Streaming Network Analytics System

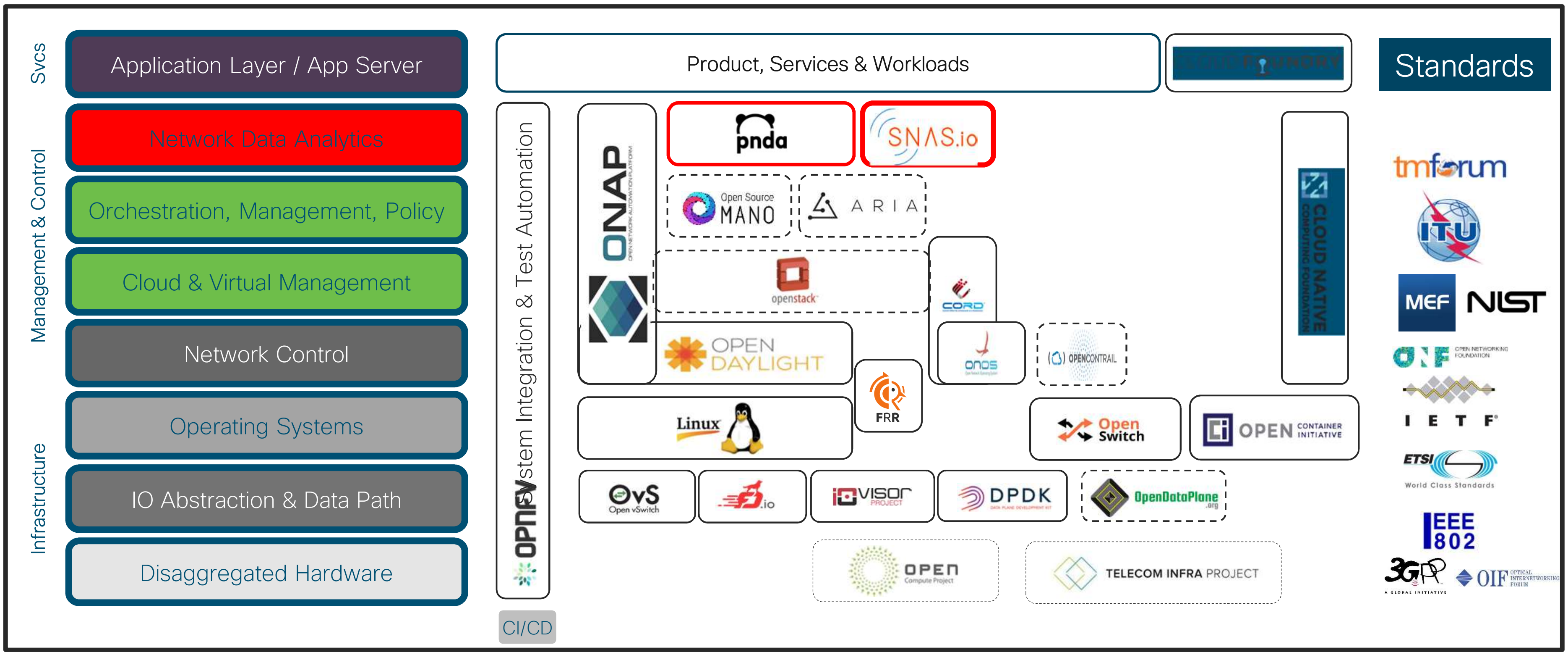
Cisco Knowledge Network Presentation

Serpil Bayraktar
Distinguished Engineer
2/28/2018

Streaming Network Analytics System
SNAS

- An open source project under Linux Foundation Networking Umbrella
- A framework to collect, track and access tens of millions of routing objects (routers, peers, prefixes) in real time
- Allows you to interact, visualize, and analyze routing data in a simple way

Linux Foundation Open Source Networking Stack



Automation of Network + Infrastructure + Cloud + Apps + IOT

Linux Foundation Hosted

Outside Linux Foundation

Background

Network Analytics Data Types

Types of data used for analytics:

- Device stats/logs/error counters/queue statistics etc.
  - SNMP/Pull Model
  - Telemetry/Push Model
- IP traffic information and statistics
  - Netflow/sFlow/IPFIX
- Routing/control/topology data*
  - IGP/Internal network topology
  - BGP/Variety of reachability information (services)

*SNAS

Device vs Network View

Device View:

