BGP Scanner

Isolario BGP-MRT Data Reader: C library & tool

Alessandro Improta
alessandro.improta <at> iit.cnr.it

Registro.it
L'anagrafe dei domini .it

Isolario Project

Alpha Cogs
Grata rerum novitas

IIT

Last episode in Tampere...

Objective: push more ASes to join

The more the ASes, the more the completeness of public BGP data

Isolario - The Book of Islands

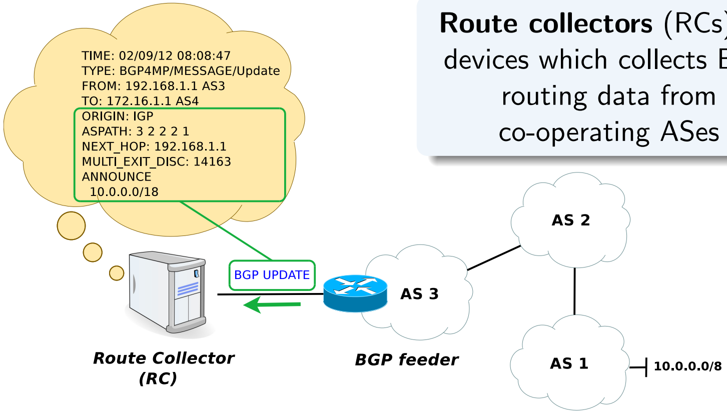
"where we discuss about all islands of the world, with their ancient and modern names, histories, tales and way of living..."

Benedetto Bordone
(Italian cartographer)

Approach: Do-ut-des

- Participants open at least one v4/v6 BGP session with Isolario providing their **full** routing table
- In change, Isolario offers **real-time** and **offline** applications based on the aggregation of every routing information collected

What is a BGP route collector?



Route collectors (RCs) are devices which collects BGP routing data from co-operating ASes

- Multi-Threaded Routing Toolkit format (RFC 6396)
- Maintains a routing table (RIB) with the best routes received
- Dumps the content of the RIB and received UPDATEs periodically

Route collecting projects

University of Oregon Route Views Project

Route Views was conceived as a tool for Internet operators to obtain real-time information about the global routing system from the perspectives of several different backbones and locations around the Internet. It collects BGP packets in MRT format since 2001
http://www.routeviews.org

RIPE NCC Routing Information Service (RIS)

The RIPE NCC collects and stores Internet routing data from several locations around the globe, using RIS. It collects BGP packets in MRT format since 1999
https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

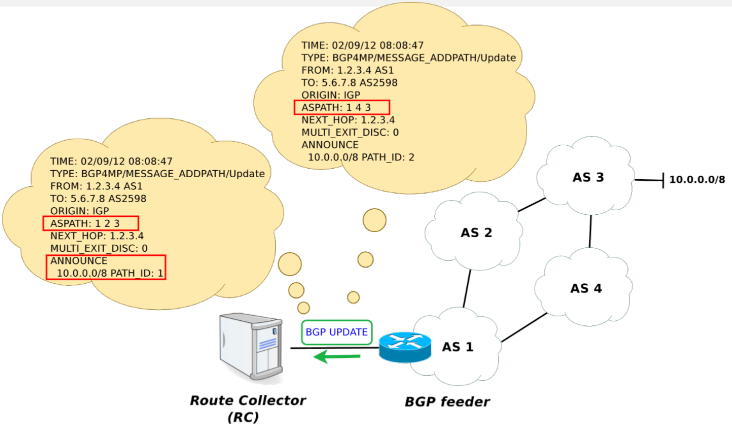
Packet Clearing House (PCH)

PCH is the international organization responsible for providing operational support and security to critical Internet infrastructure, including Internet exchange points and the core of the domain name system. It operates route collectors at more than 100 IXPs around the world and its data is made available in MRT format since 2011
https://www.pch.net/resources/Raw_Routing_Data

Isolario

Isolario is a route collecting project which provides inter-domain real-time monitoring services to its participants. It collects BGP packets in MRT format since 2013, and supports ADDPATH (RFC 7911) since 2018
https://www.isolario.it

