Roma Tre University
24th Jun 2016, Rome, IT

BGPStream 2016

Chiara Orsini, Alistair King, Alberto Dainotti
alberto@caida.org

Center for Applied Internet Data Analysis
University of California, San Diego
MEASURING BGP

Why?

BGP is the central nervous system of the Internet

BGP’s design is known to contribute to issues in:

• Availability

• Performance

• Security

Need to engineer protocol evolution!
MEASURING BGP

Why?

Defining problems and make protocol engineering decisions through realistic evaluations is difficult also because we know little about the structure and dynamics of the BGP ecosystem!

• AS-level topology

• AS relationships

• AS interactions: driven by relationships, policies, network conditions, operator updates
MEASURING BGP

two issues - somehow related

1. Literature shows that we need more/better data
   • more info from the protocol/routers

Attempts to generate more info (not much traction in the past):
• RFC 4384 BGP Communities for Data Collection
• draft-ymbk-grow-bgp-collector-communities
MEASURING BGP

two issues - somehow related

1. Literature shows that **we need more/better data**
   • more info from the protocol/routers, more collectors,

Attempts to generate more info
*(not much traction in the past)*:
• RFC 4384 BGP Communities for Data Collection
• draft-ymbk-grow-bgp-collector-communities
MEASURING BGP

two issues - somehow related

1. Literature shows that **we need more/better data**
   • more info from the protocol/routers, more collectors, more experimental testbeds, …

Attempts to generate more info (not much traction in the past):
• RFC 4384 BGP Communities for Data Collection
• draft-ymbk-grow-bgp-collector-communities

Inject/Receive Routes & Traffic.
PEERING - http://peering.usc.edu
MEASURING BGP

two issues - somehow related

1. Literature shows that **we need more/better data**
   • more info from the protocol/routers, more collectors, more experimental testbeds, …

2. But we also **need better tools to learn from the data**
   • to make data analysis: *easier, faster, able to cope with BIG and heterogeneous data*
   • to monitor BGP in near-realtime
   • tightening data collection, processing, visualization, …
1. Literature shows that **we need more/better data**
   - more info from the protocol/routers, more collectors, more experimental testbeds, …

2. But we also **need better tools to learn from the data**
   - to make data analysis: easier, faster, able to cope with BIG and heterogeneous data
   - to monitor BGP in near-realtime
   - tightening data collection, processing, visualization, …
INSPIRING PROJECTS (1/2)
IODA: Detection and Analysis of Internet Outages

• Country-level Internet Blackouts during the Arab Spring

  Dainotti et al. “Analysis of Country-wide Internet Outages Caused by Censorship”
  IMC 2011

• Natural disasters affecting the infrastructure

  Dainotti et al. “Extracting Benefit from Harm: Using Malware Pollution to Analyze the Impact of Political and Geophysical Events on the Internet”
  SIGCOMM CCR 2012

In collaboration with Roma Tre
INSPIRING PROJECTS (1/2)

**IODA: Detection and Analysis of Internet Outages**

Country-wide Internet outages in Iraq that the government ordered in conjunction with the ministerial preparatory exams - Jul 2015

![Graph showing internet outages](image)
INSPIRING PROJECTS (1/2)

IODA: Detection and Analysis of Internet Outages

Outage of AS11351 (Time Warner Cable LLC)
September 30, 2015
BEFORE IODA
post-event manual analysis

Egypt, Jan 2011
Government orders to shut down the Internet

4 months of work

Dainotti et al. “Analysis of Country-wide Internet Outages Caused by Censorship” IMC 2011
In Dec. 2014 we made it possible for anybody to follow the North Korean disconnection almost live

https://charthouse.caida.org/public/kp-outage
INSPIRING PROJECTS (2/2)

Hijacks: detection of MITM BGP attacks

www.caida.org/funding/hijacks/

- S: source (poisoned)
- D: destination (hijacked prefix)
- A: attacker

Legend:
- Green: normal path
- Red: hijacked path
- Orange: normal path used to complete the attack
IODA SYSTEM DIAGRAM
(toy diagram)
• A software framework for **historical** and **live** BGP data analysis

• Design goals:
  - Efficiently deal with large amounts of distributed BGP data
  - Offer a time-ordered data stream of data from heterogeneous sources
  - Support near-realtime data processing
  - Target a broad range of applications and users
  - Scalable
  - Easily extensible