- Data is collected from each device **after** it is processed by the device

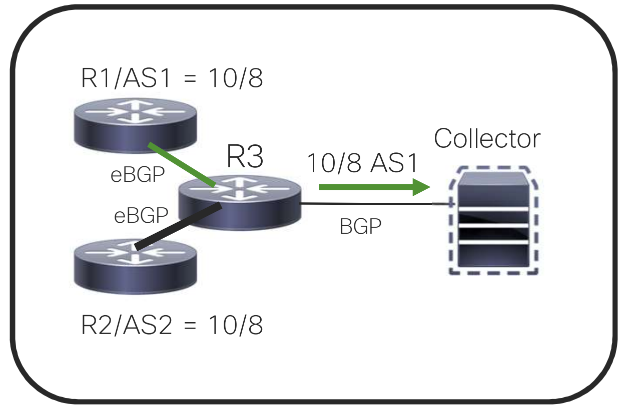
Network View*:

- Data is collected from each device **before** it is processed (as received from the network)

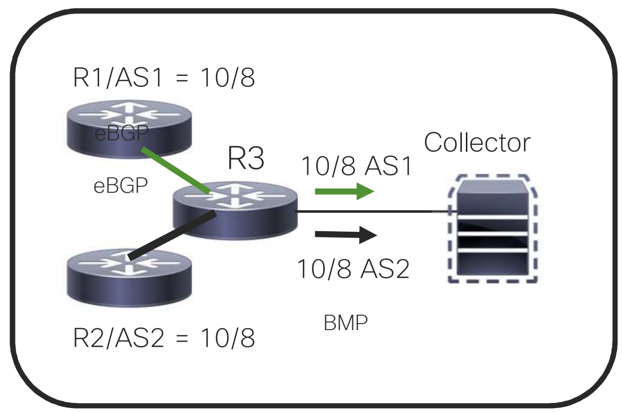
*SNAS

Example:
Device vs Network View

Device View



Network View*



*SNAS

Stateful Nature of Routing Protocols

- Stateful = Processing data based on the state information of the client
- Routing protocols are session based
- Routers maintain the state and parameters of each session
- An initial exchange of full routing table is followed by individual updates as needed
- Routing data can be viewed as a "stream" between two devices which agreed on some session parameters and only during the life of the session*
- This is very different than "stateless" or snapshot data where data is not associated with session information.

