# BGP FlowSpec Services

## beyond DDOS mitigation

Wholesale Winery Tour - 05/2023

nicola modena - CCIE #19119 / JNCIE-SP #986
nicola@modena.to - @nmodena

# Agenda

- BGP FlowSpec origins & typical DDOS scenario
- Architecture & Configuration
- BGP-FS Service 1 – flow based egress engineering
- BGP-FS Service 2 – bidirectional traffic steering
- BGP-FS Service 3 - NFV

## About me

Nicola Modena - CCIE #19119 R&S (15Y) / JNCIE-SP #986 Emeritus

Independent Network Architect

More than 25 years experience designing and implementing service provider and large enterprise networks.

https://linkedin.com/in/nmodena

1

# BGP FlowSpec

Origins and typical DDOS scenario

# BGP FlowSpec

*«Dissemination of Flow Specification Rules» [for IPv6]*

Defined in RFC5575 (2009) up by RFC7674, RFC8955 for IPv4, RFC8955 for IPv6

some draft exist for specific functions ( if-group / persistence / SR )

in a nutshell:

- Distributed PBR (Policy Based Routing)
- Signaled with BGP with a dedicated AFI/SAFI
- Mostly used for DDOS mitigation

NOTE: FlowSpec  <is not> OpenFlow <and> <is not> NetFlow

## BGP FlowSpec

### FLOW SPECIFICATION

Src/Dst Address/Subnet
Src/Dst Port/Range
IP Protocol
ICMP Type/Code
TCP Flags
Packet Lenght
DSCP Value
Fragment Bits

### ACTION

Traffic Rate Bytes/Packets
Drop [rate = 0]
Send to VRF
Set DSCP
Sample
Redirect NH

Example: Drop all UDP traffic sourced from port 123 & dest IP 192.0.0.0/24

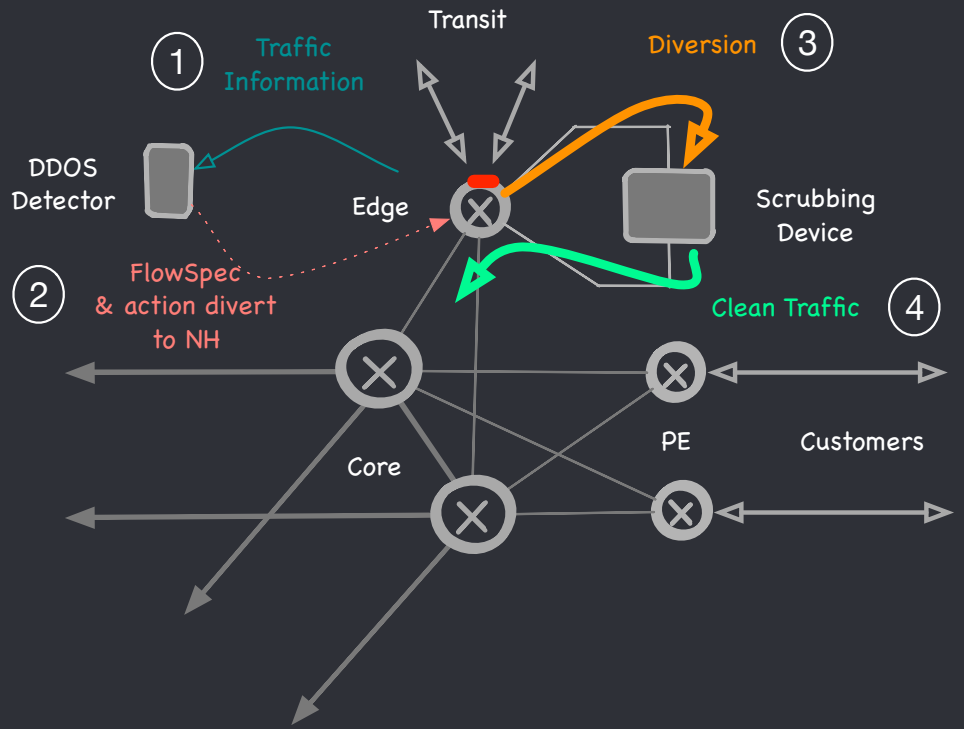
# BGP FlowSpec on Edge Router

1) Traffic info to DDOS detector
2) mitigate DDOS via BGP-FS

   Volumetric DDOS
   -> FlowSpec + action DROP

   Application DDOS
   -> FlowSpec + action Divert (NH)

