AutoNetkit:
Simplifying Large Scale, Open-Source Network Experimentation

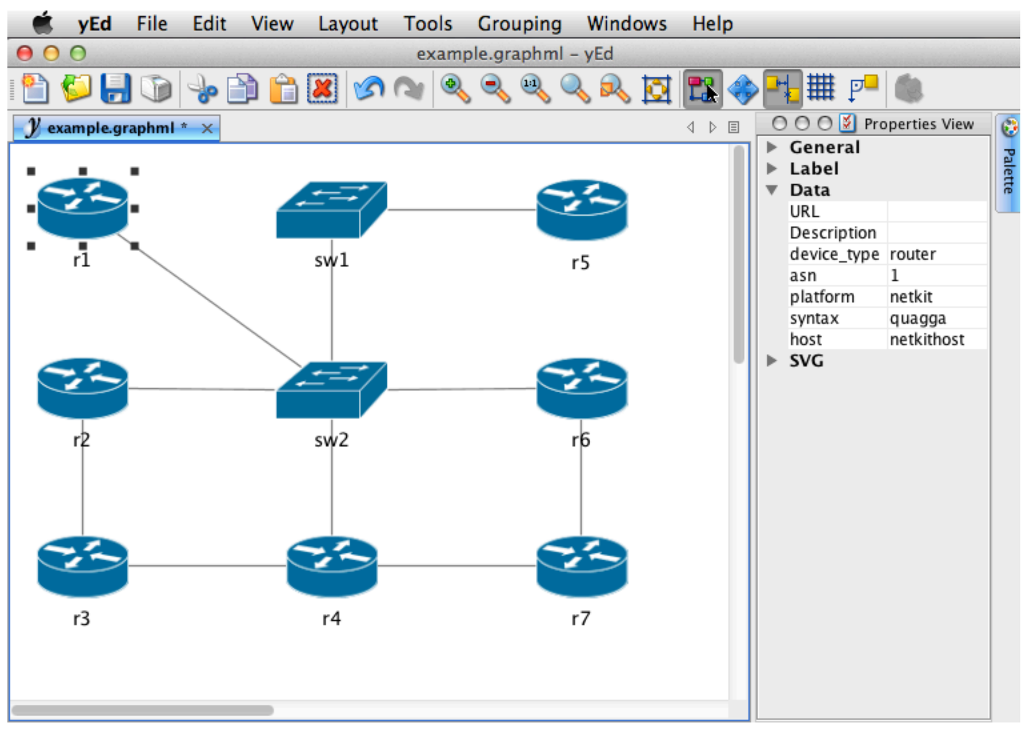
Simon Knight [1] Matthew Roughan [1] Askar Jaboldinov [2] Olaf Maennel [2] Iain Phillips [2]
[1]The University of Adelaide, Australia; [2]Loughborough University, United Kingdom

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.



