link faults. All these research works made a strong assumption that the RCA system can instrument probes or obtain measurements from any pairs of nodes in the network to obtain both packet-level routing path and measurement information. In an earlier study, the decision tree model was used to predict if a request will succeed over a flawed network system [14]. Bayesian inference was demonstrated to be efficient for fast diagnosis when the causal relationship model is established in a large

This work was supported by the US National Science Foundation under Grant OAC-1839900.

Internet system [15]. A recent study focused on a source based measurement framework to diagnose the issues in the remote application services and therefore does not directly address the network components [16].

When developing an efficient RCA system, there are multiple dimensions in the design space. Different from the recent work in gray or hard failure RCA in a production data center or single domain network, we target the large scale networked systems where there are more limitations in obtaining complete wide-area network information. The accurate network topology and routing information is normally unavailable as the *ping* and *traceroute* are always turned off in many domains due to security concerns from the service providers [17]. It is also not realistic to deploy and operate a comprehensive probe instrumentation and measurement system in this typical multi-domain environment with continuous network coverage like in the data center networks [18].

In this work, we *take an application-centric view, rather than the infrastructure-centric view, to develop a machine learning RCA system to localize the data integrity errors*. Our basic idea is to formulate the targeted RCA problem as a multi-class classification problem, where the potential root causes can be classified using flow level measurements provided by the application layer, augmented by limited network information. And we assume only passive measurement of the flows between the end hosts is possible from the application layer.

We specifically use a workflow management system (WMS) [6], a popular Internet-scale distributed application, as the targeted system. WMS facilitates the in-order execution of jobs in workflows and includes large amounts of interdependent data transfers, storage functions, and computation tasks. These tasks are often distributed over distributed hardware, software, and data resources located in different facilities nationwide or globally. Inevitably, frequent system failures and reliability issues caused by errors and faults from underlying subsystems have been serious concerns for the WMS community.

The rest of this paper is organized as follows. We first define the network system integrity error RCA problem with incomplete system and measurement information in Section II. We present a machine learning solution approach and the model selection in Section III. We identified that using the network-wide aggregated data flow as the input, training data balancing, and a Top-k accuracy metric can significantly improve the classification accuracy. A high-fidelity system emulation environment we built in a cloud testbed is introduced in Section IV. The performance evaluation results are presented in Section V. The paper is concluded in Section VI.

II. RCA for Network Integrity Errors

A network system can be represented as a simple graph $G(V, E)$ where V is the set of nodes that includes H end hosts and R routers. E is the set of links. Due to scalability or privacy constraints, deploying monitoring capabilities at every point of the targeted large-scale network is not possible. Therefore, the problem can be concisely modeled as a bipartite mapping graph between the path level flows and its network components as shown in Fig. 1.

We further observe that one component failure (e.g., L_j) will cause multiple paths to be erroneous while one flow error (e.g., F_i) may be caused by multiple component failures. And because the component error probability is normally low ($10^{-3} - 10^{-6}$), it would require a large number of file transfers to catch a few corrupted files. Our ultimate goal is to localize the possible root causes in node or link errors by inference from the observed flow level abnormalities only at the end hosts, which can be translated to learning the mappings in such a graph.

In general, the fundamental network RCA problem is to use the path-level measurement to deduce the faulty components that can be concisely modeled by the formula 1. A path P_i that a file transfer (flow) i traverses consists of origin node H_i^o, a set of links where each $e_i^{\tilde{j}} \in E$ has two interfaces e_i^t and e_i^r on the route, and destination node H_i^d. As our main concern is if a file is corrupted rather than packet losses, the file characteristics, F_i, *e.g.*, the source and destination nodes, file size, transfer time, retransmission, etc. all become part of the data features. Some of the features are continuous variables and some are categorical variables. As the components ($\tilde{j}$) of P_i that file transfer i traverses is unknown except for its two end hosts H_i^o and H_i^d, the model needs to implicitly learn the file routes in order to infer the root causes. As a result, we can not use the traditional stochastic learning approach.

