Causation mining in network logs

Satoru Kobayashi
University of Tokyo
sat@hongo.wide.ad.jp

Kensuke Fukuda
NII/Sokendai
kensuke@nii.ac.jp

Hiroshi Esaki
University of Tokyo
hiroshi@wide.ad.jp

ABSTRACT

Syslog is a valuable information source to detect unexpected or anomalous behavior in a large scale network. However, pinpointing failures and their causes is not an easy problem because of a huge amount of system log data in daily operation. This preliminary study discusses a method extracting failures and their causes from network syslog data. The main idea of the method relies on causal inference that reconstructs causality of network events from a set of the time series of events. Causal inference allows us to reduce the number of correlated edges by chance, thus it results in better accuracy than a traditional cross-correlation based approach. We demonstrate a potential ability of the causal inference algorithm with a case study, applying it to real network syslog data.

1. INTRODUCTION

Maintaining a large-scale network reliably has been a fundamental requirement in network management. It is however not an easy task in reality because of highly distributed, ever-evolving, and heterogeneous nature of operational networks [2]. One of the effective ways to track network status is to deploy monitoring agents in the network and collect log information corresponding to a change of status. In operational networks, syslog [1] is widely used for such purpose. The detailed log messages allow us to better understand failures and their causes. Nonetheless, it is usually hard for network operators to identify them because of a large number of system log messages produced by a large set of network devices (e.g., routers, switches, and servers).

To this end, various approaches have been taken for improving network monitoring and diagnosis with log messages. A simple approach is clustering log messages related to a network event (e.g., failure) into a correlated group, and analyzing the group in detail. One of the problems of log analysis is that co-occurrence of log messages does not always mean causal relations. Some existing works achieve contextual analysis by inferring causal relations among events in system logs [5, 6, 12]. However, appearance of network log messages is discrete and sparse; it makes us difficult to identify causality of events even with these approaches.

This study aims at extracting causal relations beyond co-occurrences in log messages in order to identify important network events and their causes. Causal relations in log messages help detect root causes of trouble, or enable further analysis leveraging on causalities [8,9]. There are some issues when applying cause inference algorithms to the network logs; (1) A large-scale network is composed of multiple vendor's devices, and various types of messages appeared in the network. (2) Occurrence of messages is discrete and sparse that is not assumed in causal inference algorithms. (3) All of the detected causalities are not necessarily important, in the context of network management.

To overcome these issues, we discuss a mining algorithm built on causal inference. We apply our proposed algorithm to syslog messages collected from a R&E network in Japan [11]. We obtain a reasonable number of causal relations with our approach. With case studies, we demonstrate the effectiveness of our approach.

2. METHODOLOGY

2.1 PC algorithm

The key idea of our proposal is to detect causality of two given events in network logs. The causality is a special case of the co-occurrence defined by a positive correlation coefficient. A popular approach to detect the causality is to remove pseudo correlation with conditional independence. Many causal inference algorithms assume a direct acyclic graph (DAG) of events corresponding to the causality of events. Root causes of events appear in such DAGs.

We leverage PC algorithm [3,10] to generate causality graphs, DAGs of causal relations, from system logs. PC algorithm investigates causal inferences among nodes (e.g., events) efficiently based on a set of event time series. The causal inferences are estimated with the idea of conditional independence.

PC algorithm consists of four steps.

1. Construct a complete (i.e., fully-connected) undirected graph from nodes (events).

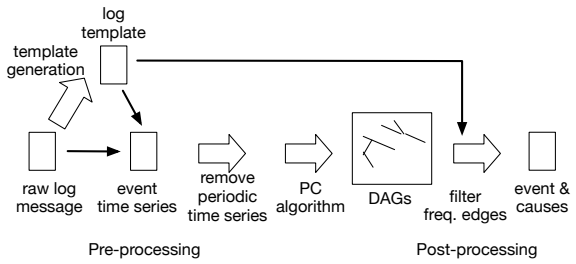

Figure 1: Processing flow

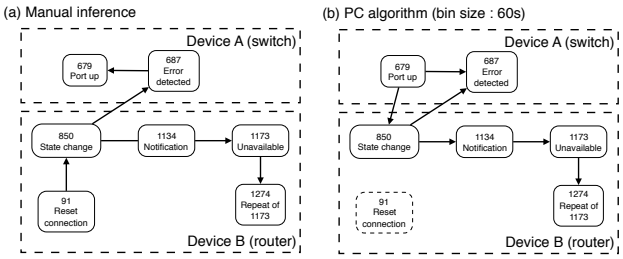

Figure 2: Ground truth and detected causalities

2. Detect and remove edges without causality by checking conditional independence.

3. Determine edge direction based on V-structure; this is a rule for deciding directions of three connected nodes.

4. Determine edge direction with orientation rule; this is a rule for generating DAGs.

It is note that undirected edges can be remained when enough information for the tests is not available in PC algorithm.