Isolario numbers with ADD-PATH (Feb 21st, 2019)



```
TIME: 02/09/12 08:08:47
TYPE: BGP4MP/MESSAGE_ADDPATH/Update
FROM: 1.2.3.4 AS1
TO: 5.6.7.8 AS2598
ORIGIN: IGP
ASPATH: 1 4 3
NEXT_HOP: 1.2.3.4
MULTI_EXIT_DISC: 0
ANNOUNCE
  10.0.0.0/8 PATH_ID: 2
```

```
TIME: 02/09/12 08:08:47
TYPE: BGP4MP/MESSAGE_ADDPATH/Update
FROM: 1.2.3.4 AS1
TO: 5.6.7.8 AS2598
ORIGIN: IGP
ASPATH: 1 2 3
NEXT_HOP: 1.2.3.4
MULTI_EXIT_DISC: 0
ANNOUNCE
  10.0.0.0/8 PATH_ID: 1
```

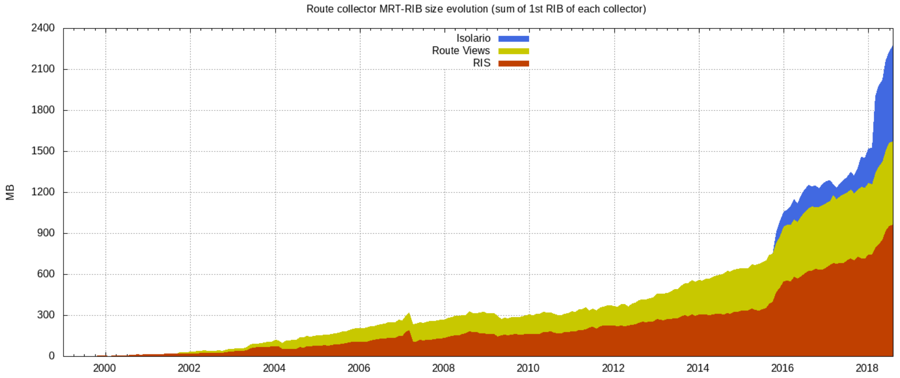
Isolario participants in ADD-PATH

- **# ASes:** 23
- **# sessions configured:** 54
  - **# IPv4:** 25
  - **# IPv6:** 29

Isolario full feeders in ADD-PATH

- **# ASes:** 94
  - **# IPv4:** 80 (**Total:** 151)
  - **# IPv6:** 82 (**Total:** 161)

MRT data is getting bigger and bigger...



What is the problem?

- Tools available are either slow, outdated or miss ADD-PATH handling
- Usually no way to filter packets

Tools available to parse MRT data

Several languages: C, C++, Python, Perl, Java, OCaml

Tool	Lang	RIB	updates	IPv6	ADD PATH	Last updated
bgpdump	C	✓	✓	✓	✓	2018-03-02
bgpdump2	C	✓	×	✓	×	2016-08-10
bgpparser	C++	✓	✓	✓	×	2015-04-11
bgpreader	C	✓	✓	✓	×	2018-07-13
gobgp (MRT)	Go	-	-	-	-	2015-02-21
mabo	OCaml	✓	✓	✓	×	2017-06-26
mrtparse	Python	✓	✓	✓	×	2018-06-27
PyBGPdump	Python	✓	✓	×	×	2007-01-15
Java-MRT	Java	✓	✓	✓	×	2013-02-09
zebra-dump-parser	Perl	✓	✓	✓	×	2016-11-07

Solution: `goto c`

The fact is, that is **exactly** *the kinds of things that C excels at. Not just as a language, but as a required* **mentality**. *One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*

Linus

C language benefits

- May be easily wrapped (C++, Python, LUA)
- Close to "metal"

Not only ANSI C: C99

- Allows dynamic allocation on stack ($\rightarrow$ **zero-copy**)
- Improves code readability

Solution: `goto c`

The fact is, that is ***exactly*** *the kinds of things that C excels at. Not just as a language, but as a required* ***mentality****. One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*

Linus

C language benefits

- May be easily wrapped (C++, Python, LUA)
- Close to "metal"

Not only ANSI C: C99

- Allows dynamic allocation on stack ($\rightarrow$ **zero-copy**)
- Improves code readability

Solution: `goto c`

The fact is, that is **exactly** *the kinds of things that C excels at. Not just as a language, but as a required* **mentality***. One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*

Linus

C language benefits

- May be easily wrapped (C++, Python, LUA)
- Close to "metal"

Not only ANSI C: C11

- Allows optimization for multithreading
- Thread local/atomic variables

BGP Scanner: Isolario MRT-BGP data reader

Don't worry!

