RIPE

ARouteServer

IXP Automation Made Easy

Pier Carlo Chiodi - https://pierky.com/

RIPE74 - Budapest, Hungary

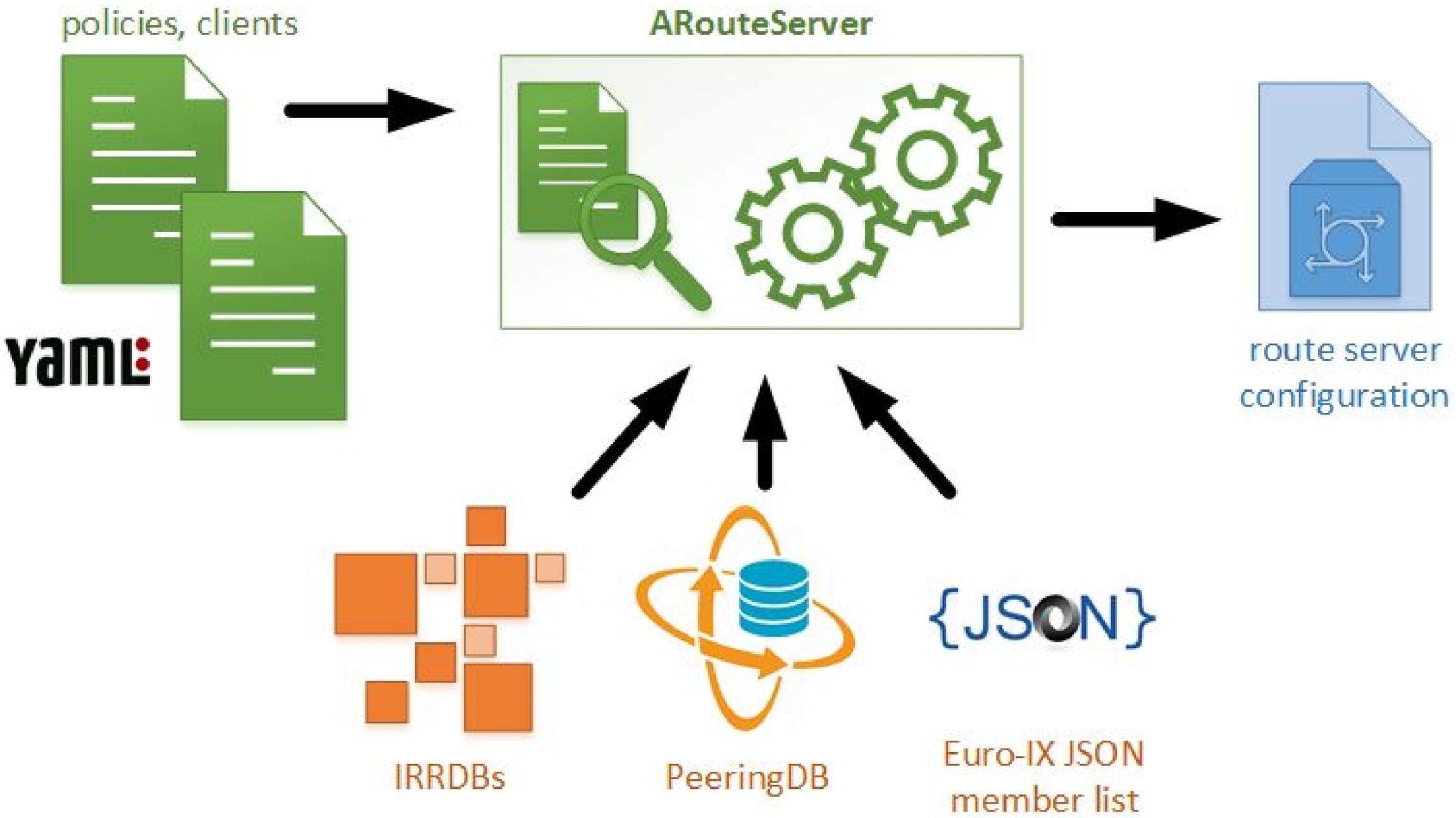
RIPE Connect Working Group

May 10, 2017

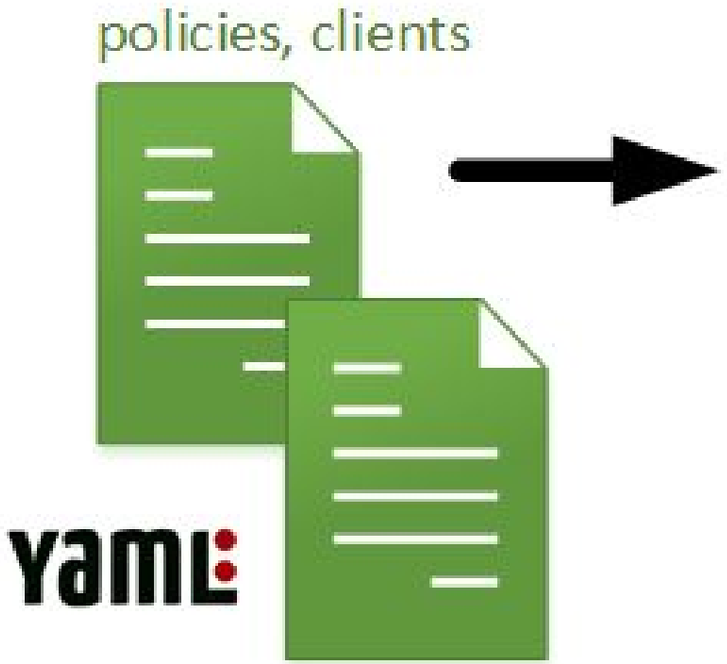
Overview

- "A route server redistributes BGP routes received from its BGP clients to other clients according to a prespecified policy" (RFC7948)
- ARouteServer automatically builds configs for route servers...
  - feature-rich configurations!
- … and it also allows to validate them
  - Docker / KVM -based "live test" framework
- Currently, BIRD and OpenBGPD are supported

How does it work



How does it work



- Two YAML files are used to configure syntax-agnostic high-level policies and clients definitions

```
cfg:
  rs_as: 999
  router_id: "192.0.2.2"
  filtering:
    next_hop:
      policy: "same-as"
  blackhole_filtering:
    policy_ipv4: "rewrite-next-hop"
    ...
```

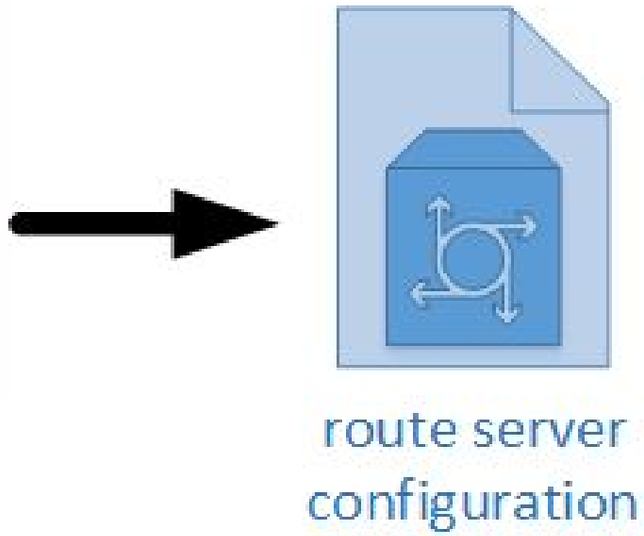
```
clients:
  - asn: 111
    ip:
    - "192.0.2.11"
    - "2001:db8:1:1::11"
    irrdb:
      as_sets:
        - "AS-AS111MAIN"
  ...
```

How does it work

- External sources are used to gather additional info:
  - IRRDBs for filters based on prefixes and origin ASNs
  - PeeringDB for max-prefix limits
  - Euro-IX JSON member list files to build clients list automatically

How does it work

- Finally, the route server configuration is automatically generated
- Custom scripts can be used to test it and to deploy it to the route server




```
$ arouteserver bird --ip-ver 4 -o /etc/bird/bird4.new && \
    bird -p -c /etc/bird/bird4.new && \
    cp /etc/bird/bird4.new /etc/bird/bird4.conf && \
    birdcl configure
```

Configuration

- Policies can be set on a global scope
- Clients inherit global policies
  - client-specific options override inherited ones
