Detecting Configuration Errors Via Pattern Mining

Jyotirmay Chauhan
Colgate University
jchauhan@colgate.edu

Devon Lee
Colgate University
dlee@colgate.edu

Emily Yu
Colgate University
eyu@colgate.edu

Aaron Gember-Jacobson
Colgate University
agemberjacobson@colgate.edu

I. Introduction

Researchers have developed numerous tools for detecting errors in network configurations. Many tools construct a model of a network's control plane—based on configurations, routing protocol standards, and vendor documentation—and check whether the model conforms to operator-specified forwarding policies—e.g., reachability and waypointing requirements. However, designing the model and enumerating a set of policies to check are both complex and error-prone tasks [2, 3].[1]

A few error detection tools have avoided these issues by focusing on the content and structure of the configurations themselves. For example, rcc [5] checks whether BGP configurations conform to common best practices, Minerals [7] infers association rules for interface and BGP configurations, and SelfStarter [6] infers templates for packet and route filters. The latter two embrace the "outliers are bugs" philosophy [4], and flag deviations from rules or templates as errors.

Based on our examinations of numerous networks' configurations, arduous attempts to mine networks' specifications, and conversations with network engineers, we wholeheartedly support applying the "outliers are bugs" philosophy to network configurations. However, we have discovered that network configurations contain even more subtle patterns than these prior works [6, 7] elucidate.

In this talk, we present illustrative examples of the subtle patterns embedded in network configurations, and we discuss our efforts to automatically detect (deviations from) these patterns using contrast set learning and link prediction.

II. Subtle Configuration Patterns

Through copious manual examination of the configurations from seven campus networks and two research & education (R&E) networks, we have uncovered three classes of subtle configuration patterns that are overlooked by prior works [6, 7]. The first half of our talk describes each class of patterns and provides illustrative examples drawn from the configurations of three different networks—although such patterns occur in many of the networks we examined.

Keywords. We first observe patterns in the "human readable" names and descriptions in configurations. For example, one network allocates a network management VLAN for each building, and includes the keyword "management" in the description of such VLANs. All devices with such VLANs also contain an ACL whose description contains the same keyword; this ACL is applied to (almost) all VLANs with the keyword and no VLANs without the keyword. Furthermore, the keyword does not appear in the names/descriptions of any other ACLs or categories of VLANs. Hence, this keyword can be used to discern whether a specific ACL is applied to the appropriate VLANs.

Reference counts. Benson et al. introduced the idea of inter- and intra-device configuration references [1]: e.g., the application of an ACL to a VLAN creates an intra-device reference from the VLAN to the ACL. We observe that the frequency of references to certain entities indicates the entity's role and the way in which we expect it to be used. For example, in one network, the layer-3 links between core and distribution routers are each assigned a unique VLAN. Each VLAN is only allowed on the two interfaces that physically connect a pair of routers. In contrast, all other VLANs are allowed on several interfaces: e.g., a department's VLAN is allowed on interfaces that connect to department switches as well as physically connected interfaces on core and distribution routers. Furthermore, OSPF is only active on the aforementioned category of VLANs. Hence, the number of interfaces that refer to a VLAN can be used to discern whether the VLAN should run OSPF.

Mutual references. We also observe patterns based on the presence of mutual references. For example, one network defines a unique ACL for each VLAN containing end hosts. The ACL only accepts packets whose source IP address falls within the subnet assigned to the VLAN, in order to prevent source spoofing. Thus, there is a reference from the VLAN to the ACL, and both the ACL and the VLAN have a reference to the same subnet. Hence, if we observe a VLAN that references an ACL, but the VLAN and the ACL do not refer to the same subnet, then we have detected an error. This class of patterns is similar to the notion of "mutual friends" in a social network.

These illustrative examples suggest there are likely other subtle patterns embedded in configurations, and uncovering these classes of patterns through manual and/or automated analysis is part of our ongoing work.

III. Pattern Mining

The second half of our talk discusses the techniques we are developing to detect patterns such as those mentioned in the first half of the talk. Our efforts our focused on two pattern mining techniques: contrast set learning are link prediction.

Contrast Set Learning (CSL). CSL is a form of association rule mining which attempts to identify meaningful differences

[1] Mined specifications may be incorrect due to configuration errors.

between separate groups. These differences can take the form of a variety of identifying attributes present in networks: e.g. subnet addresses, keywords, types/names of ACLs applied, etc. The guiding principle behind CSL is that a group of these attributes adjusted to the right threshold can help distinguish between different groups of entities. For example, the inclusion of the keyword "management" in a VLAN's description is a meaningful difference between VLANs that have a specific ACL applied versus those that do not. Any deviations from this rule—e.g., a VLAN with the keyword in its description but no ACL applied—can be flagged as potential errors.