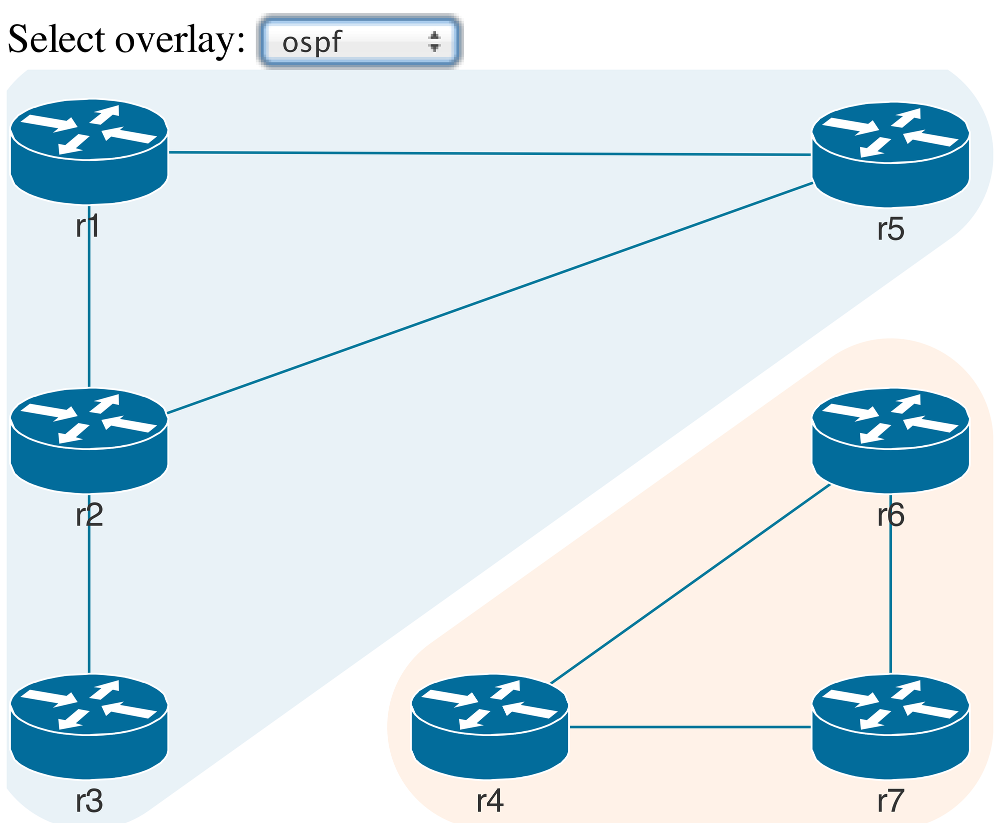
High-Level Network Design

- 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

for devices in G_phy.groupby('asn').values(): # iBGP full-mesh per AS
  rtrs = [d for d in devices if d.is_router] # filter routers
  G_ibgp.add_edges_from((s,t) for s in routers for t in routers if s != t)

G_ebgp.add_edges_from(e for e in G_in.edges() if e.src.asn != e.dst.asn)

ank.aggregate_nodes(G_ospf, G_ospf.nodes('is_switch')) # Merge switches
ank.explode_nodes(G_ospf, G_ospf.nodes('is_switch')) # Switches to edges