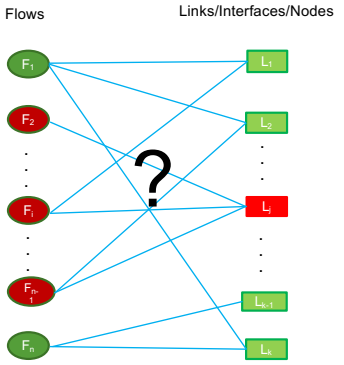

Fig. 1: RCA Bipartite Graph

$$Probability(File\ i\ corrupted) = f(F_i, \prod_{\tilde{j} \in P_i} Probability(device\ \tilde{j}\ is\ faulty)) \quad (1)$$

The majority of recent studies assume flow information between any pair of nodes in the network is available because they focus on data center networks where they have full ownership and their modern routers allow originating and receiving probing data. This is very important because, as shown in [10], there exists a minimal set of source-destination pairs to guarantee successful pinpointing of link errors in the network. They further assume the routes for all flows are known, i.e., the mapping represented in Fig. 1. However, both assumptions do not hold for our targeted Internet environment due to obvious administrative constraints, complex topology, and the lack of monitoring coverage. In the Internet-scale network, traffic routing and forwarding paths are hard to obtain because of frequent topology and policy changes, the widely adaptation of multi-path routing, and disabled *Ping* and

Traceroute in many places [13]. Furthermore, deployment of a monitoring infrastructure to conduct system-wide active and passive monitoring is administratively and costly prohibitive.

What makes the presented RCA problem uniquely challenging is that the flow routes, *i.e.*, the mapping relationships in (1) are unknown in our targeted network environment. However, the file level measurement and characteristics give us extra data features that could enhance the model accuracy, which is the case that we'll show in the evaluation.

In summary we made more restrictive but more realistic assumptions in this study in that (1) only the data file transfer information including integrity error states can be obtained at the end hosts from the application layer; (2) only the physical nodes (or abstract representation of domains) and their interfaces are known to us, but the network topology and traffic routing are unknown.

III. Machine Learning for RCA

We model our problem as a multi-class classification problem where the labels are defined as all the end host nodes and network interfaces that may incur integrity errors. The features are flow level characteristics that include source, destination, size, transfer time, throughput, whether a flow is corrupted, missed, or retried, etc. In general, the training process takes as input two arrays: an array X of size $[n_{samples}, n_{features}]$ holding the training samples, and an array y of class labels of size $[n_{samples}]$. The total number of labels L equals to the number of the link interfaces and the end hosts in the network.

We say a file transfer succeeds when it incurs no integrity errors. In this study, we focus on RCA analysis with failure data only, *i.e.*, the data sets used in training only contains those corrupted files transfers with fault labels that are extracted from the raw data collected from the experiment that consists of all data transfers. This non-probabilistic approach not only bears its own technical merit but also reflects a realistic situation where only failure events are reported in applications in order to reduce the measurement overhead and storage cost. Due to the low probability nature of the integrity errors, it is not economically viable to save all data transfer monitoring statistics as the majority of them are successful flows. This not only means that a large amount of data transfer flows with good network coverage are indispensable, but also suggests efficient fault injection mechanism is needed, to generate sufficient labeled training data.

There are a large number of different supervised ML models and associated parameter tuning methods. Our data sets introduce mixed numerical and categorical features as well as data imbalance. In addition, we want to obtain the class membership probability estimates, not just the most likely single class in the result. We evaluated several model variants using the popular Scikit-learn library [19], and found the random forest model results in the best performance in terms of RCA accuracy and training time. Decision tree is a natural choice to multi-class classification as the multiple leaves represent the labels. It supports a $predict\_proba$ method that gives the class membership probability estimates. Its main advantages include fast prediction time and excellent model explainability. In this study, we tried several ensemble methods based on randomized decision trees or random forests. By fitting over multiple randomized decision trees built from randomized samples, the random forest model with the right hyper parameters achieves higher accuracy and controls overfitting better.

A. Aggregated flows and inference accuracy

One key observation is that an erratic link may cause integrity errors on all data transfer paths traversing it. While the training data is collected in the form of individually labeled flows, the inference can be done in the unit of all flows that are corrupted at a time since we only consider the single failure scenario. The models are trained with the labeled flow data. For inference, we consider two different methods: *Flow* that just uses individual flow as the input and *Aggregated Flow* for which we generated a new data set that all corrupted file transfer flows at a time are aggregated as one input data sample. In the former case, the accuracy is computed on a per flow base. In the latter case, if all the flows in a data set are labeled by L labels, they will be aggregated into L samples to be tested against the trained model. The total number of correct label inference divided by L is defined as the accuracy.

