TOWARDS A PROGRAMMABLE TRAFFIC ENGINEERING ENGINE

GRNOG 9

Kostas Zorbadelos

kzorba AT nixly DOT net

OUTLINE

- Problem description
- BGP role
- Solution components
- IP/MPLS Reference Network simulation
- BGP policy design
- Configuration management
- LAB Demo

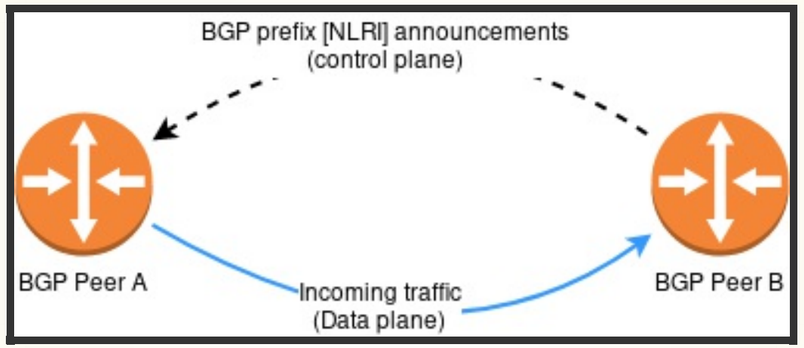
PROBLEM DESCRIPTION

- Targetted mainly in ISP networks
- Downstream traffic is dominant
- IP Network with multiple points of presense / geographically dispersed
- Providing customer transit services
- Having multiple Internet transit providers and peers (in IXes or PNIs)
- Varying costs in transit capacity, submarine capacity can also be involved

- Need to optimize incoming traffic streams and distribute them among available capacity
- Do it reliably, without errors
- Do it quickly, even real time, depending on current traffic conditions
- A link failure (especially in submarine capacity) would need proper action to bypass the failure
- Economics are also involved in transit services

BGP ROLE

- BGP is ***the*** exterior routing protocol between ASes



- BGP is used extensively for traffic engineering (why do you think the FIRT is so big?)

BGP TRAFFIC ENGINEERING TRICKS

- Lots of tricks for traffic engineering
- Communities in announcements that affect multiple aspects upstream (LOCAL_PREFERENCE, further announcements to upstream peers, even blackholing)
- ASPATH prepending
- More specifics (<– quite effective)
- MED (not much used in customer - transit)
- Many times trial and error since you actually have no control outside of your AS

SOLUTION COMPONENTS

- Manual configuration on routers is cumbersome
- Inconsistent configuration, error-prone
- Routers involved could be many, fast reaction not possible
- Configuration could be performed by network operators **or** even a program without human involvement
- Ideally vendor neutral (multiple vendor equipment in many networks)

- Standardized BGP policy configuration generated by automation tools highly desirable
- Tagging of prefixes (BGP communities) affects policy
- Centralized configuration point
- Configuration management tool (salt-sproxy + NAPALM) to enforce state to all edge routers
- Only need to think (or generate) the proper tags in the routes to get the desired outcome
- Design flexibility, all traffic engineering tricks should be supported

IP/MPLS REFERENCE NETWORK SIMULATION

- A lab environment was necessary for the development
- Close emulation of the production IP network
- Multiple peerings, transit providers / peers / customers
- Lab setup a challenging task with limited resources
- The majority of the development time was spent on the network emulation environment
- The rest was policy design and automation

