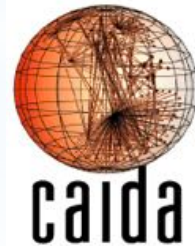


BGP STREAM

A framework for BGP data analysis



Alberto Dainotti, Alistair King, **Chiara Orsini**, Vasco Asturiano

chiara@caida.org

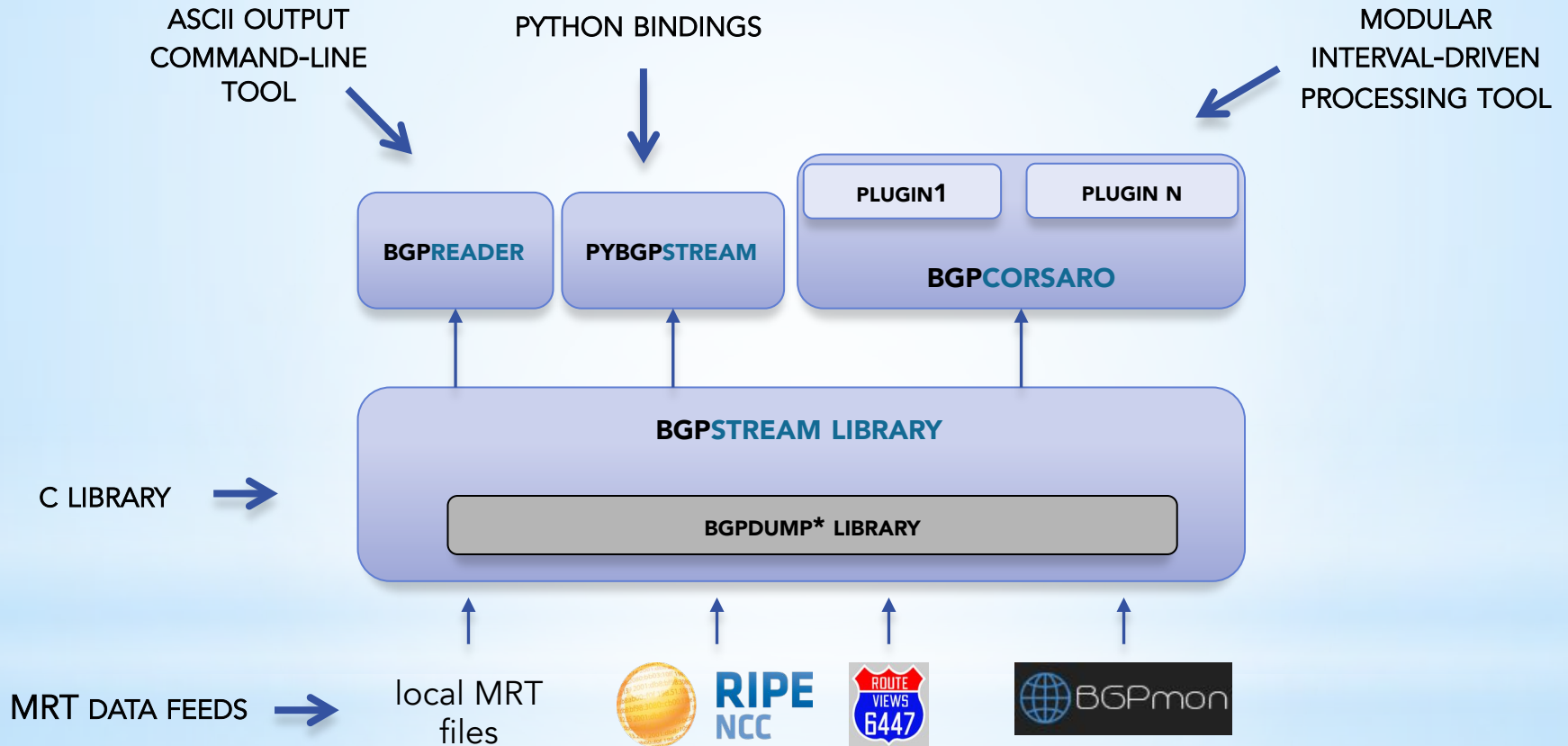
BGPSTREAM

A software framework for the historical analysis and real-time monitoring BGP data

Goals →

- * generate a sorted stream to support maintaining a BGP "state" over time
 - * abstract from underlying data sources
 - * filters BGP data based on user needs
 - * tag unreliable BGP data
 - * support real-time
-
- * work in progress, soon to be released as open-source
 - * *v1 release expected for this summer*

BGPSTREAM framework



Data Feeds

Transparent access to several *annotated* MRT data sources:

- * Previously-downloaded local files

- * Real-time stream from:

- * Colorado State's BGPmon (all Route Views) [work-in-progress for release v1]

- * RIPE RIS [discussion in-progress]

- * Historical and continuous download from RIPE RIS and RouteViews projects



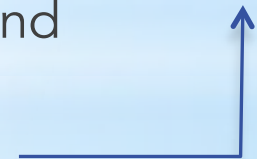
- * 13 active collectors
- * RIBS every 8 hours
- * Updates every 5 minutes



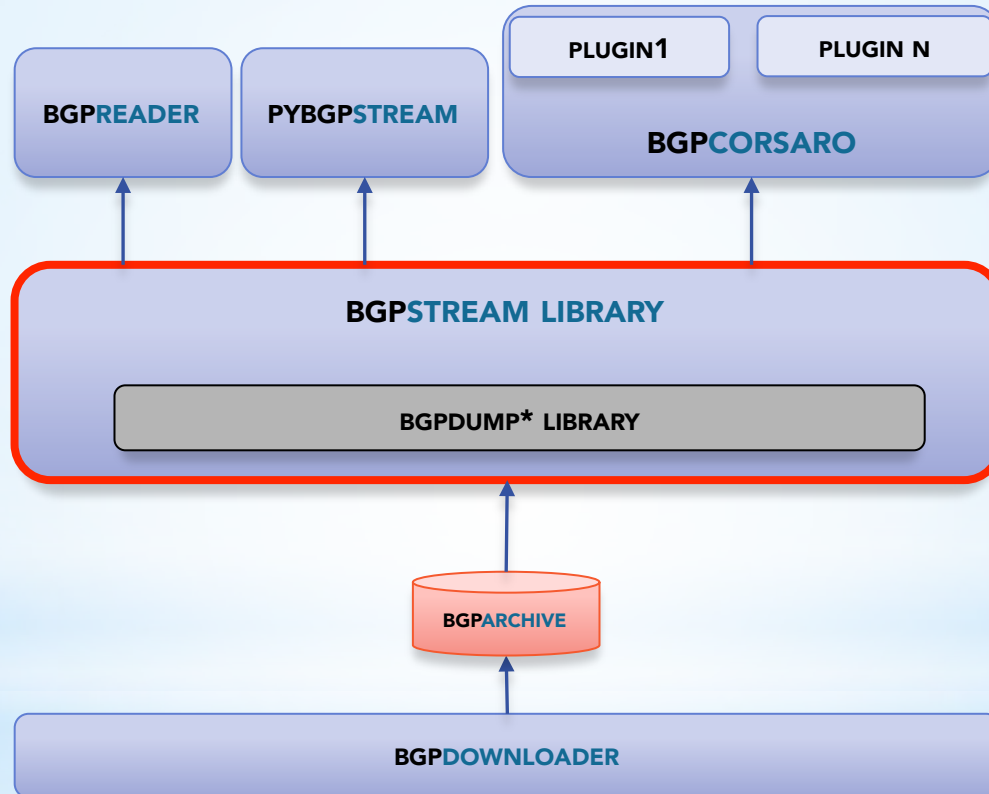
- * 17 active collectors
- * RIBS every 2 hours
- * Updates every 15 minutes

