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

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 \textit{protocol engineering} decisions through realistic evaluations is difficult also because \textbf{we know little about the structure and dynamics of the BGP ecosystem!}

- \textbf{AS-level topology}
- \textbf{AS relationships}
- \textbf{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
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, ...

libBGPdump
https://bitbucket.org/ripencc/bgpdump
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, …
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
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
INSPIRING PROJECTS (1/2)

IODA: Detection and Analysis of Internet Outages

Outage of AS11351 (Time Warner Cable LLC)
September 30, 2015

---

# of Prefixes
# of Unique Source IPs

---

Center for Applied Internet Data Analysis
University of California San Diego

www.caida.org/funding/ioda/
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
Last Christmas we made it possible for anybody to follow the North Korean disconnection almost live.

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

Hijacks: detection of MITM BGP attacks

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

- normal path
- hijacked path
- normal path used to complete the attack

www.caida.org/funding/hijacks/
• 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
it’s real!

- bgpstream.caida.org
- download it! (version 1.0)
- active development - github.com/caida/bgpstream
- Docs & Tutorials
- paper under submission at NSDI ’16 (tech report on web site)
- people are using it!
- coordination with RouteViews, Colorado State BGPMon, RIPE NCC
- BGP Hackathon in February
BGPSSTREAM

different applications and development paradigms

Ease of use

Efficiency

BGPREADER
command line tool

PYBGPSTREAM
C Python bindings

BGPCORSARO
tool + plugins

LIBBGPSTREAM
C API

Center for Applied Internet Data Analysis
University of California San Diego
TERMINOLOGY

background and naming conventions

- Adj-RIB-Out etc. [RFC 4271]
- Collectors: RIB and Updates dumps
- VPs
- Partial vs Full-feed VPs
- ...
RouteViews and RIPE RIS collectors (~31) save:
- RIB dumps every 2 and 8 hours
- Updates dumps every 15 and 5 minutes
- a full-feed VP (in 2015)
  - has a an Adj-RIB-Out with ~550k routes
  - generates ~1.5K updates every 5 minutes
- RIB and Updates dumps are saved in the Multi-Threaded Routing Toolkit (MRT) binary format [RFC6396]
  - 10KB -100MB for RIB dumps (compressed)
  - 1KB -10MB for Updates dumps (compressed)
- RouteViews and RIPE RIS archives date back to 2001 and 1999 respectively
- The full archives of compressed files are about 8.9TB and 3.7TB, currently growing at the rate of 2TB per year
LIBBBGPSTREAM API

**BGP data stream**

- BGP data stream: `<collector projects (e.g., Route Views, RIPE RIS), list of collectors, dump types (RIB/Updates), time interval start and either time interval end or live mode)>.
  - A stream can include dumps of different type and from different collector projects.
  - A stream is made of *BGP records*, which can be decomposed in *BGP elems*
LIBBGPSTREAM PULL MODEL

based on the Broker

• the library implements a “client pull” model
  - efficient data retrieval without potential input buffer overflow (i.e., data is only retrieved when the user is ready to process it)
  - supports live mode
• iteratively alternates between:
  - meta-data queries to the Broker
  - and opening and processing the returned data

• historical mode: the stream ends when the Broker returns an empty set
• live mode: the query mechanism is blocking. If the Broker has no data available, a polling cycle will begin, periodically re-issuing the request to the Broker
C API

specifying a stream

```c
int main(int argc, const char **argv) {
    bgpstream_t *bs = bgpstream_create();
    bgpstream_record_t *record = bgpstream_record_create();
    bgpstream_elem_t *elem = NULL;
    char buffer[1024];

    /* Define the prefix to monitor for (2403:f600::/32) */
    bgpstream_pfx_storage_t my_pfx;
    my_pfx.address.version = BGPSTREAM_ADDR_VERSION_IPV6;
    inet_pton(BGPSTREAM_ADDR_VERSION_IPV6, "2403:f600::", &my_pfx.address.ipv6);
    my_pfx.mask_len = 32;

    /* Set metadata filters */
    bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "rrc00");
    bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_COLLECTOR, "route-views2");
    bgpstream_add_filter(bs, BGPSTREAM_FILTER_TYPE_RECORD_TYPE, "updates");

    /* Time interval: 01:20:10 - 06:32:15 on Tue, 12 Aug 2014 UTC */
    bgpstream_add_interval_filter(bs, 1407806410, 1407825135);

    /* Start the stream */
    bgpstream_start(bs);
```
• A BGP record encapsulates 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>