The main challenges in CSL is identifying meaningful rules. Our implementation of CSL automatically ranks rules based on precision and recall, but two rules with the same precision and recall may not be equally meaningfully. For example, a rule based on a specific VLAN's subnet address is less meaningful than a rule based on keyword which applies to multiple VLANs. An additional challenge is determining which features to extract from configurations. Currently, we are "hand-engineering" features based on domain knowledge, but we plan to investigate ways to automatically extract configuration features—e.g., based on embeddings [8].

Link prediction. This method represents a router configuration as a graph, where different types of nodes represent different configuration components, and links define relationships between components [1]. For example, given VLAN 10 with the ACL 2 applied, our graph would contain an VLAN node representing VLAN 10 with a link labeled "out" to an ACL node representing ACL 2.

We predict missing links based on node similarity. Similar to friend suggestion algorithms used by social media networks, we calculate the similarity of two nodes as a weighted average of the number of neighbors they have in common, with the percent similarity of each type of node (i.e. interface, VLAN, subnet, ACL) weighted according to a hyperparameter. The computed similarity of two nodes is compared against a threshold (another hyperparamter), and if nodes are sufficiently similar, we predict links should be added to all neighbors of one node that are not neighbors of the other. For example, if two VLANs have all of the same keywords, but one VLAN applies an ACL that the other does not, then we predict a link should be added from the other VLAN to the ACL.

The primary challenges with this approach are: tuning the hyperparameters, generating suggestions for missing links for node pairs with low similarity, and predicting links to remove. We are currently working on solving these challenges.

IV. Conclusion

In summary, our analysis of network configurations indicates they are rife with patterns that can exploited to flag configuration errors. Contrast set learning and link prediction are two promising approaches for detecting these patterns, and we are actively working to address the challenges associated with these approaches.

References

[1] T. Benson, A. Akella, and D. Maltz. Unraveling the complexity of network management. In *Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[2] R. Birkner, T. Brodmann, P. Tsankov, L. Vanbever, and M. T. Vechev. Metha: Network verifiers need to be correct too! In *NSDI*, 2021.

[3] R. Birkner, D. Drachsler-Cohen, L. Vanbever, and M. T. Vechev. Config2spec: Mining network specifications from network configurations. In *NSDI*, 2020.

[4] D. Engler, D. Y. Chen, S. Hallem, A. Chou, and B. Shelf. Bugs as deviant behavior: A general approach to inferring errors in systems code. In *SOSP*, 2021.

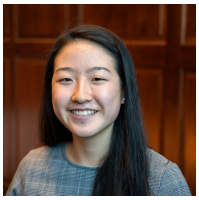
[5] N. Feamster and H. Balakrishnan. Detecting BGP configuration faults with static analysis. In *NSDI*, 2005.

[6] S. K. R. K., A. Tang, R. Beckett, K. Jayaraman, T. D. Millstein, Y. Tamir, and G. Varghese. Finding network misconfigurations by automatic template inference. In *NSDI*, 2020.

[7] F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb. Detecting network-wide and router-specific misconfigurations through data mining. *IEEE/ACM Trans. Netw.*, 17(1):66–79, 2009.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

Jyotirmay Chauhan is a Junior undergraduate student at Colgate University double majoring in Computer Science and Mathematical Economics. He has conducted research in network verification since May 2020. Jyotirmay is a residential community leader, an analyst in Colgate's Investment group, and previously worked as a consultant in Colgate's Information Technology Services.



Devon Lee is a Senior undergraduate student at Colgate University majoring in Computer Science and minoring in Geography. She has conducted research in network verification since May 2021. Devon is co-president of Colgate's Women in Computer Science Club and has been a computer science teaching assistant for Colgate's Introduction to Computing course since January 2020.

Emily Yu is a Junior undergraduate student at Colgate University majoring in Computer Science and minoring in Peace & Conflict Studies and Educational Studies. She has conducted research in network verification since May 2021. Emily is a computer science teaching assistant for Colgate's Introduction to Computing course and previously was a Technical Content Developer for the Bit Project, which provides cutting-edge open source learning experiences to prepare students for careers in tech.

Aaron Gember-Jacobson is an Assistant Professor of Computer Science at Colgate University. He received Ph.D. (2015) and M.S. (2011) degrees in Computer Science from the University of Wisconsin-Madison and a B.S. (2009) in Computer Science from Marquette University. Aaron's research focuses on efficiently detecting, localizing, and correcting errors in network configurations, especially through the use of graph-based models. Aaron enjoys teaching, especially his First Year Seminar course called 'The Unreliable Internet.'