- Client list can be automatically generated from Euro-IX member list JSON file
  - this allows an easy integration with IXP-Manager
- Custom, site-specific behaviours and configurations can be implemented using “hooks” and local files

Features: routes filtering

- NEXT_HOP enforcement: strict or same AS
- min/max prefix length & max AS_PATH length
- AS_PATH sanitation
  - leftmost ASN
  - private/invalid ASNs
  - “transit-free” ASNs
- RPKI validation and filtering
- bogons and IRRDBs-based filters (prefixes & origin ASNs)

Features: functions

- Blackhole filtering support
  - optional NEXT_HOP rewriting
- Route propagation control (via BGP communities)
  - announce / do not announce to any / specific peer
  - prepend to any / specific peer
  - add NO_EXPORT / NO_ADVERTISE to any / specific peer
- Full list of features available on GitHub

Testing configurations

- Live test framework included
  - built-in / custom scenarios define policies and expected results
  - Docker containers (and KVM VM for OpenBGPD) virtualize the route server and its clients; a Python API interacts with them
  - route server configuration is generated by ARouteServer
  - Python test cases are run to verify that expectations are met

```
def test_071_blackholed_prefixes_as_seen_by_enabled_clients(self):
  for inst in (self.AS2, self.AS3):
    self.receive_route(inst, "203.0.113.1/32", self.rs,
                       next_hop="192.0.2.66",
                       std_comms=["65535:666"])
```

Project status

- Actively developed! Looking for testers and reviewers
- Feedback from real life is strongly needed and encouraged

Source code and examples available on GitHub:

- https://github.com/pierky/arouteserver

Full documentation:

- https://arouteserver.readthedocs.io/

Pier Carlo Chiodi - https://pierky.com/

Questions?