AutoNetkit:
Simplifying Large Scale, Open-Source Network Experimentation

Simon Knight 1 Matthew Roughan 1 Askar Jalabdinov 2 Olaf Maennel 2 Iain Phillips 2
1 The University of Adelaide, Australia; 2 Loughborough University, United Kingdom

Introduction

- Build and deploy large-scale emulated network experiments in minutes.
- High-level design benefits of Software Defined Networking, with existing hardware, software, and protocols.
- Based on network abstractions which hide low-level details: save time, reduce errors, and conduct reproducible network experiments.
- Generate configurations and deploy to emulation environments: run real router software inside virtual machines, realistic and affordable experimentation.
- Part of ongoing project to simplify network management using formal methods.

Visual Capture

- Most network designs start on a whiteboard or in a diagram tool such as Visio, and are then manually transcribed to a network description.
- We automate this: draw network in a graphical editor, save as GraphML, use directly as network description — build your network directly from a diagram.
- Add custom nodes and edge attributes, e.g. device type, ASN, or link speed.

High-Level Network Design

- Design networks, not devices.
- Built on Python: use standard syntax to work with attributes.
- Quick and easy configuration, extend to configure new protocols or services.
- G.in.update(G_in.nodes(‘is_router’, platform=‘netkit’), syntax=‘quagga’) # default

Live Feedback

- Real-time plotting of overlay graphs using D3.js: live feedback on topology design.

Automated Resource Allocation

- Automatic handling of tedious and error-prone low-level details such as IP Addresses.

Abstract Network Model

- Network description: read from GraphML, CSV, JSON into G0 graph.
- Build user-defined graphs such as G0 or Gbgp from G0. Extensible to support new design patterns and protocols.
- Compile overlay graphs into Network Information Database: device-based representation of network, ready to push into configuration templates.

Extensible Configuration

- Generate configuration files from NIDB using plain-text templates.
- Separation of configuration syntax and semantics.
- Easily configure new devices, or network services such as DNS.

Build, Deploy, Measure

- Supports Quagga, Junos, IOS and C-BGP, through Netkit, Junosphere and Dynagen.
- Automatically push out a new network configuration to an emulation host.
- Automated data collection from emulated network: e.g. routing tables and traceroutes.
- Rapid iteration: modify topology or design, configuration, deployment, measurement automatically applied.

Getting and Using AutoNetkit

- Python-based: runs on Linux, Mac OS X, Windows.
- Installation and usage information on website.
- Open-Source: BSD Licence, available on GitHub.
- High-Performance: under 7 seconds to configure 1400 router multi-AS network with OSPF and BGP, including IP addressing and route-reflector IBGP hierarchy.
- Verified Bad-Gadget routing oscillation [1]: drew network in < 10 mins, multi-platform configuration, deployment and measurement. Measured oscillation on IOS and Junos, but not Quagga — due to Quagga implementation of BGP decision process. Realism of emulation over simulation: can expose real bugs or implementation decisions.

References and Acknowledgements


This project was supported by the Australian Government through an Australian Postgraduate Award, Australian Research Council Discovery Grants (DP109503 and DP1095043) and Cisco through Grant 11-R949356(4). We also would like to thank Hung Nguyen, Nick Falkner, Joel Coopey, and Michael Romansicz for their helpful input, and the Netkit, Dynogen and Junosphere development teams.

www.autonetkit.org