BGPSTREAM

a framework for historical analysis
and real-time monitoring of BGP data

Chiara Orsini, **Alistair King,**
Danilo Giordano, Vasileios Giotsas,
Alberto Dainotti

alistair@caida.org
CAIDA, UC San Diego

BGPSTREAM

BGP data analysis for the masses

- Open source libraries, APIs and tools for live and historical BGP data analysis

- Simple API
- Versatile
- Facilitates reproducibility and repeatability
- Realtime monitoring

- Stable: *https://bgpstream.caida.org*

caida

MOTIVATION

Why BGPStream?

- BGP research and monitoring is important
- Lots of existing BGP measurement data
  - Route Views and RIPE RIS have >15 years of data (16TB)
- **BUT**, distinct lack of good tooling for processing/analyzing BGP data
  - State of the art?

MOTIVATION

Why BGPStream?

- BGP research and monitoring is important
- Lots of existing BGP measurement data
  - Route Views and RIPE RIS have >15 years of data (16TB)
- **BUT**, distinct lack of good tooling for processing/analyzing BGP data
  - State of the art?

```
wget http://archive.org/xyz/abc/file.mrt
```

MOTIVATION

Why BGPStream?

- BGP research and monitoring is important
- Lots of existing BGP measurement data
  - Route Views and RIPE RIS have >15 years of data (16TB)
- **BUT**, distinct lack of good tooling for processing/analyzing BGP data
  - State of the art?

```
wget http://archive.org/xyz/abc/file.mrt
bgpdump -m file.mrt | my_parser.py
```


caida

MOTIVATION

Why BGPStream?

- BGP research and monitoring is important
- Lots of existing BGP measurement data
  - Route Views and RIPE RIS have >15 years of data (16TB)
- **BUT**, distinct lack of good tooling for processing/analyzing BGP data
  - State of the art?

```
wget http://archive.org/xyz/abc/file.mrt
bgpdump -m file.mrt | my_parser.py
```