3) traffic submitted for Scrubbing
4) Valid traffic re-injected into backbone



# BGP-FlowSpec on distributed infrastructure

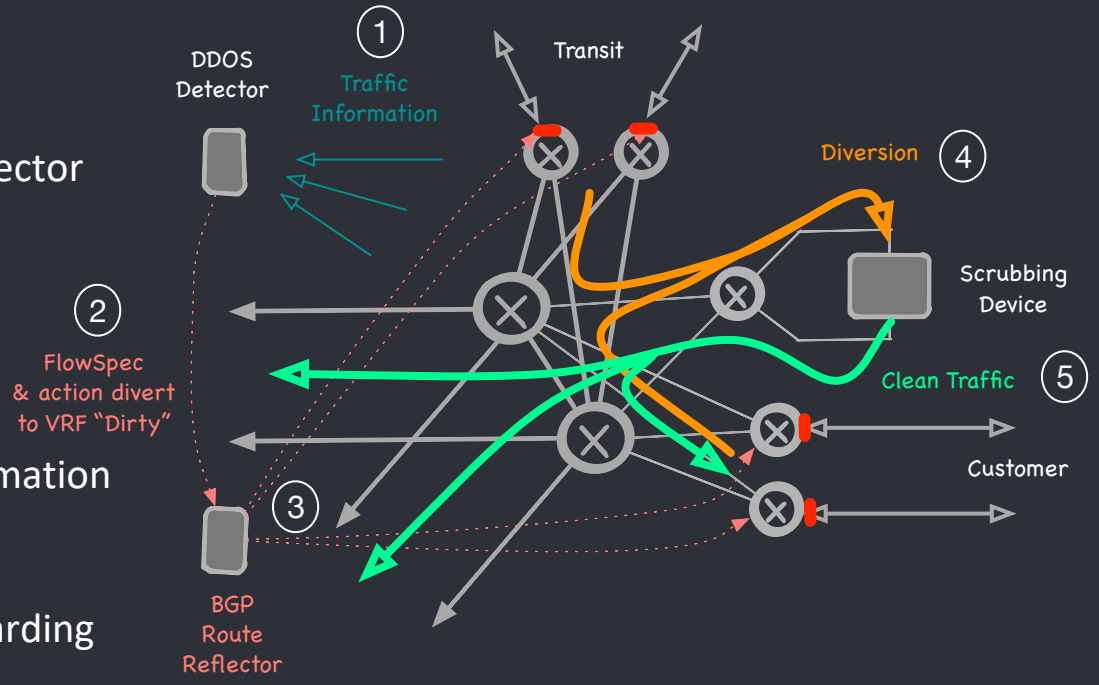
1) Traffic Information to DDOS detector

2) Flow description to RR
   -> set DSCP/EXP to Scavenger
   -> divert to «Dirty» VRF

3) RR distribuite Flow/Action information

4) Traffic dropped or submitted to
   Scrubbing device via MPLS forwarding

5) valid traffic reinjected into backbone
   to reach final destination




NOTE: diversion policy must be applied ONLY on EDGE interfaces to prevent traffic loops

