National Research Council of Italy

Isolario Project

IIT

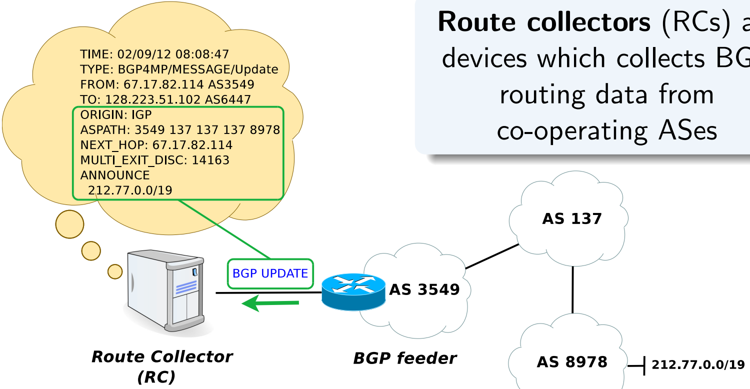
ICE

Interactive Collector Engine

Luca Sani

luca.sani@iit.cnr.it

BGP route collectors



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

A route collector

- Maintains a routing table (RIB) with the best routes received
- Dumps the content of the RIB and received UPDATEs periodically

BGP route collector projects

University of Oregon Route Views Project

Route Views was originally 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 since 1997, in MRT format since 1997
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
https://www.isolario.it

Real-time requirements

Off-line analysis of data . . .

Route collectors were originally conceived as a tool for network administrators to **obtain information** about the Internet inter-domain routing status

. . . vs real-time monitoring

Depending on the application, some information must be obtained on-the-fly, i.e. without the delay caused by **storing** the data

Example

Check the routes a feeder is using to reach a given portion of IP space

Check MRT data?

- RIB snapshots are available every 2-8 hours
- Update messages are available every 5-15 minutes

Real-time requirements

Is it possible to perform real-time queries on the RC RIB?

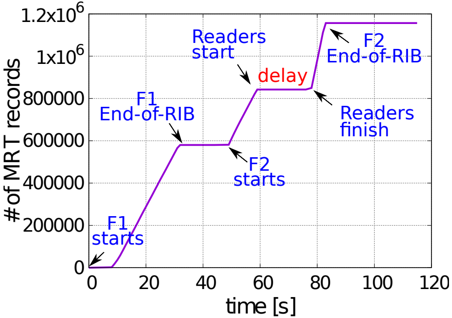
Current situation

Typically RCs run general-purpose routing software, e.g. Quagga

Cons

- The collection process is affected by the queries because most RC software is **single-threaded**
- Overhead in terms of CPU and memory usage due to BGP specs (e.g. BGP decision process)

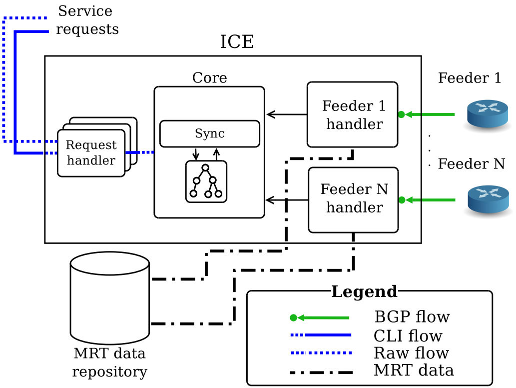
Example with Quagga



- $t = 0$ Feeder F_1 starts a RIB transfer (ends $t = 35$, $\sim$ 580k prefixes)
- $t = 45$ Feeder F_2 starts another RIB transfer
- $t = 60$ **ten** F_1 full table read operations are issued sequentially

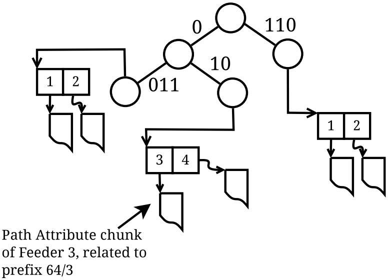
- Data collection is delayed of about 20 seconds
- After the read requests, packets arrives with higher rate

ICE: an Interactive Collector Engine



- Each BGP session is handled by a dedicated set of threads
- Each service request is handled by a dedicated thread
- Readers and writers sync on the RIB according to the classic readers-writers paradigm