BGPDOWNLOADER

- * Perl program
- * ~20 mins average delay
- * meta data into a **BGPARCHIVE** (mySQL DB)
- * MRT files stored on hard disk



BGPSTREAM framework



BGPSTREAM library

- ① access the MySQL **BGPARCHIVE** and select files based on
 - * project
 - * type
 - * collector
 - * time
- ② use a modified version of **BGPDUMP** [1] to open group of dump files in parallel
- ③ extract **BGPRECORDS** from these files, i.e. wrappers around the **BGPDUMP ENTRY** format
- ④ marshal the **BGPRECORDS** according to their timestamp
- ⑤ optionally unwrap **BGPRECORDS** and extract atomic BGP information called **BGPELEMS**

BGP RECORD

- * PROJECT
- * BGP TYPE
- * COLLECTOR
- * DUMP TIME
- * DUMP POSITION
- * RECORD TIME
- * RECORD STATUS
- * **BGPDUMP ENTRY**

BGPARCHIVE metadata (common to entire dump)

position of entry in dump

- START
- MIDDLE
- END

time associated with the **BGPDUMP ENTRY**

status of **BGP RECORD**

- VALID
- CORRUPTED RECORD
- EMPTY SOURCE
- CORRUPTED SOURCE

set of MRT formatted entries

BGP RECORD → BGPELEM

- * PROJECT
- * BGP TYPE
- * COLLECTOR
- * DUMP TIME
- * DUMP POSITION
- * RECORD TIME
- * RECORD STATUS

*** BGPDUMP ENTRY**



BGP ELEM

* TYPE
* TIMESTAMP
* PEER IP ADDRESS
* PEER AS NUMBER
* IP PREFIX
* NEXT HOP
* AS PATH
* OLD STATE
* NEW STATE

	RIB entry	Announcement	Withdrawal	State message	
	✓	✓	✓	✓	} Common fields
	✓	✓	✓	✓	
	✓	✓	✓	✓	
	✓	✓	✓		} Type-dependent fields
	✓	✓			
	✓	✓			
				✓	
				✓	

BGPSTREAM library

```
#include "bgpstream_lib.h"

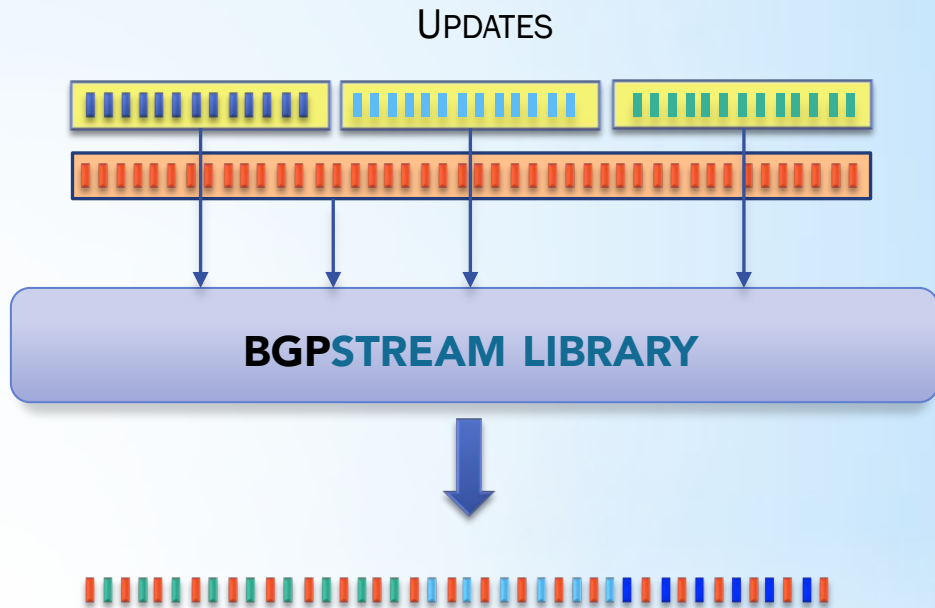
int main(int argc, char *argv[])
{
    bgpstream_t * bs = bgpstream_create();
    bgpstream_record_t *rec = \
        bgpstream_create_record();

    bgpstream_add_filter(bs, BS_COLLECTOR, "rrc00");
    bgpstream_add_filter(bs, BS_COLLECTOR, "route-views2");
    bgpstream_add_filter(bs, BS_BGP_TYPE, "updates");

    bgpstream_add_interval_filter(bs, BS_TIME_INTERVAL,
        "1410285600",
        "1412886500");

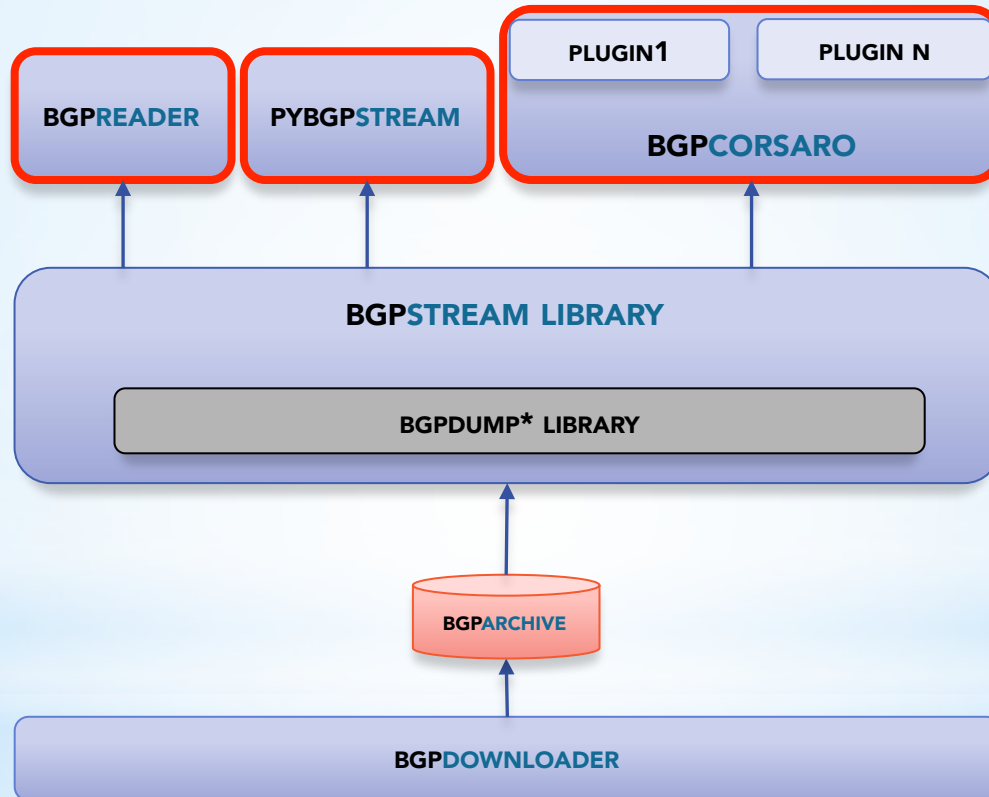
    int init_res = bgpstream_init(bs);
    while(bgpstream_get_next_record(bs, rec) > 0)
    {
        // [[ USE BGPRECORD HERE ]]
    }

    bgpstream_close(bs);
    bgpstream_destroy_record(rec);
    bgpstream_destroy(bs);
    return 0;
}
```

