NV: An intermediate language for network verification

Introduction. Network devices often rely on distributed protocols, such as BGP, to make routing decisions. Network operators can enforce routing policies (that may express security, economic or other concerns) by configuring what routing protocols devices execute, and how they process routing messages. These configurations are expressed in low-level, vendor specific languages. Combined with the distributed nature of routing protocols, reasoning about the correctness of the configurations is a daunting task for operators. Network verification [1, 4] and simulation tools [3] have been proposed to aid operators. Additionally, as those techniques often face scaling problems, researchers have suggested ways [2] to simplify the complexity of networks.

Regardless of the transformation or reasoning principles used, one needs to parse the original network configurations as provided by operators. To tackle the range of vendor-specific configurations, Batfish [3] uses a vendor-agnostic representation of routing configurations for common protocols, and provides a translation from each vendor's language to Batfish's representation. Subsequent analysis such as compression [2], simulation [3] or verification [1, 4] can be then performed on top of this representation.

Batfish has been an indispensable tool for network researchers thanks to its ability to parse a wide range of configurations from different vendors. Unfortunately, its intermediate language (IR) falls short of many language design goals. First, at 105 different expressions and 23 statements Batfish's IR is *massive*. This is a symptom of other problems in the design of the IR. In particular, the expressions and structures used are *specialized* to routing protocols. For example, instead of a set operation that specifies the field of the attribute to be changed along with its new value, Batfish uses a different expression to set the local preference of a BGP attribute, a different expression to set the MED value, and so on. As such, expressions cannot be *composed* to build other more complex operations. Besides the explosion in the size of the IR, this poses another issue: many desirable transformations cannot be expressed within Batfish's IR. For instance, replacing the `AS Path` attribute of BGP with its length can often improve simulation performance without loss of precision. Yet, this transformation cannot be expressed within Batfish's current AST, because one cannot alter the type of the `AS Path` or the operations on it. Moreover, understanding the semantics of the language requires deep knowledge of routing protocols and the intricacies of vendor specific implementations.

Finally, some effects of executing a protocol are not expressed in the configurations, but are left *implicit* and it's up to the backend (e.g. the simulator) to correctly capture them. This makes it difficult to implement new analyses of configurations, as one has to correctly implement any implicit effects operations may have.

NV: A flexible IR for control plane configurations. To overcome these limitations, we propose a typed intermediate language, called NV. NV allows the user to specify the topology of a network, the type of the routing messages exchanged, and functions that define how each device processes these messages. The key design points of NV are its compact size, the compositionality of its expressions, and the use of standard programming language constructs (similar to the ones of ML based languages). We have implemented two different backends to NV, a BDD-based simulator that simulates the message exchange procedure of distributed routing protocols, and a SMT-based logical encoding that can verify properties of the converged state of a network. Furthermore, to improve the performance of such techniques we have implemented some common compiler optimizations such as constant unfolding, inlining and partial evaluation. The small size and the use of standard constructs with well-defined semantics facilitates the implementation of such optimizations.

NV is designed to be an IR, but also a verification framework. NV includes two key features to support this role: 1. symbolic variables that denote unknowns in the network, 2. assertions to be verified about the network's converged state. For instance, a symbolic variable can model a potential link failure, or a routing announcement from an external peer.

Finally, for NV to be useful, it must be able to (at least) model commonly used routing protocols, such as BGP and OSPF. One of the challenges we faced is to find a language that is sufficient to model in detail these protocols, but that we can also efficiently compile to BDDs or logical formulas to be verified by an SMT solver. Currently, we can translate a number of protocol configurations from Batfish to NV, including eBGP and OSPF, and we are working towards supporting more complicated protocols such as iBGP.

Related Work. The design of NV is partly inspired from routing algebras [5, 6]. Routing algebras were originally devised to reason about convergence properties of protocols, but the main goal of NV is to enable modelling of protocols and reasoning about routing properties such as reachability, way-pointing, and fault tolerance.

REFERENCES

[1] R. Beckett, A. Gupta, R. Mahajan, and D. Walker. A general approach to network configuration verification. In *SIGCOMM*, August 2017.

[2] R. Beckett, A. Gupta, R. Mahajan, and D. Walker. Control plane compression. SIGCOMM '18, pages 476–489, 2018.

[3] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein. A general approach to network configuration analysis. In *NSDI*, 2015.

[4] A. Gember-Jacobson, R. Viswanathan, A. Akella, and R. Mahajan. Fast control plane analysis using an abstract representation. In *SIGCOMM*, 2016.

[5] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 1–12, August 2005.

[6] J. a. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, October 2005.