When BGP meets Big-Data
The Internet is very much ‘alive’

- Millions of BGP events occurring every day
  - 15 Routers Monitored
  - 410 active peers (both IPv4 and IPv6)
  - ~120,000,000 Prefixes Advertised
  - ~950,000 events per day from a single transit peer
  - ~202,000,000 changes per day
  - ~6,000,000,000 changes per month

- How do we extract ‘signal’ from ‘noise’?
- Can we apply techniques from other domains in this pursuit?
The Internet is very much ‘alive’

- If we know the questions we want to ask, how do we ask them?
- Enhance traditional dampening and suppression with analytics
Five Monitoring Points in BGP

- **Adj-RIB-In (Pre-Policy)**
  - Peer-A MPBGP
  - Peer-B MPBGP

- **Adj-RIB-In (Post-Policy)**
  - Post rib

- **Adj-RIB-Out (Pre-Policy)**
  - Adj-RIB-Out (Pre-Policy)
  - Peer-C MPBGP

- **Adj-RIB-Out (Post-Policy)**
  - Selection/Use

- **Local-RIB**
  - rib

- **Static**
  - ISIS
  - OSPF
  - EPE

- Distribute/BGP-LS

© 2017 PNDA a Linux Foundation Collaborative Project. All rights reserved.
Five Monitoring Points in BGP

- Adj-RIB-In (Pre-Policy)
- Peer-A MPBGP
- Peer-B MPBGP
- Post rib
- Distribute/BGP-LS
- Static
- ISIS
- OSPF
- EPE

- Adj-RIB-In (Post-Policy)
- Post rib

- Selection/Use
- Adj-RIB-Out (Pre-Policy)
- Adj-RIB-Out (Post-Policy)
- Peer-C MPBGP

- Adj-RIB-Out (Pre-Policy)
- rib

- Adj-RIB-Out (Post-Policy)

- Local-RIB
- rib

References:
- RFC7854
- BMP
- draft-evens-grow-bmp-local-rib
- draft-evens-grow-bmp-adj-rib-out

© 2017 PNDa a Linux Foundation Collaborative Project. All rights reserved.
SNAS Architecture

Router

BMP Feed

collector

TCP Listener

Connection Thread

Ingress Buffer

Parser (BMP & BGP)

RAW (Native BGP)

Textual (JSON/CSV)

Produce

Kafka

database

MariaDB

DB Connection

SQL Transformation

Consumer

API

Web UI

© 2017 PNDA a Linux Foundation Collaborative Project. All rights reserved.
SNAS Architecture

**Collector**
- TCP Listener
- Connection Thread
- Ingress Buffer
- Parser (BMP & BGP)
  - RAW (Native BGP)
  - Textual (JSON/CSV)
- Produce

**Database**
- MariaDB
- DB Connection
- SQL Transformation
- Consumer

**Kafka**

**Router**
- BMP Feed

**App**

**API**

**Web UI**

© 2017 PDNA a Linux Foundation Collaborative Project. All rights reserved.
SNAS Architecture

**Collector**
- TCP Listener
- Connection Thread
- Ingress Buffer
- Parser (BMP & BGP)
  - RAW (Native BGP)
  - Textual (JSON/CSV)
  - Produce

**Database**
- MariaDB
- DB Connection
- SQL Transformation
- Consumer

**Router**
- BMP Feed

**Kafka**

**API**

**Web UI**

**App**

© 2017 PNDa a Linux Foundation Collaborative Project. All rights reserved.
E2E architecture

- Encoding app required to perform ‘avro’ encoding of BMP data
- BGP App runs as Spark batch job, running periodically
- Can be converted to a Spark ‘streaming’ application for near-real-time processing
What does this give us?

SNAS.io gives us the ability to record the dynamics of the Internet
PNDA platform enables -

- ‘Raw’ event recording capability, with horizontal scaling (HDFS)
- Run analysis over very large data-sets with parallelism
- Ask questions of the aggregate data about the Internet
  - Ask specific question
    - Per-prefix
    - Per-AS
    - Per AS-Path
Top-N analysis
Path stability
AS Connectivity - FLAG
AS Connectivity – Deutsche Telekom
Prefix to Path history

Path Len = 5

Path Len = 9
AS Path variance – 6939 to 8386

Shortest path – 3 hops
Longest path – 28 hops
Longest unique AS path – 5
Unique paths - 9
Largest prepend count – 17x
Prepend variation – [7-17]
Path with most updates – via AS1273