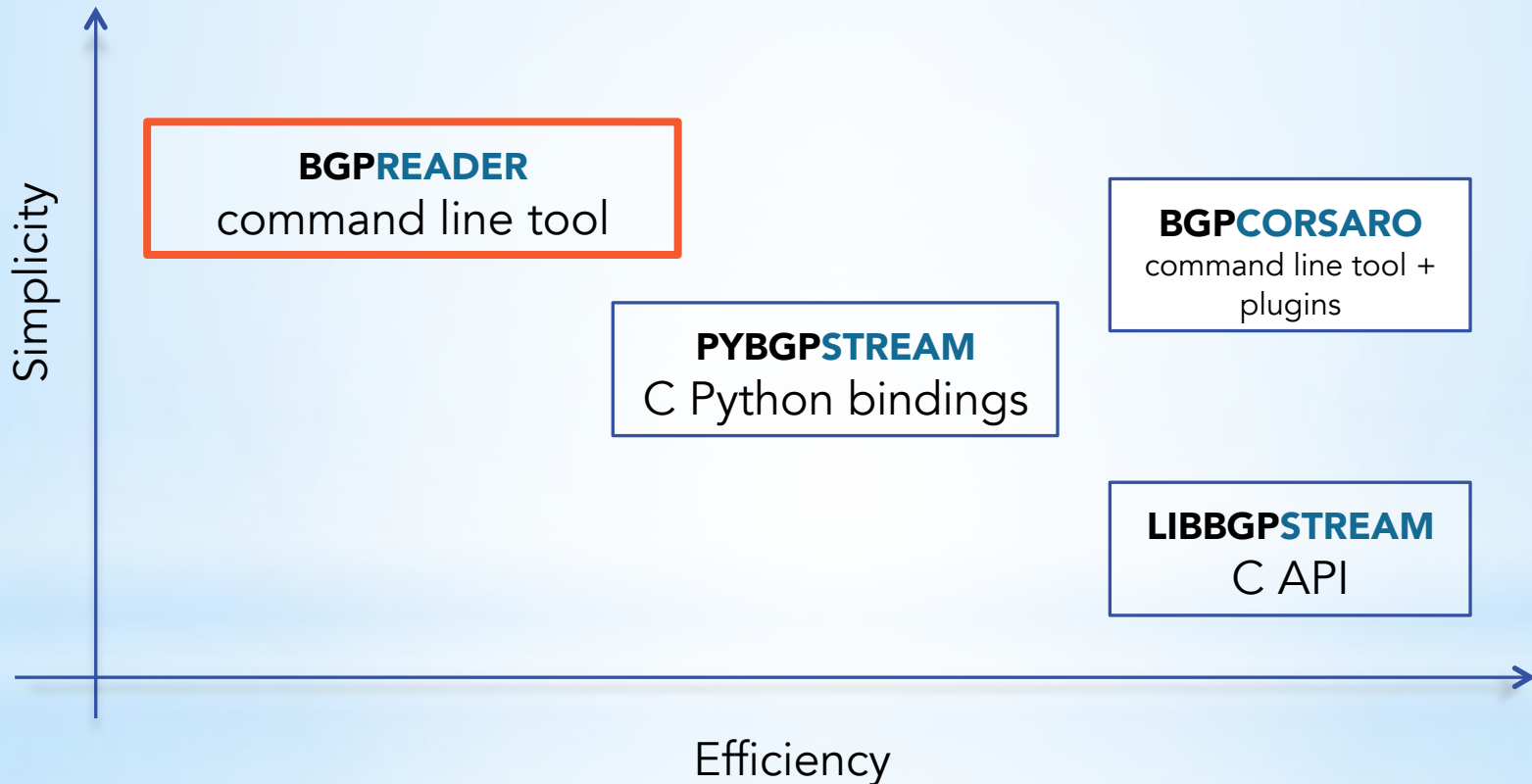


- * rely on metadata to decide how many dumps to open in parallel
- * sort based on **BGPRECORD** time

BGPSTREAM framework



BGPSTREAM just a C library?



BGPREADER

Metadata filters

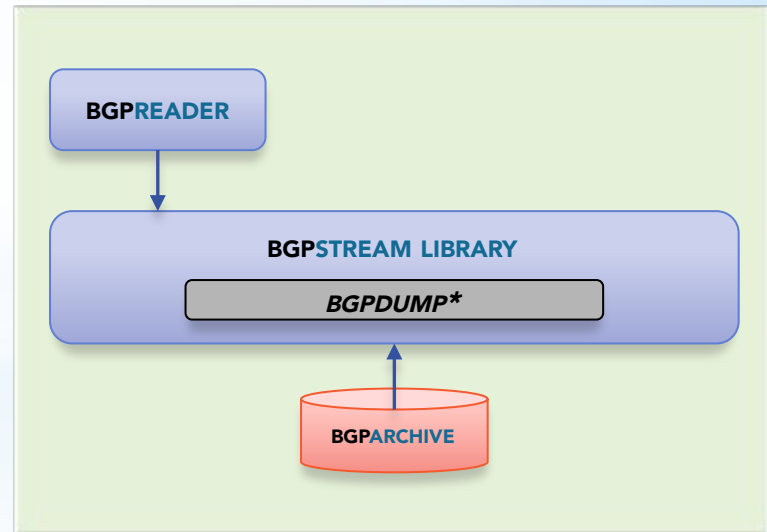


```
$ bgpreader -C rrc00 -C rrc03 -W1407808260,1407808440 -T updates -m
```