Center for Applied Internet Data Analysis
University of California San Diego
• 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)

### Field Type Function

<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);
            fprintf(stdout, "%s\n", buffer);
        }
    }
}
```

Listing 1

BGPStream prefix monitoring

5.1 libBGPStream API

5.2 Interface to Meta-Data and Data Providers

5.3 Data structures and error checking

PStream uses the latter to set the figure the stream (lines 15-19) and then iteratively request
the Broker returns an empty dump file set, the system
situates between making meta-data queries to the Broker
cess simply by setting the end of the time interval to
measurement data and a systematic organization of the
fines a BGP data stream by the following parameters:
BGP information into data structures. The API de-
measurement data and a systematic organization of the
fines a BGP data stream by the following parameters:
BGP information into data structures. The API de-
measurement data and a systematic organization of the
fines a BGP data stream by the following parameters:
RECORD-LEVEL SORTING

When

- When:
  - when reading dumps from more than one collector (inter-collector sorting)
  - when a stream is configured to include both RIB and Updates dumps (intra-collector sorting)
TOOLS/APIs

continued..
PYBGPSTREAM

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)
BGPCORSARO

Example: monitor your own address space on BGP

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

Hijacking of AS137 (GARR) - Jan 2015*

GLOBAL MONITORING

IODA, HIJACKS, etc.

• need to maintain a live global view (i.e., for each and every VP) of BGP reachability information updated with fine time granularity (e.g., few minutes)

• We implement 3 mechanisms:
  1. A solution to accurately reconstruct the observable LocRIB of each VP
  2. A synchronization mechanism
  3. Analysis modules to manipulate data from a BGP view
GLOBAL MONITORING

IODA, HIJACKS, etc.

![Diagram showing BGPView system components and interactions](image-url)

1. **BGPStream framework deployment**
   - For each collector, run an instance of BGPStream
   - Collect routing tables

2. **BGPView Server**
   - Pushes hash tables to BGPViewServer
   - Merges hash tables into partial BGP views
   - Performs sanity checking and correction

3. **BGPView Consumers**
   - Verifies occurrence of route leaks, spots new events
   - Tracks AS paths containing particular AS
   - Useful in many applicative scenarios

### Diagram Explanation
- **Time Series Monitoring**
- **BGPView publication**
- **Routing-tables plugin**
- **Shadow cell**
- **Main cell**
- **RIB dump records**
- **Prefix reachability-attributes**
- **Cell indexes**
- **Sliding window**
- **BGPStream record fields**

### System Components
- **BGPCorsaro**
- **BGPViewServer**
- **BGPViewConsumers**

### Key Concepts
- **Per-country outages detection**
- **Per-AS outages detection**

### Operations
- Every minute (Section 7.2)
- Running one BGPCorsaro instance per collector
- Multiple collectors: through synchronization buffer
- Pushes data to a system called BGPViewServer
- Merges into partial BGP views
- Publishes data on average one BGP view per minute

### Data Structures
- Multi-dimensional hash table
- Inserts and lookups with average time complexity of \(O(1)\)
- Reduces memory footprint

### Procedure
- Globally observable LocRIB
- Fully observable at time of injection
- Enables efficient and accurate routing table reconstruction

### Additional Observations
- IODA, HIJACKS, etc.
BGPVIEWSERVER

buffering partial/complete BGP views

• At the end of a 1-minute time bin, each BGPCorsaro instance pushes data (the reconstructed routing table) to the BGPViewServer.

• Such data is merged into a partial BGP view corresponding to its time bin.

• A BGP view is considered complete when all the BGPCorsaro instances have contributed to it.
BGPVIEWSERVER

sliding window

• we buffer partial BGP views in a **sliding window** based on their time bins

• the window slides each time data from a new bin arrives

• we publish a BGP view either