# 2 Architecture & Configuration

# BGP FlowSpec enabled Backbone

1) DDos Detector

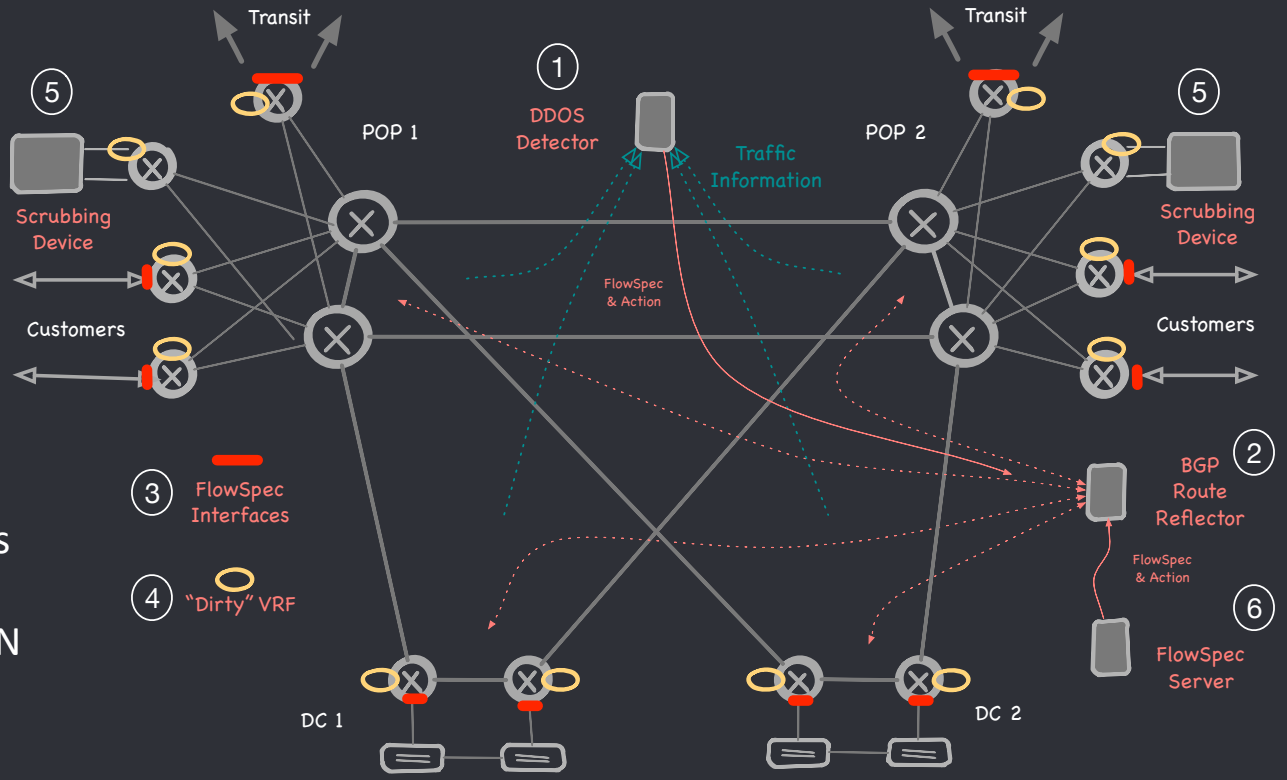
2) RR with FS AFI/SAFI

3) FlowSpec Interfaces

4) «Dirty» MPLS L3VPN

5) Scrubbing devices

6) OPT FlowSpec server for custom policy injection



# Router (client) configuration

**IOS XR**

```
!*** enable AFI/SAFI ***
router bgp $ASN$
   ...
   address-family ipv4 flowspec
   address-family ipv6 flowspec
   !
   neighbor $RR$
    ...
    address-family ipv4 flowspec
      route-policy FLOWSPEC4-FILTER-IN in
      maximum-prefix 1000 95 discard-extra-paths
    !
    address-family ipv6 flowspec
      route-policy FLOWSPEC6-FILTER-IN in
      maximum-prefix 1000 95 discard-extra-paths
!!
!*** activate on the platform ***

flowspec
   local-install interface-all
!
!*** disable on specific interfaces ***

interface XXXX
   ipv4 flowspec disable
   ipv6 flowspec disable
```

**Junos**

```
/*** enable AFI/SAFI ***/
protocols {
    bgp {
        group iBGP {
             import [.. FLOWSPEC-FILTER-IN ]
             family inet {
                flow {
                     accepted-prefix-limit {
                         maximum 1000;
                    }
                }
                family inet6 {
                   flow {
               [...]
}}
/*** activate on the platform ***/
routing-options {
    flow {
        interface-group 1 exclude;
        term-order standard;
}}

/*** disable on specific interfaces ***/

interfaces XXXX unit 0 family inet filter group 1
interfaces XXXX unit 0 family inet6 filter group 1
```

## BGP FlowSpec Server / Controller

FlowSpec Server / Controller to inject custom policy

ExaBGP - https://github.com/Exa-Networks/exabgp

GoBGP - https://github.com/osrg/gobgp)

...