caida

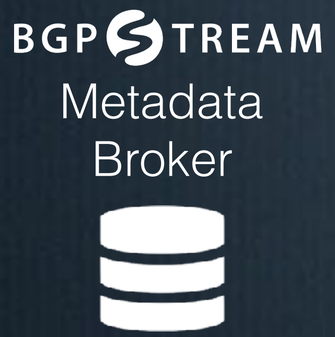
THE BGPSTREAM FRAMEWORK

An overview

caida

THE BGPSTREAM FRAMEWORK

An overview



caida

THE BGPSTREAM FRAMEWORK

An overview

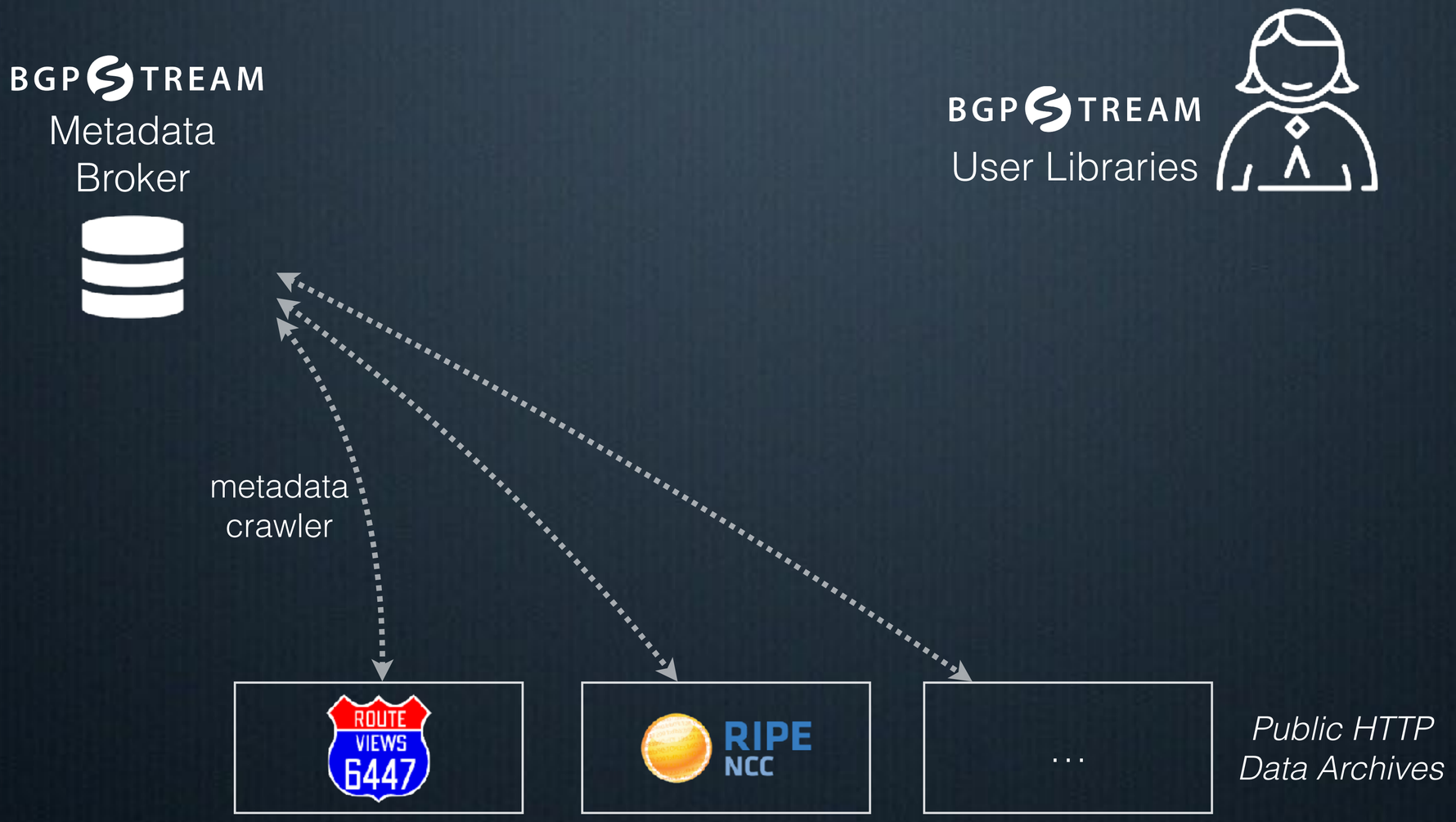
BGPSTREAM
Metadata
Broker

BGPSTREAM
User Libraries

caida

THE BGPSTREAM FRAMEWORK

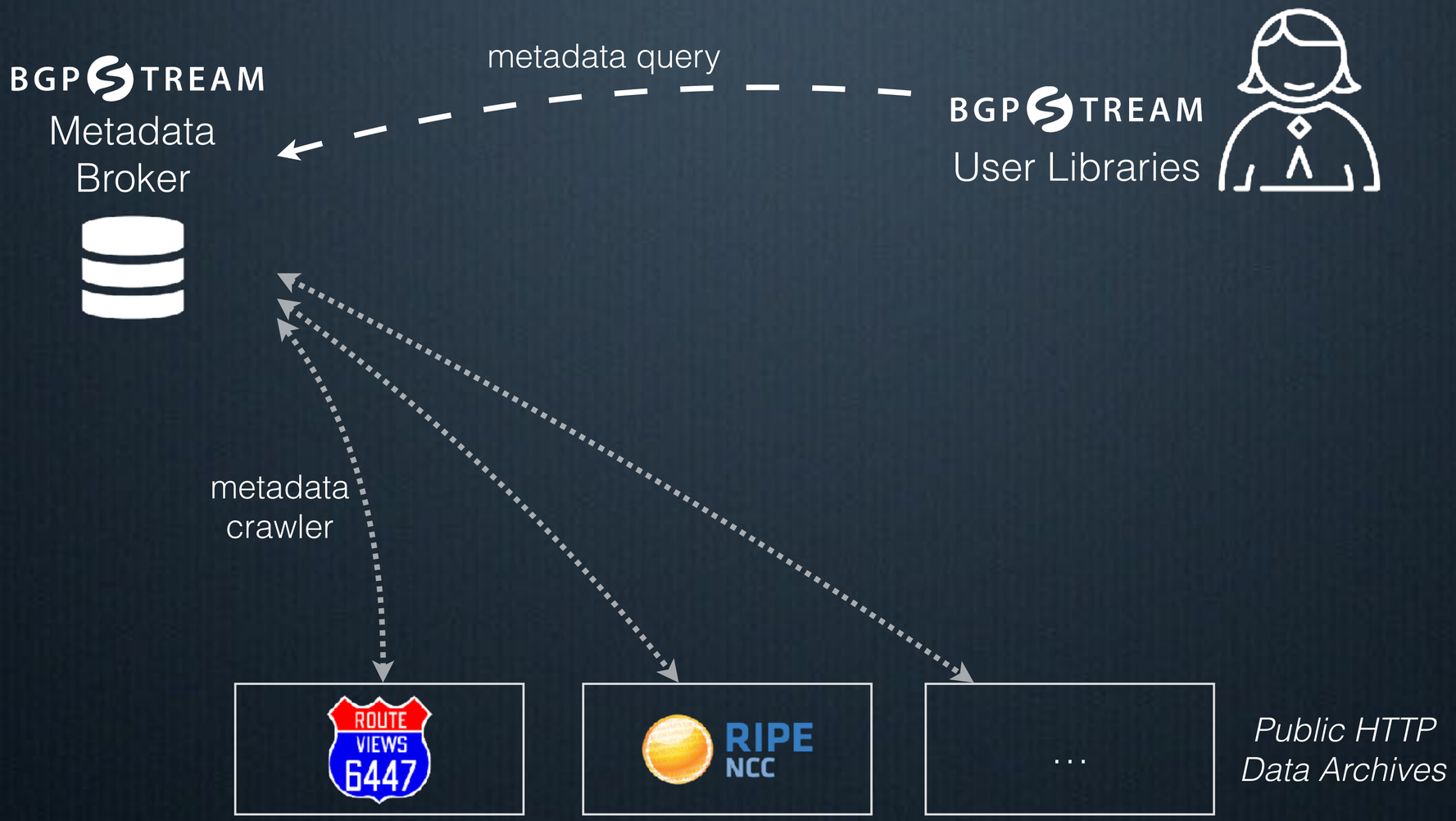
An overview



caida

THE BGPSTREAM FRAMEWORK

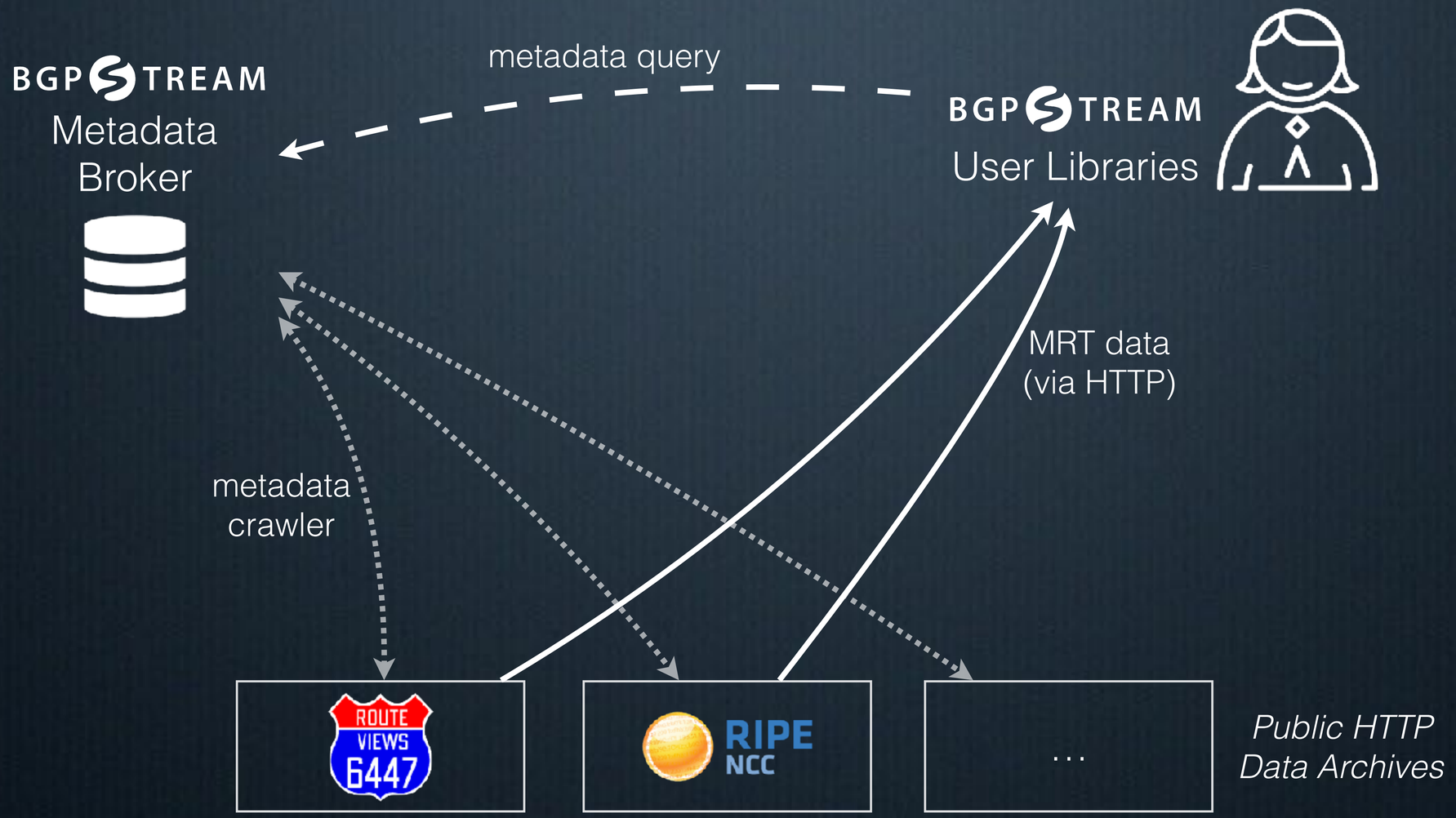
An overview



caida

THE BGPSTREAM FRAMEWORK

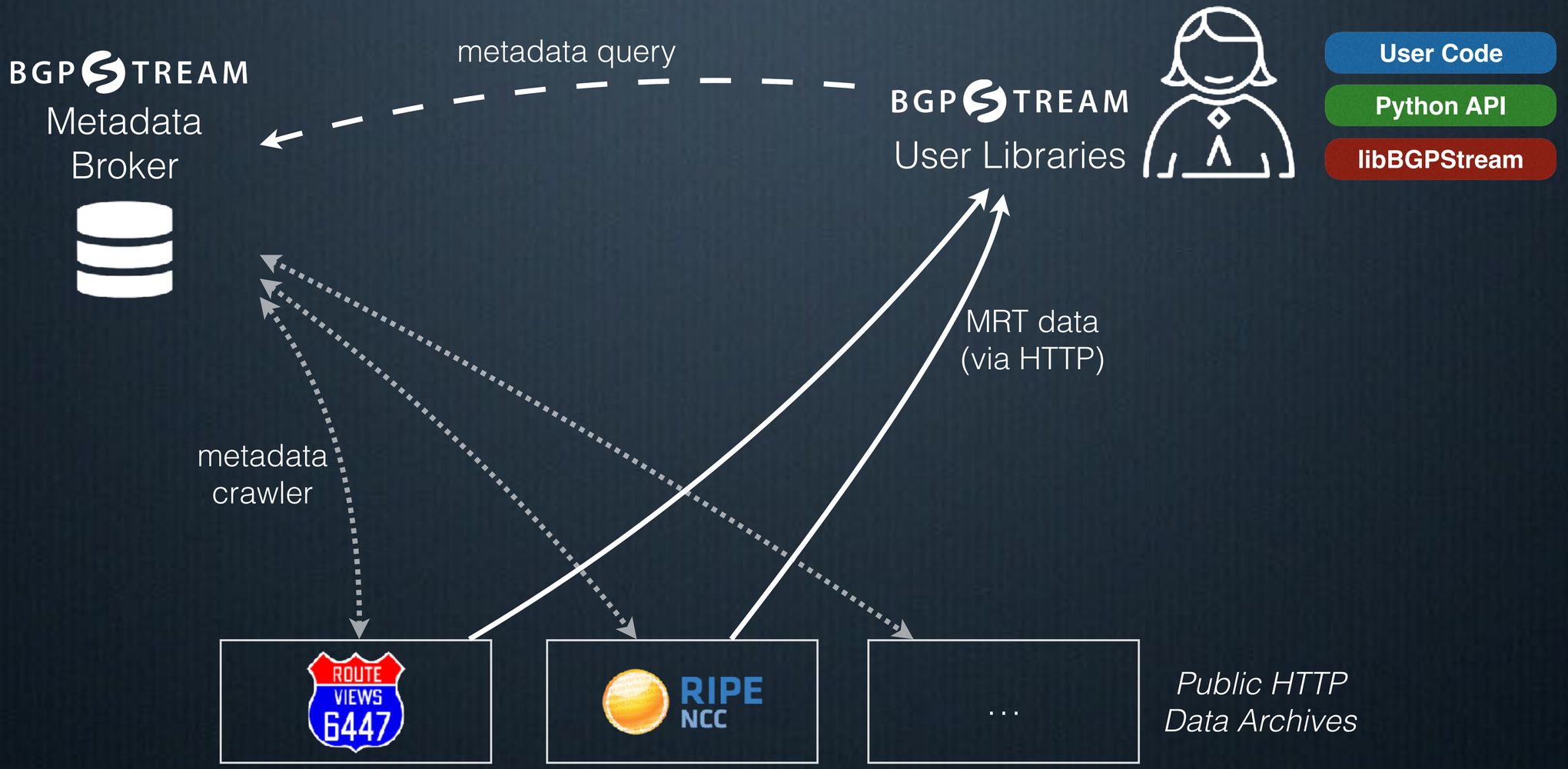
An overview



caida

THE BGPSTREAM FRAMEWORK

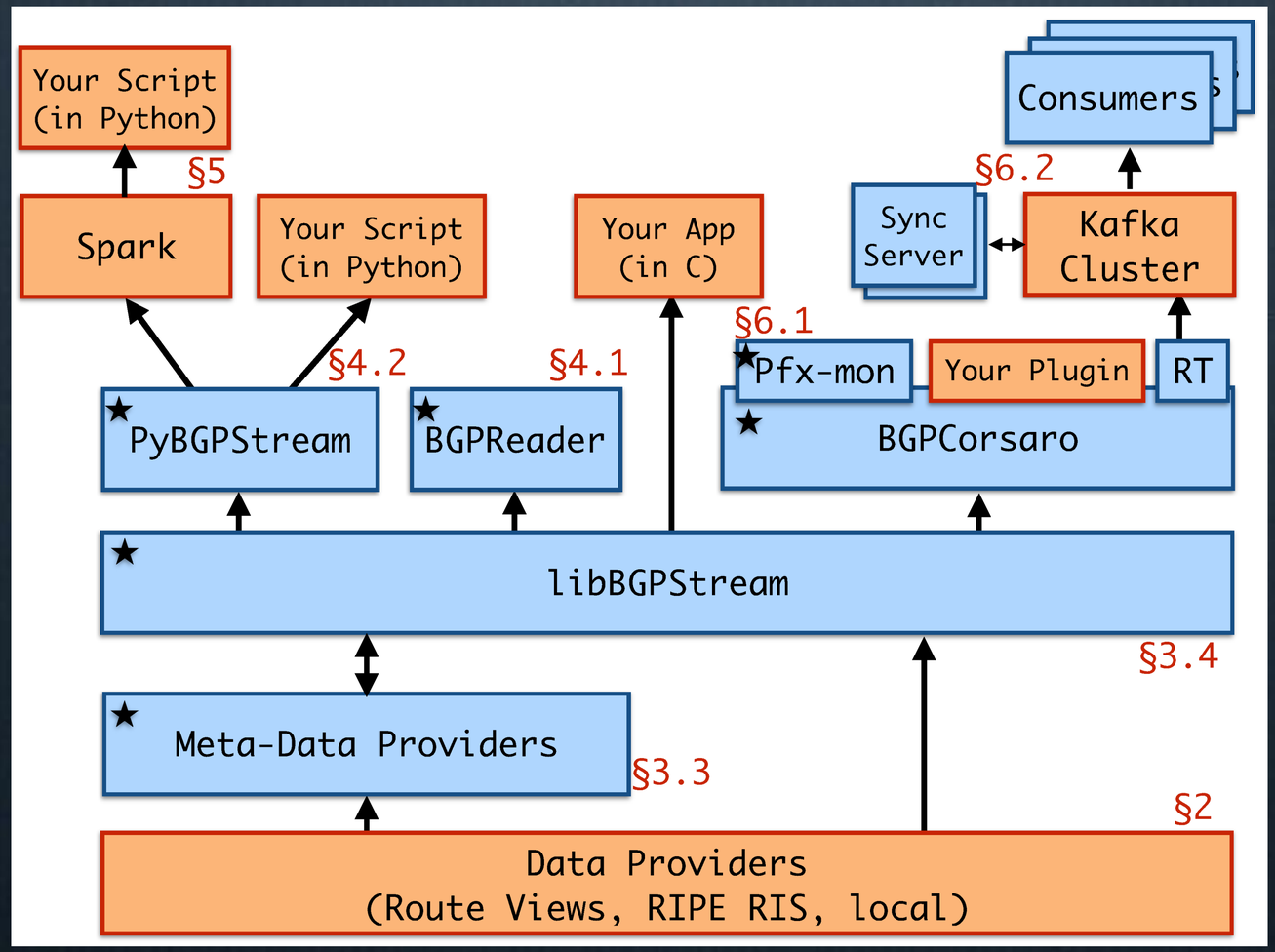
An overview



caida

THE BGPSTREAM FRAMEWORK

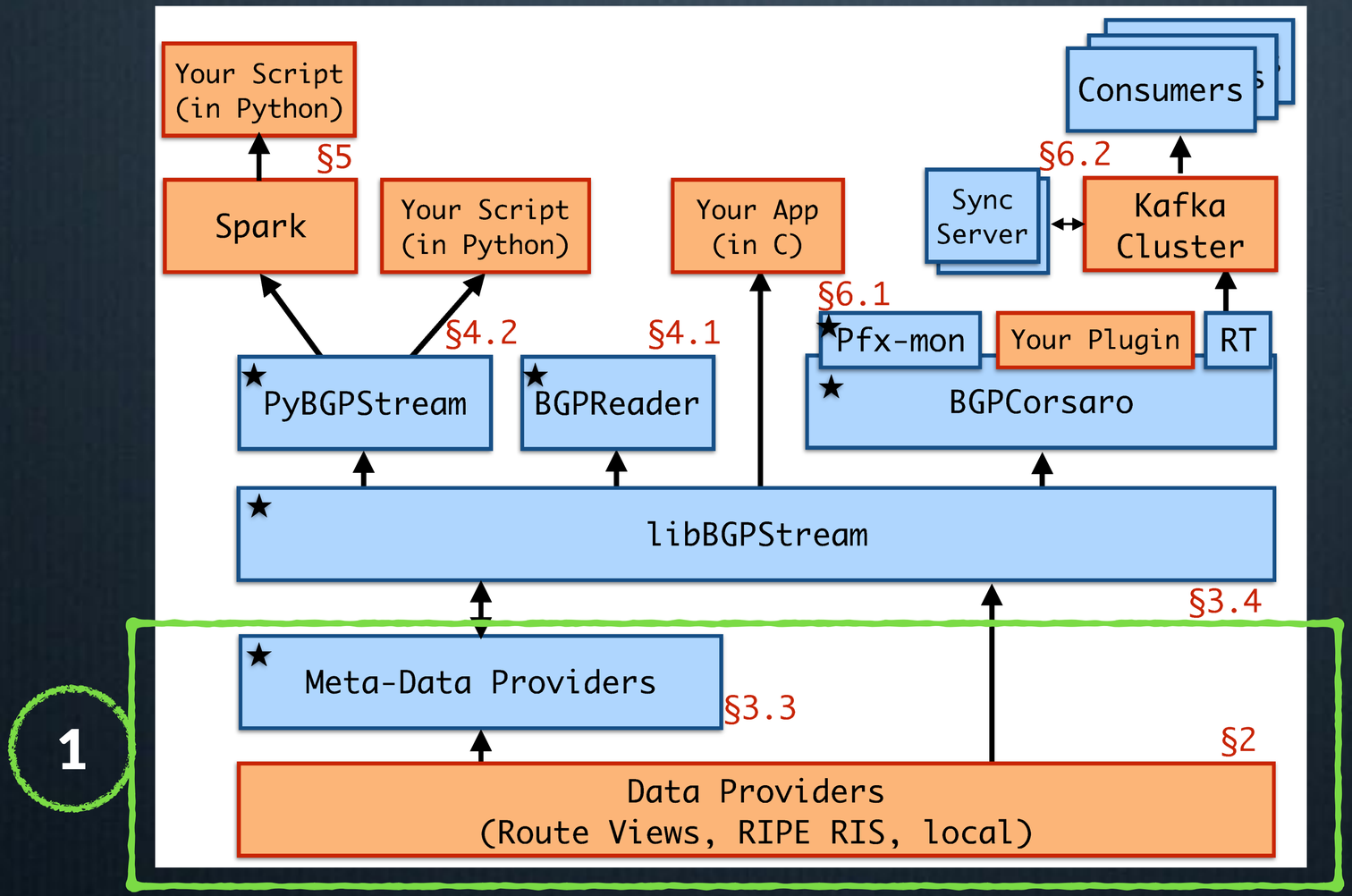
Stacked view



caida

THE BGPSTREAM FRAMEWORK

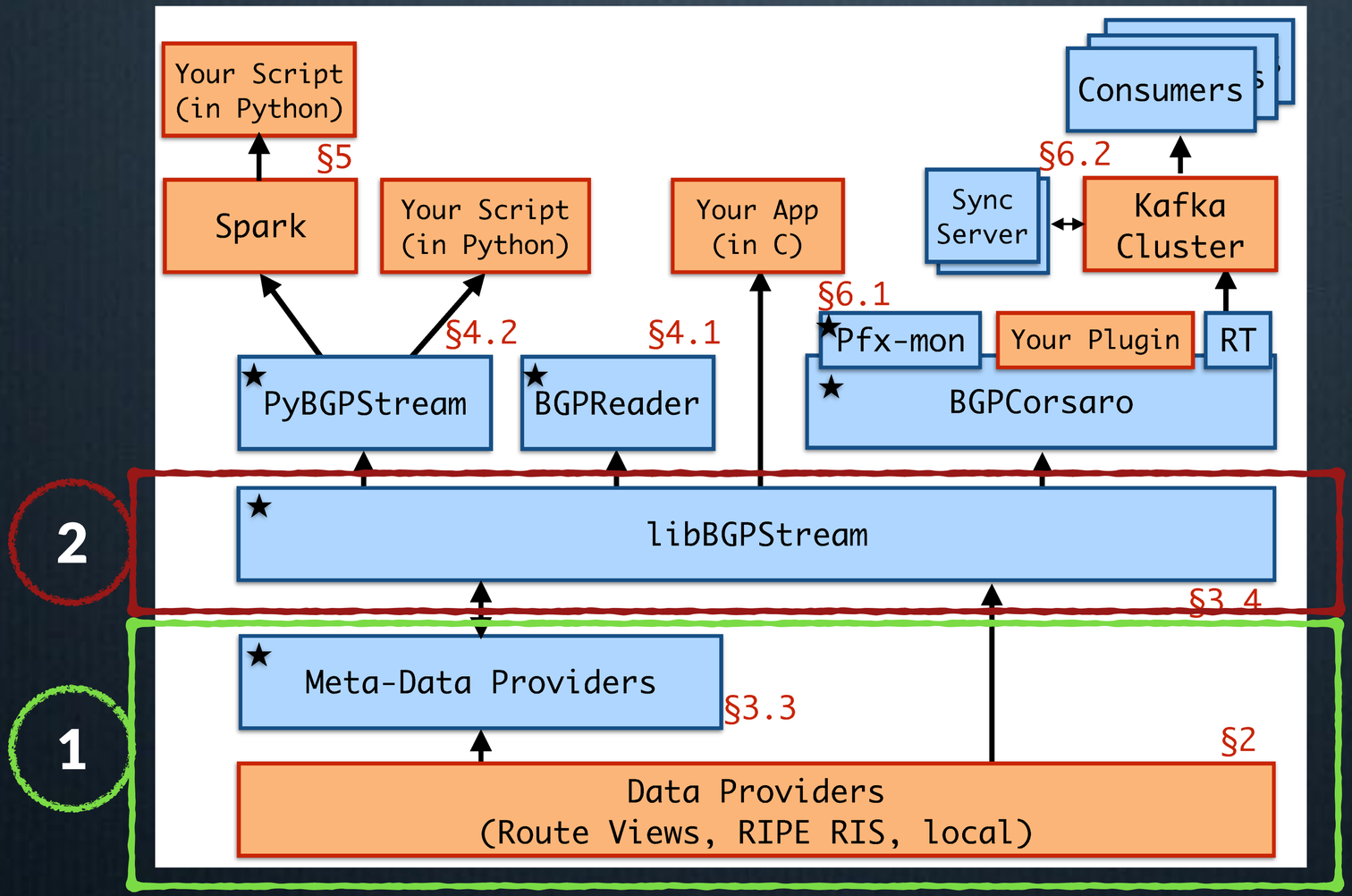
Stacked view



caida

THE BGPSTREAM FRAMEWORK

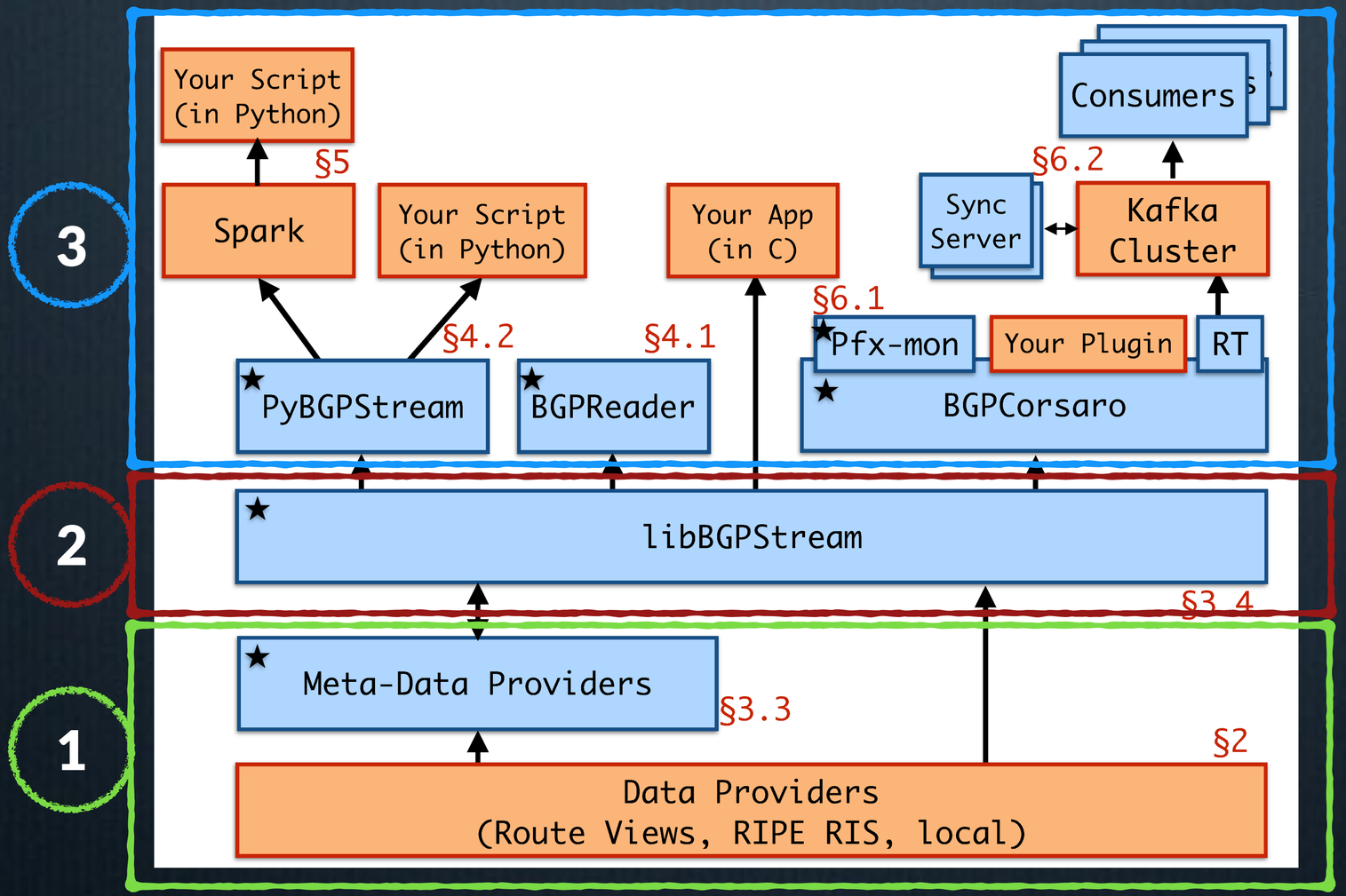
Stacked view



caida

THE BGPSTREAM FRAMEWORK

Stacked view

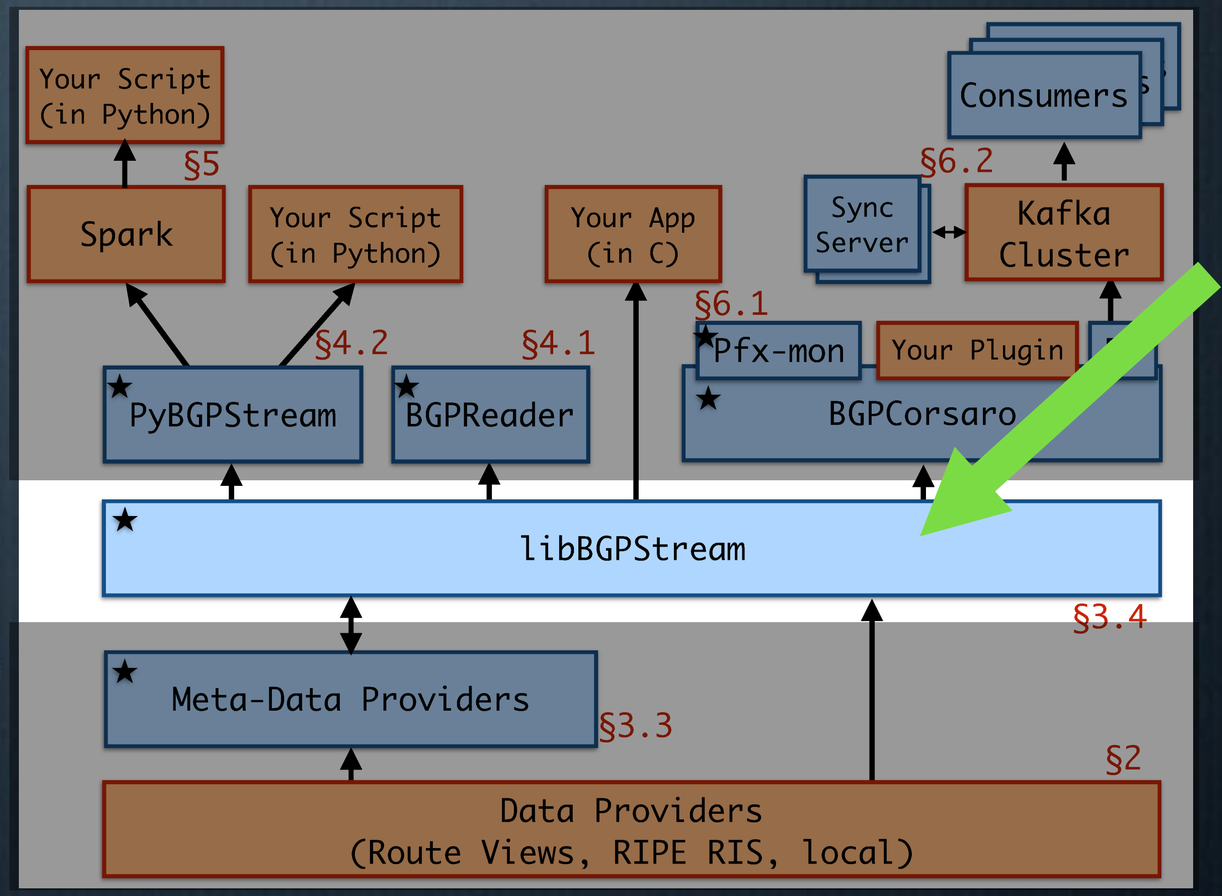


caida

BGPSTREAM USER LIBRARY

libBGPStream

- Issues queries to metadata broker
- Retrieves data *directly* from Data Providers
  - Currently supports MRT (RFC 6396)
- De-multiplexes data from many sources into a single stream
- Provides time-ordered sorting



caida

RECORDS & ELEMS

Extracting information from MRT

- *BGPStream Record*:
  - Encapsulates an MRT record
  - Adds metadata (e.g. collector)
- **MRT records (may) contain info for multiple peers/prefixes**
  - E.g. routes to a single prefix from multiple peers in a RIB dump
- **Records are decomposed into *BGPStream Elems:***
  - E.g. prefix announcement from a single peer

BGPStream Record

Field	Type	Function
project	string	project name (e.g., Route Views)
collector	string	collector name (e.g., rrc00)
type	enum	RIB or Updates
dump time	long	time the containing dump was begun
position	enum	first, middle, or last record of a dump
time	long	timestamp of the MRT record
status	enum	record validity flag
MRT record	struct	de-serialized MRT record

BGPStream Elem

Field	Type	Function
type	enum	route from a RIB dump, announcement, withdrawal, or state message
time	long	timestamp of MRT record
peer address	struct	IP address of the VP
peer ASN	long	AS number of the VP
prefix*	struct	IP prefix
next hop*	struct	IP address of the next hop
AS path*	struct	AS path
community*	struct	community attribute
old state*	enum	FSM state (before the change)
new state*	enum	FSM state (after the change)

caida

C API

Specifying a stream

```c
#include <bgpstream.h>

int main(int argc, char **argv)
{
  /* Allocate memory for a bgpstream instance */
  bgpstream_t *bs = bs = bgpstream_create();
  /* Allocate memory for a re-usable bgprecord instance */
  bgpstream_record_t *record = bgpstream_record_create();
  /* To hold a pointer to a BGPStream elem */
  bgpstream_elem_t *elem = NULL;

  /* Select bgp data from RRC06 and route-views.jinx collectors only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc06");
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views.jinx");

  /* Process updates only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");

  /* Select a time interval to process:
   * Sun, 10 Oct 2010 10:10:10 GMT -  Sun, 10 Oct 2010 11:11:11 GMT */
  bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

  /* Start the stream */
  bgpstream_start(bs);

```

caida

C API

Specifying a stream

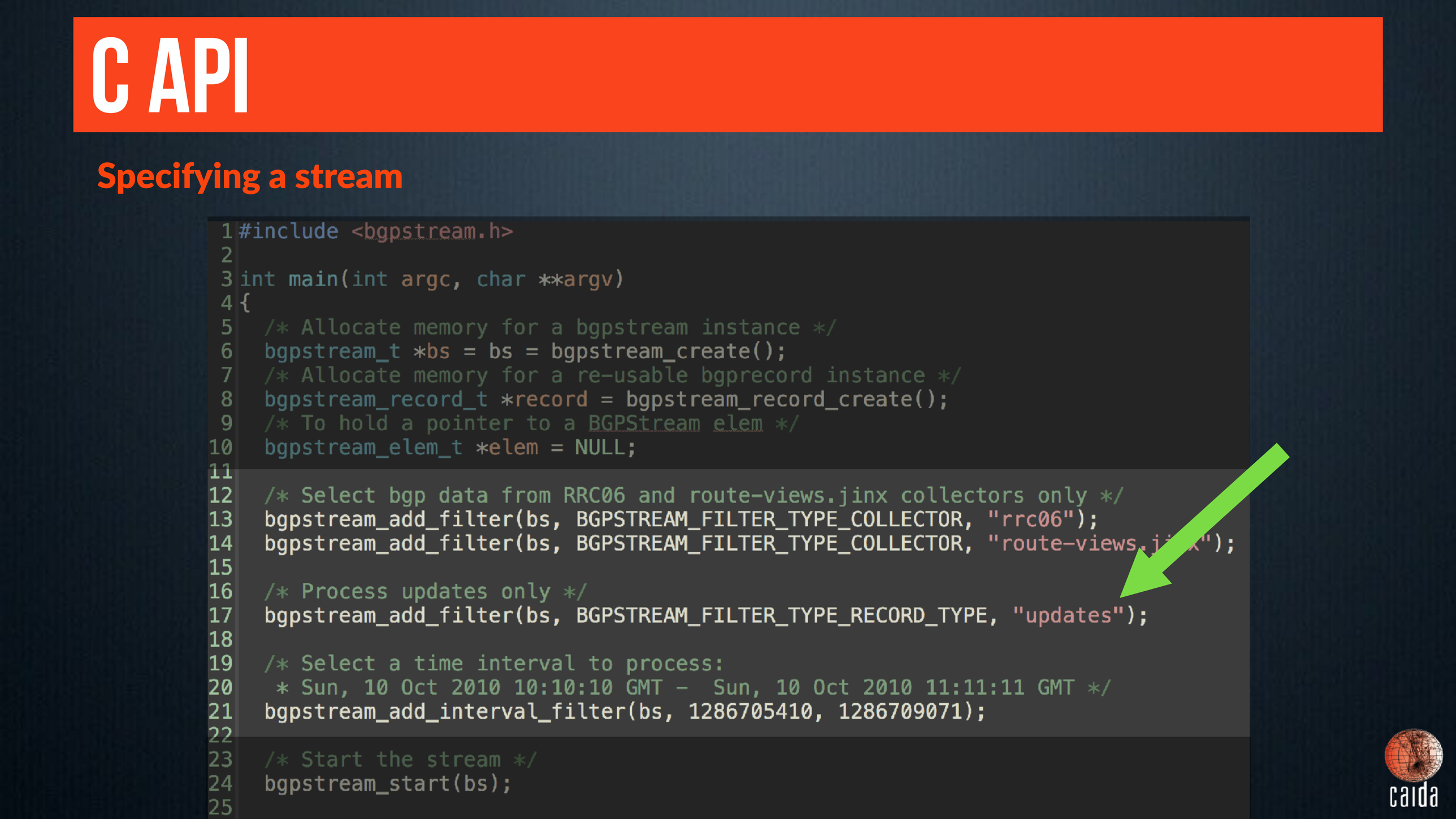
```c
#include <bgpstream.h>

int main(int argc, char **argv)
{
  /* Allocate memory for a bgpstream instance */
  bgpstream_t *bs = bs = bgpstream_create();
  /* Allocate memory for a re-usable bgprecord instance */
