RDL: A programmatic approach to generating router configurations

RIPE Routing WG, May 15 2014

Benno Overeinder
RDL: The background

- ENGRIT: Extensible Next Generation Routing Information Toolset
- Improve Internet routing security and stability
- Multi-pronged approach, RDL is one aspect
- Other aspects will focus on authentication, etc
- NLnetLabs has done much work with DNS
- RDL development done by Per Bilse (EUnet, AS286)
RDL: The rationale

- Global turnover $dozens of millions per hour
- Even small problems can be very costly
- Router configuration is inherently low level
- Large number of only moderately related detail
- Limited or no verification tools
- Limited scope for inter-ISP routing management
RDL: The idea

• A high level Routing Documentation Language

• Dual purpose:

• 1) Architecture independent generation of BGP config:
  - RDL->Cisco, RDL->Juniper, RDL->BIRD
  - C->68k, C->x86_64, C->ARM

• 2) Description and publication of routing policies:
  - Enable automated verification and proofing
  - Improve exchange of information between peers
RDL: Not RPSL NG NG

• RDL will reuse data sources also used by RPSL:
  – Some objects
  – Publication/repository means, where feasible

• But, more importantly:
  – RDL to describe BGP topology
  – RDL to cover both iBGP and eBGP peerings
  – RDL to fully qualify and identify routing policies
RDL: Also not YANG (RFC6020)

- YANG is geared for NETCONF
- YANG and NETCONF are generally focused on physical Device Configuration and Management
- YANG is itself low level and riddled with detail
- RDL is for humans
- RDL is focused on a logical and abstract BGP view, independent of underlying network and devices
- YANG could be a compilation target for RDL
RDL: What is a policy?

- Much confusion between Policy and Enforcement Action
- A policy is Thieves will be prosecuted
- An enforcement action is Arrest Nosey Parker
- Existing tools and approaches focus on enforcement actions
- Quickly degenerate into route filter mechanics
RDL: Policies in 3D

- A routing policy as seen by RDL has three dimensions to it:
  - Where it applies: topological location
  - When it applies: NLRI attributes
  - What to do: filtering and attribute manipulation
- Think of it as similar to a piece of legislation, eg speed limits: Where, When, What
- These three aspects jointly describe a given policy in its entirety
RDL: A policy example

- Policy: My AS will not announce bogons
- RDL's 3D approach:
  - Where: all peerings with foreign ASs
  - When: prefix is in list of bogons
  - What: block it
- RDL's BGP topology description is the key to specifying the Where of a policy
- the Where is statically analysed and applied when generating configurations
- The When and the What are done by the routers
RDL: The language

• Designed specifically for the purpose of describing BGP topologies simply and intuitively
• Free form curly brace, recursive, and concatenative syntax, allowing quick and easy specification of objects and their location
• Borrows inadvertently and disrespectfully from several unusual languages
• Fully dynamically typed and declaration free
RDL: BGP topology

- RDL describes BGP topology by way of three logical components:
  - Zones – may contain other zones, and routers
  - Routers – may contain one or more BGP peers
  - Peers
- Structure similar to file system directories
- Each object has a number of attributes
- Attributes may be inherited from lexical scope
- iBGP is configured automatically
hibernia = new(zone) . {
    .asn = 5580;

    EU = new(zone) . {
        NL = new(zone) . {
            ams1 = new(router) . {
                .address = 134.222.1.1;
                ripe = new(peer) . { 1.2.3.4, 3333 };
            }
        }
    }

    US = new(zone) . { .... };

    APAC = new(zone) . { ... };
};
RDL: What's in a zone

- Zones are containers for similar policies
  - often significant geographical correlation
  - should be chosen to reflect the reality of your network, not the other way around (your network is the ground, the zone map is the map)
  - you decide what your zone map should be, it is there to help you
  - again: RDL is all about BGP topology
  - the zone map identifies reference points for policies
RDL: Policy example

- Policy descriptions follow the topology format

\[
\text{nobogons} = \text{new}(\text{policy}) . \{ \\
\quad .\text{where} = \text{export peer.asn} != \text{peer.router.asn}; \\
\quad .\text{when} = \text{nlri.prefix} & \text{bogons}; \\
\quad .\text{what} = \text{reject}; \\
\};
\]

\[
\text{bogons} = \{ 0.0.0.0/8^+, 10.0.0.0/8^+, 100.64.0.0/10^+, \ldots \};
\]

- Policy syntax is experimental/undecided
- Probably a good idea to stick to general syntax of RDL
hibernia = new(zone) . {
    .asn = 5580;
    RR1 = new(router) . { 134.222.12.1 };
    RR2 = new(router) . { 134.222.14.1 };
    EU = new(zone) . {
        .ibgp = { RR1, RR2, "localmesh" };
        NL = new(zone) . {
            ams1 = new(router) . { 134.222.1.1 } . { ... };
        };
    };
    US = new(zone) . { .ibgp = { RR1, RR2, "localmesh" }; ... };
};
RDL: Unusual Example II

- Policy: de-prioritise all EU routes in US
- RDL to the rescue:

```ruby
EUexport = new(policy) . {
    .where = import peer.zone <= US && peer.remote.zone <= EU;
    .when = ;
    .what = local-preference = 90;
};
```

- RR1 and RR2 are route reflectors and are therefore transparent
RDL: Unusual Example III

Changing iBGP to full mesh requires only a few edits:

hibernia = new(zone) . {
    .asn = 5580;
    RR1 = new(router) . { 134.222.12.1 }; 
    RR2 = new(router) . { 134.222.14.1 }; 
    EU = new(zone) . { 
        .ibgp = { RR1, RR2, "localmesh" }; 
    };
    NL = new(zone) . { 
        ams1 = new(router) . { 134.222.1.1 } . { ... };
    };
    US = new(zone) . { .ibgp = { RR1, RR2, "localmesh" }; ... };
};
RDL: Unusual Example IV

- And now RDL's pièce de résistance
- Recall the policy: de-prioritise all EU routes in US

```
EUexport = new(policy) . {
    .where = import peer.zone <= US && peer.remote.zone <= EU;
    .when = ;
    .what = local-preference = 90;
};
```

- Absolutely nothing needs to be done for the iBGP change.
- Jus' like that!
RDL: Policies for the future

- As shown, policies are generally modular and independent of underlying topology detail.
  - Eg adding or deleting a normal router requires no change to any regular policies, internal or external.

- What if two peers could exchange policies before peering? “Here's mine, I've got yours, thank you”.
  - Eg “I want a default, and no US routes”; plug in and peer.

- Can't be done “Jus' like that!”, will require either coordination on parameters or a higher level meta-description.

- Outside current scope of project, but not at all impossible.
Acknowledgments

- Job Snijders and Andreas Polyrakis
  - valuable discussions
  - providing use cases for RDL
  - reviewing the previous versions of RDL
RDL: Nirvana?

RDL is all about **not configuring routers**, but documenting and programming the AS.

Open source project and open discussions
http://lists.rpsl.net/mailman/listinfo/progress

ENGRIT + admin: benno@nlnetlabs.nl

RDL: pgb@bgpinnovations.com