B. Top-k classification accuracy

Since we assume training data from data transfer flows only between the end hosts, it doesn't satisfy the necessary condition presented in [10]. The conventional classification on a single label inference from the training models performs relatively poorly in terms of accuracy and F-score. In practice, it would be very useful if the model can produce a small set of highly likely causes for the operators to zoom in. Most of the ML models, when used to infer a test sample, actually generate the probability distribution over all the classes. Therefore we can use a Top-k Accuracy metric in evaluation, for which a prediction is defined as correct as long as the set of k labels of highest probability in the classification results contains the correct label of the sample. Both decisions tree and BN models natively support the classification probability output.

C. Features

As explained on Equation 1, for a data sample, the feature set could consist of both path features and file transfer features. In our model, we only consider the features that are possible for the application to collect at the end hosts. So only the end host information is included for the path features because we assume the other network elements on the file transfer paths are unknown. The file transfer features include both numerical characteristics like file size and transfer throughput , and categorical features like correctness of integrity checksum and presence of retransmission.

The impacts of the file transfer features are two-folded. On one hand, the bigger file size may incur a higher probability of file corruption and lower throughput may imply more TCP retransmission caused by corrupted packets, which may help with the RCA performance. On the other hand, different

machine learning models perform differently when dealing with a mixture of numerical and categorical features. In reality, there are always engineering and policy limits on obtaining certain features for application users in a network. Therefore it is important to study the model performance when only a subset of features are available. Therefore we study two scenarios of different feature sets: *No File Features* when the numerical file size and transfer throughput information is not available and *All Features* when it is available.

D. Error Asymmetry and Data Imbalance

A leading factor that affects the performance of multi-class classification models is the data set imbalance. When data samples from certain classes (called *majority classes*) outnumber those from other classes, the trained models will be highly skewed toward the majority classes, which will significantly lower the prediction accuracy. By the nature of integrity error simulation, the file transfer failures caused by faulty network interfaces are more frequent than those caused by the faulty end hosts. And between the two interfaces on a link, the one on the receiving side of a file transfer over TCP has a much higher chance to corrupt the file than the one on the transmitting side. Therefore the raw data we collected is oversampled on a subset of the network interface classes and significantly undersampled on the end host classes.

E. Classification Granularity

The wide area network system may cover multiple administrative domains. For example, a subset of core routers or border routers may belong to one service provider and another subset of routers and end hosts may belong to a campus site or a data center. It would be very useful if RCA can localize the failure to a particular domain accurately so the domain administrators can further locate the faulty components with more powerful debugging tools. For the classification model training and inference, this means to aggregate multiple related labels to form *super labels* according to certain criteria. This idea could lead an efficient multi-granularity classification framework, which is not difficult to implement on top of the base model, but could be very appreciated in reality. As pointed out by the reference, inefficiency of traditional RCA approaches largely comes from the so called *blame game*, *i.e.*, back and forth communication and reasoning between administrators from different domains to identify who is responsible for the tedious debugging in her domain.

IV. Network Emulation and Fault Injection

In order to obtain sufficient labeled training data and make experiments efficient and repeatable to study the proposed RCA system, we created a high-fidelity emulation environment in the NSF ExoGENI cloud testbed [20] that can automatically create a virtual network system with virtual machines (VMs), bootstrap the network routing, initiate data transfers, inject arbitrary integrity errors into the virtual router interfaces and end hosts, and collect training data.

To inject controlled errors into any network elements, we added a new utility in Chaos Jungle [3], [21], which can corrupt arbitrary file(s) in nodes in addition to packets out of network interfaces according to a given probability. In this way, labeled data can be generated to train the machine learning models.

At the last step, all the raw data will be processed and stored in the final result database files with predefined feature columns. Each database entry represents one data transfer with features of the file name, file size, origin, sink, access router, integrity error or not, etc. However, the forwarding path is unknown as it is controlled by the routing control plane process. The final result is exported to a Jupyter notebook environment where all the ML-based data analysis is performed.

