Synthesis of Fault-Tolerant Distributed Router Configurations

Kausik Subramanian
University of Wisconsin-Madison
Madison, WI, USA
sskausik08@cs.wisc.edu

Loris D'Antoni
University of Wisconsin-Madison
Madison, WI, USA
loris@cs.wisc.edu

Aditya Akella
University of Wisconsin-Madison
Madison, WI, USA
akella@cs.wisc.edu

CCS CONCEPTS

• **Networks → Network manageability**; **Routing protocols**;

KEYWORDS

Zeppelin; Synthesis; Fault Tolerance; Network Management; Routing protocols; Hierarchical network control plane

ACM Reference Format:
Kausik Subramanian, Loris D'Antoni, and Aditya Akella. 2018. Synthesis of Fault-Tolerant Distributed Router Configurations. In *SIGMETRICS '18 Abstracts: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems Abstracts, June 18–22, 2018, Irvine, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3219617.3219652

ABSTRACT

Programming networks to correctly forward flows according to user- and application-induced policies is difficult and error-prone [9, 17]. At least three common characteristics of network policies are to blame: (1) A network may need to satisfy several types of policies, including reachability (i.e., which endpoints can communicate), isolation (i.e., which flows cannot share links), service chaining (i.e., which "middleboxes", e.g., firewalls or load balancers, must be traversed), resilience (e.g., number of available backup paths), and traffic engineering (e.g., minimizing average link utilization). (2) The network must provide certain guarantees in the event of failures. Ideally, every set of forwarding paths (i.e., data plane) installed in the network should conform to the above policies, otherwise performance, security, or availability problems may arise. (3) Most policies are global—i.e., they concern end-to-end paths, not individual devices/links.

The global nature of network policies is one motivation for software-defined networking (SDN). SDN allows paths to be centrally computed over a global view of the network. However, ensuring paths are correctly computed and installed in the presence of failures is difficult in practice, even if the SDN controller is distributed [1]. Traditional control planes rely on distributed protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) to compute paths; these protocols typically employ variants of least cost path computation, and react to failures by recomputing least cost paths and installing forwarding state in routers that induces the paths. In contrast to centralized SDN, traditional control planes offer greater fault tolerance; but, determining the appropriate distributed realization of policies is hard [3].

Our high-level goal is to develop a system to *automate the process of creating a correct and failure-resilient distributed realization of policies in a traditional control plane.* To be useful, such a system must satisfy a few key requirements. (1) It must handle a wide range of commonly-used policies—including reachability, service chaining, and traffic engineering—to meet applications' diverse security and compliance requirements. (2) It must ensure that configurations are resilient to network malfunctions such as link failures. (3) It must provide support for realizing hierarchical control planes—where a network is split into several "domains" atop which a hierarchy of intra- and inter-domain control plane configurations is deployed—to ensure scalability for large networks [10]. (4) To improve manageability and network cost-effectiveness [5, 7], it must ensure that configurations obey certain general rules-of-thumb [4], e.g., limiting the number of lines of configurations and the use of certain configuration constructs.

Thus, our work contrasts with prior efforts which suffer from one or more drawbacks: they generate SDN- or BGP-specific control planes for a limited range of policies (e.g., peering) [2, 3, 8, 11–13, 16]; do not attempt to be resilient to failures [6]; do not support generating hierarchical control planes [3]; and do not enable generation of simple-to-manage network configurations.

The problem of synthesizing router configurations for which the distributed control plane is resilient to failures and generates policy-compliant paths is computationally hard. Even generating a set of policy-compliant paths for an SDN is computationally hard—e.g., enforcing isolated paths is NP-complete. While it is possible to develop algorithms for individual policies, accommodating multiple policies and the above requirements is extremely difficult.

An attractive possibility, motivated by recent progress in program synthesis, is to use Satisfiability Modulo Theories (SMT) solvers to automatically search the space of distributed configurations for a suitable "solution"— i.e., concrete router configurations that satisfy input policies and the aforementioned four requirements. SMT solvers provide support for constraint solving for propositional logic (SAT) and linear rational arithmetic (LRA); these powerful theories can encode properties on network paths and configurations. Thus, an SMT-search based approach can, in theory, provide support for multiple policies and different manageability requirements. This approach also decouples the requirements from the underlying search, and thus, could be extended to support new policies.