Junos / IOS-XR ( crpd / XRd )

## BGP FlowSpec policy on ExaBGP

example: protect 192.0.0.0/24 from an NTP amplification attack

1) define peering configuration to RR

2) enable AFI/SAFI

3) define FLOW

4) define ACTION

**ExaBGP**

```
neighbor $route-reflector$ {                ## 1
    router-id $local-ip$;
    local-address $local-ip$;
    local-as $ASN$;
    peer-as $ASN$;
    group-updates false;

    family {                                ## 2
        ipv4 flow;
    }

    flow {
        route  ntp-ddos {
            match {                         ## 3
                destination 192.0.0.0/24;
                source-port 123;
                protocol udp;
            } then {                        ## 4
                discard;
            }
        }
    }
}
```

# BGP Flowspec on Junos client

Junos

```
nmodena@MX01> show route protocol bgp table inetflow.0

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.0/24,*,proto=17,srcport=123/term:2
                   *[BGP/170] 00:05:34, localpref 100, from 172.16.1.15
                      AS path: I, validation-state: unverified
                       Fictitious

nmodena@MX01> show firewall | find flows

Filter: __flowspec_default_inet__
Counters:
Name                                                Bytes              Packets
192.0.0/24,*,proto=17,srcport=123                       0                    0
```

- FlowSpec policy definition received with BGP
- Automatically translated in firewall filter

# Best Practice

**Implement import policy to prevent Control-Plane interruptions**

ML, AI and expecially humans can be very smart creating policy ☺

es. prevent traffic filtering to TCP 179 from trusted source.. (Bridging Gap Protocol ☺ )

**Organize and tag FlowSpec policies with custom communities**

in order to filter/apply policy only on specific devices type ( es: internal, external )

**Read carefully device capacity and limit the number of entry accepted**

typically from a few hundred to a few thousand entries

flowspec rules are implemented in HW like ACL

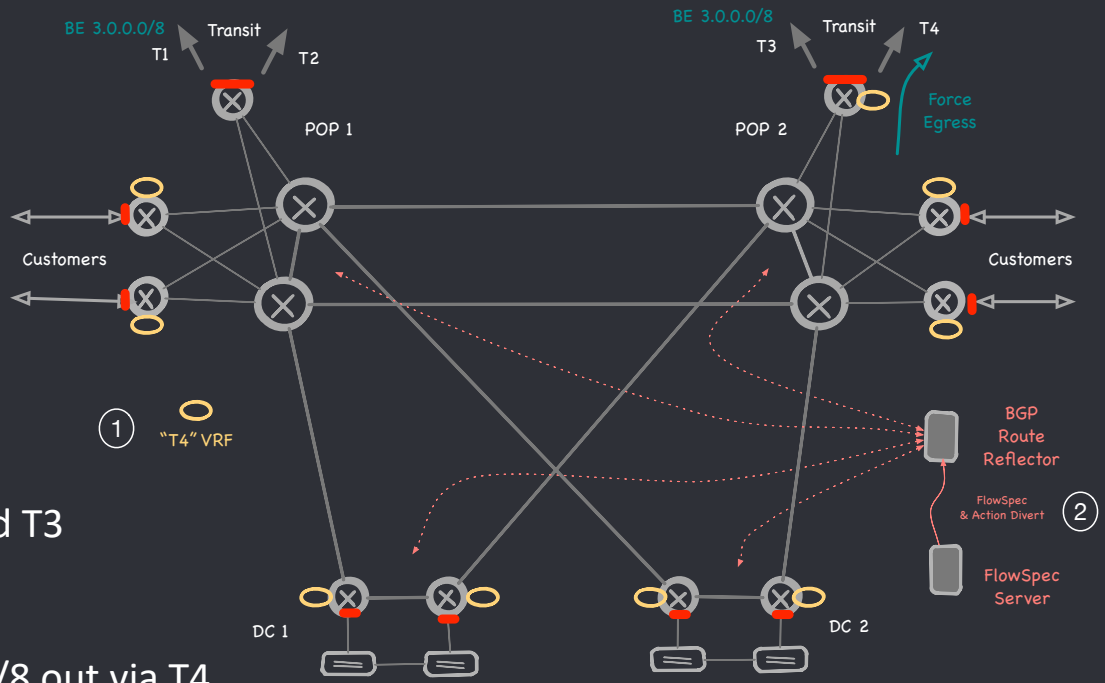
limit max accepted prefix per AFI/SAFI AFTER import-policy enforcement