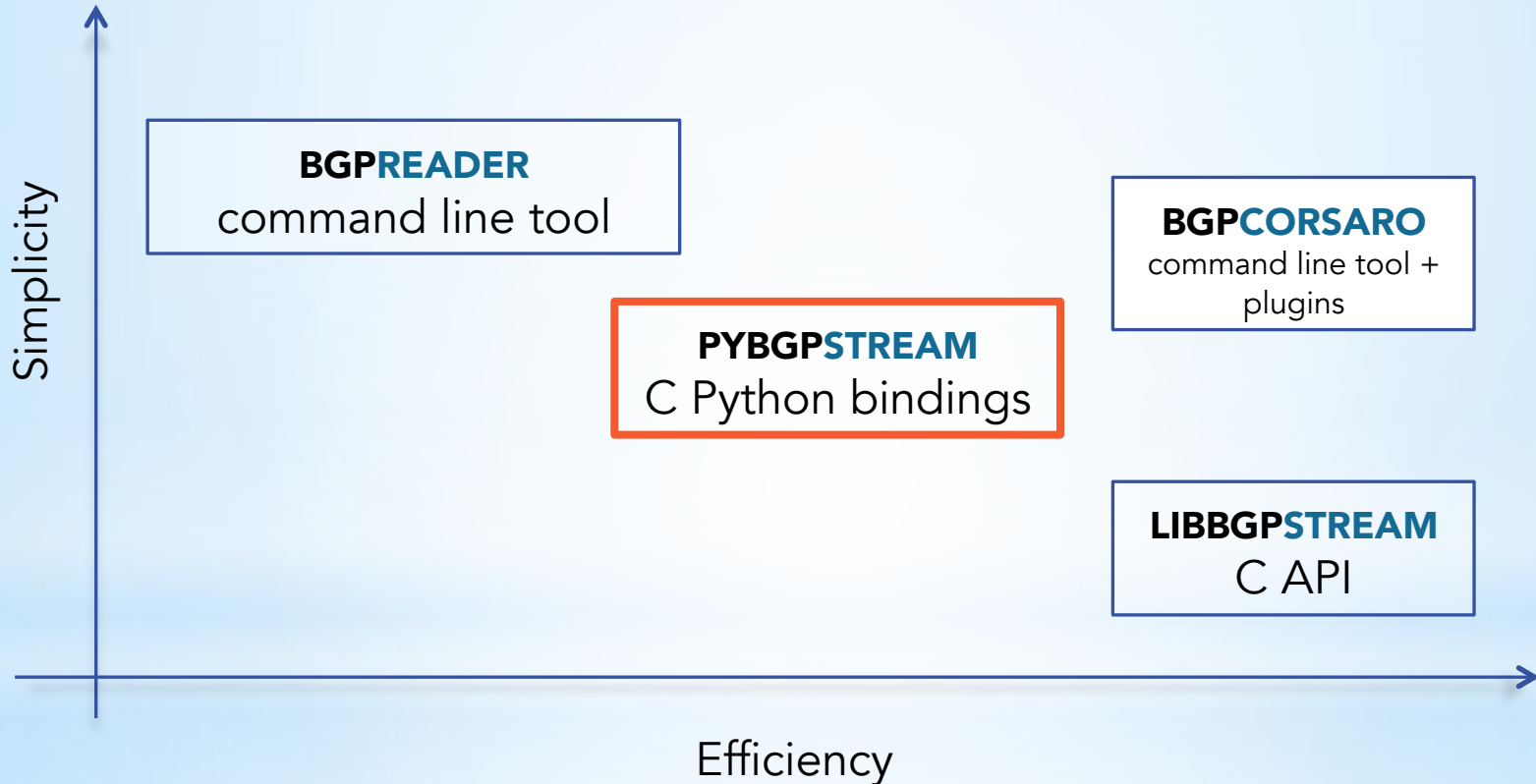
```
...  
1407808270|195.69.145.167|6453|A|202.70.88.0/21|195.69.145.167|6453 3549 9304 23752|23752||  
1407808270|218.189.6.2|9304|A|202.70.88.0/21|218.189.6.2|9304 6453 23752|23752||  
1407808270|12.0.1.63|7018|A|202.70.88.0/21|12.0.1.63|7018 6453 23752|23752||  
1407808270|195.69.145.167|6453|A|202.70.64.0/21|195.69.145.167|6453 23752|23752||  
1407808270|193.0.0.56|3333|A|202.70.88.0/21|193.0.0.56|3333 1257 6453 23752|23752||  
1407808270|195.69.144.200|12859|A|202.70.88.0/21|...  
1407808270|213.200.87.254|3257|A|190.55.32.0/20|...  
1407808270|213.200.87.254|3257|A|186.23.96.0/20|...  
1407808270|213.200.87.254|3257|A|190.55.48.0/20|...  
1407808270|213.200.87.254|3257|A|186.23.240.0/20|...  
1407808270|213.200.87.254|3257|A|186.23.160.0/20|...  
1407808270|213.200.87.254|3257|A|186.23.208.0/20|..  
...
```



- * **BGPDUMP** compatible output
- * **BGPREADER** output



BGPSTREAM just a C library?



PYBGPSTREAM

- * Python bindings
- * same API exported in C
- * no functionalities are lost

The screenshot shows the documentation for the `_pybgpstream` module. The page title is `_pybgpstream`. The main content describes the API of the `_pybgpstream` module, a low-level (almost) direct interface to the `libbgpstream` library. For most uses, the `pybgpstream` module should be used instead.

The `BGPStream` class is defined in `_pybgpstream.BGPStream`. The BGP Stream class provides a single stream of BGP Records.

The `add_filter(type, value)` method adds a filter to an unstarted BGP Stream instance. Only those records that match the filter(s) included in the stream.

If multiple filters of the **same** type are added, a record is considered a match if it matches **any** filters. E.g. if `add_filter('project', 'routeviews')` and `add_filter('project', 'ris')` are used, then records are from either the *Route Views*, or the *RIS* project will be included.

If multiple filters of **different** types are added, a record is considered a match if it matches **all** filters. E.g. if `add_filter('project', 'routeviews')` and `add_filter('record-type', 'updates')` are used, records that are both from the *Route Views* project, **and** are *updates* will be included.

Parameters:

- `type (str)` – The type of the filter, can be one of `project`, `collector`, `record-type`
- `value (str)` – The value of the filter

Raises:

- `TypeError` – if the type or value are not basestrings
- `ValueError` – if the type is not valid

The `add_interval_filter(start, stop)` method adds an interval filter to an unstarted BGP Stream instance. Only those records that fall within the given interval will be included in the stream.

If multiple interval filters are added, then a record is included if it is inside **any** of the intervals.

Parameters:

- `start (int)` – The start time of the interval (inclusive)
- `stop (int)` – The end time of the interval (exclusive)

PYBGPSTREAM

What's the AS topology seen by collector Y?

