Route Shepherd: Stability Hints for the Control Plane

Alexander J. T. Gurney
University of Pennsylvania
Philadelphia, USA

Xianglong Han
University of Pennsylvania
Philadelphia, USA

Yang Li
University of Pennsylvania
Philadelphia, USA

Boon Thau Loo
University of Pennsylvania
Philadelphia, USA

ABSTRACT

The Route Shepherd tool demonstrates applications of choosing between routing protocol configurations on the basis of rigorously-supported theory. Splitting the configuration space into equivalence classes allows the identification of which parameter combinations lead to protocol stability, and which do not. This ahead-of-time analysis generates a predicate, in the form of a combination of linear integer inequalities, which can be used in several complementary ways by downstream applications. Examples presented include warning operators about errors in advance, recovery from protocol oscillation, plotting a series of safe parameter changes, and understanding the dynamics of the routing system.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*

General Terms

Management, Reliability

Keywords

Border Gateway Protocol, routing policy, stable path problems, partial specification

1. INTRODUCTION

We present the Route Shepherd toolkit, which aims to provide analysis and generation of safe routing configurations based on well-established routing theory [4]. Network operators often have to reconfigure their routing setups, in order to perform traffic engineering, update ISP policies, or perform a live migration of one protocol to another [2, 5, 7, 8]. A significant challenge facing operators is the large array of configuration choices that could be applied. Many of these have non-obvious interactions: the chosen paths are computed in a complex fashion, depending on the values of these settings and the timing of network events. Sometimes, particular combinations of parameters can result in persistent oscillation, or loss of connectivity. These situations are dependent on the precise numeric values of the parameters (e.g. link weights) involved—a very small change can have huge effects—and it can be hard to see how to salvage a proposed network design once such problems occur.

SIGCOMM'12, August 13–17, 2012, Helsinki, Finland.
ACM 978-1-4503-1419-0/12/08.

As an example, operators might want to adjust link weights for their internal network, so that performance metrics such as congestion can be minimized [5, 7]. However, it is hard to make such changes with confidence, since one has to take multi-protocol and cross-domain interaction into account. Individual changes can be simulated, one at a time, but this technique can only sample a few points out of an enormous configuration space. A misstep can be costly, since not all anomalies disappear after parameters are reverted [3]. Moreover, one cannot revert a problematic change made in somebody else's network.

To address the above challenges, the Route Shepherd toolkit provides a mechanism to explore many routing configurations at once, and summarize the results for the benefit of operators. Route Shepherd aims to be able to deal with configurations not one-by-one, but in equivalence classes, giving a higher-level perspective. In this way, unsafe configurations can be avoided en masse, as the operator is presented with the information at the earliest opportunity. In the demo, we will show how this can be done in the case of changes to a single parameter at a time, by highlighting ahead of time the *safe and unsafe ranges* for that value. That is, based on the current state of the rest of the network, we can tell which changes to this parameter will lead to oscillation and which will not. This is done *without* the need to simulate all of the different possibilities.

Based on the same data, we can also plot *safe courses* in the configuration space—going from one complete parameter set to another, while avoiding transient anomalies, or recovering from an unsafe configuration by making a sequence of changes to bring the network back into compliance.

2. SYSTEM OVERVIEW

The goal of our system is to allow network operators to understand the properties of a wide range of configuration possibilities, in a lightweight fashion. The essence of our tool is a pipeline (see Figure 1) for identifying *representative* configurations and analyzing only these.

From the original network configuration (1), we build a combinatorial model of the possible routing outcomes (2). This intermediate representation is based on the idea of *partial configuration*, where some parameters are treated as variables [4]. From this, we identify representative configurations, each one associated with a set of parameter bounds (3). The basic idea here is that only certain parameter changes affect the routing outcome, and so equivalent changes can be grouped together. Our underlying theory gives us confidence that these representatives do indeed cover only safe configurations. However, not all of the representatives deemed unsafe will actually diverge, since our theory is not precise enough to include all nuances of protocol behavior. We therefore introduce two further filtering steps (4): numerically impossible configura-

