Internet Routing Registry daemon version 4

...

Job Snijders
NTT Communications
job@ntt.net
@jobsnijders

Sasha Romijn
DashCare
sasha@dashcare.nl
@mxsash

Why IRRd v4

- IRRd v3 is an organically grown, 20 year old code base, mostly in **`C, perl, flat text file backend`**
- Reliability issues with irrd2/irrd3 (fix is to restart the daemon…)
- Absolutely critical to NTT's daily operations - **All** NTT Communications' prefix-filters come from irrd instances!

```
job@vurt irrd$ cloc .
     189 text files.
     185 unique files.
      28 files ignored.

github.com/AlDanial/cloc v 1.74  T=2.25 s (71.9 files/s, 36938.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                               92           6645           9205          33967
Perl                            10            812            877          12451
Bourne Shell                     4            993           1308           9687
C/C++ Header                    35            722            549           3608
yacc                             1            326            111           1453
make                            20            168             63            313
-------------------------------------------------------------------------------
SUM:                           162           9666          12113          61479
```

Why IRRd v4

Looking forward

- No possibility for innovation, extensions
- IRRd v3 is ‘dumb’ middleware, literal transposition between input and output - need to move to ‘smart’ middleware
- For routing security road map see slides 11-18: http://www.lacnic.net/innovaportal/file/3135/1/lacnic30_snijders_routing_security_roadmap.pdf

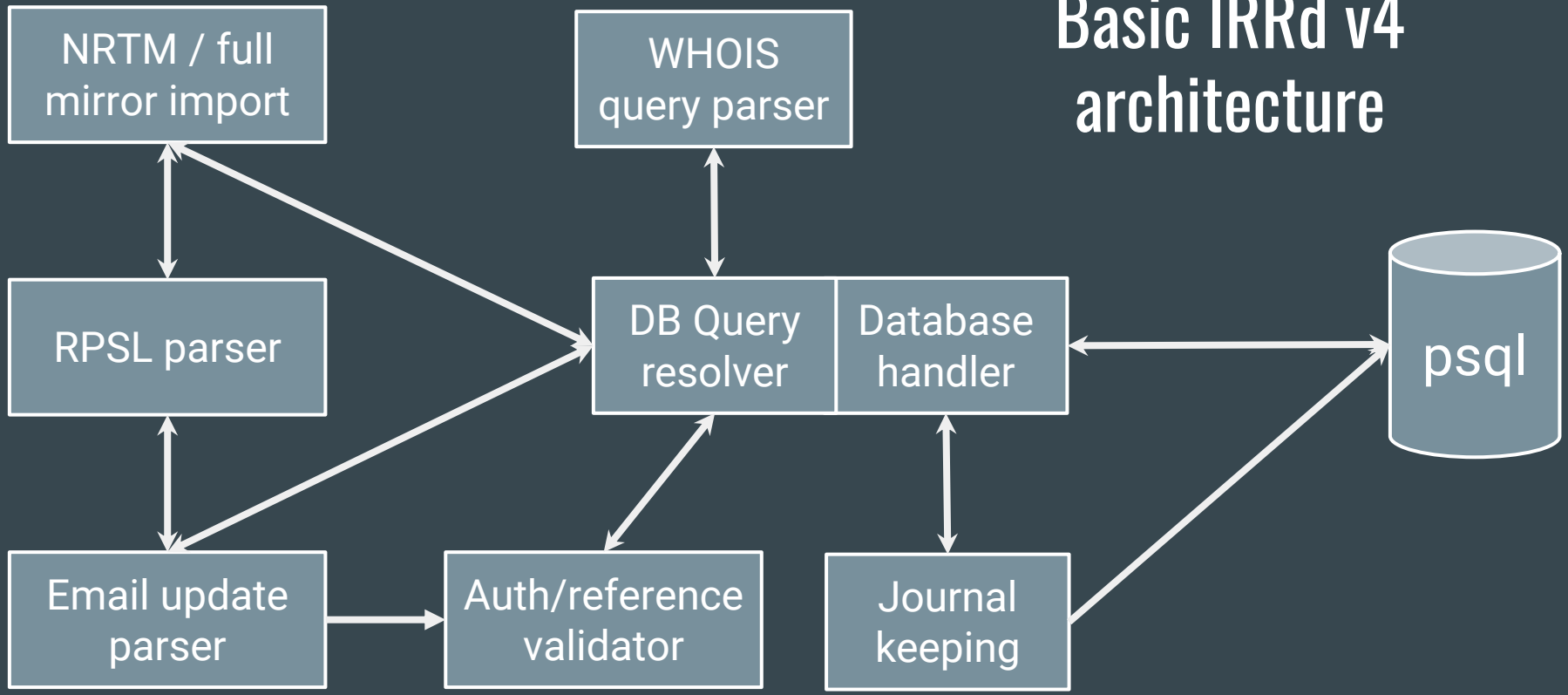
Goals for IRRd v4

- Single modern architecture with extension options
- Codebase that is well documented, consistent, maintainable
- Extensive test suite
- QA checks comparing to current `rr.ntt.net`