- collector rrc00 on Thu, 30 Apr
 - 1 RIB file
 - 8,205,994 RIB entries
 - 108,197 unique AS adjacencies

2m:09s

- all RIS collectors on Thu, 30 Apr
 - 13 RIB files
 - 57,690,921 RIB entries
 - 164,739 unique AS adjacencies

15m:18s

```
#stream.add_filter('collector', 'rrc00')
```

```
1#!/usr/bin/env python
2
3from _pybgpstream import BGPStream, BGPRcord, BGPElem
4
5stream = BGPStream()
6rec = BGPRcord()
7
8as_topology = set()
9rib_entries = 0
10
11# Select datasource
12stream.set_data_interface('mysql')
13
14# Apply filters
15stream.add_filter('project', 'ris')
16stream.add_filter('collector', 'rrc00')
17stream.add_filter('record-type', 'ribs')
18# Wed, 29 Apr 2015 23:50:00 GMT -> Thu, 30 Apr 2015 00:10:00 GMT
19stream.add_interval_filter(1430351400, 1430352600)
20
21stream.start()
22
23# Process data
24while(stream.get_next_record(rec)):
25    elem = rec.get_next_elem()
26    while(elem):
27        rib_entries += 1
28        path = elem.fields['as-path']
29        ases = path.split(" ")
30        for i in range(0, len(ases)-1):
31            if(ases[i] != ases[i+1]):
32                as_topology.add(tuple(sorted([ases[i], ases[i+1]])))
33        elem = rec.get_next_elem()
34
35    # Output results
36print "Processed ", rib_entries, " rib entries"
37print "Found ", len(as_topology), " AS adjacencies"
```


PYBGPSTREAM

What is the number of MOAS (multi origin AS) prefix events observed in a 3 hours period?

- 1 collector: rrc00
 - 1 RIB file + 36 update files
 - 3,824 MOAS events
- all RIS collectors (13)
 - 13 RIB files + 468 update files
 - 6671 MOAS events

4m:57s

53m:16s

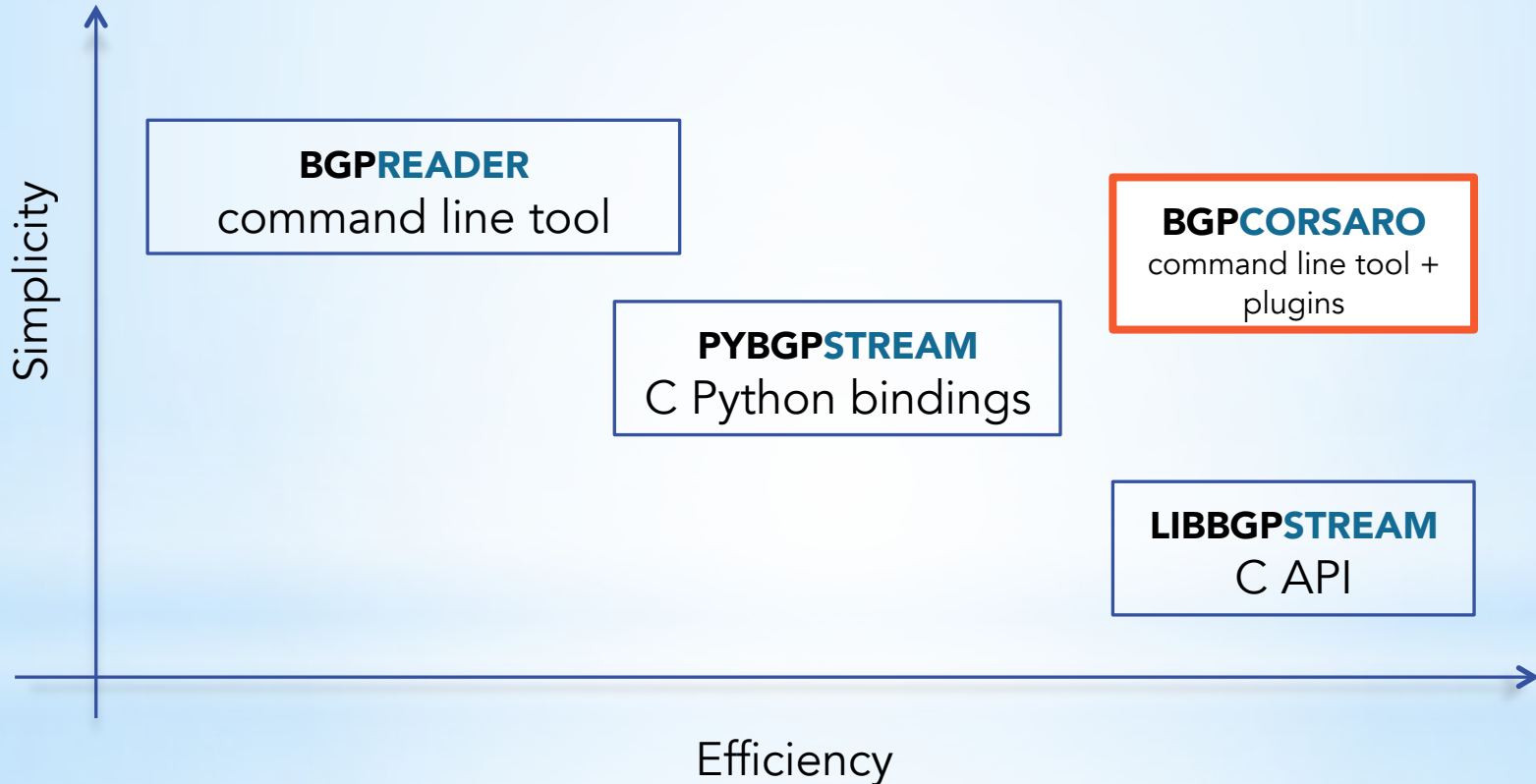
What if I want to do it in real time?