• Paper under submission at IMC ’16
  *Orsini, King, Giordano, Giotsas, Dainotti*
  (older tech report on web site)
**bgpstream.caida.org**
- download it! (version 1.1)
- active development - [github.com/caida/bgpstream](https://github.com/caida/bgpstream)
- Docs & Tutorials
- lots of people are using it!
- coordination with RouteViews, Colorado State BGPMon, RIPE NCC
- BGP Hackathon last February, NANOG Hackathon in June, …
- Funding from Cisco to collaborate and natively support OpenBMP
1. A web service ("BGPStream Broker")
   - enables SIMPLE **access** to LOTS of heterogeneous BGP sources
2. **LibBGPStream**:
   - Acquires the data and provides to upper layers a realtime stream of BGP data
   - makes it SIMPLE to **process** data from LOTS of heterogeneous BGP sources
3. Command-line tools and APIs in C and Python
/** Start the stream */
bgpstream_start(bs);

Listing 1 shows sample code that uses the BGPStream API to print out all announced and withdrawn prefixes from the stream (lines 15-19) and then iteratively requests new records to process them. A BGPStream API program that uses the BGPStream API would look like this (using the protocol described in Section 4), and opening multiple files in parallel from a single process, using the version of libBGPStream that we extended to: (i) read remote paths (HTTP and HTTPS), (ii) support opening and reading from multiple files in parallel from a single process, and (iii) signal the event of a corrupted read.

To open MRT dumps, we use a version of libBGPStream that supports registering multiple MRT files and either opening them at the same time or opening them one at a time. The libBGPStream user API provides the essential library functions for processing, or libBGPStream receives an interrupt for processing, or libBGPStream receives a RIB dump. Information related to a single prefix may group elements of the same type but related to different VPs or prefixes, such as routes to the same destination.

For example, the BGPStream record to connect to a RIB dump, information related to a single prefix from different collector projects (e.g., Route Views, RIPE RIS), lists prefixes that may group elements of the same type but related to different VPs connected to a RIB dump. Multiple MRT records may contain a de-
A BGP record encapsulate an MRT record

Dumps are composed of multiple MRT records, whose type is specified in their header
-an update message is stored in a single MRT record, but multiple update messages can be in the same MRT record (see next slide)

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>project</td>
<td>string</td>
<td>project name (e.g., Route Views)</td>
</tr>
<tr>
<td>collector</td>
<td>string</td>
<td>collector name (e.g., rrc00)</td>
</tr>
<tr>
<td>type</td>
<td>enum</td>
<td>RIB or Updates</td>
</tr>
<tr>
<td>dump time</td>
<td>long</td>
<td>time the containing dump was begun</td>
</tr>
<tr>
<td>position</td>
<td>enum</td>
<td>first, middle, or last record of a dump</td>
</tr>
<tr>
<td>time</td>
<td>long</td>
<td>timestamp of the MRT record</td>
</tr>
<tr>
<td>status</td>
<td>enum</td>
<td>record validity flag</td>
</tr>
<tr>
<td>MRT record</td>
<td>struct</td>
<td>de-serialized MRT record</td>
</tr>
</tbody>
</table>
An MRT record may group elements of the same type but related to different VPs or prefixes
- e.g., routes to the same prefix from different VPs (in a RIB dump record)
- e.g., announcements from the same VP to multiple prefixes, but sharing a common path (in a Updates dump record)

**libBGPStream decomposes a record into a set of individual elements (BGPStream elems)**

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

* denotes a field conditionally populated based on type
C API

while loop

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

/* Read the stream of records */
while (bgpstream_get_next_record(bs, record) > 0) {
    /* Ignore invalid records */
    if (record->status != BGPSTREAM_RECORD_STATUS_VALID_RECORD) {
        continue;
    }
    /* Extract elems from the current record */
    while ((elem = bgpstream_record_get_next_elem(record)) != NULL) {
        /* Select only announcements and withdrawals, */
        /* and only elems that carry information for 2403:f600::/32 */
        if ((elem->type == BGPSTREAM_ELEM_TYPE_ANNOUNCEMENT ||
             elem->type == BGPSTREAM_ELEM_TYPE_WITHDRAWAL) &&
             bgpstream_pfx_storage_equal(&my_pfx, elem->prefix)) {
            /* Print the BGP information */
            bgpstream_elem_snprintf(buffer, 1024, elem);
            printf(stdout, "%s\n", buffer);
        }
    }
}
```
$ bgpreader -w 1445306400,1445306402 -c route-views.sfmix
R|B|1445306400|routeviews|route-views.sfmix
R|R|1445306400|routeviews|route-views.sfmix|32354|206.197.187.5|1.0.0.0/24|206.197.187.5|32354 15169|15169|||
R|R|1445306401|routeviews|route-views.sfmix|14061|2001:504:30::ba01:4061:1|2c0f:ffd8::/32|
R|R|1445306401|routeviews|route-views.sfmix|32354|206.197.187.5|1.0.0.0/24|206.197.187.5|32354 15169|15169|||
...
Example: studying AS path inflation

How many AS paths are longer than the shortest path between two ASes due to routing policies? (directly correlates to the increase in BGP convergence time)

Listing 2