Development principles

- BSD 2-Clause licensed
- Complete rewrite
- Python 3.6 & PostgreSQL
- Planned for future expansion
- Using third party projects over inventing things ourselves
- Strict requirements on data validation and consistency
- “100%” coverage unit tests
- Extensive documentation

Basic IRRd v4 architecture



```
class RPSLAsSet(RPSLObject):
  fields = OrderedDict([
      ("as-set", RPSLSetNameField(
      primary_key=True, lookup_key=True, prefix="AS")),

   ("members", RPSLReferenceListField(
      lookup_key=True, optional=True, multiple=True,
      referring=["aut-num", "as-set"])),

   ("tech-c", RPSLReferenceField(
      lookup_key=True, optional=True, multiple=True,
      referring=["role", "person"])),

   ("mnt-by", RPSLReferenceListField(lookup_key=True,
      multiple=True, referring=["mntner"])),
  ..., ..., ...])
```

Metadata extraction (as-set)

```
rpsl_pk        = AS-RIPENCC
source         = RIPE
object_class   = as-set
parsed_data    = {
   "as-set": "AS-RIPENCC",
   "mnt-by": ["RIPE-NCC-MNT"],
   "source": "RIPE",
   "tech-c": "DUMY-RIPE",
   "admin-c": "DUMY-RIPE",
   "members": ["AS3333", "AS2121", "AS12654"]}
```

Metadata extraction (route)

```
parsed_data = {"route": "193.0.0.0/21",
               "mnt-by": ["RIPE-NCC-MNT"],
               "origin": "AS3333",
               "source": "RIPE"}
ip_version  = 4
ip_first    = 193.0.0.0
ip_last     = 193.0.7.255
ip_size     = 2048
asn_first   = 3333
asn_last    = 3333
```

Challenges

(all solved)

- Almost everyone deviates from RFCs in slightly different ways
- RPSL has a few strange features, and doesn't always fit common storage methods
- Invalid objects occur in many databases (improved)
- NRTM is barely documented and inconsistently implemented
- Interdependent email updates

RPSL has many exciting features

```
inetnum:          192.0.2.0 - 192.0.2.255
source:           DEMO

-------------------------------------------

inetnum:          192.0.2.0

 # documentation prefix
+-
 192.0.2.255 # end
source:           DEMO
```

... and all kinds of invalid objects

```
aut-num: AS65536
notify:  EXAMPLE-MNT
notify:  our address is info@example.com
source:  DEMO
```

That’s not an AS number

```
route:  192.0.2.0/24
origin: 64496
source: DEMO
```

ASDOT never left

```
as-set:  AS-EXAMPLE
members: AS64496, AS64497
members: AS1.0, AS1.1
source:  DEMO
```

Note that only the ASCII character set can be used.

```
mntner:             EXAMPLE-MNT
upd-to:             ***@example.com
upd-to:             ***@example.com
auth:               MD5PW # Filtered
mnt-by:             EXAMPLE-MNT
source:             DEMO
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ERROR: Line 3: encountered empty line in the
middle of object: []
```

Note that only the ASCII character set can be used.

FIXED

```
mntner:       EXAMPLE-MNT
upd-to:       ***@example.com

upd-to:       ***@example.com
auth:         MD5-PW # Filtered
mnt-by:       EXAMPLE-MNT
source:       DEMO
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

ERROR: Line 3: encountered empty line in the middle of object: []

bold _italics_ ~strike~ `code` ```preformatted``` >quote

Incomplete objects in NRTM (not only RADB)

```
%START Version: 3 radb 3917323-3917323

DEL 3917323

route:          116.86.232.0/21
descr:          StarHub Ltd - NGNBN Network
origin:         AS55430

%END RADB
```

Statistics

(based on deployment like rr.ntt.net)

- 2.6 million RPSL objects
- One NRTM update per 40s
- Average object 427 bytes
- Largest object is 3.4MB
- 1.6 GB PostgreSQL data, 1.4 GB PostgreSQL indexes
- Log of 330.000 rr.ntt.net queries to run QA tests/comparisons

Current IRRDv4 status

Phase 1: replace existing IRRD with same features

- Project ~75-80% complete
- ~4000 lines of code
- ~3000 lines of 214 tests
- (Mostly) completed:
  - RPSL parsing and validation
  - Object storage, indexing, search and journal keeping.
  - IRRD/RIPE whois queries
  - Mirroring other sources
- Logo coming soon

Plans for phase 2

- Use of RPKI to negate conflicting IRR information
  - (ala RIPE 2018-06 Policy Proposal)
- Business rules to clean up / reject objects
  - Bogon prefixes & ASNs, special prefixes
- Integration with or validation against authoritative RIR databases

Plans for phase 2

- HTTPS API
- NRTMv4
- Better query language, perhaps GraphQL?
- …. (your input here) ….

Thank you!

https://github.com/irrdnet/irrd4

...

Job Snijders
NTT Communications
job@ntt.net
@jobsnijders

Sasha Romijn
DashCare
sasha@dashcare.nl
@mxsash