- end time in future
- just add one more line of configuration!

```
stream.set_blocking()
```

```
1#!/usr/bin/env python
2
3from _pybgpstream import BGPStream, BGPRecord, BGPElem
4
5# pfx -> list of peers observing the prefix
6pfx_peers_asn = {}
7# signature {collector}{ip}{asn} -> id
8info_id = 1
9# last id assigned
10last_id = -1
11
12# get id associated with current peer
13def get_peer_id(col, ip, asn):
14    global info_id
15    global last_id
16    if col not in info_id:
17        info_id[col] = {}
18    if ip not in info_id[col]:
19        info_id[col][ip] = {}
20    if asn not in info_id[col][ip]:
21        last_id = last_id + 1
22        info_id[col][ip][asn] = last_id
23    peer_id = info_id[col][ip][asn]
24    return peer_id
25
26# add prefix peer origin
27def add_prefix_peer_origin(ts, pfx, peer, asn):
28    global pfx_peers_asn
29    global peer_info
30    moas = 0
31    already_there = 0
32    if pfx not in pfx_peers_asn:
33        pfx_peers_asn[pfx] = {}
34    for p in pfx_peers_asn[pfx]:
35        if p != peer and pfx_peers_asn[pfx][p] != asn:
36            moas = 1
37            if pfx_peers_asn[pfx][p] == asn:
38                already_there = 1
39    if int(peer) not in pfx_peers_asn[pfx]:
40        pfx_peers_asn[pfx][int(peer)] = asn
41    else:
42        pfx_peers_asn[pfx][int(peer)] = asn
43    if already_there == 0 and moas == 1:
44        print ts, "MOAS", pfx, pfx_peers_asn[pfx]
45
46# remove prefix peer (i.e. this prefix is not seen by this peer anymore)
47def remove_prefix_peer(pfx, peer):
48    global pfx_peers_asn
49    global peer_info
50    if pfx not in pfx_peers_asn:
51        return
52    if peer not in pfx_peers_asn[pfx]:
53        return
54    del pfx_peers_asn[pfx][int(peer)]
55
56# remove peer (i.e. this peer is not active anymore)
57def remove_peer(peer):
58    global pfx_peers_asn
59    for pfx in pfx_peers_asn:
60        if int(peer) in pfx_peers_asn[pfx]:
61            del pfx_peers_asn[pfx][int(peer)]
62
63stream = BGPStream()
64rec = BGPRecord()
65stream.set_data_interface('mysql')
66stream.set_data_interface_option('mysql', 'db-host', 'loki-ge')
67stream.set_data_interface_option('mysql', 'db-port', '3306')
68stream.set_data_interface_option('mysql', 'db-user', 'bgpstream')
69stream.add_filter('project', 'ris')
70stream.add_filter('collector', 'rrc00')
71stream.add_interval_filter(1430351400, 1430352600)
72stream.start()
73
74while(stream.get_next_record(rec)):
75    elem = rec.get_next_elem()
76    while(elem):
77        # get peer information
78        peer_id = get_peer_id(rec.collector, elem.peer_address, elem.peer_asn)
79        # apply bgp message
80        if(elem.type == 'R' or elem.type == 'A'):
81            path = elem.fields['as-path']
82            ases = path.split(' ')
83            add_prefix_peer_origin(rec.time, elem.fields['prefix'], peer_id, ases[-1])
84        if(elem.type == 'W'):
85            remove_prefix_peer(elem.fields['prefix'], peer_id)
86        if(elem.type == 'S' and elem.fields['new-state'] != "established"):
87            remove_peer(peer_id)
88        # get next element
89        elem = rec.get_next_elem()
90
```

BGPSTREAM just a C library?

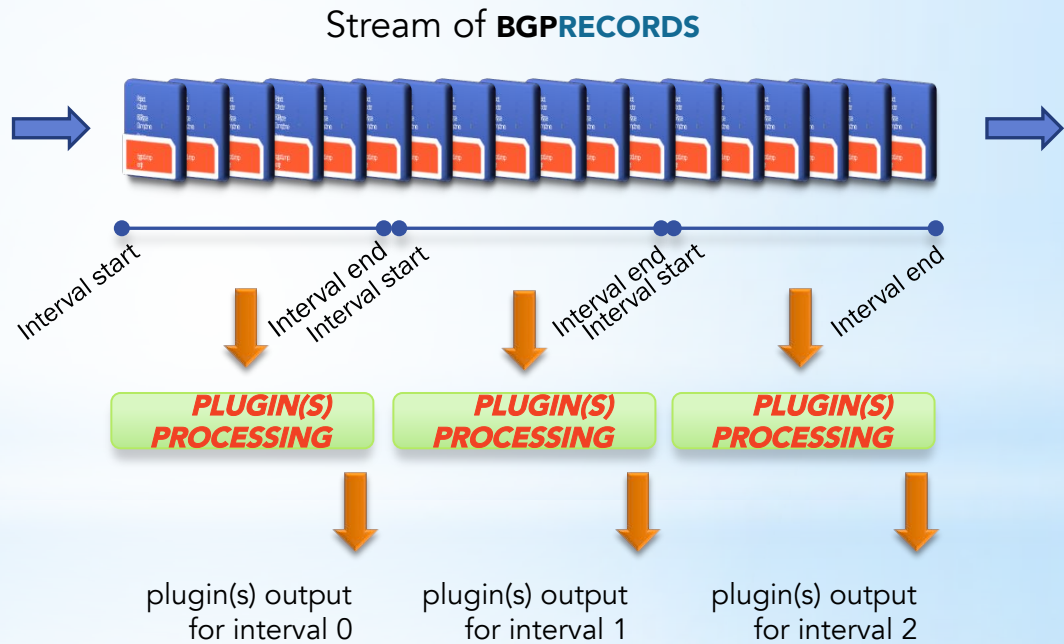


BGPCORSARO

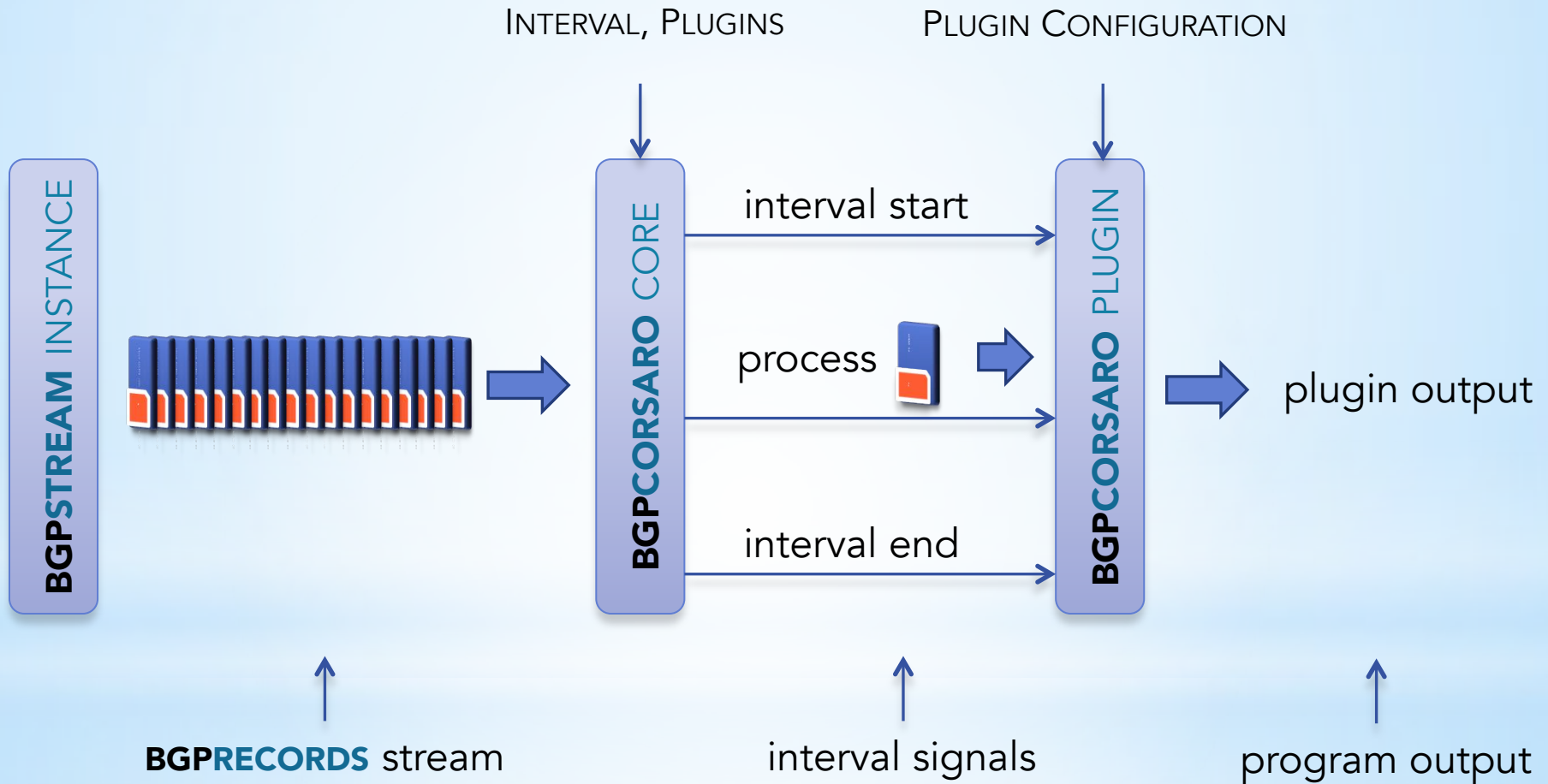
* C tool that transforms a stream of **BGP RECORDS** into a set of structures and metrics representative of specific time intervals