You do not have to use C

MRT-BGP library

- A highly optimized low-level reusable library
- Optimized to achieve high throughput
- Multi-thread friendly
- Memory friendly

The real star of the show: BGP Scanner tool

- Comes with all the benefits of the low-level C library
- Good old `grep` friendly output
- Can be piped to other tools
- Supports `gz`, `bz2`, `xz` and raw
- Powerful filtering features

Wait... what?

Filtering

- Peer IP, Peer AS
- Subnets, supernets, related, exacts
- Peer Index
- AS path regexp and loop detection
- Can be configured with template files and/or directly by command line

Example

```
bgpscanner -s 192.65.0.0/16 -p "174 137" rib.20180701.1400.bz2

=|192.65.131.0/24|7018 174 137 137 137 2598|12.0.1.63|i|||7018:5000 7018:37232|12.0.1.63 7018|1530186440|1
=|192.65.131.0/24|6539 577 174 137 137 137 2598|216.18.31.102|i||||216.18.31.102 6539|1529912797|1
=|192.65.131.0/24|701 174 137 137 137 2598|137.39.3.55|i||||137.39.3.55 701|1529397335|1
=|192.65.131.0/24|3741 174 137 137 137 2598|168.209.255.56|i||||168.209.255.56 3741|1529593095|1
=|192.65.131.0/24|11686 174 137 137 137 2598|96.4.0.55|i||||96.4.0.55 11686|1530338943|1

...
```

Subnets of 192.65.0.0/16 crossing 174 137 link

It can do a lot more...

```
Available options:
-a <feeder AS>
        Print only entries coming from the given feeder AS
-A <file>
        Print only entries coming from the feeder ASes contained in file
-d
        Dump packet filter bytecode to stderr (debug option)
-e <subnet>
        Print only entries containing the exact given subnet of interest
-E <file>
        Print only entries containing the exact subnets of interest contained in file
-f
        Print only every feeder IP in the RIB provided
-i <feeder IP>
        Print only entries coming from a given feeder IP
-I <file>
        Print only entries coming from the feeder IP contained in file
-l
        Print only entries with a loop in its AS PATH
-L
        Print only entries without a loop in its AS PATH
-o <file>
        Define the output file to store information (defaults to stdout)
-p <path expression>
        Print only entries which AS PATH matches the expression
-P <path expression>
        Print only entries which AS PATH does not match the expression
-r <subnet>
        Print only entries containing subnets related to the given subnet of interest
-R <file>
        Print only entries containing subnets related to the subnets of interest contained in file
-s <subnet>
        Print only entries containing subnets included to the given subnet of interest
-S <file>
```

Benchmarks: BGP data evolution scenario

Test machine

- Intel(R) Core(TM) i7-4790K 4.00GHz
- RAM 16GB
- Samsung SSD 850 EVO 500GB
- Debian Stretch

Data sources

- Route Views `route-views6`
- RIS `rrc00`
- Isolario `Korriban`

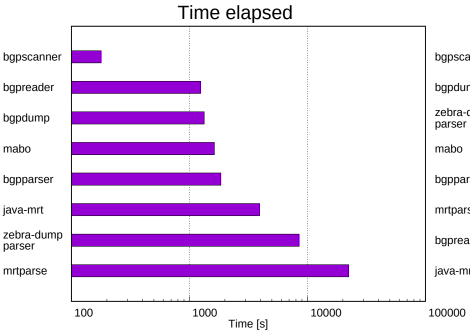
Benchmark phases (∀ collectors)

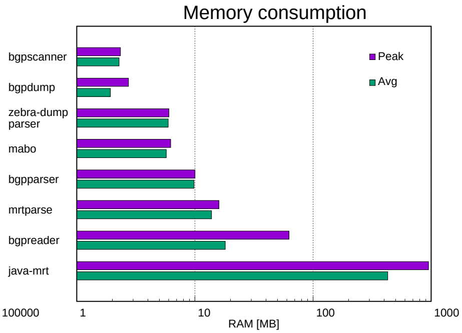
1. Download first RIB of July, 2018
2. Download all updates of July, 2018
3. Decompress
4. Run 10 times each MRT tool
5. Compute average results of runs for each metric