  bgpstream_record_t *record = bgpstream_record_create();
  /* To hold a pointer to a BGPStream elem */
  bgpstream_elem_t *elem = NULL;

  /* Select bgp data from RRC06 and route-views.jinx collectors only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc06");
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views.jinx");

  /* Process updates only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");

  /* Select a time interval to process:
   * Sun, 10 Oct 2010 10:10:10 GMT -  Sun, 10 Oct 2010 11:11:11 GMT */
  bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

  /* Start the stream */
  bgpstream_start(bs);
```

caida

C API

Specifying a stream

```c
#include <bgpstream.h>

int main(int argc, char **argv)
{
  /* Allocate memory for a bgpstream instance */
  bgpstream_t *bs = bs = bgpstream_create();
  /* Allocate memory for a re-usable bgprecord instance */
  bgpstream_record_t *record = bgpstream_record_create();
  /* To hold a pointer to a BGPStream elem */
  bgpstream_elem_t *elem = NULL;

  /* Select bgp data from RRC06 and route-views.jinx collectors only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc06");
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views.jinx");

  /* Process updates only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");

  /* Select a time interval to process:
   * Sun, 10 Oct 2010 10:10:10 GMT -  Sun, 10 Oct 2010 11:11:11 GMT */
  bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

  /* Start the stream */
  bgpstream_start(bs);

```

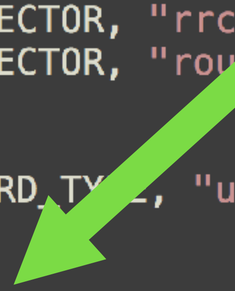


caida

C API

Specifying a stream

```c
#include <bgpstream.h>

int main(int argc, char **argv)
{
  /* Allocate memory for a bgpstream instance */
  bgpstream_t *bs = bs = bgpstream_create();
  /* Allocate memory for a re-usable bgprecord instance */
  bgpstream_record_t *record = bgpstream_record_create();
  /* To hold a pointer to a BGPStream elem */
  bgpstream_elem_t *elem = NULL;

  /* Select bgp data from RRC06 and route-views.jinx collectors only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc06");
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views.jinx");

  /* Process updates only */
  bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");

  /* Select a time interval to process:
   * Sun, 10 Oct 2010 10:10:10 GMT -  Sun, 10 Oct 2010 11:11:11 GMT */
  bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

  /* Start the stream */
  bgpstream_start(bs);

```



caida

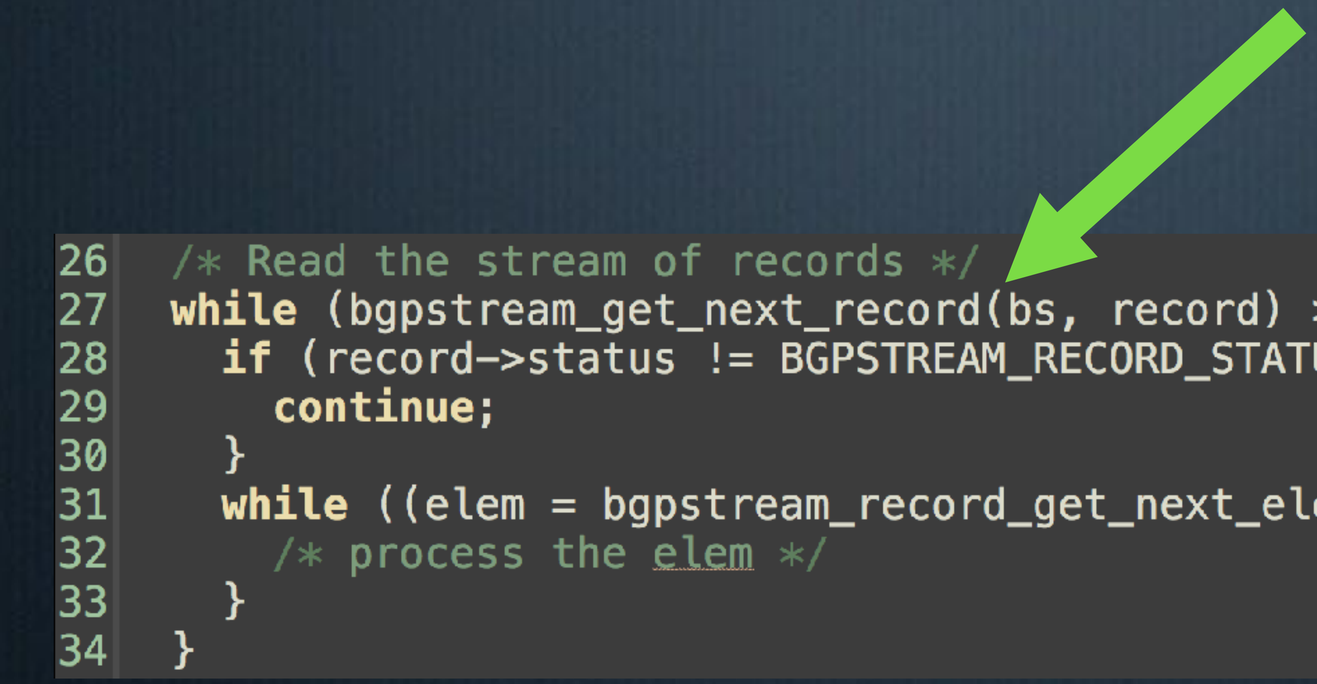
C API

Consuming the stream

```
26  /* Read the stream of records */
27  while (bgpstream_get_next_record(bs, record) > 0) {
28    if (record->status != BGPSTREAM_RECORD_STATUS_VALID_RECORD) {
29      continue;
30    }
31    while ((elem = bgpstream_record_get_next_elem(record)) != NULL) {
32      /* process the elem */
33    }
34  }
```

C API

Consuming the stream

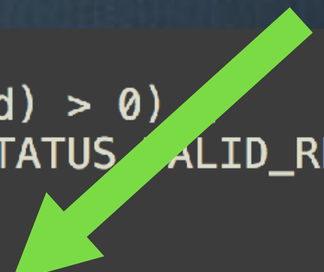


```
26  /* Read the stream of records */
27  while (bgpstream_get_next_record(bs, record) > 0) {
28    if (record->status != BGPSTREAM_RECORD_STATUS_VALID_RECORD) {
29      continue;
30    }
31    while ((elem = bgpstream_record_get_next_elem(record)) != NULL) {
32      /* process the elem */
33    }
34  }
```

C API

Consuming the stream

```
26  /* Read the stream of records */
27  while (bgpstream_get_next_record(bs, record) > 0)
28    if (record->status != BGPSTREAM_RECORD_STATUS_VALID_RECORD) {
29      continue;
30    }
31    while ((elem = bgpstream_record_get_next_elem(record)) != NULL) {
32      /* process the elem */
33    }
34  }
```