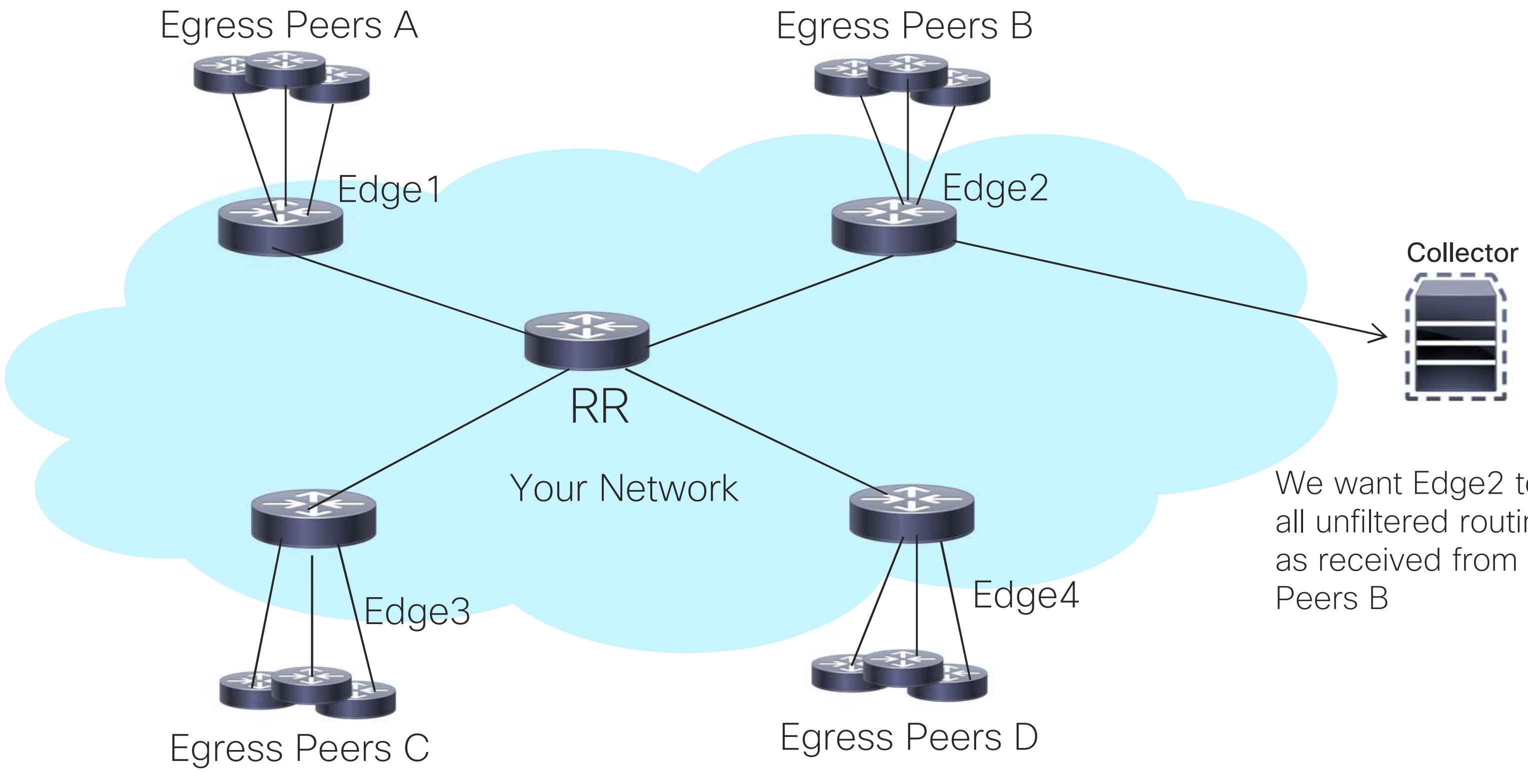
*SNAS

SNAS

Project Objectives

- Collect, store, maintain, track and expose network centric routing data for analytics applications
- Make routing data application developer friendly
- Produce lightweight packages that can run on a small server

Collect Network Wide Routing Data Securely and Efficiently



We want Edge2 to stream all unfiltered routing packets as received from Egress Peers B