* interval driven

* modular architecture based on plugins



BGPCORSARO architecture



BGPCORSARO plugins

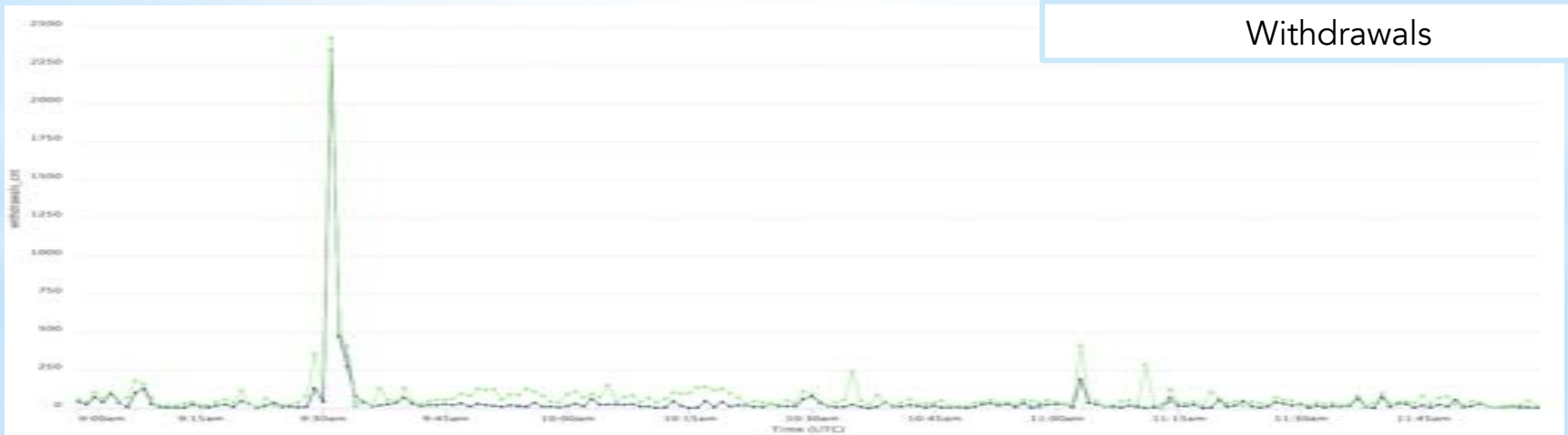
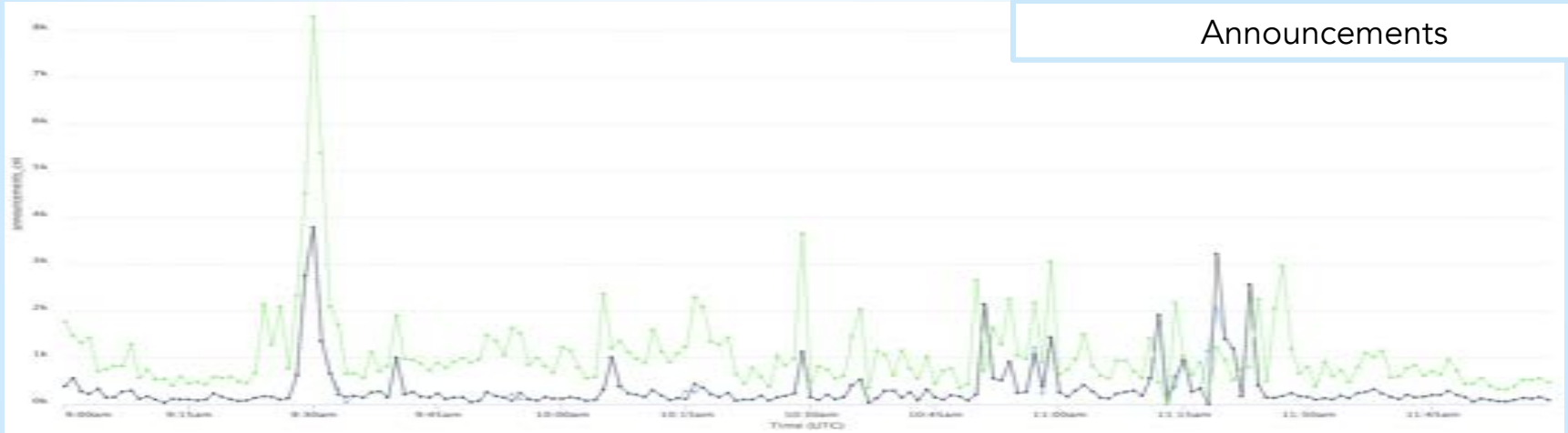
ROUTINGTABLES plugin:

- * it maintains the state and the routing table of each peer
 - * BGP finite state machine per peer
 - * RIBs and updates
 - * recover from out of order and corrupted data
- * outputs statistics every minute (of BGP time)