  - when all the BGPCorsaro instances have contributed to it (**complete view**)  
  - or when it expires, i.e., its time bin is no longer covered by the window (**partial view**)
• We **dimension the length of the sliding window** empirically (12 months observation of RV+RIS)
  • the *latency* at which data providers publish dumps
  • the *memory footprint*
    • when processing data from all Route Views and RIPE RIS collectors, a **30 minute** sliding-window buffer requires ≈60GB of memory and causes *99%* of BGP views to be published because they are complete rather than expired
The BGPViewServer is a potential bottleneck

- # collectors grows \(\rightarrow\) increase in the amount of data that the server must receive, process and publish every minute
- we architected the server to process each time bin independently of others
- multiple server instances can be run (e.g., on separate hosts), with BGPCorsaro processes distributing data amongst them in a round-robin fashion.

![Diagram of BGPViewServer and BGPCorsaro processes](image-url)
BGPVIEW CONSUMERS

demo on the browser
BGP HACKATHON - FEB 2016

theme: “live BGP measurements & monitoring”

Improve/Integrate tools to study the BGP eco-system. Target practical problems: topology, hijacks, outages, RPKI deployment, path inflation, circuitous paths, policies, relationships, visualize dynamics, …
We will provide a rich toolbox and “live” data access:

**RIPE Atlas**  **CAIDA Ark**  **Looking Glasses**

Data-plane active measurements

**Generation**  **Collection**  **Injection**  **Processing & Analysis**  **PEERING**  **CAIDA AS Rank**

**VIZ tools**

---

**REFERENCES**

1. Colorado State University. BGPmon.
5. Timely detection of suspicious BGP events enables detection systems to identify BGP hijack events.
6. Similarity we developed consumers that continuously process and publish every minute.
7. CAIDA AS Rank requires RIS collectors (31), a 30 minute sliding-window buffer.
8. Considering the trade-off of the latency at which data providers publish dumps of Internet service in the country for three hours.
9. According to the press the government ordered a complete shutdown at 5:00am UTC. Such outages have been reported by [5, 15, 18], at a lower frequency, for a period of about 3 hours, between 2:00am and 5:00am.
10. There is an observable series of outages that starts in [28], and shows the number of unique prefixes announced by 5 Iraqi ISPs. The noticeable drops, in terms of number of visible prefixes, identify a sequence of country-wide Internet service outages.
11. We analyze AS paths in the BGP views, looking for suspicious announcements (e.g., multiple unrelated ASes announcing overlapping portions of the address space), or creating a new edge in the AS graph).
12. We plan to make available, as Web services, global live BGPStream targets a broad range of applications and data providers, such as Route Views and BGPMon, as well as feedback and contributions to our platform. We also plan to make available, as Web services, global live updates of the BGP views to be published because they are complete complete rather than expired.
13. In Figure 10, we show the output of the per-country and per-AS outages consumers over a period of 1 month, selecting only the visibility of prefixes announced by each single AS. Each time a consumer selects the prefixes observed by full-feed VPs, per-country and per-AS outages (Figure 7). Both consistent with other researchers.
BGP HACKATHON
http://github.com/CAIDA/bgp-hackathon/wiki

• 6-7 February 2016 (weekend before NANOG 66)
• San Diego Supercomputer Center, UC San Diego
• Theme: live BGP measurements and monitoring
• Toolbox: BGPMon, RIPE RIS, PEERING, BGPStream, RIPE Atlas, CAIDA Archipelago, Route Views, looking glasses, AS relationships, AS Rank, Visualization tools, …

• How to contribute:
  • join us and come over to hack!
  • help teams as a domain expert
  • propose projects that hacking teams may pick
  • offer to join the jury that will assign awards

>>> bgp-hackathon-info@caida.org <<<
THANKS

bgpstream.caida.org
github.com/CAIDA/bgp-hackathon/wiki