Data Collection

BGP

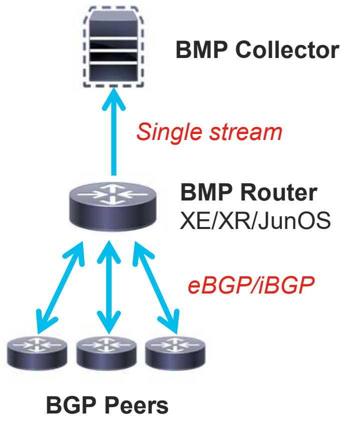
to

BMP

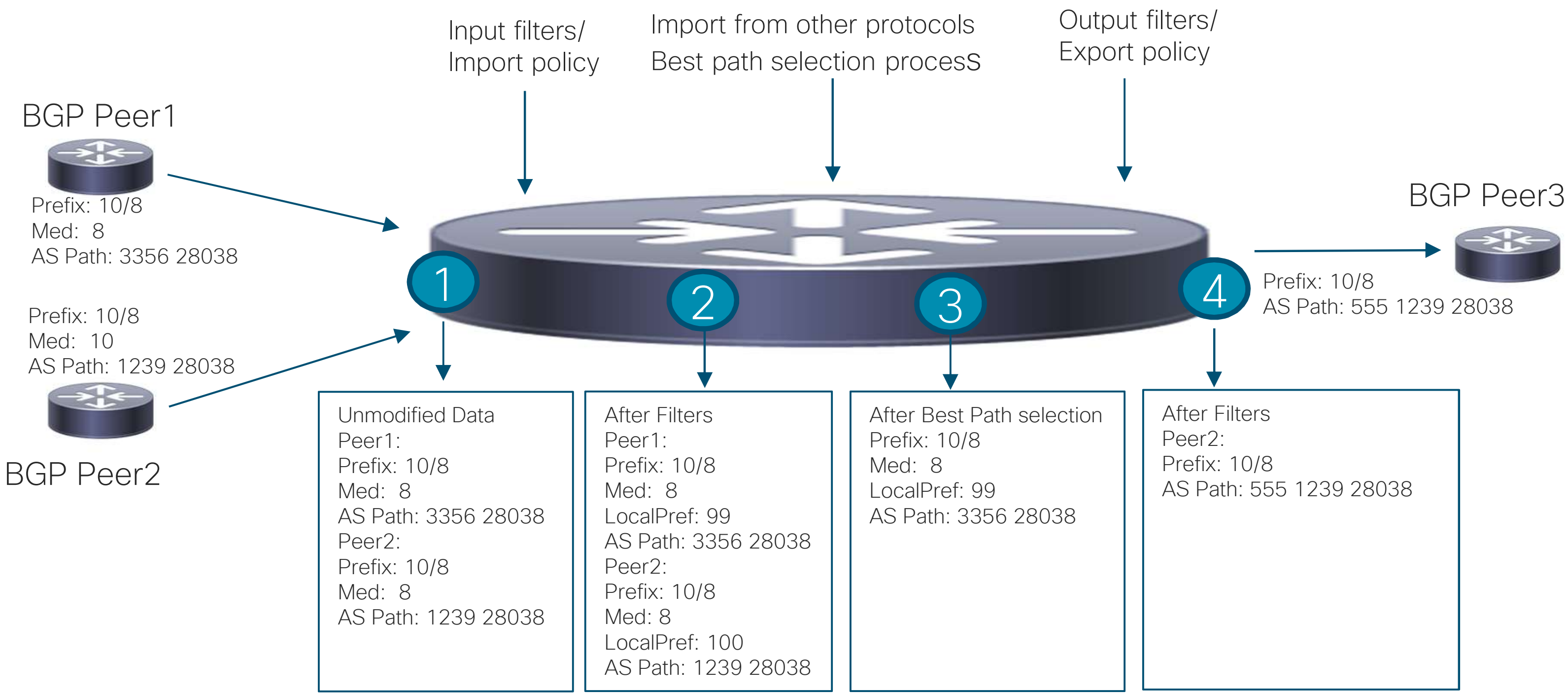
"BGP Monitoring Protocol"

What is BMP

- BGP Monitoring Protocol (BMP) encapsulates BGP messages from one or more BGP peers into a single TCP stream to one or more collectors
- Efficient, *[near]* real-time, low memory/CPU on router, little to no service impact with peering
- Simplified configuration (one-time setup) with granular controls per peer
- All address families supported
- https://tools.ietf.org/html/rfc7854



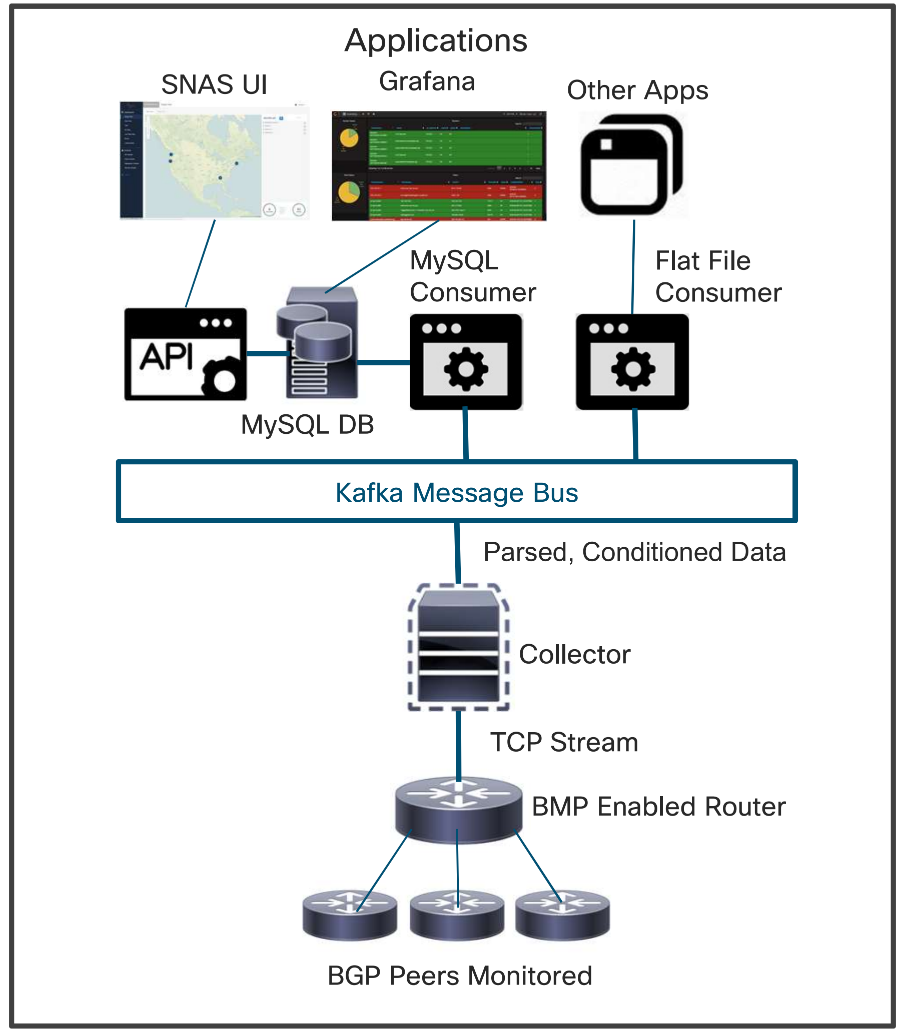
Access to Multiple Monitoring Points via BMP