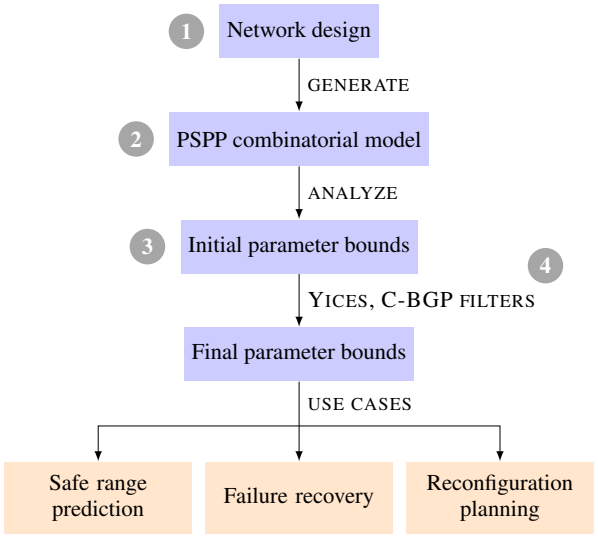

Figure 1: Tool pipeline

Figure 2: Link weight controls and predictions

tions are rejected by the Yices SMT solver [1], and any remaining configurations are screened with the C-BGP [6] solver, for the final stability test. Although we end up using simulation with C-BGP, the context of our earlier theory work means that we only have to simulate a few different examples, in order to achieve coverage of the configuration space.

We have validated the results of this tool against networks of Quagga software routers, to confirm its predictions against the actual network behavior. The demo uses several Quagga instances on a single computer, each running in a user-mode Linux virtual machine.

3. DEMONSTRATION PLAN

We demonstrate our tool in two modes: the normal interactive mode, allowing users to tweak the running system and see the consequences, and a replay mode, where human interaction is not required. We have developed a GUI that illustrates the action of the simulated network control plane, and allows users to alter certain chosen parameters. Additionally, the controls are decorated with annotations indicating *safe ranges* for each parameter, based on the current routing state; see Figure 2. These indicate which changes are potentially unsafe (red slashed range), and which are guaranteed not to result in oscillation (green range).

The interface also shows the network map, and highlights the chosen routes as they change over time. The program can be configured with a choice of different network topologies. These include simple 'toy' examples, where it is easier to comprehend why given states are safe or otherwise, and also more realistic and complex scenarios, showing why our tool is useful for these situations that are more difficult to understand without automated support.

We present the following additional use cases:

Recovery. We will show how a persistent oscillatory state can be halted by means of a minimal number of automatically chosen parameter changes. This is achieved by using the Yices solver in MAX-SMT mode.

Reconfiguration. In addition to the ability to interactively change parameters one-by-one, we show how the system can plot a series of safe changes, from one complete parameter set to another, without causing any transient routing anomalies.

Video

A demonstration video is available online at http://netdb.cis.upenn.edu/routeshepherd/.

Acknowledgements

This research is partly supported by the NSF Expeditions in Computer Augmented Program Engineering (ExCAPE) project, ITR-1138996. Further support was provided by the NSF grants CCF-0820208, IIS-0812270, CNS-0845552 and CNS-1040672.

References

[1] B. Duterte and L. de Moura. A fast linear arithmetic solver for DPLL(T)*. In *Proc. Conference on Computer Aided Verification*, 2006.

[2] P. François, M. Shand, and O. Bonaventure. Disruption-free topology reconfiguration in OSPF networks. In *Proceedings of IEEE INFOCOM*, 2007.

[3] T. Griffin and G. Huston. RFC 4264: BGP wedgies, 2005.

[4] A. J. T. Gurney, L. Jia, A. Wang, and B. T. Loo. Partial specification of routing configurations. In *Workshop on Rigorous Protocol Engineering (WRiPE)*, 2012.

[5] A. M. C. A. Koster and X. Muñoz, editors. *Graphs and algorithms in communication networks*. Springer, 2010.

[6] B. Quoitin and S. Uhlig. Modeling the routing of an Autonomous System with C-BGP. *IEEE Network*, 19(6), 2005.

[7] S. Raza, Y. Zhu, and C.-N. Chuah. Graceful network state migrations. *IEEE/ACM Transactions on Networking*, 19(4), 2011.

[8] L. Vanbever, S. Vissicchio, C. Pelsser, P. François, and O. Bonaventure. Seamless network-wide IGP migrations. In *Proceedings of ACM SIGCOMM*, 2011.