3

# use case 1 : Flows-based egress engineering

bypass routing for specific traffic flows

# Flows-based egress engineering




scenario:
3.0.0.0/8 best path via T1 and T3

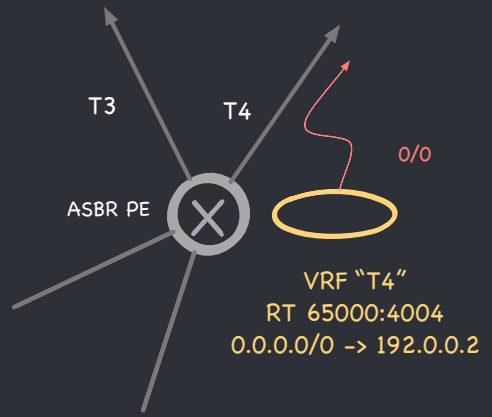
requirement:
force some traffic for 3.0.0.0/8 out via T4

1) Create a «T4» MPLS L3VPN with 0/0 pointing to T4 as next-hop
2) Distribute a FlowSpec definition to divert required traffic into VRF T4

## Flows-based egress engineering

**Junos ASBR**

```
routing-instances {
    T4 {
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 172.16.4.1;
        }}
        instance-type vrf;
        vrf-import none;
        vrf-export vrf-export-T4;
        vrf-table-label;
}}
policy-options {
    policy-statement vrf-export-T4 {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then {
            community add vrf-target-T4;
            accept;
    }}
    community vrf-target-T4 members target:65000:4004;
}
note: import interface-route with a rib-group
```




**Junos all PE**

```
routing-instances {
    T4 {
        instance-type vrf;
        vrf-target target:65000:4004;
    }
}
```

On ASBR advertise a default-route into T4 L3VPN

NOTE: Avoid  local IP lookup and provide fallback

## Flows-based egress engineering

ExaBGP

```
[...]

flow {
  route DC1-DC2-to-AWS-via-T4 {
    match {
      source 192.0.2.0/24;
      destination 3.0.0.0/8;
    }
    then {
      # install on DC 1 & DC 2
      community [65000:48001 65000:48002];

      # redirect to vrf T4 (
      redirect 65000:4004;
    }
}
```

Activate diversion defining the policy

- flow description
- optional community to control distribution
- redirect flow pointing to VRF RT 65000:4004

## Flows-based egress engineering

- Useful for probing and temporary traffic diversion
- Quick solution without backbone policy change
- VRF for most used transit can be permanently defined

  -> ( just 1 FIB entry x VRF )

NOTE:

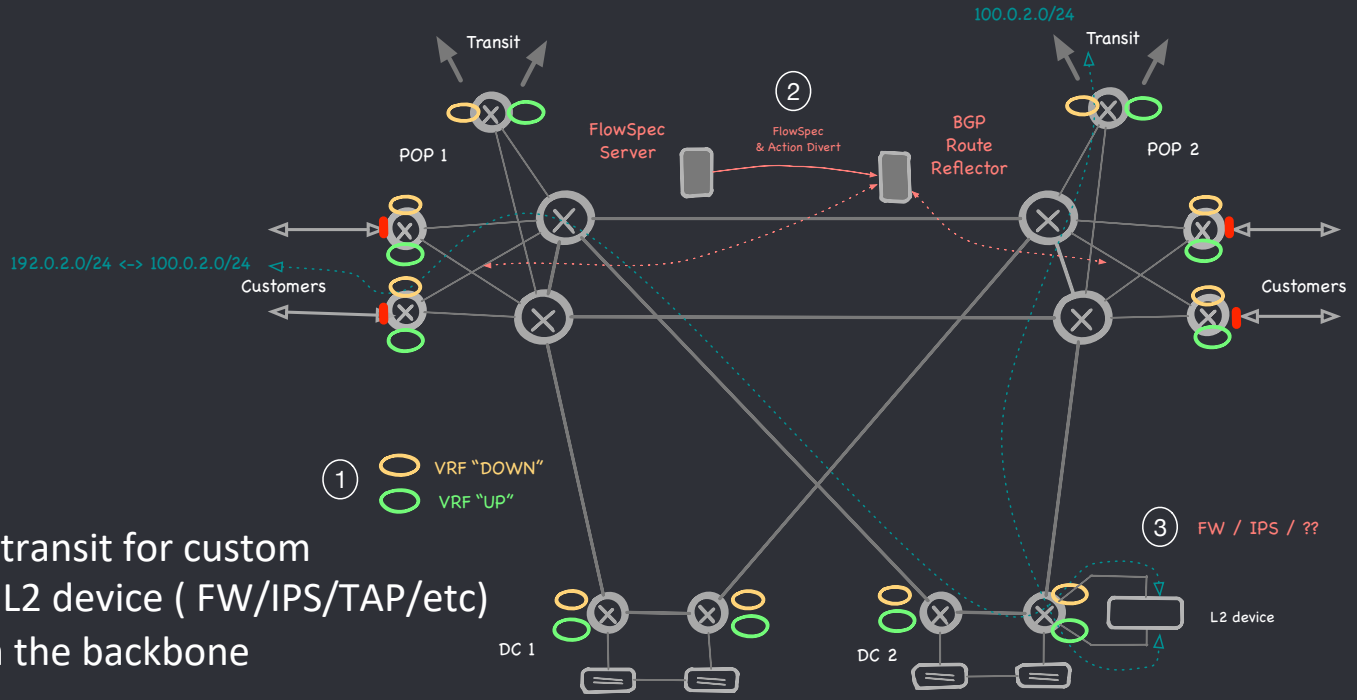
- affect only EGRESS traffic !
- check/set default platform diversion action if vrf doesn't exist

  -> ( drop -> forward )
- provide fallback if transit goes down

  -> ( floating default route )

# 4 use case 2 : bidirectional traffic steering

# Bidirectional traffic steering



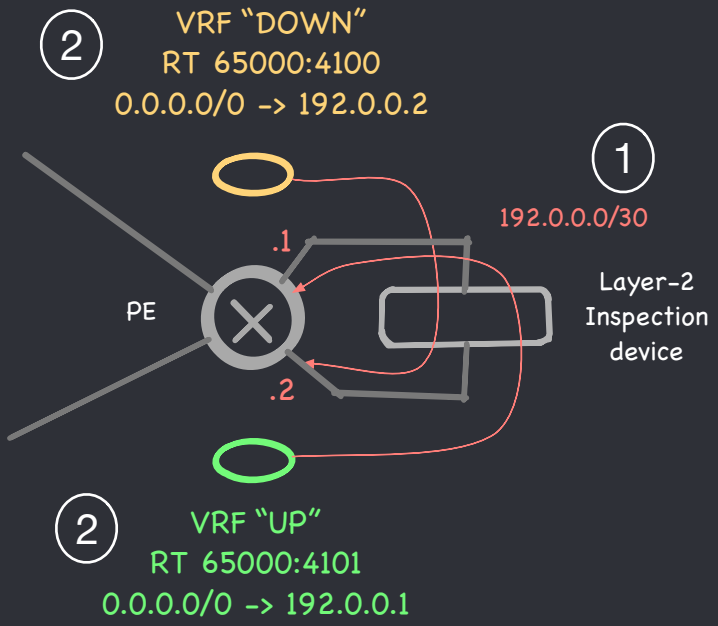

scenario:
force bidirectional transit for custom SRC/DST flow on a L2 device ( FW/IPS/TAP/etc) located anywere in the backbone