However, using SMT solvers in our context is non-trivial. To infer the concrete set of paths induced by network configurations, one has to incorporate into synthesis complex concepts—e.g., reasoning about shortest path algorithms requires constraints in complex

SIGMETRICS '18 Abstracts, June 18–22, 2018, Irvine, CA, USA

ACM ISBN 978-1-4503-5846-0/18/06.
https://doi.org/10.1145/3219617.3219652

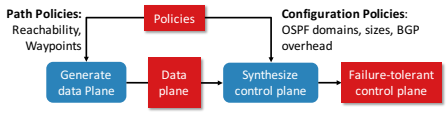

Figure 1: Two-phase process for generating a control plane with failure-tolerance properties

theories (SAT and LRA). Even with recent advances in SMT solving, approaches that directly generate configurations from policies do not scale to moderately-sized networks or sets of policies [6]. Furthermore, generating resilient control planes needs reasoning about how protocols react to failures, which further complicates an already intractable synthesis problem. Control plane hierarchy and manageability requirements further complicate the problem.

In our paper [15], we present Zeppelin, a system that overcomes the above challenges in SMT-based distributed configuration synthesis. Zeppelin uses a two-phased approach for tractability (Figure 1) that does not attempt to generate a policy-compliant control plane in a single step. First, Zeppelin uses Genesis [14] to synthesize paths—i.e., the network forwarding state—that are compliant with given policies, such as, waypoints and isolation. Zeppelin then generates intra-domain (shortest-path OSPF) and inter-domain (BGP) router configurations that induce the forwarding state synthesized by Genesis *and* provide high resilience. Zeppelin caters for moderate-sized enterprises and multi-tenant datacenters which require support for diverse policies.

We consider three settings with progressively more complex policies and resilience requirements and show how Zeppelin, using its two-phase approach, can effectively generate highly-resilient and policy-compliant solutions.

First, we consider a setting in which the operator of a hierarchically structured network wants to enforce a large diverse set of complex policies (waypoint traversal, isolation, traffic engineering, etc). The operator also desires a simple notion of *connectivity-resilience*—i.e., under most common link failures, even if operator-specified policies (like waypoints) are not enforced, majority of packets can still reach their destination. In this case, Zeppelin synthesizes OSPF configurations using linear constraint solving to compute link weights, and uses the unsatisfiable cores of failed solving attempts to judiciously place a small number of static routes. Zeppelin synthesizes BGP configurations directly from the domain mapping (i.e., which router belongs to what domain) and the paths. Using these techniques, Zeppelin generates configurations that are 10% more resilient than configurations that use only static routes.

Second, we consider a setting in which the operator wants to generate configurations which reduce the number of policy violations occurring under common link failure scenarios. We call this notion *policy-resilience*. Since synthesizing policy-resilient configuration is very difficult in the general case, we focus on a restricted class of policies: for traffic class (src-dst subnet pair), we allow the operator to specify a set of waypoints that packets must traverse before reaching their destination. We modify our linear constraints to ensure that at least two paths that traverse the waypoints have path cost (e.g., OSPF path cost) lower than any path not going through a waypoint, thus providing 1-resilience under failure. Using this, Zeppelin can generate configurations that are 140% more policy-resilient than configurations generated with our first technique (i.e., reduces policy violations under failures by 140%).

Finally, we consider a setting where the operator can specify bounds on the number and sizes of domains her control plane is organized into, and the ability to assign routers to different domains. We present a stochastic search technique that leverages this flexibility to look for ways to assign routers to different domains so that the synthesized configurations have even higher resilience. Through this, Zeppelin can further improve the resilience of the configurations by 10%. We show that Zeppelin can be naturally used to bound or optimize different configuration metrics such as static routes and BGP configurations' size to improve network manageability.

Our experiments show that, thanks to its two-phase approach, Zeppelin synthesizes connectivity-resilient configurations for medium sized datacenter topologies in < 10 *minutes* and policy-resilient configurations in under an hour. Notably, Zeppelin's performance is *2-3 orders of magnitude* faster than the state-of-art network configuration synthesis system SyNET [6], which uses a direct synthesis approach based on SMT.