Data is decompressed to eliminate decompression algorithm overhead

Route Views `route-views6` collector

RIB size	$\sum$ UPDATE size	μ UPDATE size	# files
99MB	25.65GB	8.82MB	2977

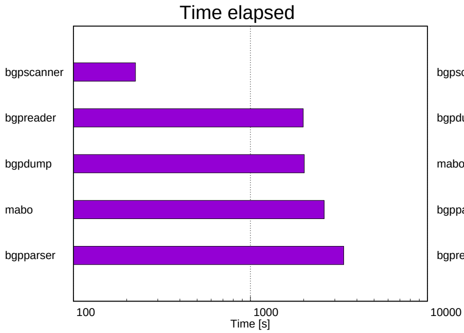


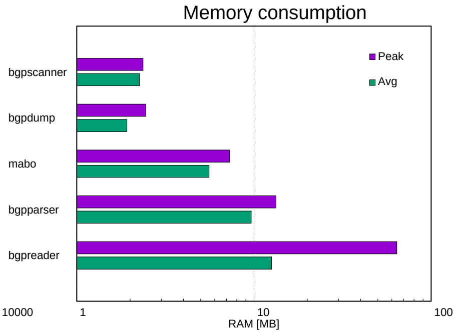


Only IPv6 feeders
(26 sessions, 24 full tables)

RIS `rrc00` collector

RIB size	$\sum$ **UPDATE size**	μ **UPDATE size**	**# files**
1.1GB	33.6GB	3.85MB	8930



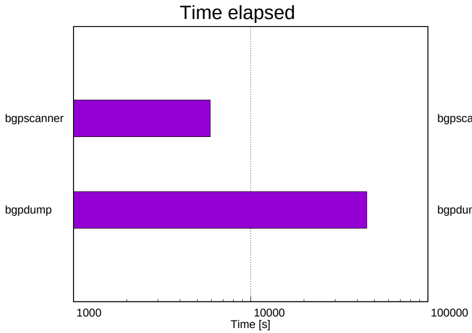


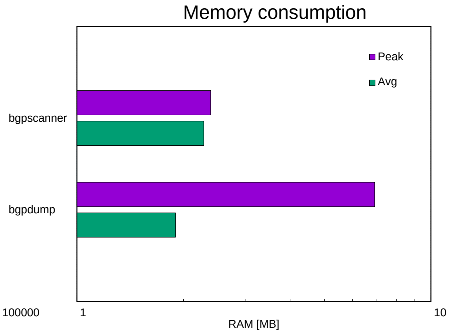
IPv4 + IPv6 feeders (39 sessions)

- 22 IPv4 sessions (21 full tables)
- 17 IPv6 sessions (14 full tables)

Isolario `Korriban` collector

RIB size	$\sum$ **UPDATE size**	μ **UPDATE size**	**# files**
5.7GB	810.64GB	92.97MB	8930





IPv4 + IPv6 feeders with ADDPATH

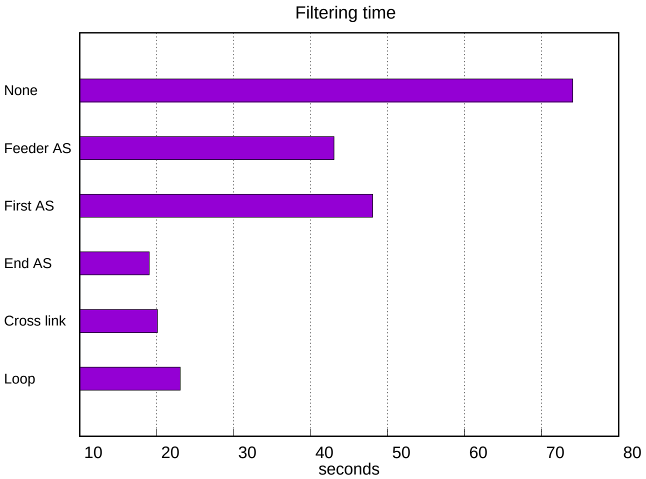
- 512 IPv4 sessions (112 full tables)
- 407 IPv6 sessions (126 full tables)

Thanks to NLNOG RING for providing 68 IPv4 and 69 IPv6 full tables!

Filtering benchmark

Data source

- Last RIB of July 2018 of `Korriban` collector
- 475MB (7.8GB uncompressed)



A possible IXP use case

Assumption

- No IXP peering LAN should be announced on the Internet
- Thus, no route collector should see any IXP peering LAN

