Automated Peering Operations

Job Snijders job@instituut.net
Coloclue / AS 8283

(don’t worry, I still also work for NTT)

What is Coloclue?

- https://coloclue.net (note: website encrypted in Dutch)
- A registered non-profit association based in the Netherlands
- 100% volunteer driven
- Present at AMS-IX & NL-IX
- A **testbed** for network technologies:
  - One of 3 RPKI validating networks worldwide (invalid->drop)
  - Only ISP to apply strict IRR filtering to all peering partners
  - First ever ISP to deploy BGP Large Communities (RFC 8092)
  - First “peering-over-github” network
  - Who knows what the future brings… ☺

Scale of peering

- 4 routers
  - BIRD, Debian/Linux
- 2 Internet Exchanges, some private peering, 3 transit
- ~ **1000 BGP sessions** (IPv4 + IPv6)
- ~ 51 megabyte of configuration per router
  - RPKI done without RTR
  - IRR filters for all peers
  - AS_PATH filters for all peers
- https://as8283.peeringdb.com/

What are **automated peering operations**?

- When two networks agree to peer
  - Computer consults PeeringDB what IXPs are common
  - Computer uses PeeringDB for maxprefix, irr-data
  - Computer sets up BGP sessions to all the IPs listed on PDB
  - Computer removes BGP sessions towards unlisted (“old”) IPs

The above process should be done regularly (we do every 11 hours)

Why automate your peering operations?

$$$$$ MONEY SAVER $$$$$$

1. Reduce coordination between humans
    1. Big time saver, we are volunteers doing this in our spare time
    2. Reduce chances of discoordination

2. Reduce human error

Automation beats Human Coordination

“The less we have to coordinate with each other, the better”

- If you (as my peer) add or remove an AMS-IX connection: do not email me, the computer will notice and configure accordingly
- If you link up to a new IXP, that we happen to have in common: don’t email me, the computer will take care of it
- If you enable/add IPv6 on an interface, no need to email

A human element remains

- Coordination on interconnection agreements
- Interconnection Policy Compliance matching
- Troubleshooting

Practical Workflow

- When a peer agrees:
  1. Peer submits a Github Pull-Request https://github.com/coloclue/peering
  2. A coloclue volunteer makes the Pull-Request if the peer doesn't like git
- Wait for jenkins (https://jenkins.io/) to start the '`update_routers.sh`' script
- Done!

- Depeering: just remove the peer from the peers.yaml file

peers.yaml example file: https://github.com/coloclue/peering

```
AS714:
    description: Apple Inc
    import: AS-APPLE
    export: "AS8283:AS-COLOCLUE”

AS46489:
     description: Twitch / Justin.tv
     import: AS-TWITCH
     export: "AS8283:AS-COLOCLUE”

AS1273:
     description: Vodafone Group
     import: AS1273:AS-CWW AS1273:AS-CWW-V6
     export: "AS8283:AS-COLOCLUE”
     ipv4_limit: 90000
     ipv6_limit: 4500
```

Why does this work?

- **Idempotency is key**
  - Can run the ’update_routers.sh’ script as many times, as long as input is the same, the output will be the same
- **The database is authoritative**
  - Should a volunteer manually configure a BGP session on one of the routers, it will be removed by ‘kees’
- **Humans are terrible at repetition**

Review “Network Automation Do’s and Don’ts”
http://instituut.net/~job/jobsnijders_nanog66_dosdontsnetworkautomation.pdf

Snippet of peer agnostic template

```
Vurt:kees job$ cat templates/peer.j2
protocol bgp {{ neigh_name }} {
    description "{{ description }}";
    neighbor {{ neigh_ip }} as {{ asn }};
    local as 8283;
    next hop self;
    receive limit {{ limit }} action restart;
    import keep filtered;
    import filter {{ filter_name }};
{%- if export_full_table %}
    export where full_table_export({{ asn }});
{%- else %}    export where ebgp_peering_export({{ asn }});
{%- endif %}
{%- if password %}
    password "{{ password }}";
{%- endif %}
{%- if gtsm %}
    ttl security on;
{%- endif %}
{%- if multihop %}
    multihop;
    source address {{ source }};
{%- endif %}}
```

https://github.com/coloclue/kees/blob/master/templates/peer.j2

Open Source Tooling

- Kees https://github.com/coloclue/kees
  - The BGP orchestration software

- Birdseye https://github.com/ecix/birdseye
  - A cool BGP Looking Glass which hooks into BIRD
  - Allow peering partners to self-help (public tooling > human coordination)

Considerations

- This approach (perhaps naively) assumes that networks want to peer at every IXP they have in common
  - Corollary: Traffic controllers should steer traffic, not BGP sessions
- The OSS toolkit does not validate Interconnection Policy:
  - Consistent announcements? Honoring MED? etc
- *(@#$&*(@#$ MD5
- What to do with down sessions?
  - Chase with unstructured emails?
  - Ignore as long as some sessions are up?
  - Notify PeeringDB that the IP may not be correct?

Questions?

- Would you put “Automated Peering Operations” on your road-map?
- How can we help each other to get there?
- Any hints on what we should do differently?