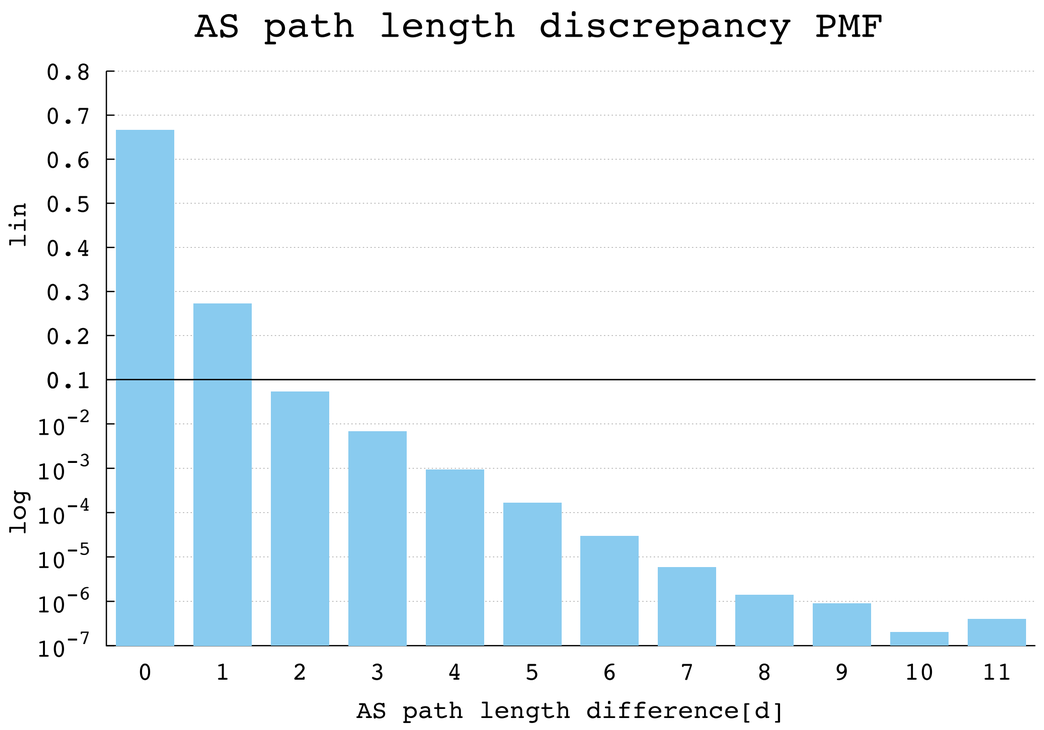


caida

PYTHON BINDINGS - CASE STUDY

Studying AS path inflation using PyBGPStream

How many AS paths are longer than the shortest path between two ASes?




```
from _pybgpstream import BGPStream, BGPRecord, BGPElem
from collections import defaultdict
from itertools import groupby
import networkx as nx

stream = BGPStream()
as_graph = nx.Graph()
rec = BGPRecord()
bgp_lens = defaultdict(lambda: defaultdict(lambda: None))
stream.add_filter('record-type','ribs')
stream.add_interval_filter(1438415400,1438416600)
stream.start()

while(stream.get_next_record(rec)):
    elem = rec.get_next_elem()
    while(elem):
        monitor = str(elem.peer_asn)
        hops = [k for k, g in groupby(elem.fields['as-path'].split(" "))]
        if len(hops) > 1 and hops[0] == monitor:
            origin = hops[-1]
            for i in range(0,len(hops)-1):
                as_graph.add_edge(hops[i],hops[i+1])
            bgp_lens[monitor][origin] = \
                min(filter(bool,[bgp_lens[monitor][origin],len(hops)]))
        elem = rec.get_next_elem()
for monitor in bgp_lens:
    for origin in bgp_lens[monitor]:
        nxlen = len(nx.shortest_path(as_graph, monitor, origin))
        print monitor, origin, bgp_lens[monitor][origin], nxlen
```

30 LINES OF PYTHON CODE

caida

PY BGPSTREAM

Python bindings

- Single script includes data specification and analysis logic:
  - Enhances reproducibility/repeatability
- All of the power of the C API, available in Python

```
bgp_lens = defaultdict(lambda: defaultdict(lambda: None))
stream.add_filter('record-type','ribs')
stream.add_interval_filter(1438415400,1438416600)
stream.start()
```

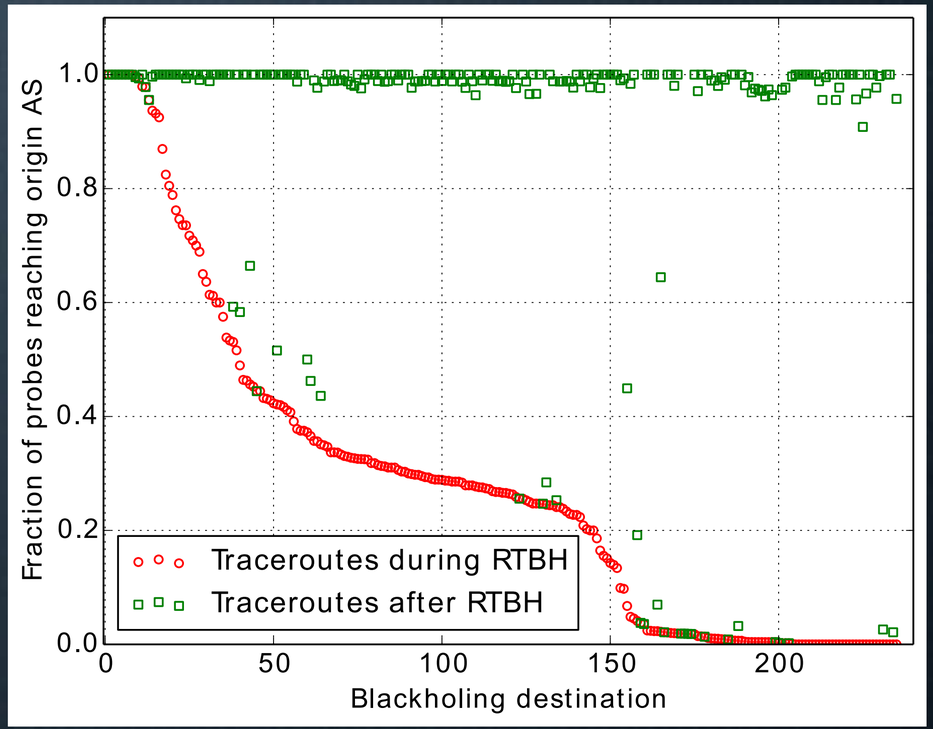
30 LINES OF PYTHON CODE

caida

PYTHON BINDINGS - CASE STUDY

Timely reactive measurements

- We monitor **community-based black-holing**
  - **Victim of DoS attack** announces prefix with special community attribute to **request neighbors drop traffic**
- We trigger traceroutes to characterize the black-holing **event** (using 50-100 probes per event)
  - probed 253 victims (90-95% of black-holing events) while black-holing in effect
- ***Combined passive control-plane and active data-plane measurements to capture and investigate transient routing policies***

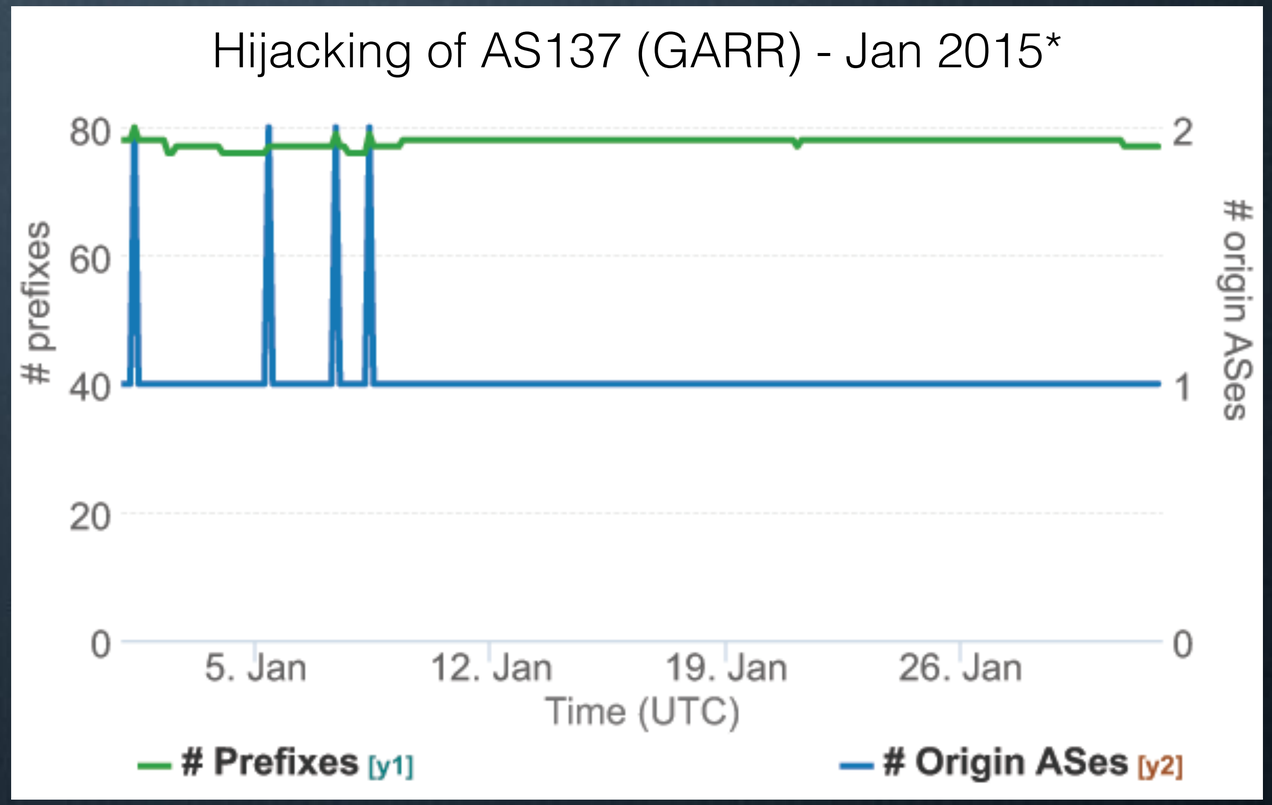


caida

BGPCORSARO

Continuous realtime monitoring

- Plugin-based tool for processing live BGP data
- Continuously **extracts derived data** from BGPStream in **regular time bins**
- Incl. *"prefix-monitor"* sample plugin
  - Monitor your own address space
  - How many prefixes/origin ASes?



*originally described by Dyn Research:
http://research.dyn.com/2015/01/vast-world-of-fraudulent-routing/

caida

BIG DATA

BGP data analysis for the 1%

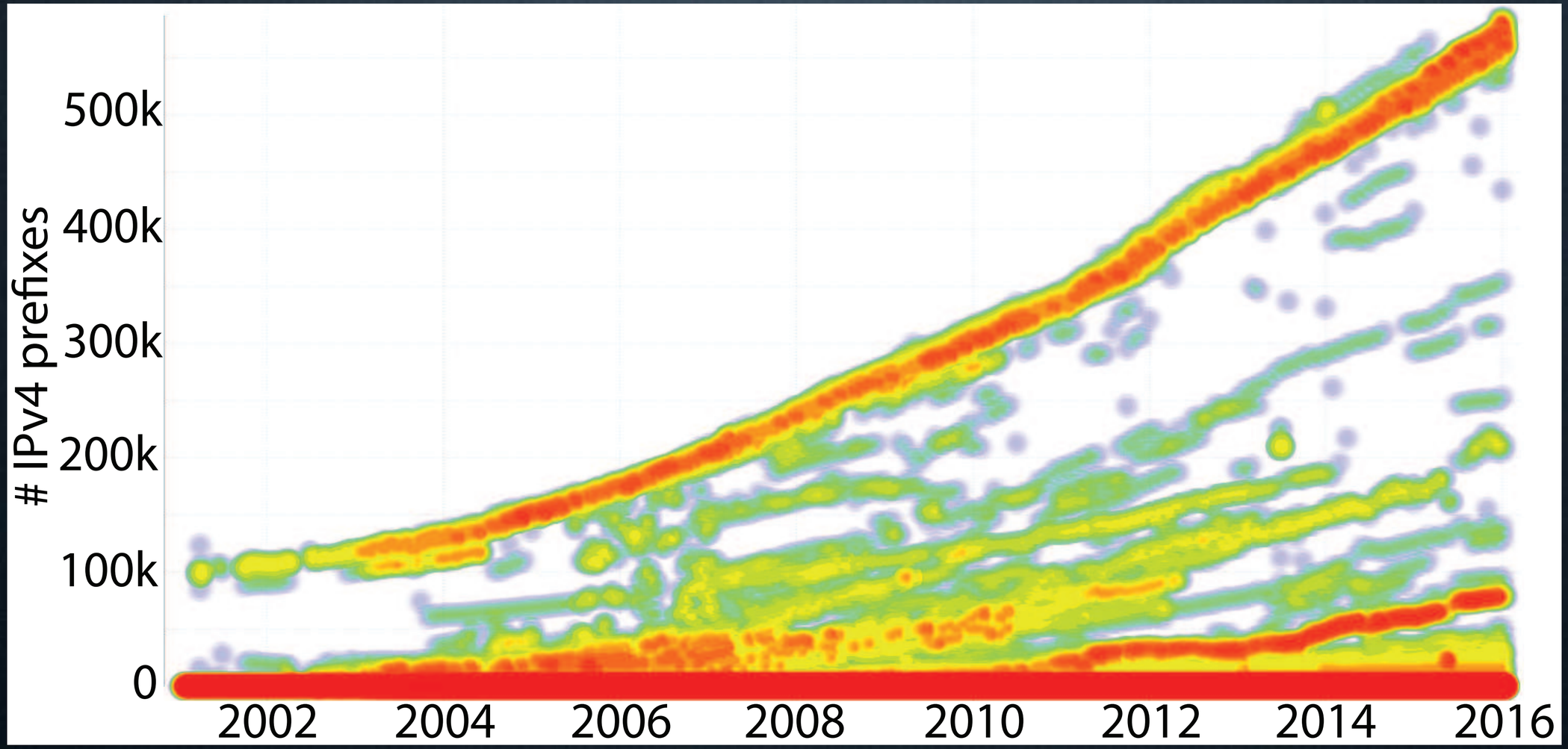
- *“Students can write scripts to analyze BGP data, but I need to do REAL analysis…”*
- We conducted a proof-of-concept study using PyBGPStream with Apache Spark:
- Analyzed 15 years of data:
  - one RIB per month
  - all Route Views and RIPE RIS collectors
  - > 3000 RIBs, ~44 billion BGPStream Elems
- See the paper for more details about lessons learned