```python
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: defaultdict(int)))
stream.add_interval_filter((1438415000, 1438416600))
stream.start()

while(stream.get_next_record(rec)):
    elem = rec.get_next_elem()
    while(elem):
        if str(elem.peer_asn)
            monitor = str(elem.peer_asn)
            hops = {k: g for k, g in groupby(elem.fields['as-path'].split(' '), lambda x: x in monitor)}
            bgp_lens[monitor][origin] += \
            min(filter(lambda x: bgp_lens[monitor][origin], len(hops)))

    elem = rec.get_next_elem()
    for elem 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
PYBGPSTREAM

Example: timely combine with active measurements

…… In the paper you’ll find a case study that uses PyBGPStream to detect blackholing (a mitigation measure against denial-of-service attacks) and triggers traceroute measurements from RIPE Atlas to better characterize the event.
The “prefix-monitor” plugin (distributed with source) monitors a set of IP ranges as they are seen from BGP monitors distributed worldwide:
- how many prefixes reachable
- how many origin ASes
- generates detailed logs

libBGPStream talks to the broker and gets the data

```python
bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc06");
bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views.jinx");
bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");
bgpstream_add_interval_filter(bs, 1286705410, 1286709071);

stream.add_filter('record-type', 'ribs')
stream.add_filter('collector', 'route-views.sfmix')
stream.add_interval_filter(1445306400,1445306402)

$ bgpreader -w 1445306400,1445306402 -c route-views.sfmix -t updates
$ bgpCorsaro -w 1445306400,1445306402 -p ris
```

Experiments can be easily reproduced: a script defines the (public) data used.
GET A LIVE STREAM

libBGPStream keeps retrieving data as it becomes available

```python
stream.add_filter('record-type',
  'ribs')
stream.add_filter('collector',
  'route-views.sfmix')
stream.add_interval_filter(1445306400,
-1)
```

```sh
$ bgpreader -c route-views.sfmix -t updates
$ bgpcorsaro -p ris
```

Experiments can be easily repeated: a script defines the (public) data used
44 Billion BGPElems processed w/ Spark + PyBGPStream

There are a few observations in this experiment: VP concentrations in data from 2,296 VPs (warmer colors representing a higher percentage of the maximum at each time bin). Aggregated data (collector and project) is depicted as grey circles. In this experiment, we also compute, at each midnight of the 1st day of the month (thus we perform a level of aggregation (VP, collector, overall), the number of ASes, the fraction of transit ASes over time has been constant! For IPv6 instead, overall there has been a fast rate: the IPv6 graph is growing fast while its edge transit portions have recently started growing at a similar pace! (Approaching the property we observed previously). Besides the slow growth in observable MOAS sets over the last 15 years.) As of January 2016, such decay has slowed down considerably, while the number of unique sets of ASes (dashed lines) and percentage of those ASes which are classified as transit – i.e., appearing in the middle of an AS path) observed for IPv4, despite the nearly-linear growth in the number of unique prefixes and ASes observed, which we use to normalize data in the other experiments.

We then calculated the number of transit ASes (ASes appearing in the middle of an AS path – (solid lines), for both IPv4 (red lines) and IPv6 (other lines).
There are a few observations in this experiment.

- The heatmap of data from 2,296 VPs (warmer colors represent a higher concentration of points) is depicted in Figure 5a.
- Only 710 VPs out of 2,296 are within 20 percentage points of the maximum at each time bin.
- RIB-outs are numerous and they significantly skew the distribution; only 710 VPs out of 2,296 are within 20 percentage points of the maximum at each time bin.
- MOAS prefixes, i.e., those showing significantly smaller Adj-RIB-outs, are numerous and they significantly skew the distribution; only 710 VPs out of 2,296 are within 20 percentage points of the maximum at each time bin.