RIB implementation: Patricia Trie



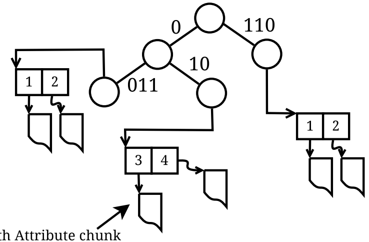
- Each node represents a subnet
- Each subnet has associated a set of path attributes, one for each feeder that announced the subnet

Readers and Writers

- Writer: feeder thread
- Reader: request thread

Writers/Readers can *W_lock*/*R_lock* both the whole trie **and** single nodes

RIB implementation: Patricia Trie



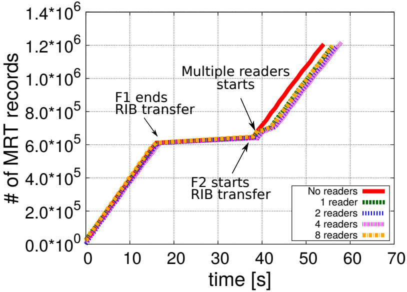
Writer

1. Checks if the node is present (**R_lock** RIB)
2. If not, inserts new node (**W_lock** RIB)
3. Inserts/updates the path attribute (**W_lock** node)

Reader

1. Checks if the node is present (**R_lock** RIB)
2. If yes, reads the node (**R_lock** node)

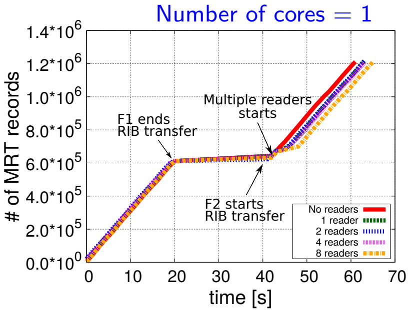
Test: delay in storing BGP packets

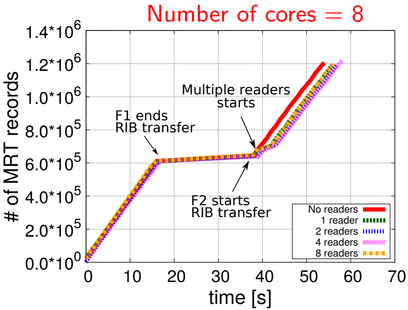


- $t = 0$ Feeder F_1 starts a RIB transfer
- $t{\sim}40$ Feeder F_2 starts a RIB transfer
- $t{\sim}41$ Multiple F_1 full table read operations are issued **simultaneously**

Thanks to the scheduler activity (and the multi-threaded design) ICE is able to write incoming packets while readers are reading

Test: delay in storing BGP packets





- $t = 0$ Feeder F_1 starts a RIB transfer
- $t \sim 40$ Feeder F_2 starts a RIB transfer
- $t \sim 41$ Multiple F_1 full table read operations are issued **simultaneously**

Thanks to the scheduler activity (and the multi-threaded design) ICE is able to write incoming packets while readers are reading

Test: delay in reading the RIB

# of readers	Before F_2 RIB transfer				During F_2 RIB transfer			
	1 core		8 cores		1 core		8 cores	
	μ	σ	μ	σ	μ	σ	μ	σ
1	2.12	0.07	2.09	0.07	2.21	0.07	2.18	0.08
2	2.15	0.10	2.14	0.10	2.19	0.09	2.16	0.08
4	2.32	0.15	2.20	0.11	2.28	0.13	2.17	0.11
8	3.66	0.06	2.15	0.14	3.73	0.09	2.13	0.15

- Similar to the previous test, but from the reader side
- How much time a reader takes to retrieve the full routing table of F_1 before and **during** F_2 table transfer?
- μ and σ are respectively the average time and the standard deviation

The time the second set of readers takes to retrieve F_1 table is very close to the time taken by the first set, confirming that multiple readers can proceeed in parallel with F_2