- PyBGPStream/Spark template script: https://github.com/CAIDA/bgpstream

caida

BIG DATA - CASE STUDIES

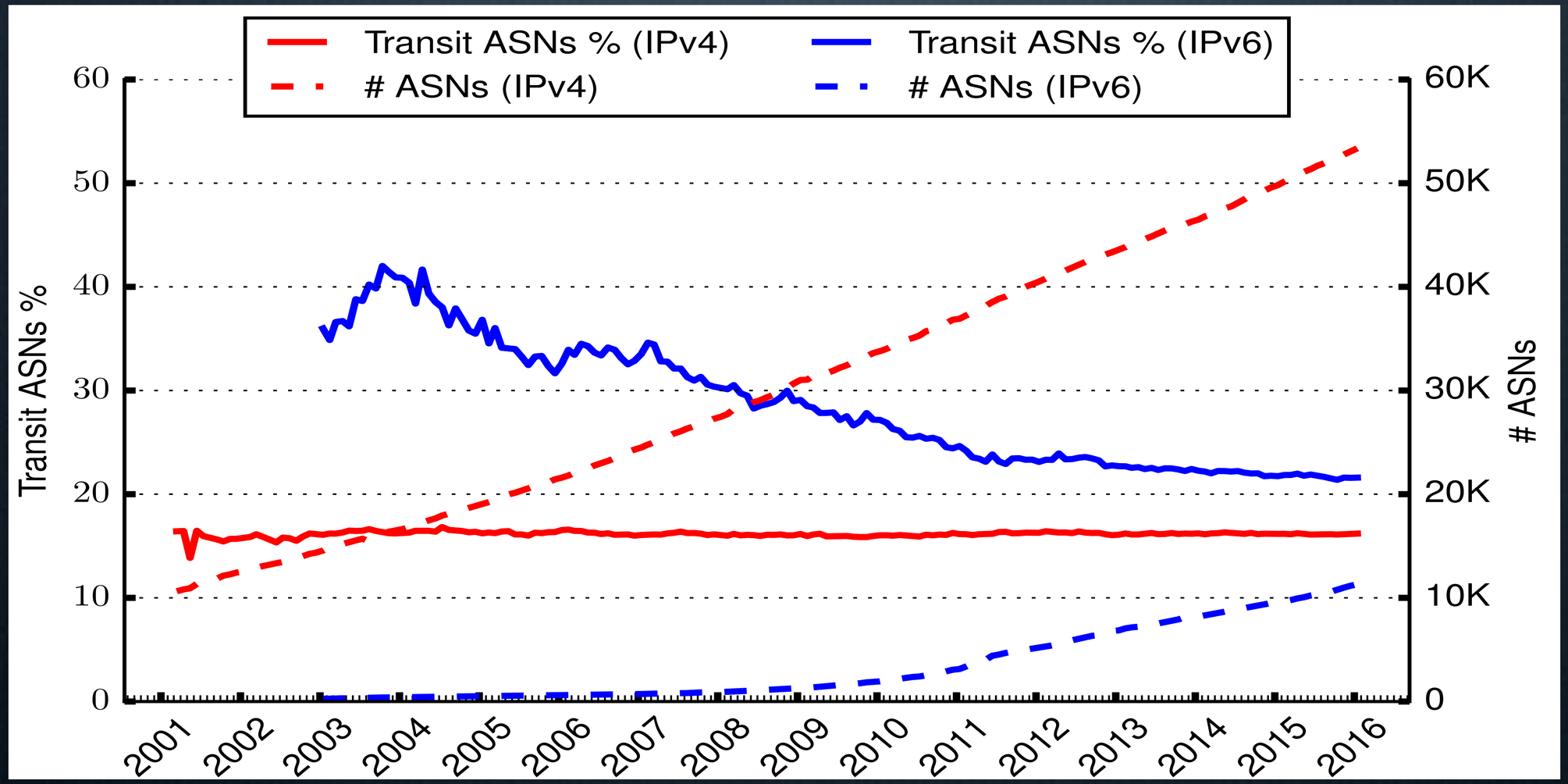
Routing table size over time



caida

BIG DATA - CASE STUDIES

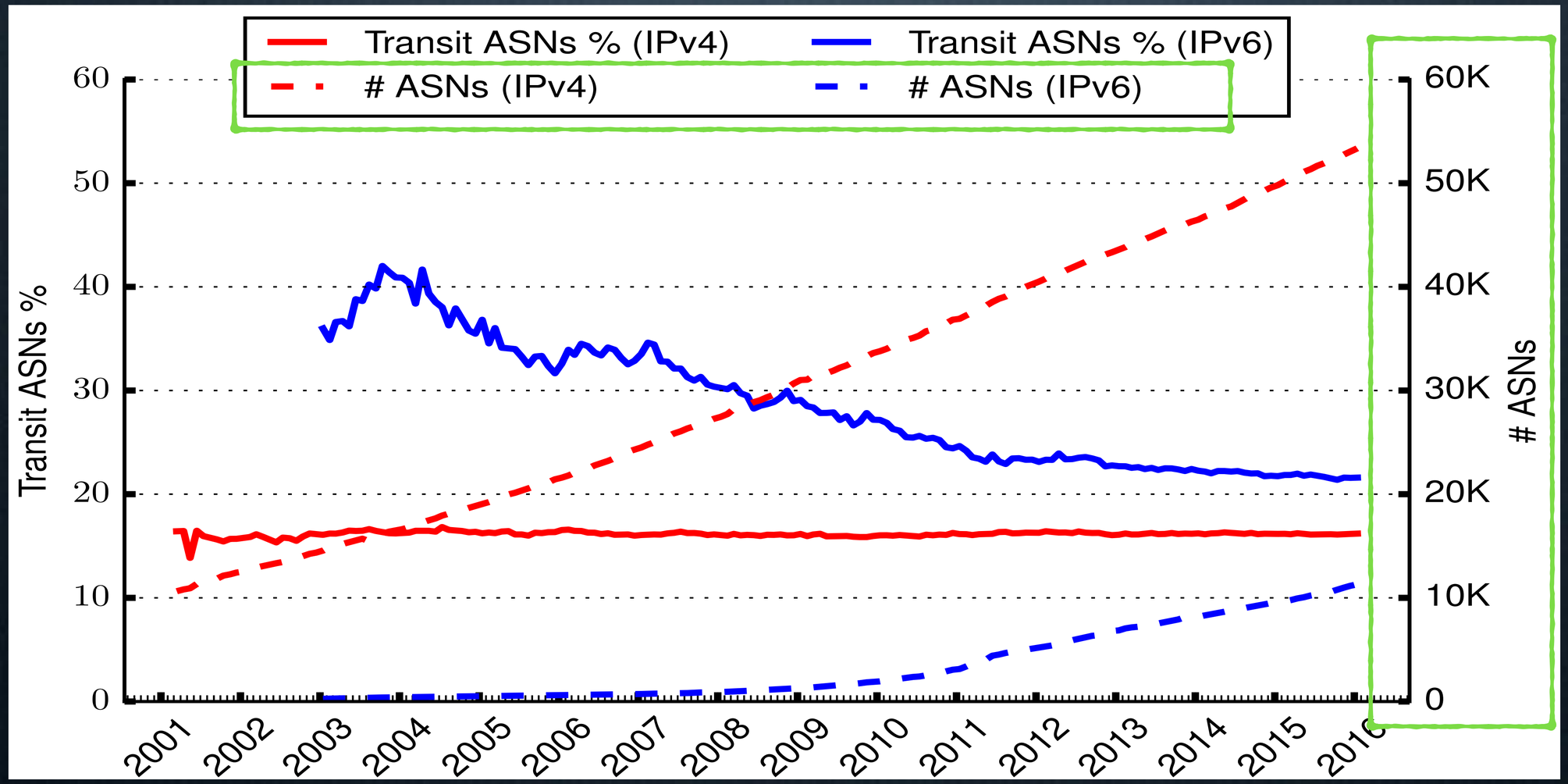
Transit ASes over time



caida

BIG DATA - CASE STUDIES

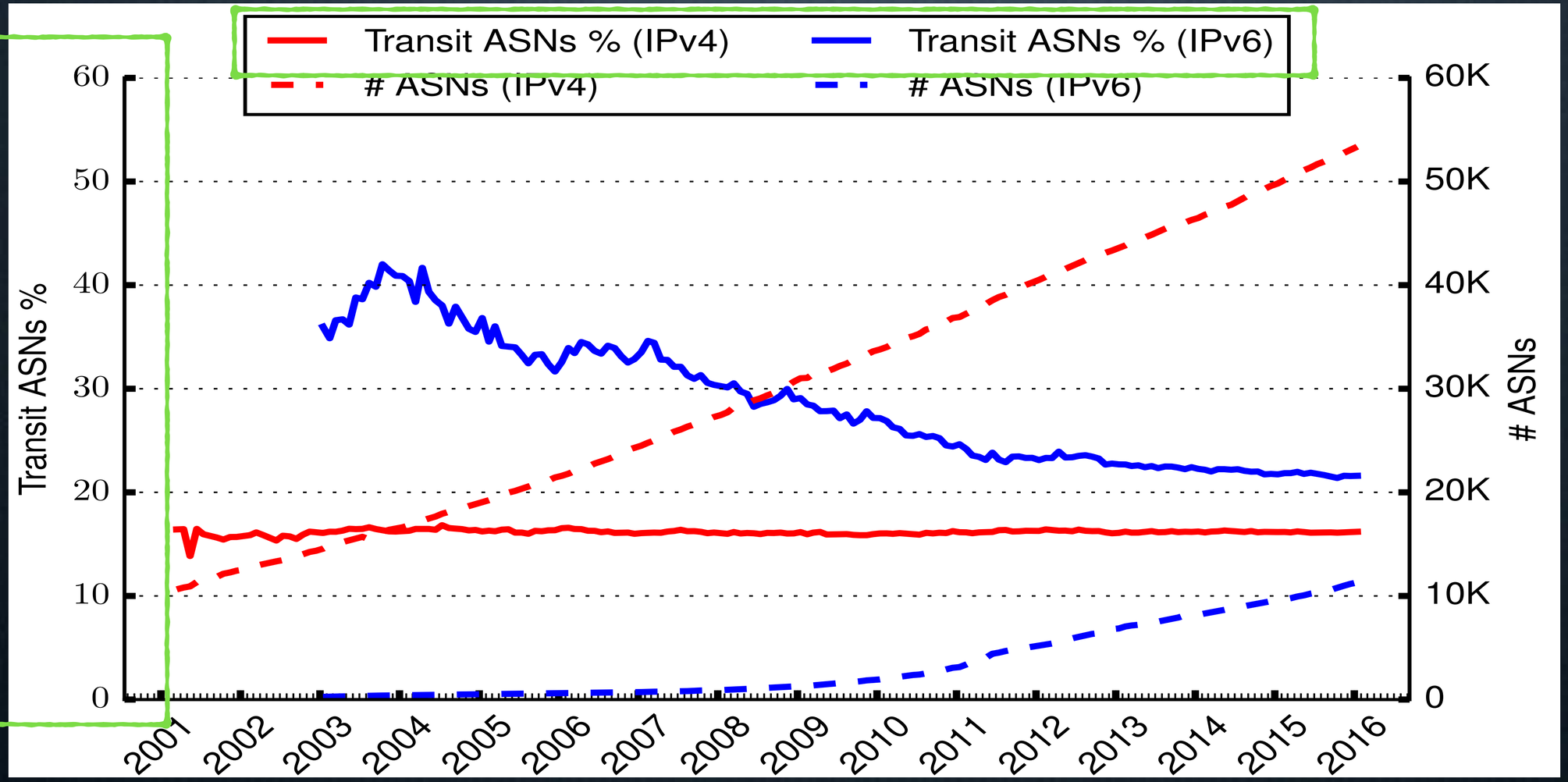
Transit ASes over time



caida

BIG DATA - CASE STUDIES

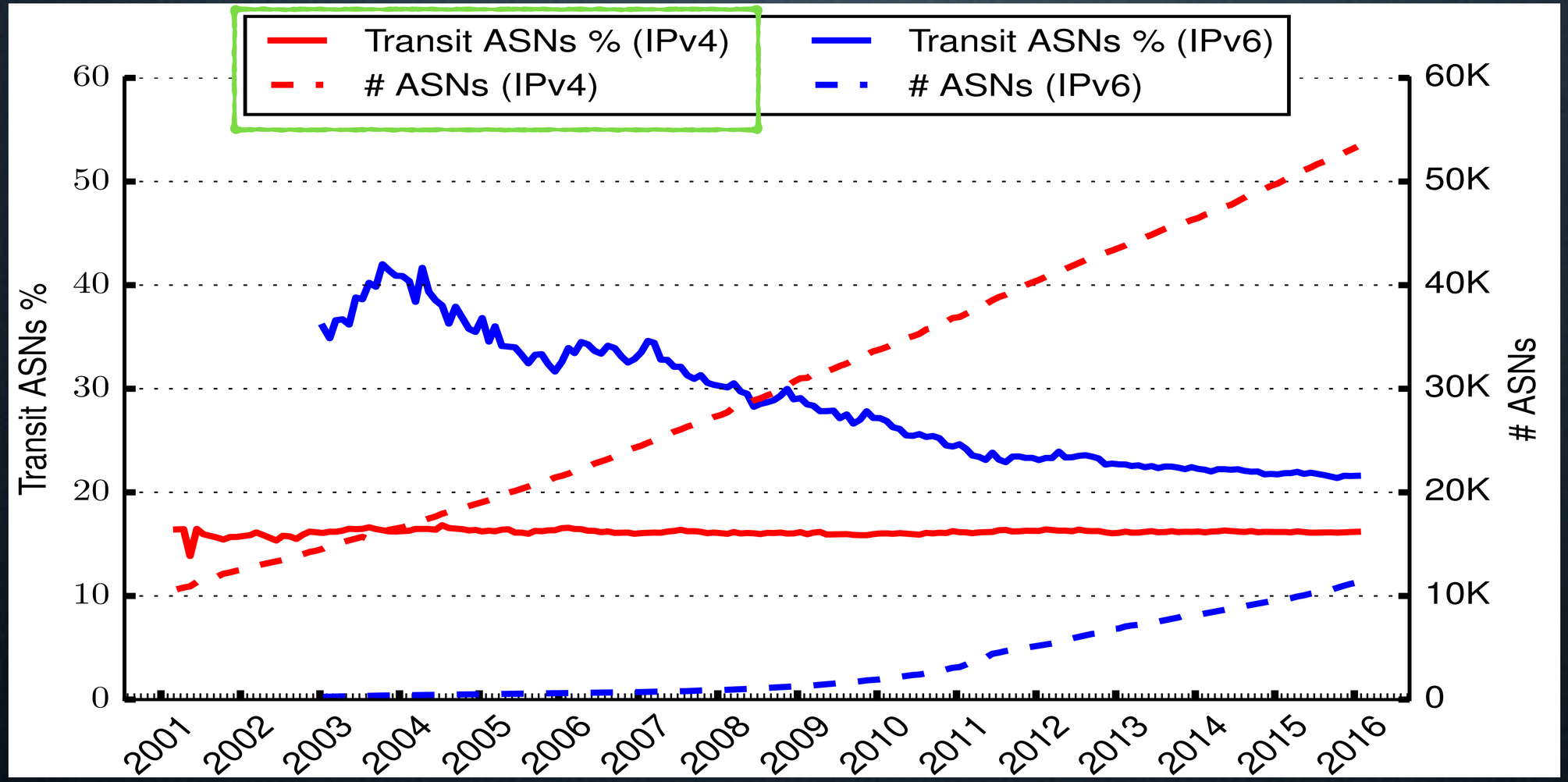
Transit ASes over time



caida

BIG DATA - CASE STUDIES

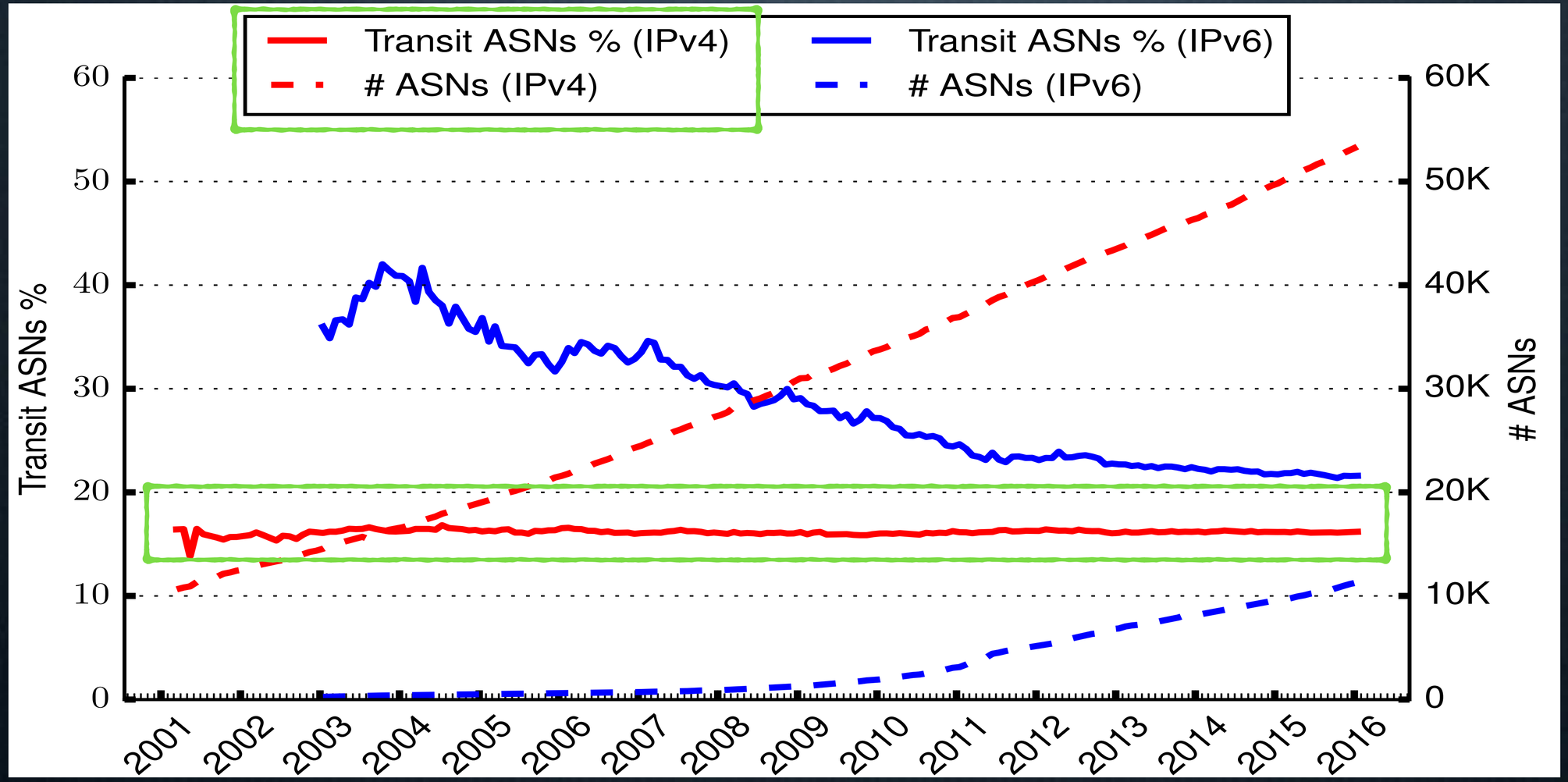
Transit ASes over time



caida