Contributions. We make the following contributions [15].

- Zeppelin, a framework for that enforces policies in "traditional" (OSPF and BGP) networks by synthesizing highly-resilient router configurations. Zeppelin uses concrete paths to guide the synthesis instead of directly generating policy-compliant configurations.
- An algorithm for synthesizing policy-compliant configurations with few statically assigned routes. This algorithm yields configurations with high network connectivity even in the presence of link failures.
- An algorithm for synthesizing policy-compliant configurations that is specialized for waypoint policies. This algorithm yields configurations that have high policy compliance even in the presence of link failures.
- A stochastic search mechanism for finding partitions of the network into multiple routing domains which yield configurations with higher resilience.
- An implementation of Zeppelin together with an evaluation of its algorithms for different workloads, and comparison with a state-of-the-art configuration synthesis tool.

REFERENCES

[1] Aditya Akella and Arvind Krishnamurthy. 2014. A Highly Available Software Defined Fabric. In *HotNets*.

[2] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*. ACM, New York, NY, USA, 113–126. https://doi.org/10.1145/2535838.2535862

[3] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitu Padhye, and David Walker. 2016. Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations. In *Proceedings of the ACM SIGCOMM 2016 Conference on SIGCOMM (SIGCOMM '16)*.

[4] Theophilus Benson, Aditya Akella, and David Maltz. 2009. Unraveling the Complexity of Network Management. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. USENIX Association, Berkeley, CA, USA, 335–348. http://dl.acm.org/citation.cfm?id=1558977.1559000

[5] Theophilus Benson, Aditya Akella, and Aman Shaikh. 2011. Demystifying Configuration Challenges and Trade-offs in Network-based ISP Services. In *Proceedings*

of the ACM SIGCOMM 2011 Conference (SIGCOMM '11). ACM, New York, NY, USA, 302–313. https://doi.org/10.1145/2018436.2018471

[6] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin Vechev. 2017. Network-wide Configuration Synthesis. In *29th International Conference on Computer Aided Verification, Heidelberg, Germany, 2017 (CAV'17)*.

[7] Aaron Gember-Jacobson, Wenfei Wu, Xiujun Li, Aditya Akella, and Ratul Mahajan. 2015. Management Plane Analytics. In *IMC*.

[8] Victor Heorhiadi, Michael K Reiter, and Vyas Sekar. 2016. Simplifying software-defined network optimization using SOL. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 223–237.

[9] Ratul Mahajan, David Wetherall, and Thomas E. Anderson. 2002. Understanding BGP misconfiguration. In *SIGCOMM*.

[10] David A. Maltz, Geoffrey Xie, Jibin Zhan, Hui Zhang, Gísli Hjálmtýsson, and Albert Greenberg. 2004. Routing Design in Operational Networks: A Look from the Inside. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*. ACM, New York, NY, USA, 27–40. https://doi.org/10.1145/1015467.1015472

[11] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. 2013. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 27–38. https://doi.org/10.1145/2486001.2486022

[12] Mark Reitblatt, Marco Canini, Arjun Guha, and Nate Foster. 2013. Fattire: Declarative fault tolerance for software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 109–114.

[13] Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert Kleinberg, Emin Gun Sirer, and Nate Foster. 2014. Merlin: A Language for Provisioning Network Resources. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT '14)*. ACM, New York, NY, USA, 213–226. https://doi.org/10.1145/2674005.2674989

[14] Kausik Subramanian, Loris D'Antoni, and Aditya Akella. 2017. Genesis: Synthesizing Forwarding Tables for Multi-tenant Networks. In *POPL*. ACM.

[15] Kausik Subramanian, Loris D'Antoni, and Aditya Akella. 2018. Synthesis of Fault-Tolerant Distributed Router Configurations. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 1, Article 22 (April 2018), 26 pages. https://doi.org/10.1145/3179425

[16] Yifei Yuan, Dong Lin, Rajeev Alur, and Boon Thau Loo. 2015. Scenario-based Programming for SDN Policies. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '15)*. ACM, New York, NY, USA, Article 34, 13 pages. https://doi.org/10.1145/2716281.2836119

[17] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. 2012. *A Survey on Network Troubleshooting*. Technical Report TR12-HPNG-061012. Stanford University.