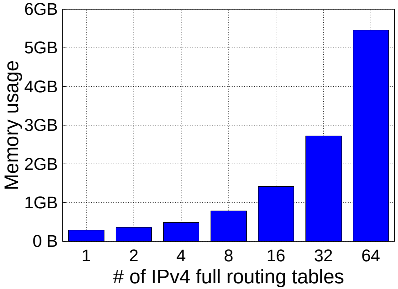
Test: delay in reading the RIB

# of readers	Before F_2 RIB transfer				During F_2 RIB transfer			
	1 core		8 cores		1 core		8 cores	
	μ	σ	μ	σ	μ	σ	μ	σ
1	2.12	0.07	2.09	0.07	2.21	0.07	2.18	0.08
2	2.15	0.10	2.14	0.10	2.19	0.09	2.16	0.08
4	2.32	0.15	2.20	0.11	2.28	0.13	2.17	0.11
8	**3.66**	0.06	2.15	0.14	**3.73**	0.09	2.13	0.15

- Similar to the previous test, but from the reader side
- How much time a reader takes to retrieve the full routing table of F_1 before and **during** F_2 table transfer?
- μ and σ are respectively the average time and the standard deviation

The time the second set of readers takes to retrieve F_1 table is very close to the time taken by the first set, confirming that multiple readers can proceeed in parallel with F_2

What about memory?



Memory consumption

- ICE uses ~ 82.4MB per feeder, ~100 feeders on a standard machine
- This means that in scenarios where the feeders are near to a thousand (e.g. route collecting, route servers) at least ten machines are needed

How to reduce memory consumption?

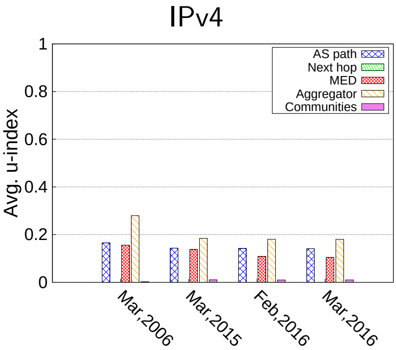
Repetitiveness of BGP data

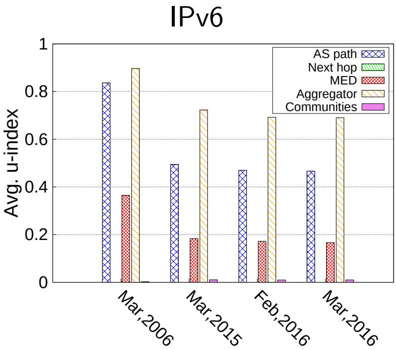
- The memory usage is mainly caused by the `PATH_ATTRIBUTE`s
- Is that possible to compress them? Are they repetitive?

- We analysed BGP data collected by Route Views, RIS and Isolario route collectors:
  - March 2nd, 2016
  - February 2nd, 2016
  - March 2nd, 2015
  - March 2nd, 2006 (only Route Views and RIS)
- We consider only full feeders data

- We computed an **uniqueness index** $u - index$ for each attribute
- Given a full feeder RIB and an attribute A, the $u - index$ is the ratio between the different values assumed by A and the number of occurrences of A

Repetitiveness of BGP data: results





- Attributes are more repetitive in IPv4 scenarios than IPv6
  - Only about one every five ASes appears in IPv6 routes
  - an AS announces on average two subnets vs ten in the IPv4 case
- NEXT_HOP is the most repetitive (of course)

Compression algorithm

Data can be compressed!

Requirements for the compression algorithm

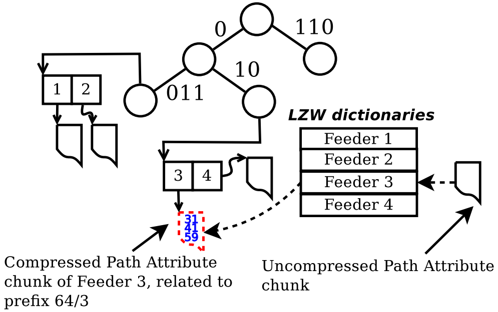
- Lossless
- Adaptive
- **Random access at record level**

Compression algorithms categories

- Entropy encoding
- **Dictionary encoding**

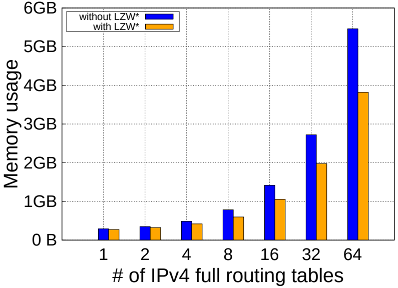
Recurring patterns are substituted with fixed-size indexes