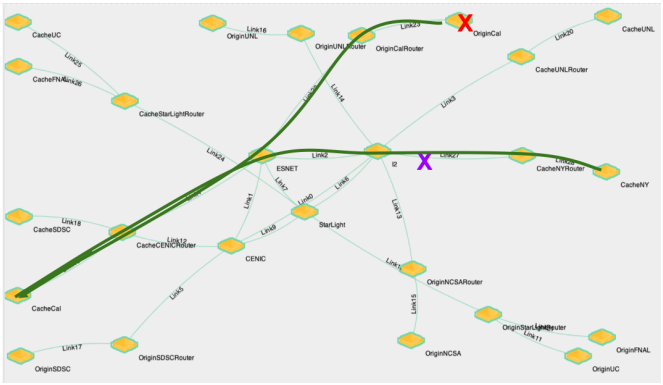

Fig. 2: Fault Injection in an Emulated Network

We use Fig. 2, an annotated screenshot from the ExoGENI GUI, to illustrate an emulated network system, a simplified version of the Open Science Grid (OSG) data federation network, a highly utilized wide area network to distribute data for high throughput scientific computing [22]. This network consists of multiple end hosts running the data transfer and computing jobs controlled by Pegasus WMS [6], and a set of routers in the middle to represent multiple forwarding domains. The unit of data transfers is a *file* of different size for which we define as a *flow* from an origin end host to a destination end host. The two cross signs, one on an end host and another one on a network link, represent the locations where we inject the data integrity errors with a probability setting. When an error is enabled (injected), all the traffic flows that pass through the faulty element will be subject to possible data corruption as the error may flip the bits of packets randomly under a predefined probability. As we have discussed, most of these corrupted packets will not be caught by TCP or the storage system. As a result, some corrupted files will successfully land at the destination end hosts and will only be detected by Pegasus at the application layer.

This emulation environment can be used to create a sandbox to generate high-fidelity RCA models for a production networked system.

V. Experiments and Evaluation

We emulated a network following the power law, i.e., 4 routing nodes in the middle to emulate the backbone domains and the rest emulate the access domains in between the backbone nodes and the end hosts.

Among the 23 nodes in this topology, we specify 6 data origins and 6 data sinks as the end hosts to transfer a batch of data files of different sizes that we randomly acquired from OSG. This file batch is transferred between every chosen pair of end hosts. We ran two sets of experiments to collect two sets of raw training data. In the first one, called *Partial*, data transfers only happen between the origins and sinks, where every origin node sends all the files in the batch to all the receiving nodes in parallel. In the second one, called *Complete*, data transfers happen between all the end host pairs. The file integrity is being checked at the receiving end host and each file transfer accounts for one data flow and therefore a data sample in a training data set. We further parallelized the data transfer process to reduce the emulation time down to about twenty-four hours for this particular network.

For each experiment, probabilistic integrity error or network impairment via the Chaos Jungle tool is injected to the 54 link interfaces and 12 end hosts in sequence with the given probability setting. For each fault injection scenario, the entire set of *Partial* or *Complete* data transfers are conducted. Each link interface or node component with fault injected represents a label. The receiving node checks if a received file is identical to its original copy via checksum and marks this data transfer as a failure data sample if checksums do not match. We treat retransmission as a separate feature for the data samples. A file could also be missed at the destination due to ultimate transfer failure which is also treated as a failure. If the checksum matches, this data transfer becomes a positive data sample in the training data set. Otherwise, it is labeled by the corresponding faulty element. The final *Complete* training data set consists of 8M data samples.

Each figure shows five different performance metrics under four scenarios. The five metrics are F1 score with per-flow inference, the accuracy with per-(*Flow*) inference, and the $Top-1$, $Top-2$, and $Top-3$ accuracy with the *Aggregated Flow* inference. The four scenarios are *Partial* and the *Complete* data sets with *All Features* or only *No File features*.