1) two MPLS L3 VPN for Downstream and Upstream traffic
2) two mirrored FlowSpec policy to divert into Down and Up VRF
3) vrf-exit points with default-route leaking trought the layer-2 device

# Bidirectional traffic steering

**ExaBGP**

```
[...]

flow {
  route CUST-UP {            <- UPSREAM TRAFFIC FLOW
    match {
      source 192.0.2.0/24;
      destination 100.0.2.0/24;
    }
    then {
      redirect 65000:4101;       // RT destination VRF
  }}
  route CUST-DOWN {          <- DOWNSTREAM TRAFFIC FLOW
    match {
      source 100.0.2.0/24;
      destination 192.0.2.0/24;
    }
    then {
      redirect 65000:4100;       // RT destination VRF
}}}
```




1) PE has 1 point-to-point link in Global Routing Table trough the L2 device
2) UP & DOWN vrf exit-points with default-route leaking trought the layer-2 device
   -> IP lookup it's performed in GRT after crossing L2 «inspection» device

# 5 use case 3 : traffic steering for NFV

## NFV with BGP FlowSpec

example:

- Analyze ALL DNS traffic for selected customers

  (es: who have subscribed for parental-control)

but also valid for other scenario:

- Intercept all web traffic to trigger redirect to a captive portal for user activation/deactivation (and block the remaining traffic)
- Insert a pool of caching proxy/waf in front of web server
- as an infrastructure for almost any NFV solution