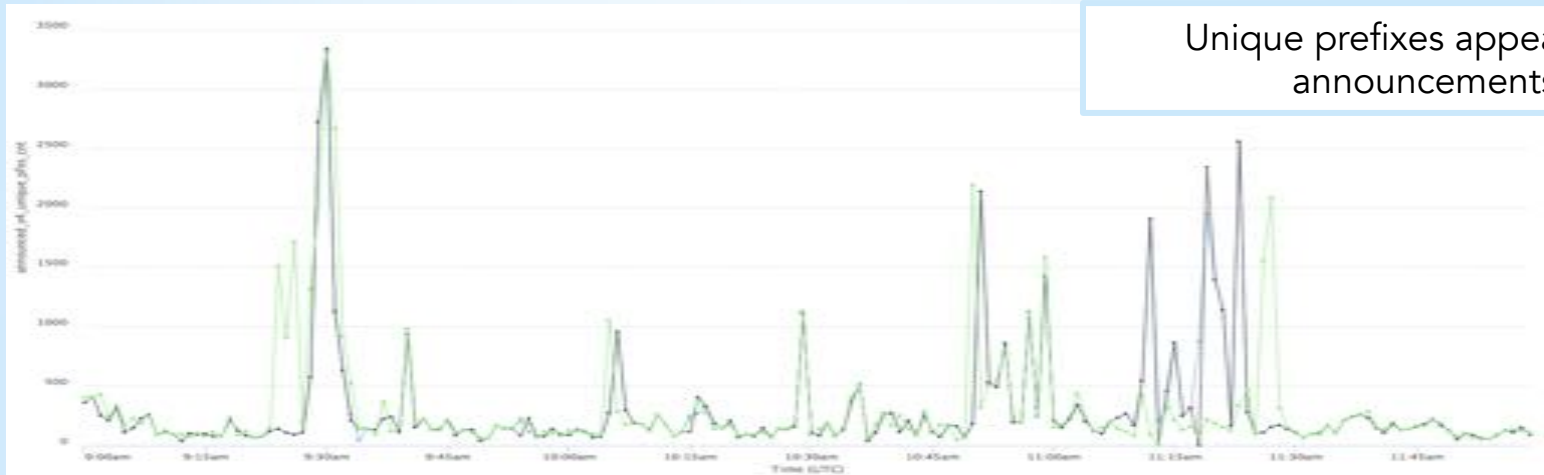
BGPCORSARO routingtables plugin



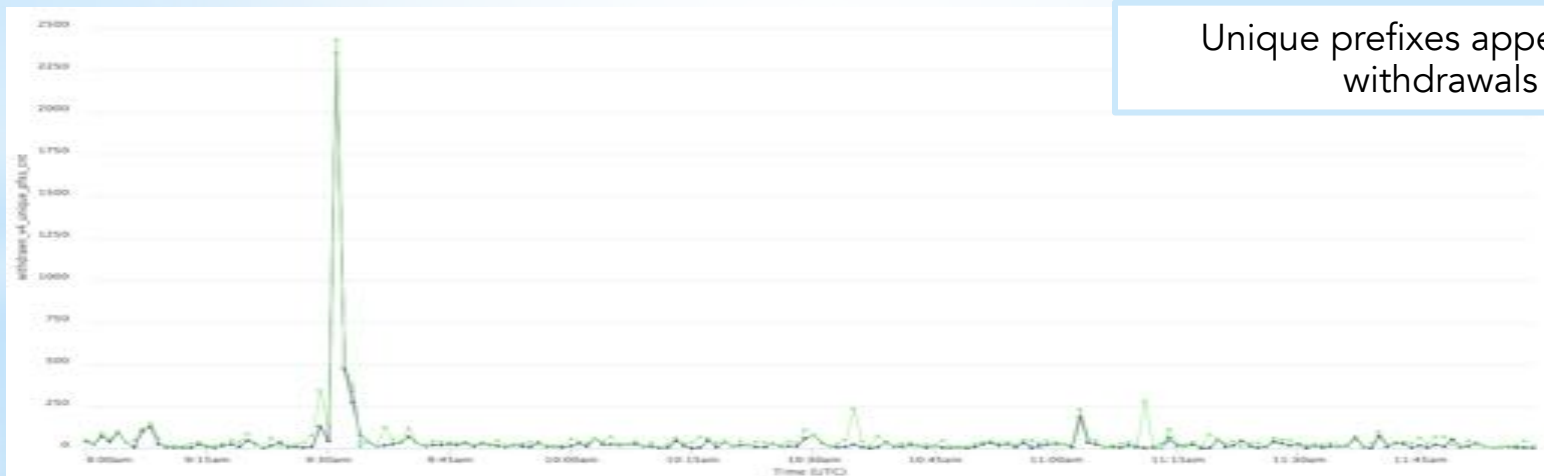
BGPCORSARO routingtables plugin



BGPCORSARO routingtables plugin

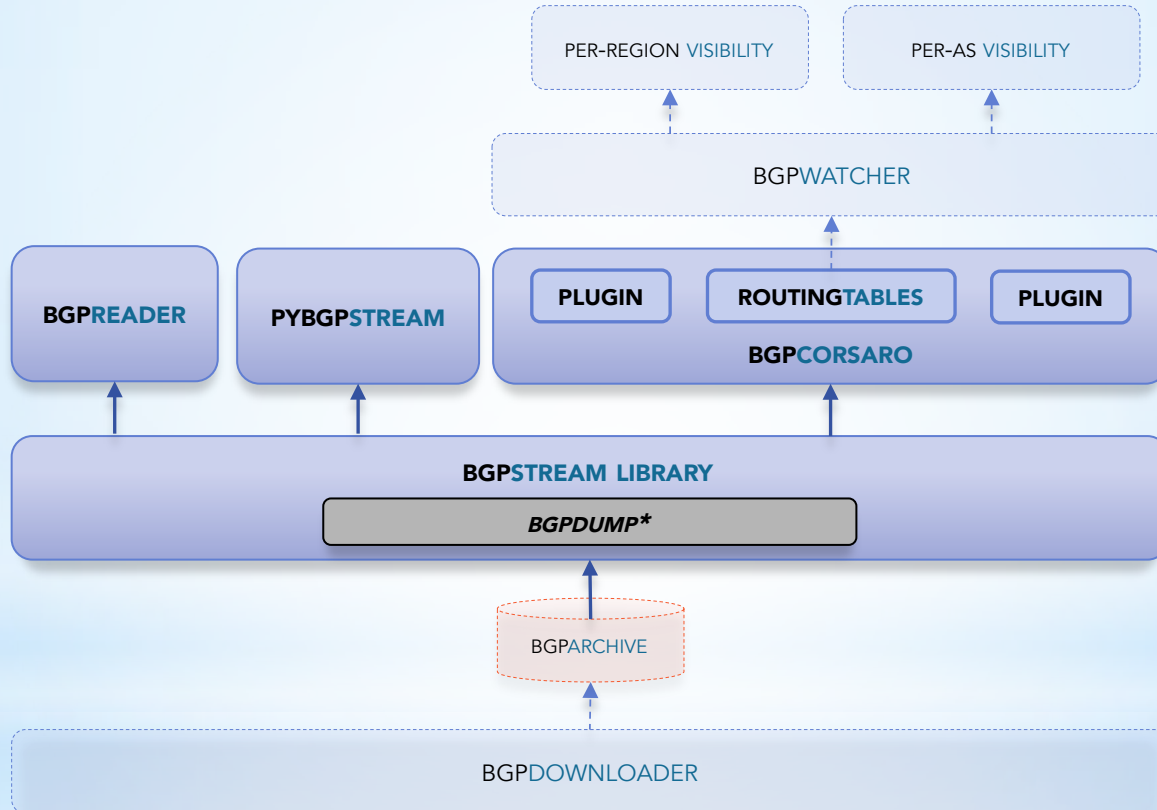


Unique prefixes appearing in announcements



Unique prefixes appearing in withdrawals

BGPSTREAM framework



QUESTIONS ?
THANKS

Chiara Orsini
chiara@caida.org