BMP Availability

BMP Availability		
Vendor	Release	Notes
IOS-XE	3.12 and above	ASR1K, CSR1000v
IOS-XR	5.2.2	ASR9K, CRS, NCS6K, XRv
NX-OS	Evergreen	N9K, N7K
JunOS	Since 10.3	MX, EX ACX (12.3)
goBGP	1.3+	
Arista/Huawei	Coming Soon	

SNAS Architecture



Collector

- Parses BGP data
- Conditions the data
- Produces the parsed (and raw) BGP data to Kafka using Kafka's topic structure
- Data produced in tab delimited format
- Highly scalable, very small footprint
- IPv4, IPv6, Labeled IPv4/v6, BGP-LS (Segment Routing), L3VPN address families supported

A Word About Parsing

- Parsing BGP data requires deep knowledge of network packet format based on standards (RFCs)
- Anyone who wants to do analytics using BGP data has to figure out how to parse it
- SNAS does the parsing for you

Example Parsed Data (REST API)


```
RouterName: "cirl-sjc16-a9k1",
RouterIP: "192.133.147.161",
LocalIP: "64.71.176.51",
LocalPort: 179,
LocalASN: 32764,
LocalBGPId: "192.133.147.129",
PeerName: "v416.core1.sjc1.he.net",
PeerIP: "64.71.176.49",
PeerPort: 8018,
PeerASN: 6939,
PeerBGPId: "216.218.252.163",
LocalHoldTime: 180,
PeerHoldTime: 180,
isUp: 1,
isBMPConnected: 1,
isPeerIPv4: 1,
isPeerVPN: 0,
isPrePolicy: 1,
LastModified: "2018-02-26 05:09:17.460999",
LastBMPReasonCode: 0,
LastDownCode: 0,
LastdownSubCode: 0,
LastDownMessage: "",
LastDownTimestamp: "2018-02-26 05:09:17.460999",
SentCapabilities: "MPBGP (1) : afi=1 safi=1 : Unicast IPv4, Route Refresh Old (128), Route Refresh (2), 4 Octet ASN (65), Graceful Restart (64)",
RecvCapabilities: "MPBGP (1) : afi=1 safi=1 : Unicast IPv4, Route Refresh (2), Route Refresh Old (128), 4 Octet ASN (65)",
as_name: "HURRICANE",
isLocRib: 0,
isLocRibFiltered: 0,
table_name: "",
peer_hash_id: "201d59fcd894ab5dcd2560199e24342e",
router_hash_id: "46b469035bf7fa7d600167e832b6dcb0",
geo_ip_start: "@G�0",
```

Example Parsed Data (Kafka API)