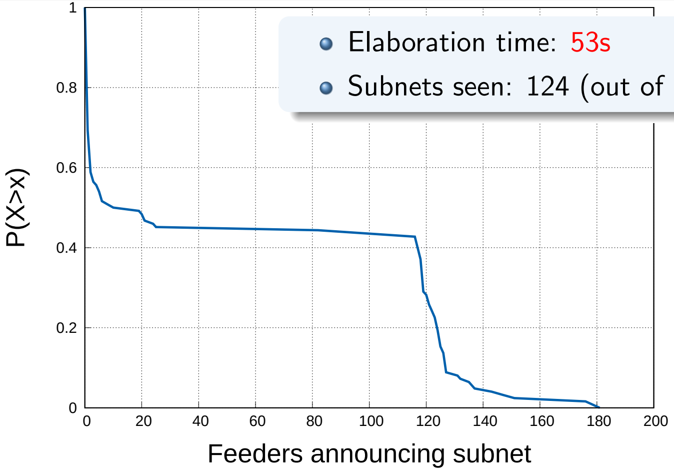
Procedure

1. Extract peering LANs via PeeringDB APIs
2. `bgpscanner -E <Peering LAN file> <RIB>`
3. Analyse the results (e.g. to check for particular LANs of interest)

Data

- Monitor: `Korriban`
- File: `rib.20190212.0000`

A possible IXP use case: results



- Elaboration time: 53s
- Subnets seen: 124 (out of 1176)

Peering LANs could be announced on purpose (for any reason)

Bgpscanner can help in identifying those which are not

How to install BGP Scanner?

BGP Scanner is open-source

- BSD license
- Available on `www.isolario.it` → Tools
- Source code on `https://gitlab.com/Isolario`

Install procedure (Source Tarball)

1. Download bgpscanner-`[version]`.tar.gz
2. `./configure && make && make install`

Install procedure (Debian package archives)

1. Download libisocore1_`[version]`.deb
2. Download bgpscanner_`[version]`.deb
3. dpkg -i *.deb

Thank you for your attention

Any question?

info <at> isolario.it
alessandro.improta <at> iit.cnr.it

https://www.isolario.it
http://www.alphacogs.com

Links and benchmark configuration

Tool	Version	Language	RIB
bgpdump	1.5.0	C	`https://bitbucket.org/ripencc/bgpdump/overview`
bgpdump2	2.0.1	C	`https://github.com/yasuhiro-ohara-ntt/bgpdump2`
bgpparser	?	C++	`https://github.com/cawka/bgpparser`
bgpreader	1.2.1	C	`https://github.com/caida/bgpstream`
gobgp (MRT)	?	Go	`https://git.2f30.org/go-bgp/`
mabo	?	OCaml	`https://github.com/ANSSI-FR/mabo`
mrtparse	1.6	Python	`https://github.com/t2mune/mrtparse`
PyBGPdump	?	Python	`https://jon.oberheide.org/pybgpdump/`
Java-MRT	?	Java	`https://github.com/paaguti/java-mrt`
zebra-dump-parser	?	Perl	`https://github.com/rfc1036/zebra-dump-parser`

Tool	Command
bgpscanner	`bgpscanner -m FILE > /dev/null 2>&1`
bgpdump	`bgpdump -m FILE > /dev/null 2>&1`
bgpdump2	`bgpdump2 -m FILE > /dev/null 2>&1`
bgpparser	`bgpparser -B FILE > /dev/null 2>&1`
bgpreader	`bgpreader -d singlefile -o rib-file/upd-file,FILE > /dev/null 2>&1`
mabo	`mabo dump FILE > /dev/null 2>&1`
mrtparse	`mrt2bgpdump.py FILE > /dev/null 2>&1`
Java-MRT	`mrt.jar org.javamrt.progs.route_btoa FILE > /dev/null 2>&1`
zebra-dump-parser	`zebra-dump-parser.pl FILE > /dev/null 2>&1`

Filtering benchmark

Filters

- BGP data announced by feeder `AS199036`
  - `bgpscanner -a "199036"`
- First AS of `AS_PATH` is `AS199036`
  - `bgpscanner -p "^199036"`
- Last AS of `AS_PATH` `AS3333`
  - `bgpscanner -p "3333$"`
- `AS_PATH` crosses link `AS174 AS3356`
  - `bgpscanner -p "174 3356"`
- Subnets of `193.0.0.0/16` or `2001:67c::/32` destined to `AS3333`
  - `bgpscanner -s "193.0.0.0/16" -s "2001:67c::/32" -p "3333$"`
- `AS_PATH` contains a loop
  - `bgpscanner -l`

PeeringDB query

SQL

```
SELECT p.prefix
FROM peeringdb_ixlan_prefix as p, peeringdb_ixlan as i, peeringdb_ix as n
WHERE p.ixlan_id = i.id AND i.ix_id = n.id;
```