2.2 System model

We apply PC algorithm to a set of event time series to generate causality graphs, DAGs with nodes of events and edges of causalities. In our approach, we define an event as a set of log messages that have a common log template that is a skeleton of log messages without variables (e.g., IPs, interface names).

Figure 1 shows processing flow of our log mining algorithm. First, we extract log templates from the original log messages to classify log messages into events. We employed a machine-learning-based log template algorithm [4]. Next, we construct a set of event time series by each log template per each device (router or switch). Then, we apply PC algorithm to a set of event time series to generate causality graphs. For evaluating the conditional independence of events, we perform a statistical test called G-square test [7], a natural extension of Chi-square test for the data. Finally, we remove frequently appeared edges from the DAGs with a threshold for easily identifying important causality.

Because of the limitation of G-square test, the event time series are generated as binary data. In addition, we remove a large number of unrelated event time series indicating strong periodicity so as to decrease false positives of coincident. These events are detected with an auto-correlation coefficient with lag of 1 hour or 24 hours for the events.

3. CASE STUDY

To evaluate the effectiveness of our approach, we use a set of backbone network logs obtained at a R&E network in Japan (SINET4 [11]). The nation-wide network consists of eight core routers, 60 edge routers, and 100 layer-2 switches composed of multiple vendors. The network consists of a large number of network devices, so we divide the data into eight subsets corresponding to a sub network with one core router, edge routers and switches that connected to the core router. We also analyze one-day-long log data because we target short-term causality instead of long-term one.

We analyze 15-months consecutive logs composed of $35,513,125$ log messages. These log messages are classified into events with $1,414$ log templates. The preprocessing reduced the number of messages to $1,483,455$. Finally, we detect $8,613$ edges from this dataset, and $1,548$ (3.4 edges/day) of them are identified as important edges after the post-processing.

These causal edges include some useful knowledge for troubleshooting. Here, we provide one example; an interface error on a router yields repeated BGP peering connections. It is not easy for operators to find the relation of these events, because the events occur over multiple devices with a certain time lag. Figure 2 shows (a) a plausible ground truth of this failure by manual inspection and (b) generated causality graphs with PC algorithm. Compared to the ground truth, PC algorithm reasonably detects causal relations though some directions of the edges are different from our intuition.

4. CONCLUDING REMARKS

In this paper, we propose a method to mine causality of network events from network log messages. The key idea of the work is leveraged on PC algorithm that reconstructs causal structures from a set of time series of events. We demonstrate the effectiveness of the causal inference algorithm in network syslog analysis by illustrating a case study.

We intend to extend the study to evaluate with a ground truth data (e.g., trouble ticket data). Another direction of the work is to integrate the timestamp information in the original logs in order to obtain more reliable results. Also, we consider to apply the proposed methodology to other network system such as data center or cloud system.

5. REFERENCES

[1] R. Gerhards. The Syslog Protocol. RFC 5244, 2009.

[2] R. Govindan, I. Minei, ahesh Kallahalla, B. Koley, and A. Vahdat. Evolve of die: High-availability design principles drawn from google's netwwork infrastructure. In *ACM SIGCOMM*, pages 58–72, 2016.

[3] M. Kalisch and P. BÃijhlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. In *The Journal of Machine Learning Research*, volume 8, pages 613–636, 2007.

[4] S. Kobayashi, K. Fukuda, and H. Esaki. Towards an NLP-based log template generation algorithm for system log analysis. In *Proceedings of The Ninth International Conference on Future Internet Technologies - CFI '14*, pages 1–4, New York, New York, USA, 2014.

[5] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large iptv network. In *ACM SIGCOMM*, pages 231–242, 2009.

[6] K. Nagaraj, C. Killian, and J. Neville. Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation - NSDI'12*, pages 1–14, 2012.

[7] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall Upper Saddle River, 2004.

[8] C. Scott, A. Panda, A. Krishnamurthy, V. Brajkovic, G. Necula, and S. Shenker. Minimizing Faulty Executions of Distributed Systems. In *Preceedings of 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 291–309, 2016.

[9] C. Scott, S. Whitlock, H. Acharya, K. Zarifis, S. Shenker, A. Wundsam, B. Raghavan, A. Panda, A. Or, J. Lai, E. Huang, Z. Liu, and A. El-Hassany. Troubleshooting blackbox SDN control software with minimal causal sequences. In *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, pages 395–406, 2014.

[10] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9:62–72, 1991.

[11] S. Urushidani, M. Aoki, K. Fukuda, S. Abe, M. Nakamura, M. Koibuchi, Y. Ji, and S. Yamada. Highly available network design and resource management of sinet4. *Telecommunication Systems*, 56:33–47, 2014.

[12] Z. Zheng, L. Yu, Z. Lan, and T. Jones. 3-Dimensional root cause diagnosis via co-analysis. In *Proceedings of the 9th international conference on Autonomic computing - ICAC '12*, page 181, 2012.