# Trim non intra-AS links
G_ospf.remove_edges_from(l for l in G_ospf.edges() if l.src.asn != l.dst.asn)

for link in G_ospf.edges(): # set defaults
  link.cost = 1
```

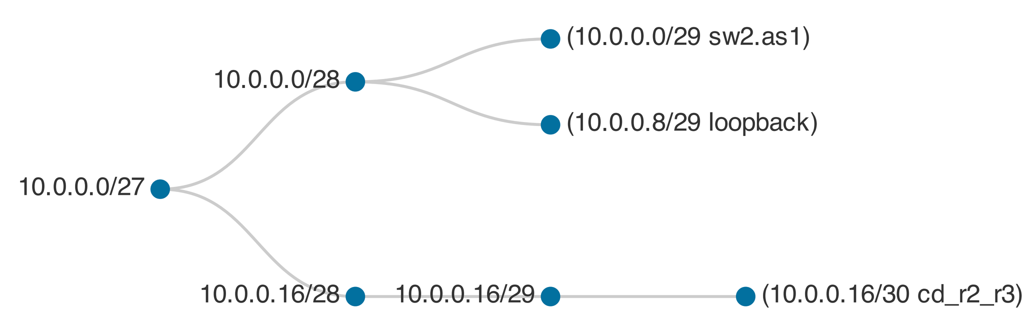
Live Feedback

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



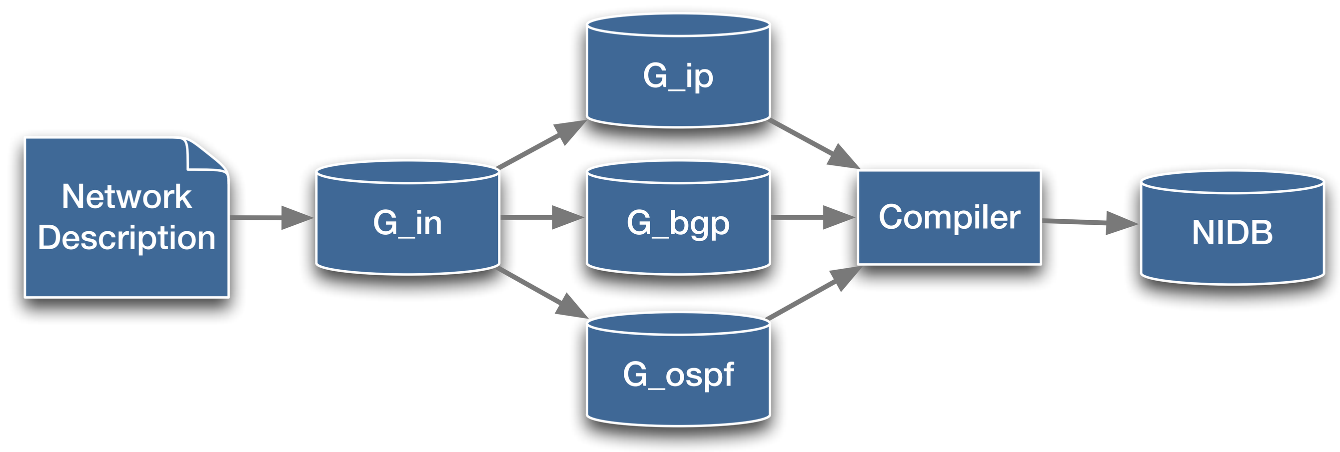
Automated Resource Allocation

- 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 G_{in} graph.
- Build user-defined graphs such as G_{ip} or G_{bgp} from G_{in}. 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.



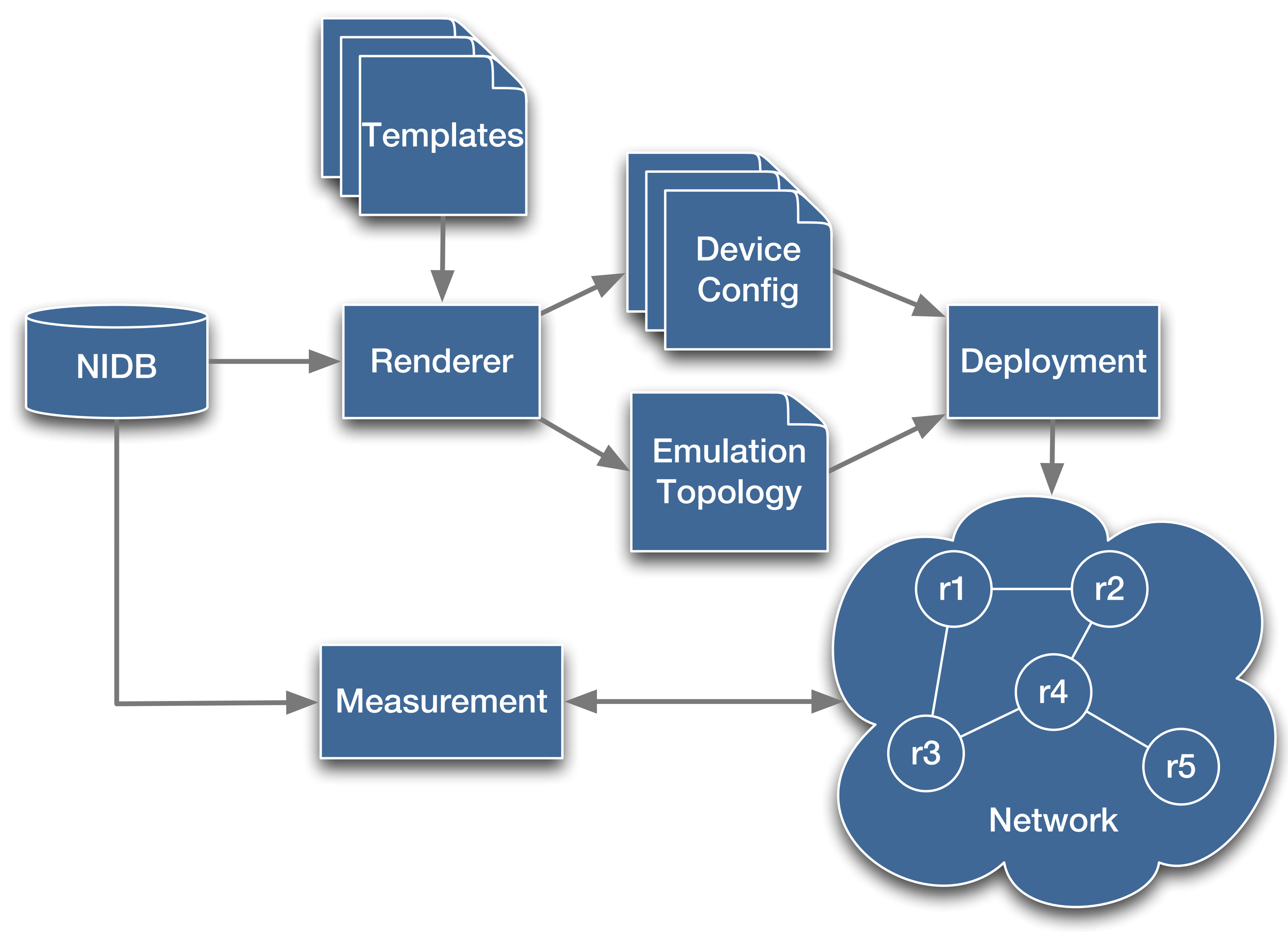
Extensible Configuration

- 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.

```
hostname ${node}
password ${node.zebra.password}
!
% for i in node.interfaces:
  interface ${i.id}
  #Link ${i.description}
  ip ospf cost ${i.ospf_cost}
  !
% endfor
router ospf
% for l in node.ospf.links:
  network ${l.network.cidr} area ${l.area}
% endfor
```

```
hostname r5.as1
password 1234
!
  interface eth0
  #Link to r5.as1 to r1.as1
  ip ospf cost 1
  !
  interface eth1
  #Link to r5.as1 to r2.as1
  ip ospf cost 1
  !
router ospf
  network 10.0.0.0/29 area 0
```

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

T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, Apr. 2002.

This project was supported by the Australian Government through an Australian Postgraduate Award, Australian Research Council Discovery Grants DP110103505 and DP0985063; and Cisco through Grant 2011-89493(3696). We also would like to thank Hung Nguyen, Nick Falkner, Joel Obstfeld, and Michael Rumsewicz for their helpful input, and the Netkit, Dynagen and Junosphere development teams.