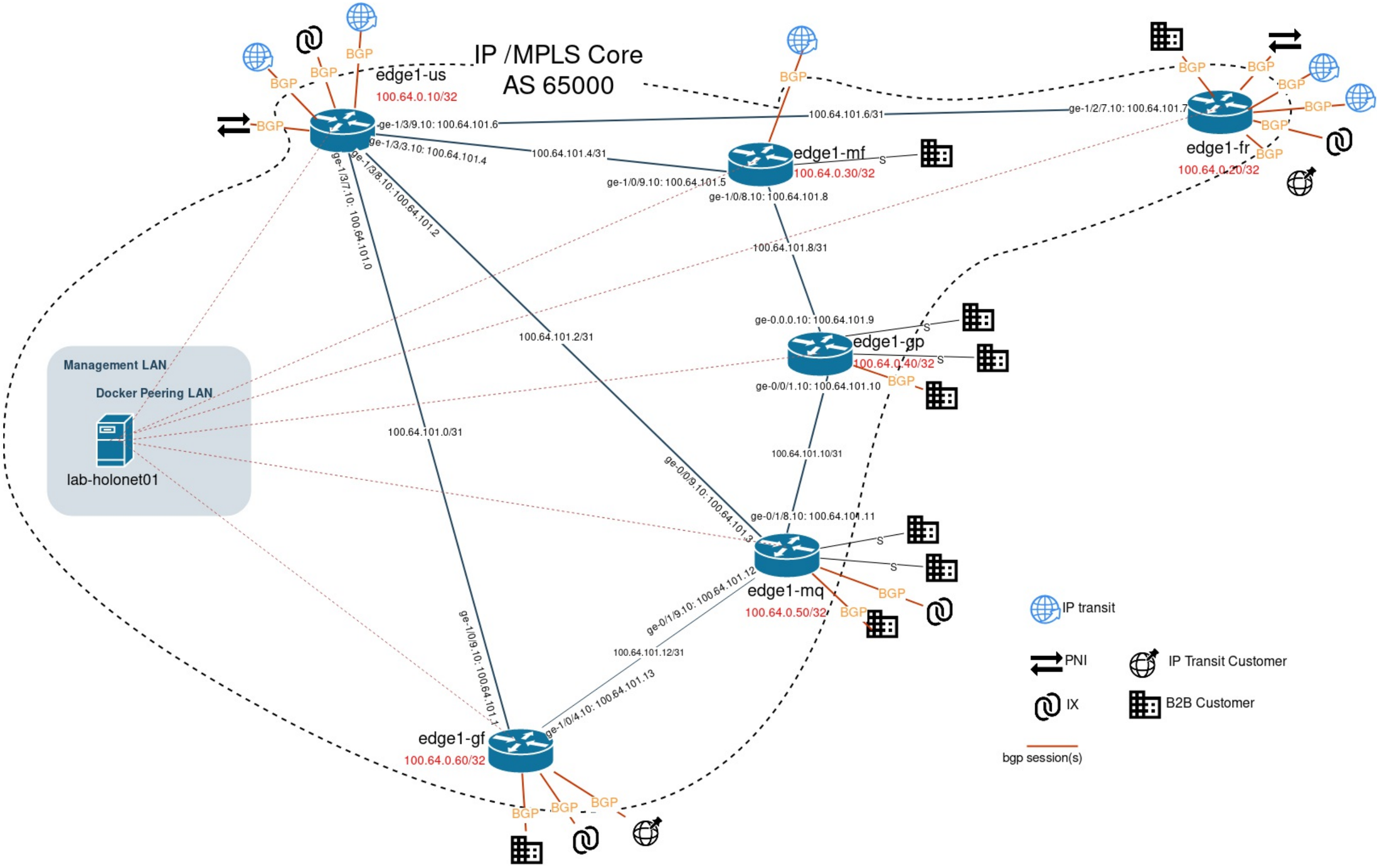
LAB RESOURCES

- Four (4) physical Juniper routers (MX 240, MX 104, 2 x MX 5)
- One (1) old server

```
IBM System x3650 M3 (withdrawn no longer available for ordering)
Intel(R) Xeon(R) CPU E5640, 4 cores 8 threads
16GB RAM
2x1TB SAS disks
```

- Emulation of full IP/MPLS core with 6 routers, 3 transit providers, 4 IXes, PNIs, transit customers, total of 21 emulated peers plus management host with Salt installation

- Juniper ***logical systems*** and ***docker*** to the rescue!
- Each lab router implemented as a logical system on the physical Juniper machines
- BGP speakers are docker containers with GoBGP (tried also ExaBGP...)
- Configuration emulates closely production network (no P routers, only PE having all the customers and peerings)
- Complete IPv4/IPv6 addressing, AS numbers and communities specification