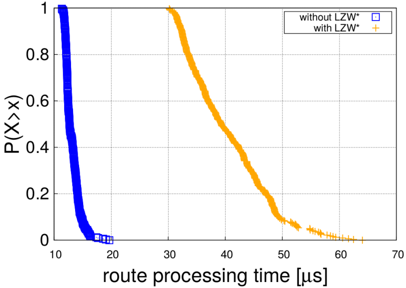
Lampel-Ziv-Welch



- One dictionary per feeder
- The dictionary is kept in memory
- The decompressor **does not** need to rebuild the dictionary (small compression variant applied here)

Results

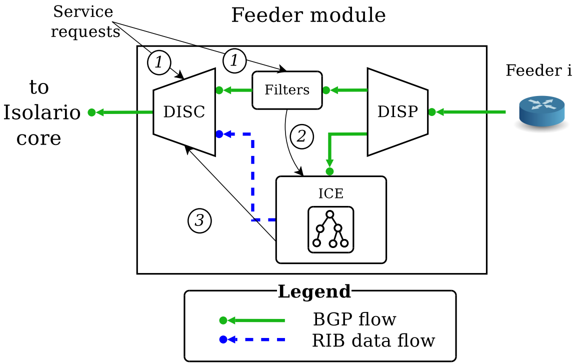




- Compression saves up to 30% of memory, i.e. 57.5MB per feeder
- Route processing time increases from 12μs-20μs to 30μs-64μs
  - Amount of time it takes a complete feeder table transfer

Isolario: an ICE use case

https://www.isolario.it



- Isolario is a route collector project which offers real-time monitoring services to its participants
- Services obtain the real-time inter-domain status of a feeder by querying ICE
- Evolution is obtained with subsequent `UPDATE` messages

Conclusion

Interactive Collector Engine

- We proposed a multi-threaded collector engine which takes care of memory consumption
- This is **not** a complete BGP daemon like Quagga or Bird
- It is designed to support real-time access to the routing table and simultaneous collection of data

Future

- Add support for...
  - `ADD_PATH` (RFC 7911)
  - `BMP` (RFC 7854)
- Route Server?
  - Add BGP decision process
  - Add import/export policies
- Use as a basic brick to implement a real-time looking glass on IXPs?
- Suggestions?

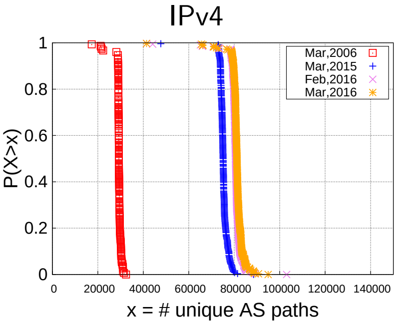
Thank you for your attention

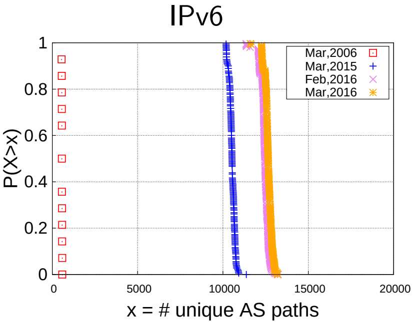
Any question?

luca.sani@iit.cnr.it
https://www.isolario.it

ICE is open-source and written in C++
It can be downloaded from **isolario.it**
(Tools section)

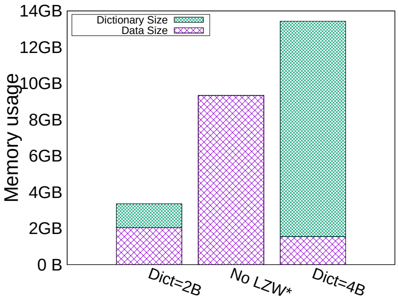
Repetitiveness of BGP data: AS path





- Each full feeder uses on average a number of distinctive AS paths which is much smaller than the average size of the full table
- Almost every full feeder uses the same **number** of distinctive AS paths
- The number of distinctive AS paths is about 1.5 times the number of ASes

Tuning the index size N



- N = 4B best compression but too large dictionary
- **N = 2B** data is still fairly compressed