Besides the slow growth in observable MOAS sets over the last 15 years, as of January 2016, though, the fraction of transit ASes is much larger than the fraction of RIB-OUTs appearing in the middle of an AS path – (solid lines), for both IPv4 (red lines) and IPv6 (blue lines). The graph in Figure 5b, shows the results of an experiment in which we find that both the Route Views and RIPE RIS repositories record the same data; however, they do not have a single full-feed peer, thus may not provide us with enough information for most experiments; instead, we find a constant number of ASes identified in the MOAS prefixes aggregated by a single collector.

The RIB-Outs of VPs are proportional to the number of distinct AS identifiers they observe.

Center for Applied Internet Data Analysis
University of California San Diego
There are a few observations in this experiment. The heatmap of data from 2,296 VPs (warmer colors represent a higher concentration of points from different VPs). Aggregated data (collector and project) is depicted as grey circles. In this experiment, we also compute, at each month, the percentage points of the maximum at each time bin. Besides the slow growth in observable MOAS sets over time, this graph highlights that to obtain a better view of data from the 15th day of the month. In this experiment, we also compute, at each month, the percentage points of the maximum at each time bin.

The chart shows the growth of IPv4 ASNs over time, aggregated into overall (top blue line) and per-collector (other lines). The IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate: the IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate; the IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate.

The chart shows the growth of IPv4 ASNs over time, aggregated into overall (top blue line) and per-collector (other lines). The IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate: the IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate.

The chart shows the growth of IPv4 ASNs over time, aggregated into overall (top blue line) and per-collector (other lines). The IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate: the IPv6 graph is growing fast while its edge transit portions have recently started growing at a fast rate.

CRUNCH BIG DATA
44Billion BGPElems processed w/ Spark + PyBGPStream

Transit ASes

- Transit ASNs % (IPv4)
- Transit ASNs % (IPv6)
- # ASNs (IPv4)
- # ASNs (IPv6)

Center for Applied Internet Data Analysis
University of California San Diego
There are a few observations in this experiment. There is a heatmap of data from 2,296 VPs (warmer colors representing a higher concentration of points from different VPs). Aggregated data (collector and project) is depicted as grey circles. (January 2016). VPs are depicted as circles with a diameter and color proportional to the number of distinct AS identifiers they observe.

Besides the slow growth in observable MOAS sets over time, this graph highlights that to obtain a better view following contributing to MOAS prefixes aggregated into overall (top blue line) and per-collector (other lines). Over the years, MOAS prefixes have been classified as transit – i.e., appearing in the middle of an AS path – (solid lines), for both IPv4 (red lines) and IPv6 (blue lines). The number of transit ASes (ASes appearing in the middle of an AS path) observed for IPv4, despite the nearly-linear growth in the number of ASes, the fraction of transit ASes over time has grown faster than transit. However, since around 2012, such decay has slowed down considerably, while growing faster than transit. In the IPv6 graph, overall there has been a steady growth in the number of unique ASNs (dashed lines) and percentage of those ASNs which are classified as transit – i.e., appearing in the middle of an AS path – (solid lines) for both IPv4 (red lines) and IPv6 (blue lines). The graph in Figure 5c, shows that the percentage points of the maximum at each time bin are numerous and they significantly skew the distribution; only 710 VPs out of 2,296 are within 20%

For IPv4, the fraction of transit ASes has increased from around 10% in 2001 to around 60% in 2016, though, the fraction of transit ASes is much larger, as many collectors as are available: the number of MOAS sets identified in the last 15 years! (Approaching the property we observed previously). RIB-outs are numerous and they significantly skew the distribution; only 710 VPs out of 2,296 are within 20%

In this experiment, we also compute, at each month, the number of unique prefixes and ASes observed, which we use to normalize data in the other experiments. We then calculated the number of transit ASes (ASes appearing in the middle of an AS path) observed for IPv4, despite the nearly-linear growth in the number of ASes, the fraction of transit ASes over time has grown faster than transit. However, since around 2012, such decay has slowed down considerably, while growing faster than transit. However, since around 2012, such decay has slowed down considerably, while growing faster than transit. However, since around 2012, such decay has slowed down considerably, while growing faster than transit. However, since around 2012, such decay has slowed down considerably, while growing faster than transit.
Live mode introduces the problem of sorting records from collectors that may publish data at variable times: trade-off between:

- size of buffers
- completeness of data available to the application
- latency

We solve this problem using Apache Kafka, Meta-data, and a Sync Server.
THANKS

bgpstream.caida.org

alberto@caida.org