BIG DATA - CASE STUDIES

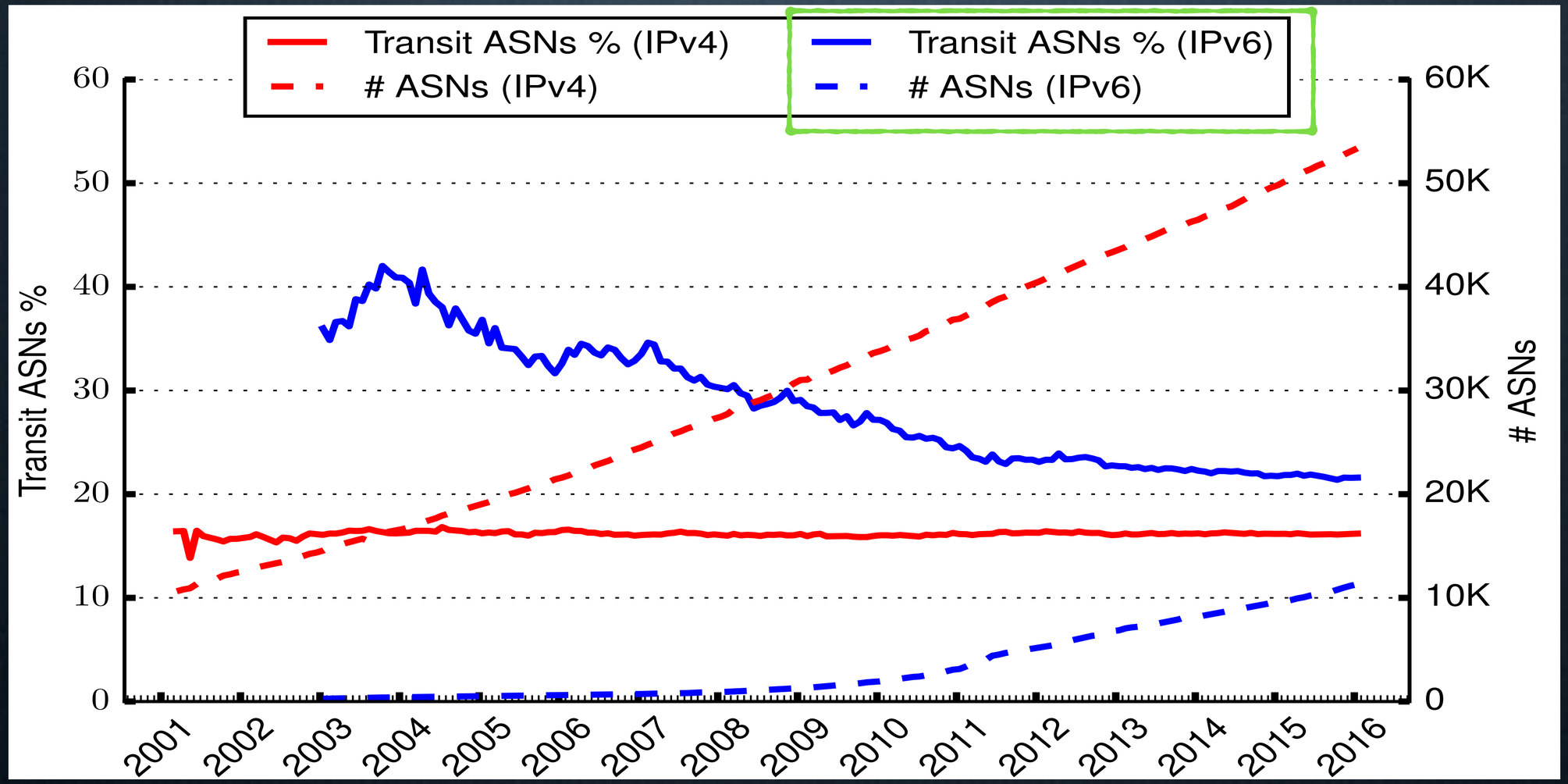
Transit ASes over time



caida

BIG DATA - CASE STUDIES

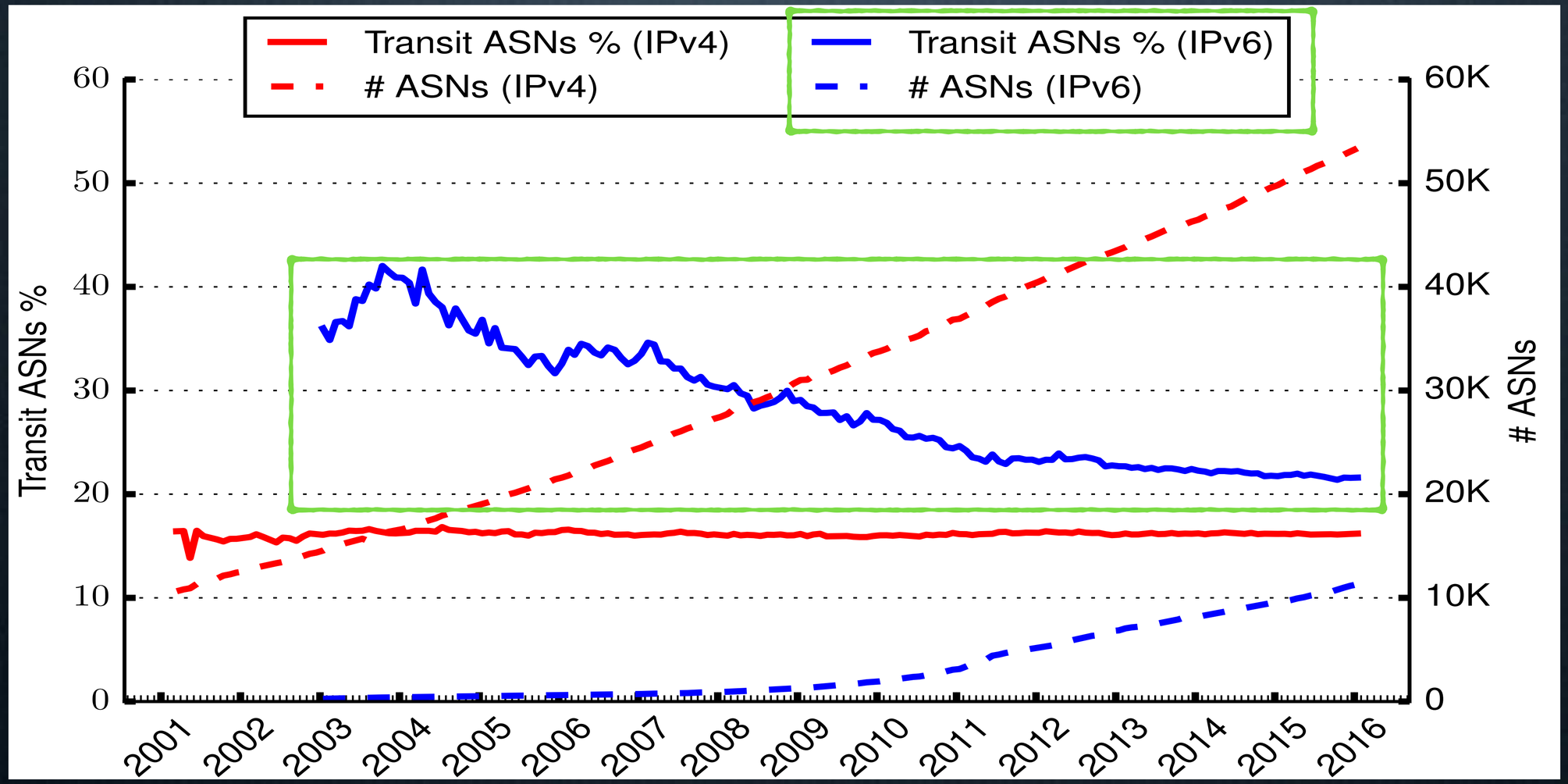
Transit ASes over time



caida

BIG DATA - CASE STUDIES

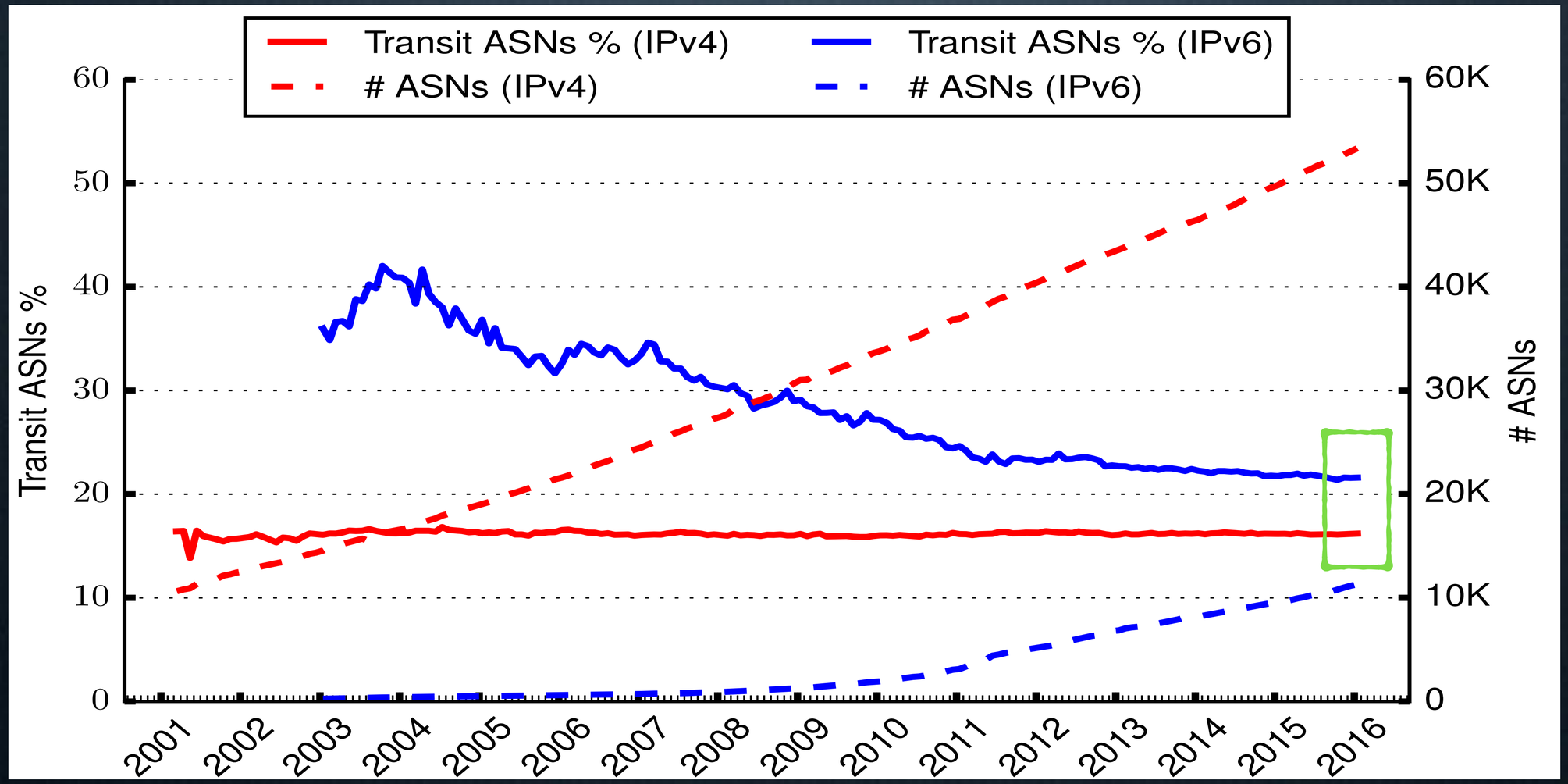
Transit ASes over time



caida

BIG DATA - CASE STUDIES

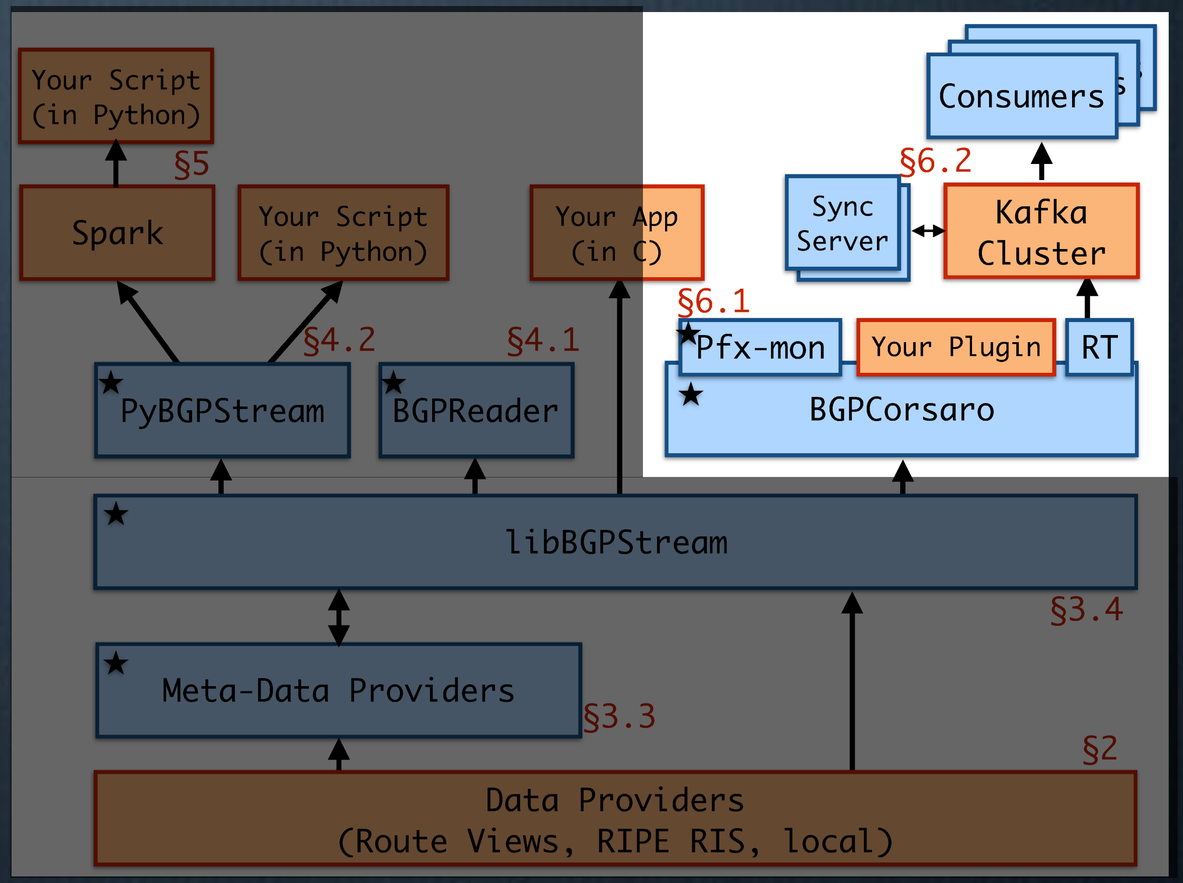
Transit ASes over time



caida

COMPLEX MONITORING INFRASTUCTURE

- E.g. realtime global monitoring for:
  - Internet outages
  - BGP hijacking attacks
- Leveraging BGPCorsaro and BGPStream
- But there are additional challenges…