IP /MPLS Core
AS 65000
edge1-us
100.64.0.10/32
edge1-mf
100.64.0.30/32
edge1-fr
100.64.0.20/32
edge1-gp
100.64.0.40/32
edge1-mq
100.64.0.50/32
edge1-gf
100.64.0.60/32
ge-1/3/9.10: 100.64.101.6
100.64.101.6/31
ge-1/2/7.10: 100.64.101.7
ge-1/3/3.10: 100.64.101.4
100.64.101.4/31
ge-1/0/9.10: 100.64.101.5
ge-1/0/8.10: 100.64.101.8
ge-1/3/7.10: 100.64.101.0
ge-1/3/8.10: 100.64.101.2
100.64.101.8/31
ge-0.0.0.10: 100.64.101.9
100.64.101.2/31
ge-0/0/1.10: 100.64.101.10
100.64.101.0/31
100.64.101.10/31
ge-0/0/9.10: 100.64.101.3
ge-0/1/8.10: 100.64.101.11
ge-0/1/9.10: 100.64.101.12
ge-1/0/9.10: 100.64.101.1
100.64.101.12/31
ge-1/0/4.10: 100.64.101.13
Management LAN
Docker Peering LAN
lab-holonet01
BGP
S
IP transit
PNI
IP Transit Customer
IX
B2B Customer
bgp session(s)

- Main routing table contains only p2p links and loopbacks (aka IGP)
- Internet service in a dedicated VRF
- IPv4 and IPv6 service over MPLS transport (IPv6 using 6vPE)
- Full mesh IBGP peerings
- Standardized BGP OUT-bound policies in EBGP peerings (for announcements)
- Proper tagging of all prefixes

BGP POLICY DESIGN

BGP COMMUNITIES

- A BGP community is a group of destinations that share a common property
- In the beginning we had standard communities [rfc1997], 32-bit/4 octet values (as-number:community-value)
- Networking enhancements, such as VPNs brought extended communities [rfc4360], 64-bit/8-octet values (type:administrator:assigned-number)

BGP LARGE COMMUNITIES

- Both regular and extended communities cannot encode two 32-bit ASN values
- BGP large communities [rfc8092] a recent development, 12 octets, three 4-byte integers
  (example 21351:602:6799)
  to overcome policy design limitations with 32-bit ASNs
- [rfc8195] Use of BGP Large Communities informational RFC giving excellent policy examples
- An IETF “blessed” way to create policies!

LOCATION COMMUNITIES (INFORMATIONAL)

Community value	Description
<ASN>:1:<LOC_CODE>	Route of ASN belongs / originates / was imported in location with code LOC_CODE

```
/* location communities */
/* code is ISO 3166 country code */
L_LOCATION_FR members large:65000:1:250
L_LOCATION_GF members large:65000:1:254
L_LOCATION_GP members large:65000:1:312
L_LOCATION_MF members large:65000:1:663
L_LOCATION_MQ members large:65000:1:474
L_LOCATION_US members large:65000:1:840
```

DO NOT ANNOUNCE TO PEER AS (ACTION)

Community value	Description
<ASN>:40:<PEER_ASN>	Do not announce route of <ASN> to <PEER_ASN>

```
L_NO_ANNOUNCE_AS65100 members large:65000:40:65100
L_NO_ANNOUNCE_AS65101 members large:65000:40:65101
L_NO_ANNOUNCE_AS65102 members large:65000:40:65102
L_NO_ANNOUNCE_AS65103 members large:65000:40:65103
L_NO_ANNOUNCE_AS65200 members large:65000:40:65200
L_NO_ANNOUNCE_AS65201 members large:65000:40:65201
L_NO_ANNOUNCE_AS65202 members large:65000:40:65202
L_NO_ANNOUNCE_AS65203 members large:65000:40:65203
L_NO_ANNOUNCE_AS65300 members large:65000:40:65300
L_NO_ANNOUNCE_AS65301 members large:65000:40:65301
L_NO_ANNOUNCE_AS65302 members large:65000:40:65302
L_NO_ANNOUNCE_AS65303 members large:65000:40:65303
```

ANNOUNCE TO PEER AS (ACTION)