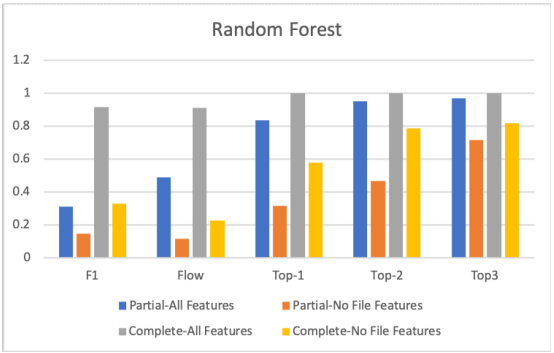

Fig. 3: Classification Accuracy with Random Forest Model

Fig. 3 presents the results from training random forest models with different data sets. Two prominent observations stand out. First, it clearly shows that the model with *Complete* data performs significantly better than the *Partial* case. This illustrates the importance of sufficient coverage of the network path information in the training data set. It also shows that, even without the full coverage of network paths between the network routers, the flows between end hosts alone can guarantee very high RCA inference accuracy. Secondly, training with *All Features* outperforms its *No File Features* counterpart by a large margin. This means the file transfer statistics can boost the inference performance. We also learned that the file transfer throughput has a bigger impact than the file size, though the result is not shown here.

When we zoom into more details, we can see that the combined *Complete-All Features* data set presents satisfactory performance even with the single flow-based testing data samples with both F1 score and accuracy reaching above 0.90. When the aggregated flow data samples are used for inference, the accuracy scores perfect 1. The next best scenario is when *All Features* presented with *Partial* data transfer, while the single flow-based inference only achieved under 0.5 accuracy, the aggregated flow-based inference doubles the accuracy and quickly narrows down the root cause to the top 2 elements. However, without more path coverage, it doesn't achieve perfect accuracy until $Top-10$. For the next two scenarios, the aggregated flow inference helps achieve better performance and *Complete* path coverage appears more important than the inclusion of file transfer features. However, the best it can achieve is a mere 80% $Top-3$ accuracy.

In order to gain more insight into the classification accuracy performance, we observed that the majority of mislabeled data in the classification inference are those with labels of end host faults in all the scenarios whose accuracy is less than 1. This is because the raw training data set is highly imbalanced due to the different impacts of the injected failures on the data files being transferred (Section III-D). There are significantly fewer integrity errors caused by the faulty end hosts, *i.e.*, significantly less labeled data in these classes.

The main techniques to solve the dataset imbalance problem are to rebalance the data via oversampling or downsampling data from different classes. Since the labeled data subsets from the end host failure classes are rather small, the pure downsampling techniques will not improve on the model performance. We focus on two representative oversampling techniques: random and SMOTE as well as two methods that combine oversampling and downsampling in SMOTETomek and SMOTEENN [23].

For the *Complete-All Features* data set, our analysis indicated that both Random and combined SMOTE methods resulted in similar perfect $Top-k$ performance but slightly reduced flow-based F1 Score and accuracy. SMOTE deteriorated all the metrics. The observations on the *Complete-No File Features* data set are similar. Since the performance of the original model is already perfect for the former case, we only present the results on the two *Partial* data sets.

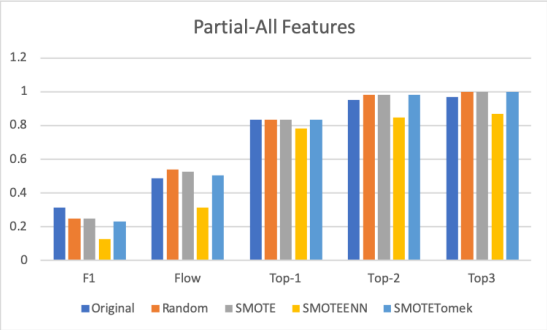

Fig. 4: Oversampling with All Features

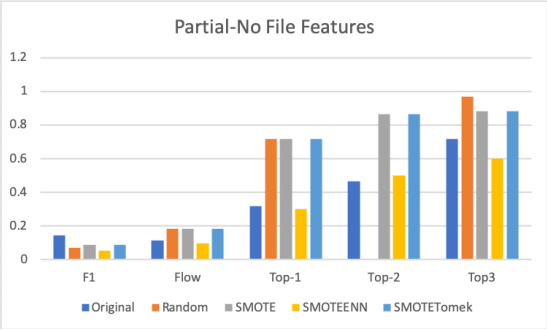

Fig. 5: Oversampling with No File Features

The results in Fig. 4 and 5 show that the three methods other than the SMOTEENN improve the performance to some extent, especially on the $Top-k$ accuracy measurement. The basic random oversampling demonstrates the best performance improvement in most cases. The main takeaway is that multiple sampling methods have to be carefully evaluated in order to find the best one.