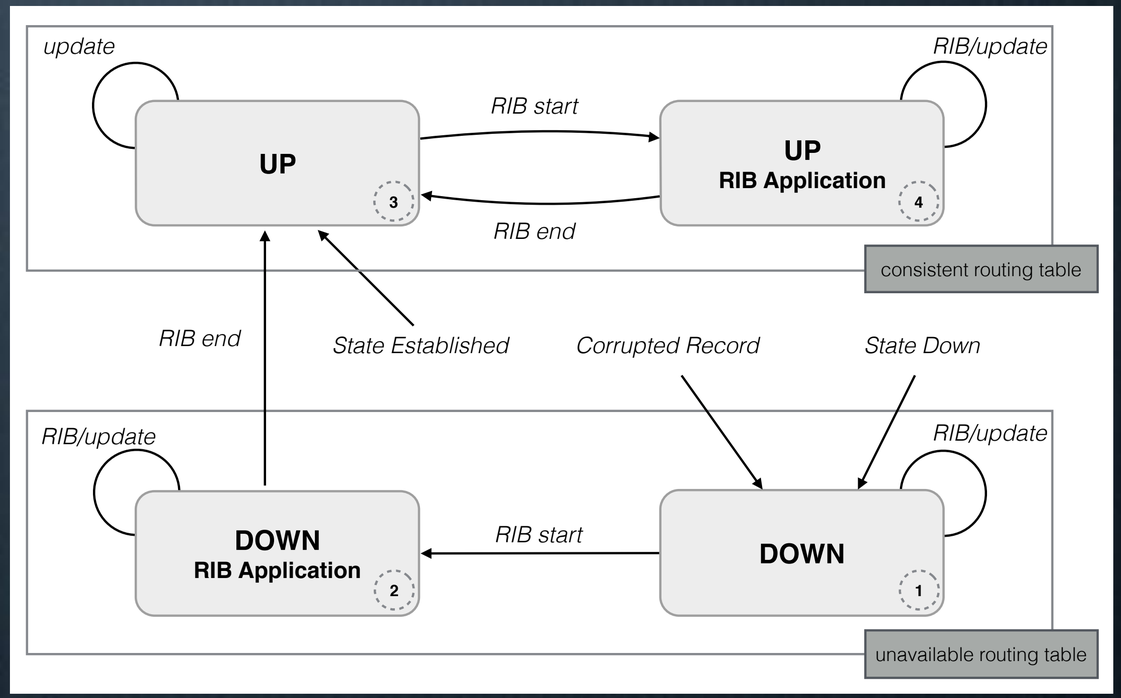


caida

ROUTING TABLES

Continuously rebuilding the state of each peer

- Goal: infer per-peer routing tables every **minute**
  - Route Views and RIPE RIS sample peer routing tables every 4, 8 **hours** respectively
- We infer intermediate states from updates
  - we use RIBs as “sync frames”
  - process modeled as a finite state machine
  - implemented as a BGPCorsaro plugin
  - error rates of 10^{-8} (RIS) and 10^{-5} (RV)

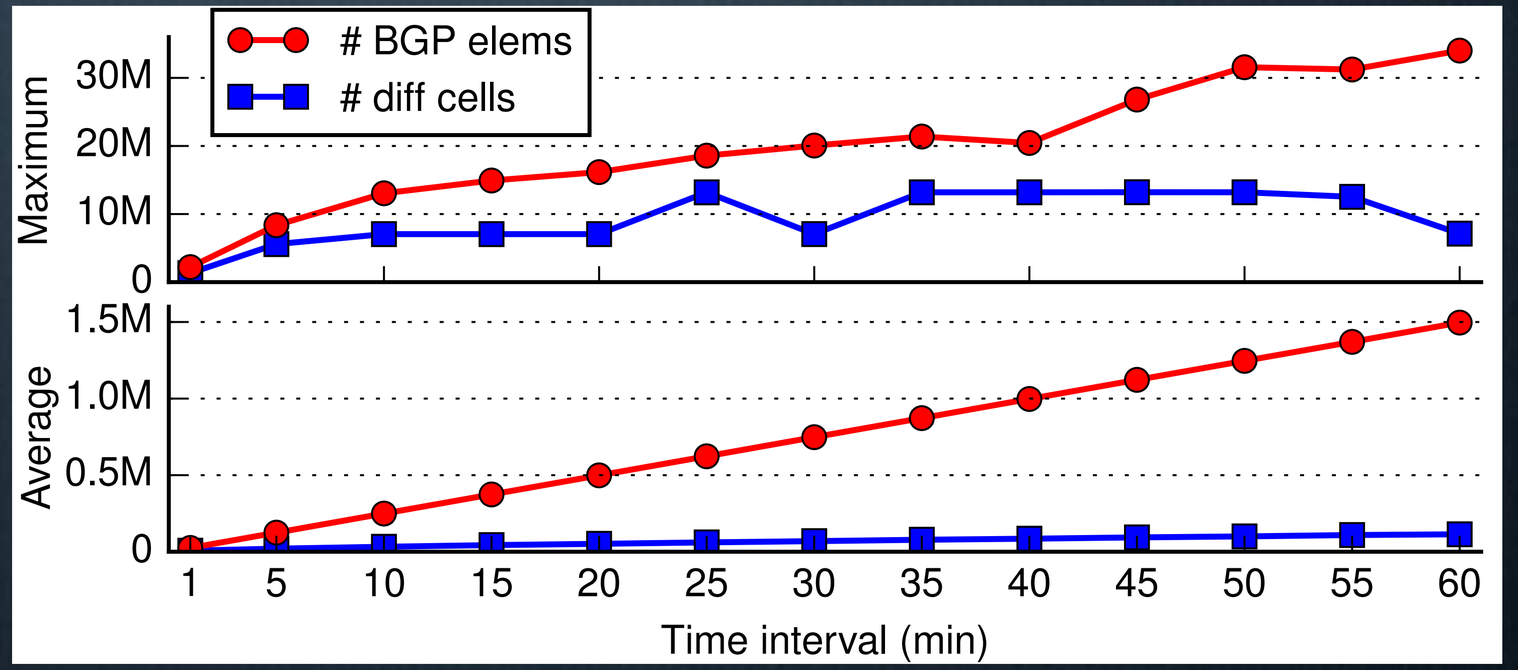


caida

DATA REDUCTION

Removing redundancy in updates

- Significant redundancy in update messages
- Output only changes between successive peer routing tables
- Reduces data volume:
  - 3x reduction at 1min compared to updates

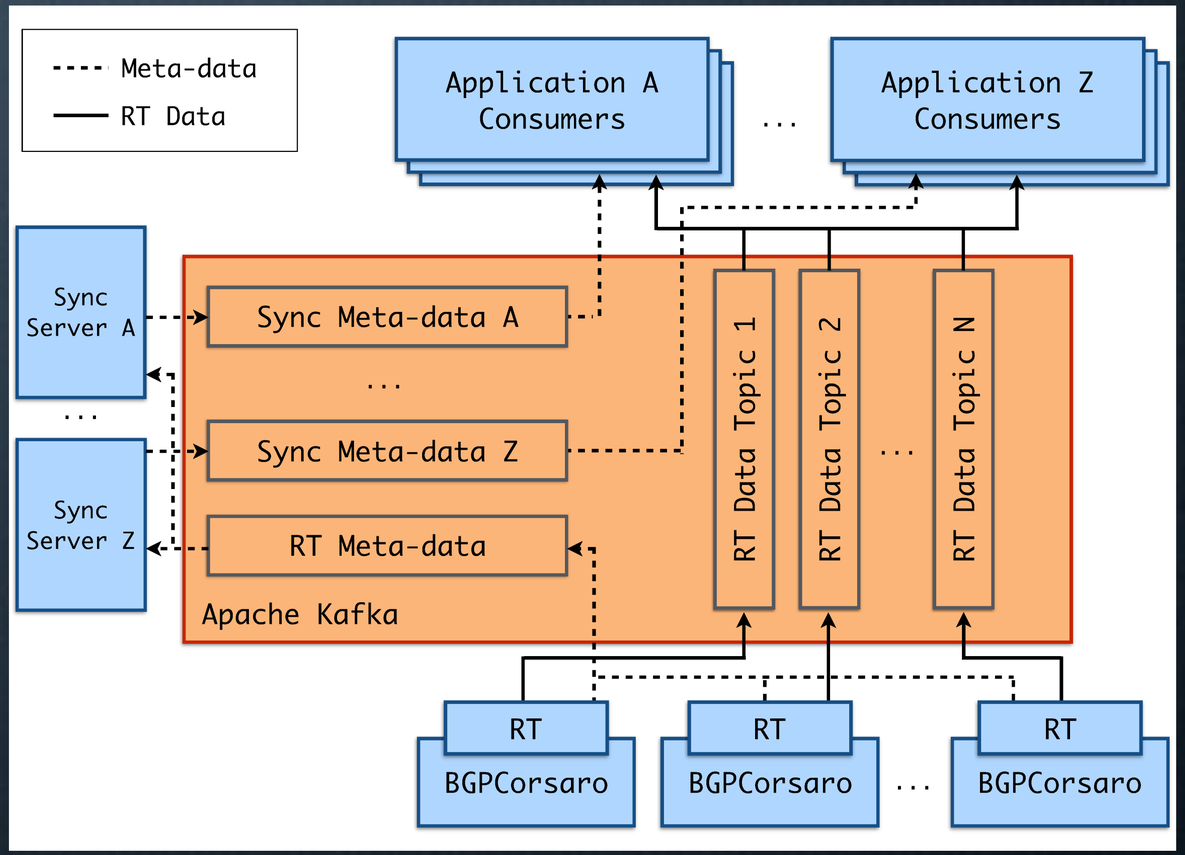


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Data from different projects and collectors is available at different times
- Need to buffer per-peer tables while we wait
- There is a trade-off:
  - Buffer size
  - Latency
  - Completeness
- Some apps need data as *soon as possible*, others as *complete as possible*