Community value	Description
<ASN>:41:<PEER_ASN>	Announce route of <ASN> to <PEER_ASN>

```
/* (to be used in combination with do not announce to location) */
L_ANNOUNCE_AS65100 members large:65000:41:65100
L_ANNOUNCE_AS65101 members large:65000:41:65101
L_ANNOUNCE_AS65102 members large:65000:41:65102
L_ANNOUNCE_AS65103 members large:65000:41:65103
L_ANNOUNCE_AS65200 members large:65000:41:65200
L_ANNOUNCE_AS65201 members large:65000:41:65201
L_ANNOUNCE_AS65202 members large:65000:41:65202
L_ANNOUNCE_AS65203 members large:65000:41:65203
L_ANNOUNCE_AS65300 members large:65000:41:65300
L_ANNOUNCE_AS65301 members large:65000:41:65301
L_ANNOUNCE_AS65302 members large:65000:41:65302
L_ANNOUNCE_AS65303 members large:65000:41:65303
```

PREPEND TO PEER AS (ACTION)

Community value	Description
<ASN>:61:<PEER_ASN>	Prepend <ASN> once to <PEER_ASN>
<ASN>:62:<PEER_ASN>	Prepend <ASN> twice to <PEER_ASN>
<ASN>:63:<PEER_ASN>	Prepend <ASN> three times to <PEER_ASN>

```
/* Examples */
L_PREPENDx1_AS65100 members large:65000:61:65100
L_PREPENDx2_AS65100 members large:65000:62:65100
L_PREPENDx3_AS65100 members large:65000:63:65100
```

DO NOT ANNOUNCE IN LOCATION (ACTION)

Community value	Description
<ASN>:400:<LOC_CODE>	Do not announce route of <ASN> to location with code <LOC_CODE>

```
/* <LOC_CODE> is ISO 3166 country code */
L_NO_ANNOUNCE_FR members large:65000:400:250
L_NO_ANNOUNCE_GF members large:65000:400:254
L_NO_ANNOUNCE_GP members large:65000:400:312
L_NO_ANNOUNCE_MF members large:65000:400:663
L_NO_ANNOUNCE_MQ members large:65000:400:474
L_NO_ANNOUNCE_US members large:65000:400:840
```

PREPEND IN LOCATION (ACTION)

Community value	Description
<ASN>:601:<LOC_CODE>	Prepend <ASN> once to all peers in location <LOC_CODE>
<ASN>:602:<LOC_CODE>	twice...
<ASN>:603:<LOC_CODE>	three times

ROUTE TYPES (INFORMATIONAL)

Community value	Description
<ASN>:3:<TYPE_CODE>	Route of ASN is of specified type

```
/* Our address space */
L_ROUTE_INTERNAL_DEFAULT members large:65000:3:10
L_ROUTE_INTERNAL_P2P members large:65000:3:101
L_ROUTE_INTERNAL_LOOPBACK members large:65000:3:110
L_ROUTE_INTERNAL_AGGREGATE members large:65000:3:111
L_ROUTE_INTERNAL_B2B_CUSTOMERS members large:65000:3:112
L_ROUTE_INTERNAL_RES_CUSTOMERS members large:65000:3:122
L_ROUTE_INTERNAL_PPP members large:65000:3:132
L_ROUTE_ANNOUNCEMENT members large:65000:3:199
```

ROUTE TYPES (CONTINUED)

```
/* Customer routes */
L_ROUTE_CUSTOMER members large:65000:3:200
/* Peering partner routes */
L_ROUTE_PEERING members large:65000:3:300
L_ROUTE_PEERING_LAN members large:65000:3:301
L_ROUTE_PEERING_PNI members large:65000:3:302
/* Transit routes */
L_ROUTE_TRANSIT members large:65000:3:400
L_ROUTE_TRANSIT_P2P members large:65000:3:401
```