We also evaluated the multi-granularity classification scheme discussed in Section III-E. For the simulated system, we combine each access router and the end hosts attached to it into one domain and aggregate the related labels into a super label. A random forest model can be trained to achieve 100% accuracy with the basic aggregated flow inference.

VI. Conclusions and Future Work

We developed a machine learning based network integrity error RCA system that leverages the end-to-end flow monitoring information from the application layer, augmented by limited network information. The impacts of different combinations of numerical and categorical data features under different realistic network and measurement assumptions on the inference accuracy are quantified via an emulated network created in an automated high-fidelity emulation environment we built. The analysis validated that high RCA accuracy to the device level can be achieved with an efficient supervised learning model even when only partial network and flow level measurement information are available.

For our future work, we plan to study networks of larger scales with different topology characteristics, multiple concurrent faults, more finely tuned ML models and possible integration with limited network monitoring information. We will further explore the multi-granularity classification framework and stochastic approaches that leverage the probability distribution characteristics of the network failures.

References

[1] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," *ArXiv 1701.08546*, 2017.

[2] J. Stone and C. Partridge, "When the crc and tcp checksum disagree," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, Aug. 2000.

[3] M. Rynge, K. Vahi, E. Deelman, A. Mandal, I. Baldin, O. Bhide, R. Heiland, V. Welch, R. Hill, W. L. Poehlman, and F. A. Feltus, "Integrity protection for scientific workflow data: Motivation and initial experiences," in *Practice and Experience in Advanced Research Computing on Rise of the Machines Learning (PEARC)*, New York, 2019.

[4] P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The achilles' heel of cloud-scale systems," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. New York, NY, USA: ACM, 2017.

[5] Q. Zhang, G. Yu, C. Guo, Y. Dang, N. Swanson, X. Yang, R. Yao, M. Chintalapati, A. Krishnamurthy, and T. Anderson, "Deepview: Virtual disk failure diagnosis and pattern detection for azure," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Renton, WA, Apr. 2018.

[6] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus: a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[7] E.-S. JUNG, R. KETTIMUTHU, and S. CHUNG, "High-performance end-to-end integrity verification on big data transfer," *IEICE TRANSACTIONS on Information and Systems*, vol. E102-D, no. 8, 2019.

[8] R. Boutaba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications, 9(1)*, 2018.

[9] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *IEEE Internet of Things Journal*, 2019.

[10] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "Netbouncer: Active device and link failure localization in data center networks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, Boston, MA, 2019.

[11] H. Herodotou, B. Ding, S. Balakrishnan, G. Outhred, and P. Fitter, "Scalable near real-time failure localization of data center networks," in *20th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA, 2014.

[12] Y. Peng, J. Yang, C. Wu, C. Guo, C. Hu, and Z. Li, "detector: a topology-aware monitoring system for data center networks," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA: USENIX Association, Jul. 2017.

[13] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, April 2018.

[14] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *International Conference on Autonomic Computing, 2004.*, May 2004.

[15] J. C. Platt, E. Kiman, and D. A. Maltz, "Fast variational inference for large-scale internet diagnosis," in *20th International Conference on Neural Information Processing Systems (NIPS)*, USA, 2007.

[16] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "Microrca: Root cause localization of performance issues in microservices," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.

[17] T. Hou, Z. Qu, T. Wang, Z. Lu, and Y. Liu, "Proto: Proactive topology obfuscation against adversarial network topology inference," in *IEEE Conference on Computer Communications (Infocom)*, 2020.

[18] C. Guo, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *SIGCOMM*. ACM, August 2015.

[19] "Scikit-learn," https://scikit-learn.org/stable.

[20] "Exogeni website and wiki," http://www.exogeni.net.

[21] "Chaos jungle," https://github.com/RENCI-NRIG/chaos-jungle.

[22] "Open secience grid (osg)," https://opensciencegrid.org.

[23] "Imbalanced-learn," https://imbalanced-learn.org/stable/.