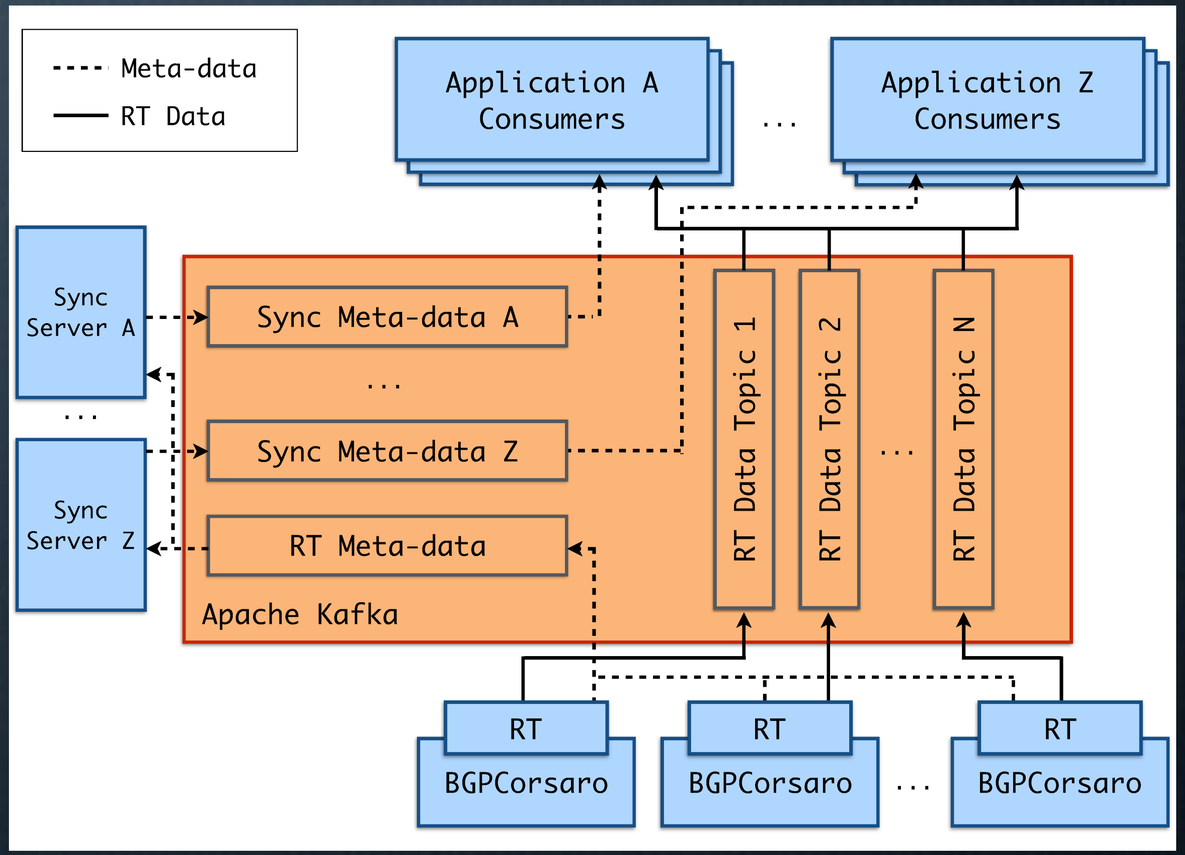


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability

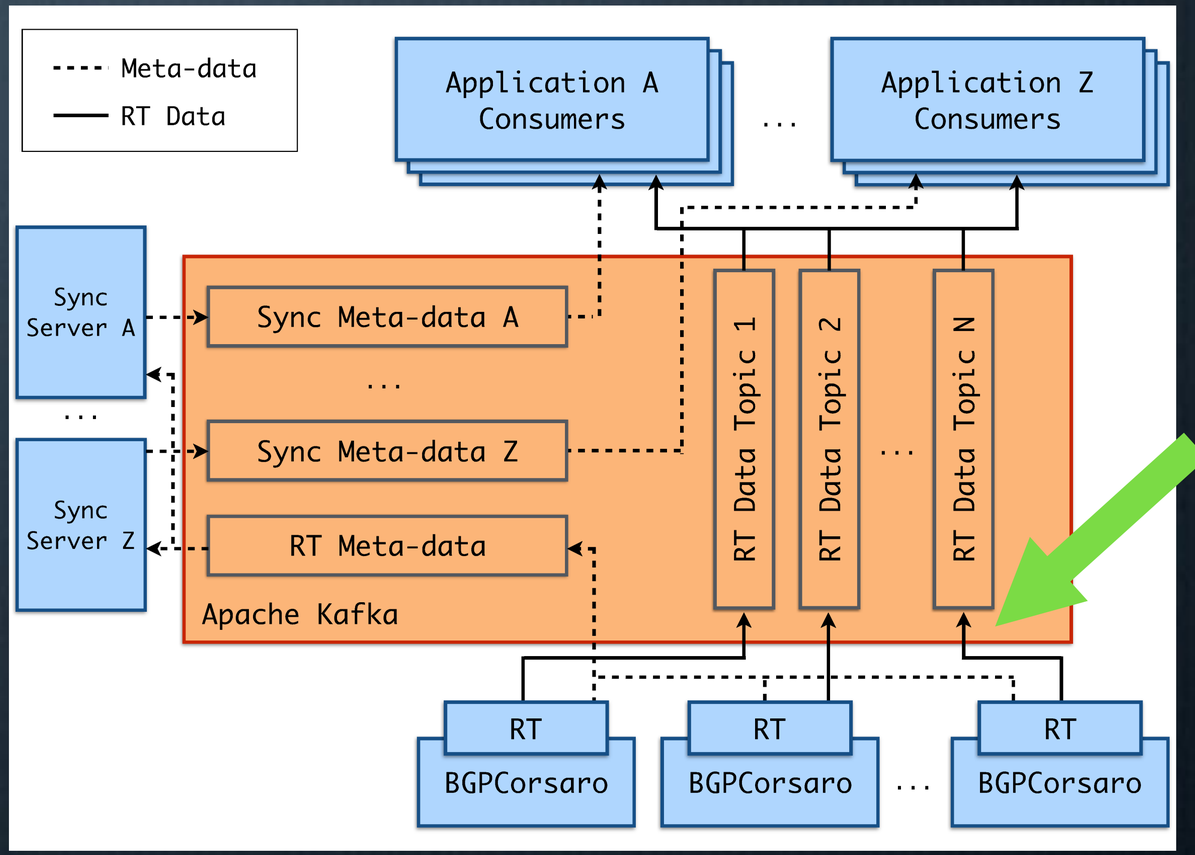


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability

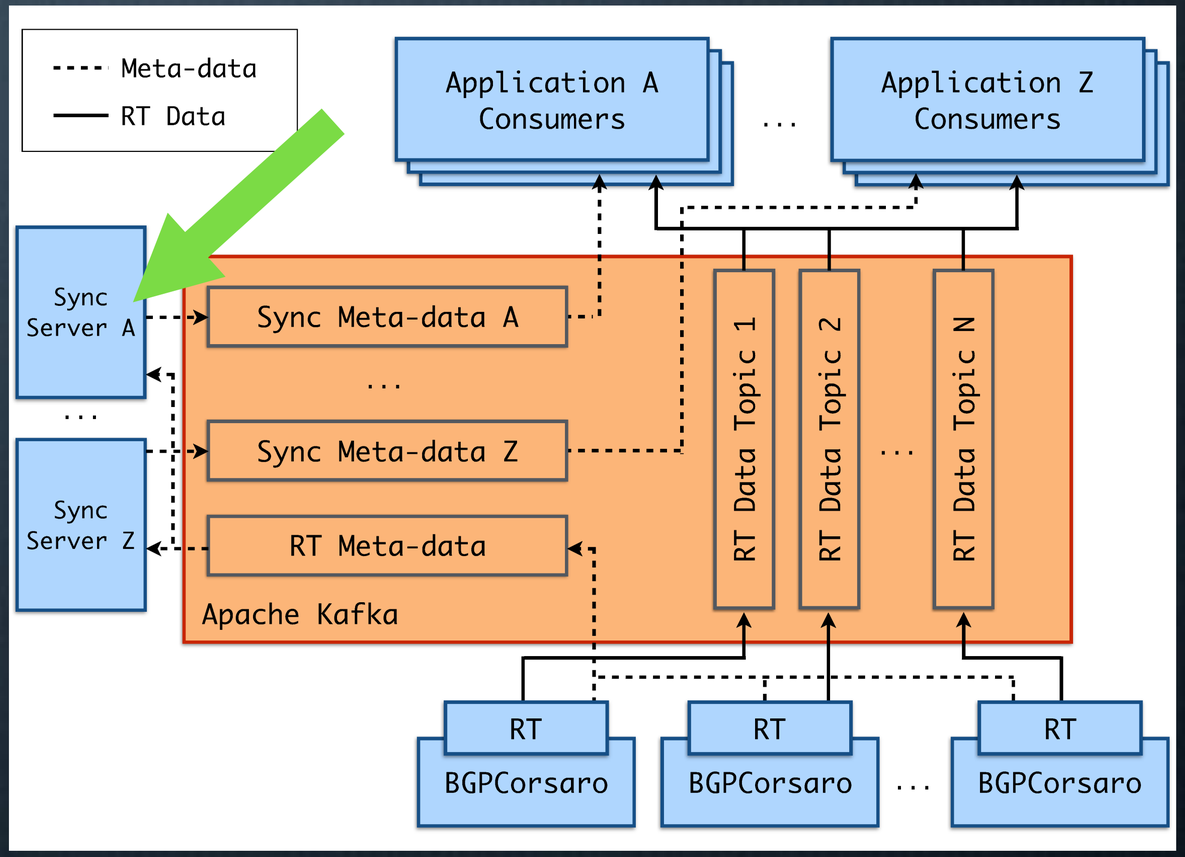


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability

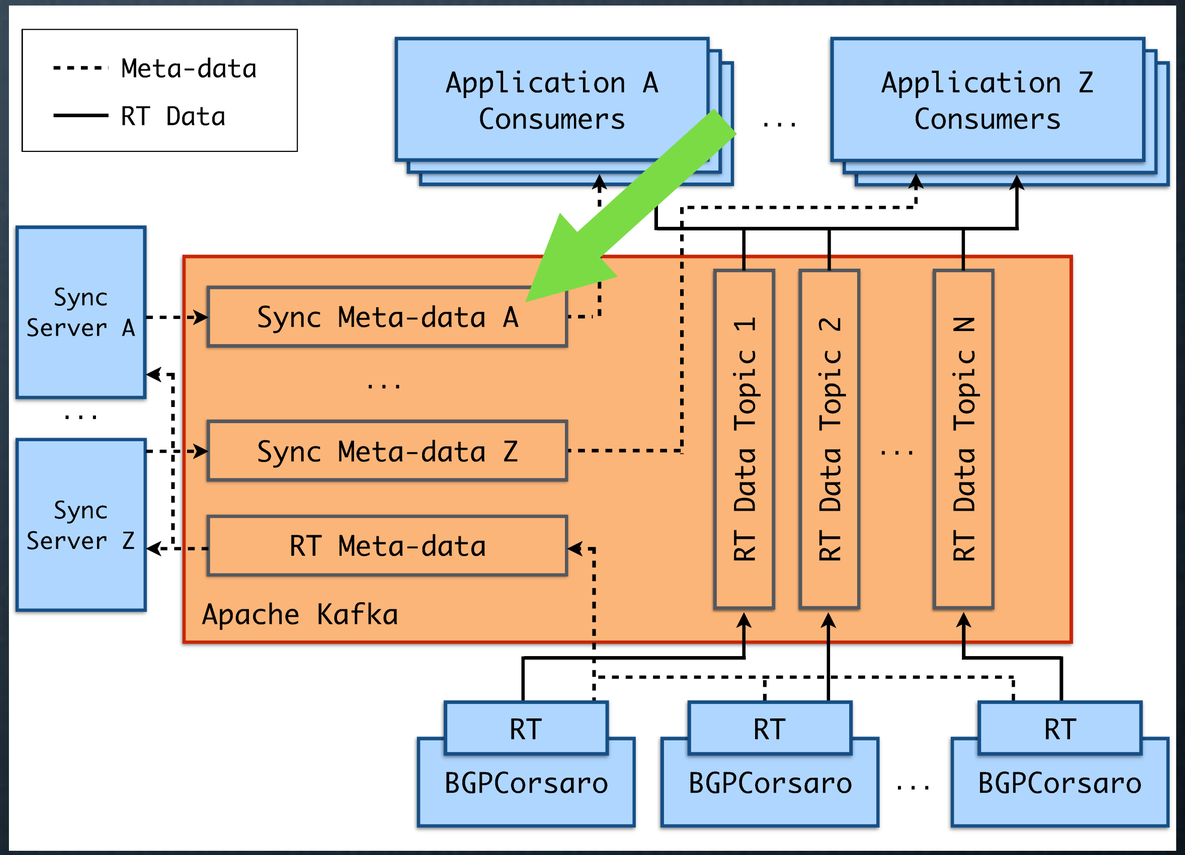


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability

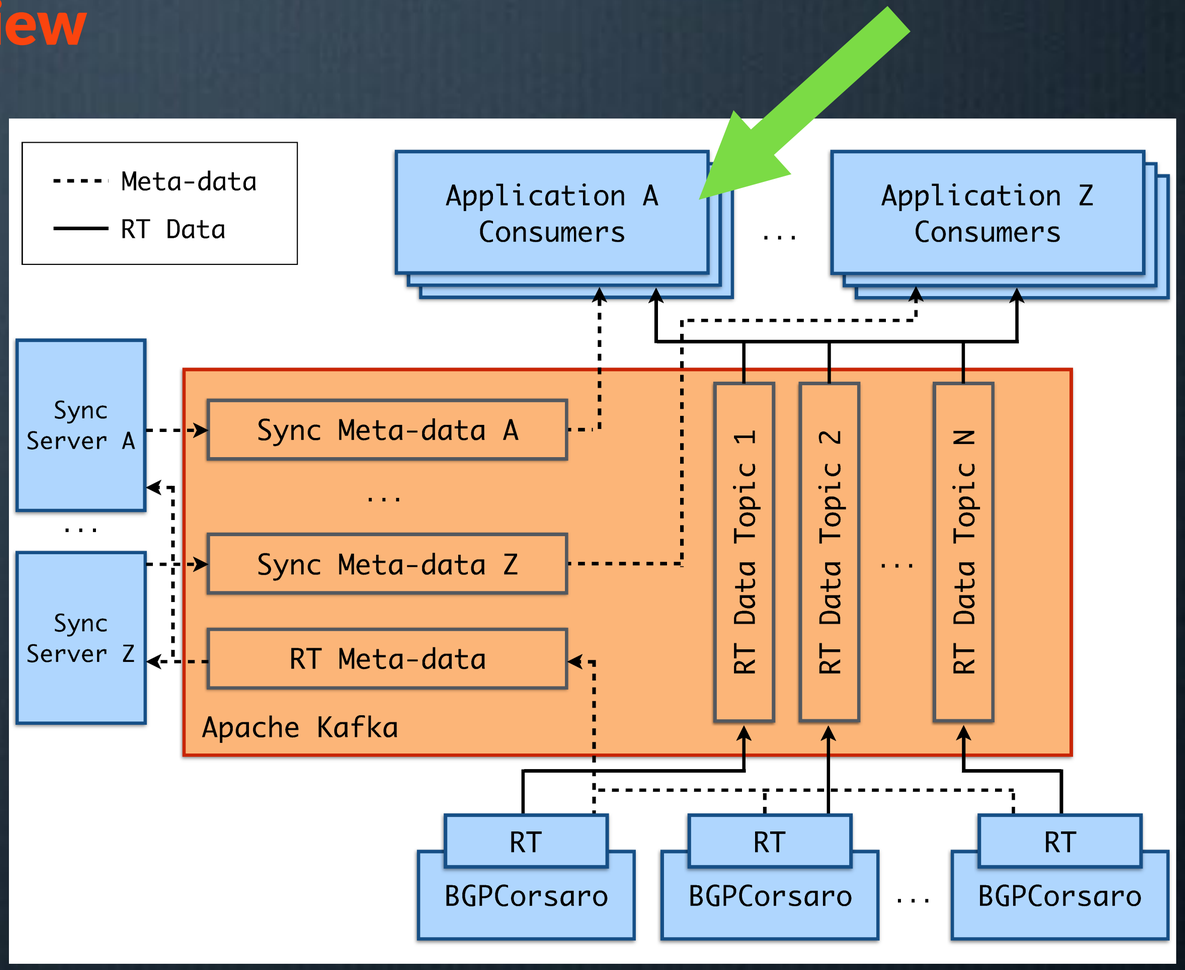


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability

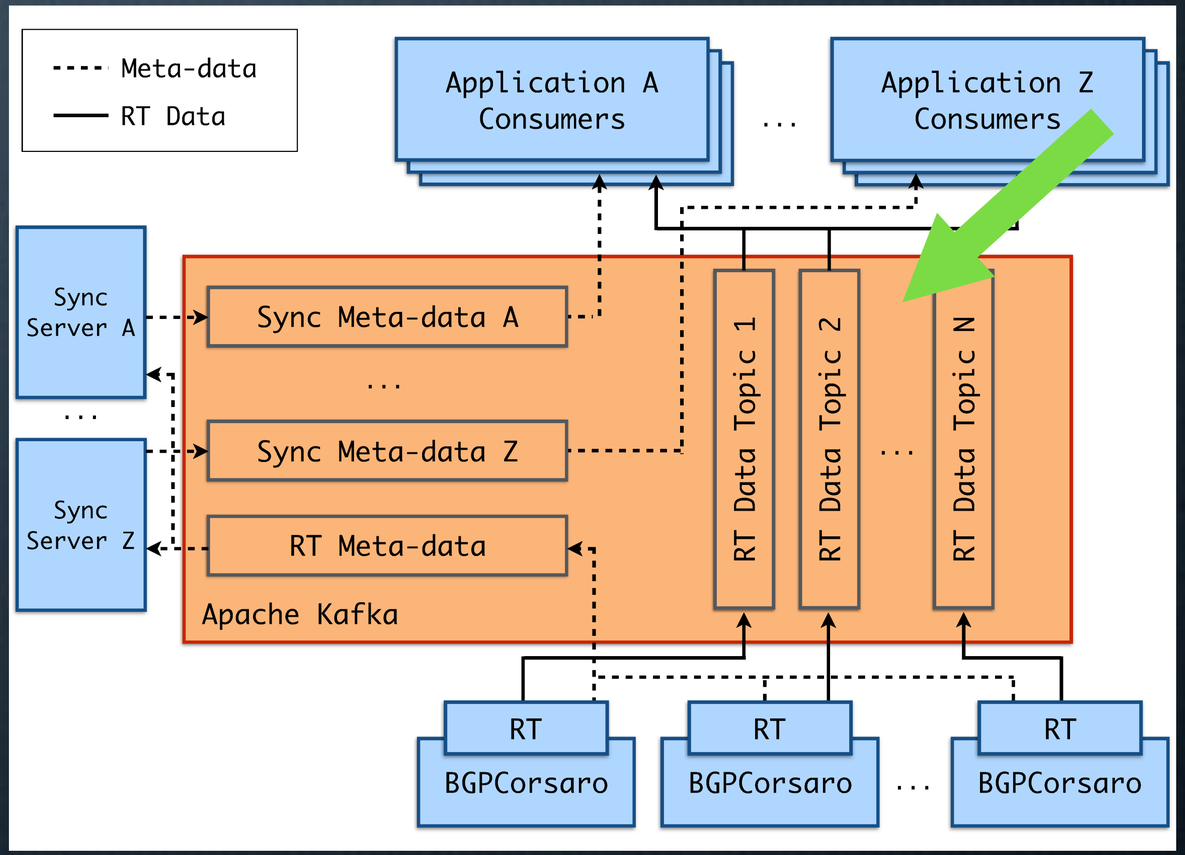


caida

SYNCHRONIZATION

Aligning distributed data into a global view

- Our solution:
  - Supports multiple applications with single architecture
  - Uses a metadata-based gating mechanism
  - Implemented using BGPCorsaro and Apache Kafka
  - Minimal per-application overhead means excellent scalability



caida

A COMMUNITY EFFORT

There's lots to be done

- **We're not alone in working to modernize BGP analysis/monitoring:**
  - OpenBMP, RouteViews, RIPE RIS Streaming, BGPmon (Colorado State)
- **Some coordination between efforts:**
  - Hosted BGP Hackathon in collaboration with Route Views/RIPE/BGPmon
  - Ongoing active collaboration with Cisco & OpenBMP developers

- **BGPStream is complementary:**
  - Allows users to easily take advantage of collection advancements
  - With little/no changes to code

DEVELOPMENT ROADMAP

Coming soon…

- v2.0 release
  - Native BMP [RFC7854]/OpenBMP support
  - Better filter interface, e.g.:
    ```
    prefix any 1.2.3.0/22 and collector rrc06 and aspath '$681_1444_'
    ```
  - Performance improvements & bug fixes

- Public OpenBMP collector *(We're looking for BMP feeds)*
- Additional Metadata Broker web app instances *(We're looking for hosts)*
- RIPE RIS Streaming support

… but, we'd happily reprioritize things based on your feedback

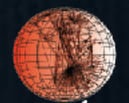
SUMMARY

Easier, faster, less error-prone BGP data analysis

- Improves repeatability, reproducibility and share-ability

- Give it a try!
  - *https://bgpstream.caida.org*
- Give us feedback!
  - *https://github.com/caida/bgpstream*

caida


caida

NEW FILTER SYNTAX

Coming Soon…

```c
#include <bgpstream.h>

int main(int argc, char **argv)
{
  /* Allocate memory for a bgpstream instance */
  bgpstream_t *bs = bs = bgpstream_create();
  /* Allocate memory for a re-usable bgprecord instance */
  bgpstream_record_t *record = bgpstream_record_create();
  /* To hold a pointer to a BGPStream elem */
  bgpstream_elem_t *elem = NULL;

  /* Apply filters using new filter API */
  bgpstream_parse_filter_string(bs, "collector rrc06 and collector route-views.jinx"
                                    " and type updates");

  /* Select a time interval to process:
   * Sun, 10 Oct 2010 10:10:10 GMT -  Sun, 10 Oct 2010 11:11:11 GMT */
  bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

  /* Start the stream */
  bgpstream_start(bs);

```


caida

DELAYS

Getting updates dumps

- 5 (RIS) and 15 (RV) minutes delay due to file rotation duration
- plus small amount of variable delay due to publication infrastructure
- But, 99% of Updates dumps are available in < 20 minutes after the dump was begun

THE CASE FOR SUPPORTING MRT

… for the moment

- *"MRT is dead, why not support a modern collection format?"*

- MRT is still the de-facto standard for BGP data collection
- Loads of *historical* MRT data
  - Route Views and RIPE RIS have >14 years of data (XXTB)
- Vast majority of *new* data collected is still MRT

- Users shouldn't have to care about collection format
- BGPStream: support for MRT internally, but other formats are coming…

BGPREADER

CLI with parseable ASCII output

- Supports all the filters that libBGPStream supports
- Drop-in replacement for *bgpdump*

BGPSTREAM.CAIDA.ORG

Real people are using it!

- **Stable**: version 1.0 was released over a year ago
- **Maintained**: version 1.1 released in Feb
- **Documented:** API documentation and tutorials at bgpstream.caida.org
- **Community involvement via GitHub:**
  - Several community-contributed Pull-Requests,
  - Incl. pending PR to add support for RPKI

FILTERS

Analyze only what you're interested in

- Time
- Collector
- Updates and/or RIBs
- Prefix*
- Community*