STANDARDIZED EBGP OUT POLICY

```
Policy name: <ASN>_<PEER>-V[46]-OUT

clean RouteTargets
/* Part 1: handle specific peer announcements
   announce or not */
if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE)
      AND (DO_NOT_ANNOUNCE_PEER) {
  reject
}
if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE)
      AND (ANNOUNCE_PEER) {
  accept
}
...
```

```
...

/* Part 2: handle specific peer announcements */
/* prepend N times */

if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE)
      AND (ROUTE_PREPEND_x_N_PEER) {
  as-path-prepend_x_N <ASN>
}

...
```

```
...

/* Part 3: Location processing */

if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE)
      AND (DO_NOT_ANNOUNCE_LOCATION) {
  reject
}

if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE)
      AND (PREPEND_x_N_LOCATION) {
  as-path-prepend_x_N <ASN>
}

...
```

```
...

/* Final: default actions */

if (INTERNAL_ANNOUNCEMENT_ROUTE OR CUSTOMER_ROUTE) {
  accept
}

DEFAULT {
  reject
}
```

CONFIGURATION MANAGEMENT

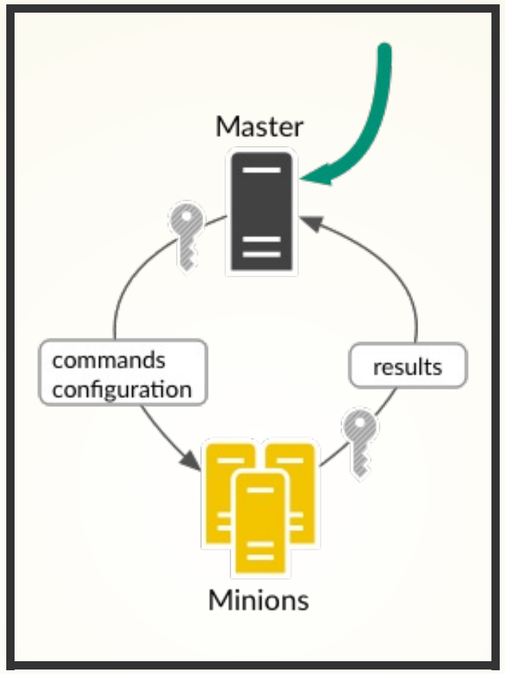
TOOLS SELECTION

- A lot of open source configuration management frameworks
- Salt (sometimes referred to as SaltStack) is an open-source software for event-driven IT automation, remote task execution, and configuration management
- NAPALM is a vendor neutral, cross-platform open source project that provides a unified API to network devices
- All Python based

WHY SALTSTACK/SALT?

- Salt is a very flexible framework
- Ansible ++
- Integrated modules for network device configuration (NAPALM one of them)
- Scalable and proven
- Event driven action capabilities
- A bit of a steep learning curve