Data recorded in a 24hr period
AS Path variance – 6939 to 8386

Shortest path – 4 hops
Longest path – 29 hops
Longest unique AS path – 6
Unique paths - 9
Largest prepend count – 17
Prepend variation – [7-17]
Path with most updates – via AS1273

Data recorded in a 24hr period
Security – Short prefix / long prefix detection

Default Route and Long prefix injection detected
## Security – Unallocated prefixes

Download data: JSON, ground truth

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Origin AS</th>
<th>Peer AS</th>
<th>AS Path</th>
<th>Advertising Router.</th>
<th>Type</th>
<th>Timestamp</th>
<th>Last Seen</th>
<th>Still in use</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>103.243.31.0/24</td>
<td>133120</td>
<td>6939</td>
<td>6939 4823 0493 0720 133120</td>
<td></td>
<td>IPv4</td>
<td>2017-04-26 13:38:00</td>
<td>2017-04-25 05:14:54</td>
<td>true</td>
<td>unallocated</td>
</tr>
</tbody>
</table>

TOTAL ITEMS: 957

---

*Observed over a 12 hour period*
### Security – Prefix drill-down

#### Download data: JSON ground truth

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Origin AS</th>
<th>Peer AS</th>
<th>AS Path</th>
<th>Advertising Router</th>
<th>Type</th>
<th>Timestamp</th>
<th>Last Seen</th>
<th>Still</th>
<th>Category</th>
</tr>
</thead>
</table>

#### AS PATHS

- 2017-04-25 15:47:52.336696000
- 103.212.178.0/24
- BMP
- eBGP
- 6939
- 15412
- FLAG-AS
- AS
- 18101
- RELIANCE-COMM...
- AS
- 55410
- VODAFONE-NET-A..
- AS
- 56124
- GETNET-IN
## Security – drill-down

Download data: JSON  ground truth

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Origin AS</th>
<th>Peer AS</th>
<th>AS Path</th>
<th>Advertising Router.</th>
<th>Type</th>
<th>Timestamp</th>
<th>Last Seen</th>
<th>Still</th>
<th>Category</th>
</tr>
</thead>
</table>

### AS PATHS

- **2017-04-25 15:48:15.963081000**
  - 138.59.180.0/23
  - **ROSTELECOM-AS** 48066
  - **RIPE-ASNBLOCK-...** 58271
  - **58934**
More specific prefix detection

- AS 12345 originates 100.100.0.0/18
- Hijacker originates 100.100.63.0/24
- Basically a needle in a large haystack, does anyone notice?
- What does RPKI show?
- Do the origin ASNs match?
- Does the less specific share the same transit set or similar as_paths?
- Does RIR have the same organization name or contacts for both origins?
- Anything out of the norm for the new originating ASN?
Potential

What can we do with large-scale collection of historical event information?

- Event impact analysis –
  - Stability
  - Security
  - Misconfiguration
  - Forensics
- Application of ML/DL to data-set
- Pattern-detection and network ‘weather forecasting’
PNDA.io – the platform
What is PNDA?

PNDA brings together a number of open source technologies to provide a simple, scalable open big data analytics Platform for Network Data Analytics

Linux Foundation Collaborative Project based on the Apache ecosystem
Where is PNDA today?

- Linux Foundation project
- Selected by MEF for Analytics function within Lifecycle Service Orchestration framework
- In service trials with two Service Providers
- One platform supporting a range of use-cases including
  - Network security – Apache Spot
  - 6CN
  - Virtualization infrastructure monitoring and analysis
  - Smart Cities
  - Smart Transportation use-cases
- Horizontally scalable platform for analytics and data processing applications
- Support for near-real-time stream processing and in-depth batch analysis on massive datasets
- Decouples data collection and aggregation from data analysis
- Consuming applications can be either platform apps developed for PNDA or client apps integrated with PNDA
- Client apps can use one of several structured query interfaces or consume streams directly.
- Leverages best current practice in big data analytics
- Simple, scalable open data platform
- Provides a common set of services for developing analytics applications
- Accelerates the process of developing big data analytics applications whilst significantly reducing the TCO
- PNDA provides a platform for convergence of network data analytics
Why PNDA?

Innovation in the big data space is extremely rapid, but combining multiple technologies into an end-to-end solution can be extremely complex and time-consuming.

PNDA removes this complexity and allows you to focus on developing the analytics applications, not on developing the pipeline — significantly reducing the effort required and time-to-value.
Where can I learn more?

- www.pnda.io
- https://github.com/pndaproject
- www.snas.io