Tab-Delimited


```
V: 1.5
C_HASH_ID: ff9618a2250eea2e6a9bee265f3340f6
L: 1030
R: 4

add	21	841958a3d10f82682d08c49e053ac26e
d235feb2ec2475ada7d4c6f86c77aeb4	200.1.1.5
39a1b23609327e8d62a6a00f5eaf9edf	19d1791539592a9f4407f6509a6b2440
200.1.1.2	100	2017-05-09 23:19:38.679570	10.0.0.2	32	1
incomplete	300	1	300	20.2.9.9	0	100
0	1	0	1	1
add	22	e59aa66d10d04b8dfbef9a245206ab9c
d235feb2ec2475ada7d4c6f86c77aeb4	200.1.1.5
39a1b23609327e8d62a6a00f5eaf9edf	19d1791539592a9f4407f6509a6b2440
200.1.1.2	100	2017-05-09 23:19:38.679570	8.0.108.0	24	1
incomplete	300	1	300	20.2.9.9	0	100
0	1	0	1	1
add	23	519aa05c3f8f0e02e758802c4a00ffbe
d235feb2ec2475ada7d4c6f86c77aeb4	200.1.1.5
39a1b23609327e8d62a6a00f5eaf9edf	19d1791539592a9f4407f6509a6b2440
200.1.1.2	100	2017-05-09 23:19:38.679570	200.1.1.9	32	1
incomplete	300	1	300	20.2.9.9	0	100
0	1	0	1	1
add	24	6c0a2b38b46166bc73ab0e1be9723791
d235feb2ec2475ada7d4c6f86c77aeb4	200.1.1.5
39a1b23609327e8d62a6a00f5eaf9edf	19d1791539592a9f4407f6509a6b2440
200.1.1.2	100	2017-05-09 23:19:38.679570	20.7.9.0	24	1
incomplete	300	1	300	20.2.9.9	0	100
0	1	0	1	1
```


Converted to Json


```
Received Message (2017-05-09 16:32:38.460859) : UNICAST_PREFIX(V: 1.5)
[
    {
        "origin": "incomplete",
        "seq": 0,
        "nexthop": "20.2.9.9",
        "as_path": " 300",
        "prefix": "200.1.1.9",
        "originator_id": "200.1.1.2",
        "isNexthopIPv4": 1,
        "prefix_len": 32,
        "isIPv4": 1,
        "origin_as": 300,
        "aggregator": "",
        "peer_ip": "200.1.1.6",
        "ext_community_list": "",
        "cluster_list": "200.1.1.6 ",
        "peer_asn": 100,
        "med": 0,
        "isPrePolicy": 1,
        "labels": "",
        "hash": "dc28679ebcdc9160c2f962dca70162f1",
        "timestamp": 1494397956000,
        "as_path_count": 1,
        "community_list": "",
        "router_ip": "200.1.1.5",
        "router_hash": "77d322c91f03a2ac40b06e30950dc418",
        "isAtomicAgg": 0,
        "base_attr_hash": "01c118be1f108b0f73eb1f736354d831",
        "isAdjRibIn": 1,
        "path_id": 0,
        "peer_hash": "22612464101869f13566a8a7a8bf17ea",
        "action": "add",
        "local_pref": 100
    },
```

Tracking Stateful Routing Data at Internet Speeds and Scales

- Maintaining state for a very large number of peers and prefixes:
  - 100s – 1000s of peers
  - A typical Internet peer carrying 700K+ IPv4 prefixes
  - Multiple monitoring points
- Processing very high number of packets with minimum delay:
  - Updates/sec during initialization (routing table dump) generates tens of thousands of updates/sec

Getting Started

Start Here

- SNAS Webpage: https://www.snas.io
- SNAS Repositories:
  - https://github.com/OpenBMP
  - https://github.com/SNAS

Installation

http://www.snas.io/gettingstarted/getting_started/

Step 1: Install AIO Container

Step 2: Install UI Container

Step 3: Start Feeding Data

Configure a router to send BMP data

Use public BGP data, Install MRT2BMP Application or

All-In-One (aio) Container

- Openbmpd - Latest collector (listening port is TCP 5000)
- MariaDB 10.2 – MySQL server (listening port TCP 3306)
- Apache Kafka 0.10.1 - High performing message bus (listening ports are TCP 2181 and 9092)
- Tomcat/DB_REST - Latest Rest interface into MySQL/MariaDB (listening port TCP 8001)
- SNAS MySQL Consumer - Latest Consumer that puts all data into MySQL

UI Container

- nginx – Web Server
- ui – SNAS UI

Router Configuration

http://www.snas.io/docs/

Configuration:

XR, XE, JUNOS

XR Configuration

```
router bgp <nnnn>
 !
 neighbor <d.d.d.d>
  bmp-activate server 1
  ...
  !
 !
!
bmp server 1
 host 10.20.254.245 port 5000
 description BMP Server - primary
 update-source GigabitEthernet0/0/0/0
 initial-delay 60
 initial-refresh delay 60 spread {number of peers * 2}
 stats-reporting-period 300
!
```

Demo Servers

http://www.snas.io/demo/

- SNAS UI
  - http://demo-rv.snas.io:8000/
- Grafana
  - http://demo-rv.snas.io:3000/