SALT-ARCHITECTURE

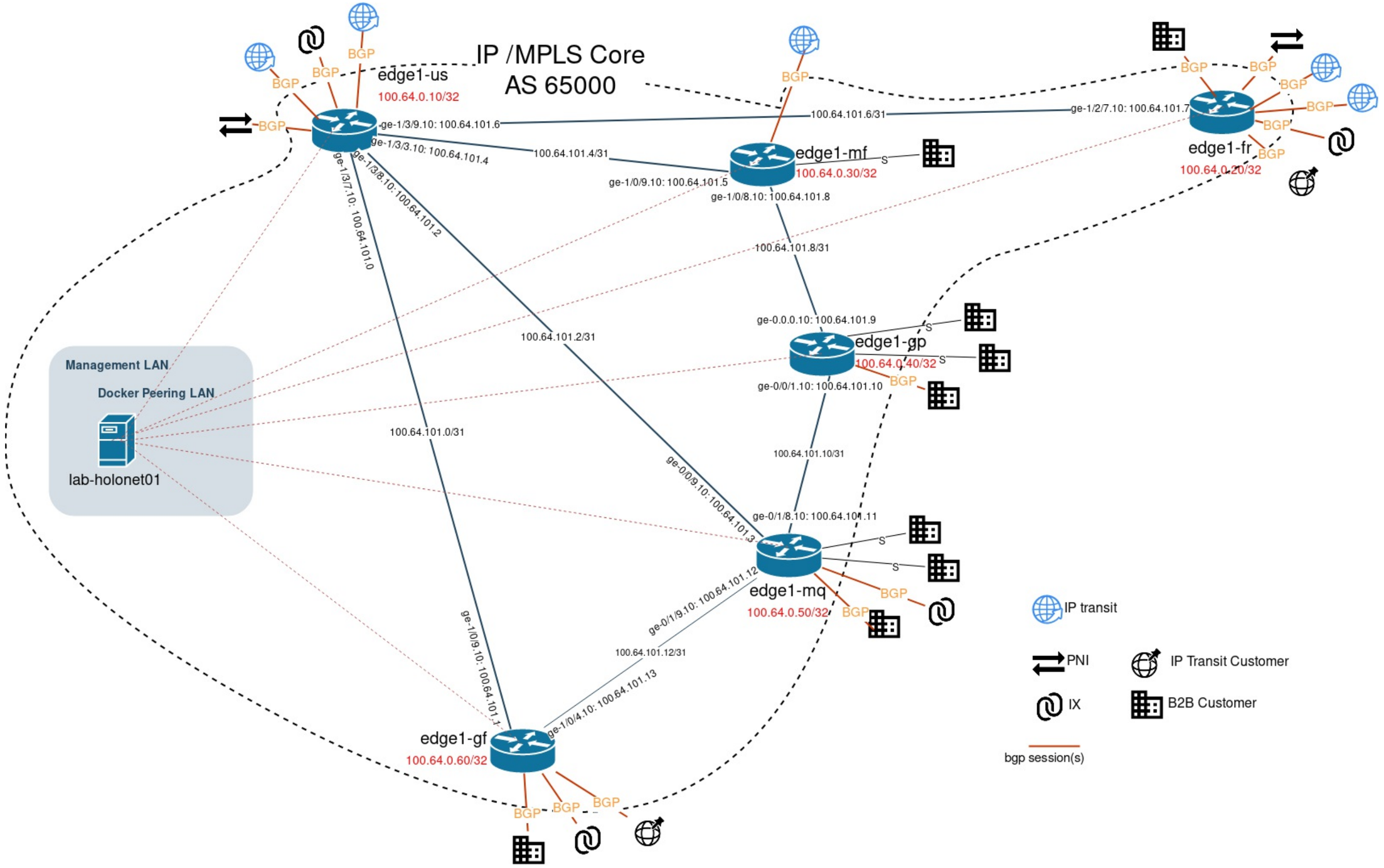


- Agents run on managed devices (minions)
- In network equipment proxy minions are used
- One process per device needed

SALT-SPROXY

- New open source project
- Not embraced by Saltstack (yet)
- Gives all Salt framework and capabilities, without the need to run and handle minion processes
- (proxy) minions are started on-demand
- salt master necessary to use API and event bus

LAB DEMO


IP /MPLS Core
AS 65000
edge1-us
100.64.0.10/32
edge1-mf
100.64.0.30/32
edge1-fr
100.64.0.20/32
edge1-gp
100.64.0.40/32
edge1-mq
100.64.0.50/32
edge1-gf
100.64.0.60/32
ge-1/3/9.10: 100.64.101.6
100.64.101.6/31
ge-1/2/7.10: 100.64.101.7
ge-1/3/3.10: 100.64.101.4
100.64.101.4/31
ge-1/0/9.10: 100.64.101.5
ge-1/0/8.10: 100.64.101.8
ge-1/3/7.10: 100.64.101.0
ge-1/3/8.10: 100.64.101.2
100.64.101.8/31
ge-0.0.0.10: 100.64.101.9
100.64.101.2/31
ge-0/0/1.10: 100.64.101.10
100.64.101.0/31
100.64.101.10/31
ge-0/0/9.10: 100.64.101.3
ge-0/1/8.10: 100.64.101.11
ge-0/1/9.10: 100.64.101.12
ge-1/0/9.10: 100.64.101.1
100.64.101.12/31
ge-1/0/4.10: 100.64.101.13
Management LAN
Docker Peering LAN
lab-holonet01
BGP
S
IP transit
PNI
IP Transit Customer
IX
B2B Customer
bgp session(s)

Terminal recorded demo

REFERENCES

1. RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)
2. Juniper Routing Policies for BGP Communities
3. salt-sproxy https://github.com/mirceaulinic/salt-sproxy
4. NAPALM: Network Automation and Programmability Abstraction Layer with Multivendor support
5. RFC 8092: BGP Large Communities Attribute
6. RFC 8195: Use of BGP Large Communities