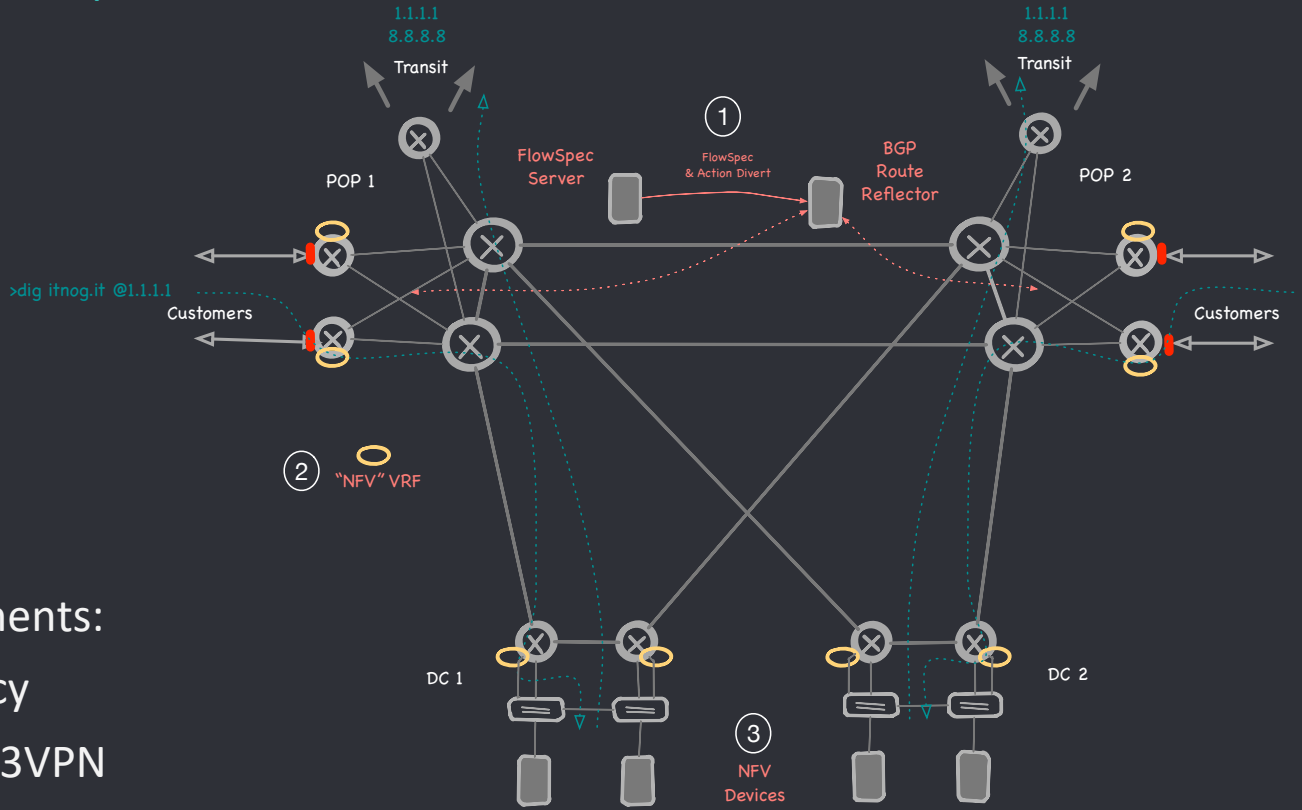
## NFV with BGP FlowSpec

**Service Provider Class Solutions :**

- Dynamic & Flexible -> BGP FlowSpec
- Load Balance -> BGP Multipath
- Proximity -> BGP path selection (IGP Metric)
- Reliable -> BGP for HA
- Scalable -> BGP can scale ?

Guess what my favorite protocol is?

# NFV with BGP FlowSpec




Solution Components:

1) FlowSpec policy
2) «NFV» MPLS L3VPN
3) NFV Devices

# NFV with BGP FlowSpec

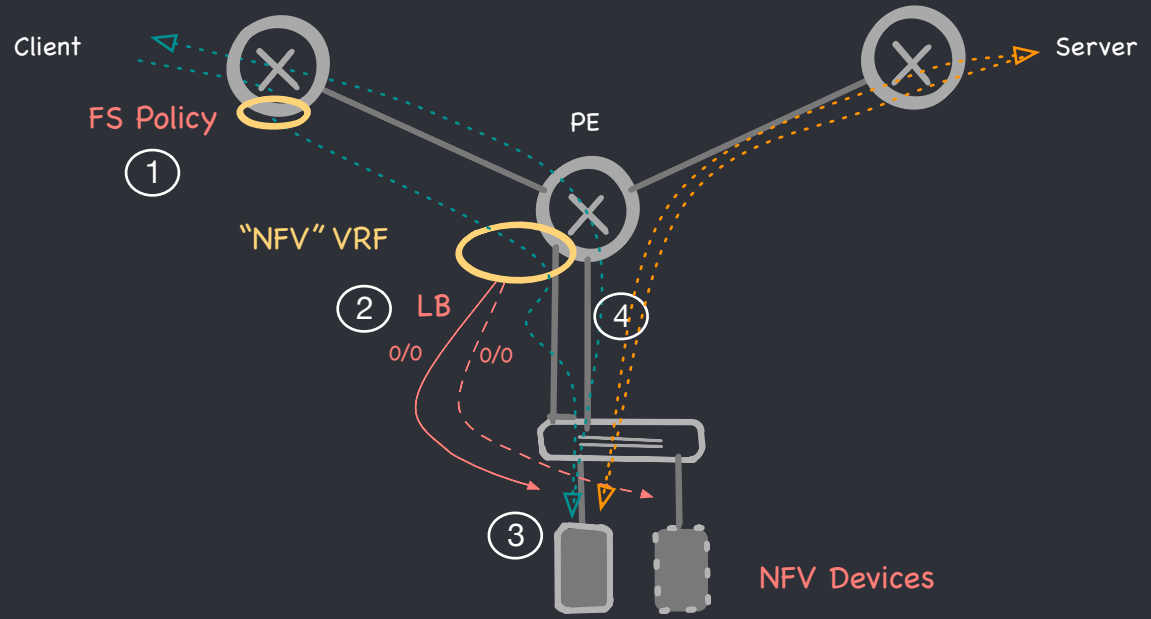
ExaBGP

```
[...]

 flow {
    route  parental-control-pool-1 {
        match {
            source 100.64.0.0/16;
            destination-port 53;
            protocol udp;
        }
        then {
            # install on BNG 1 & BNG 3
            community [65000:48011 65000:48012];

            # redirect to NFV
            redirect 65000:4010;
}}}}
```

Activate the diversion defining the policy

- flow description
- optional community to control distribution
- redirect flow pointing to VRF RT 65000:4010

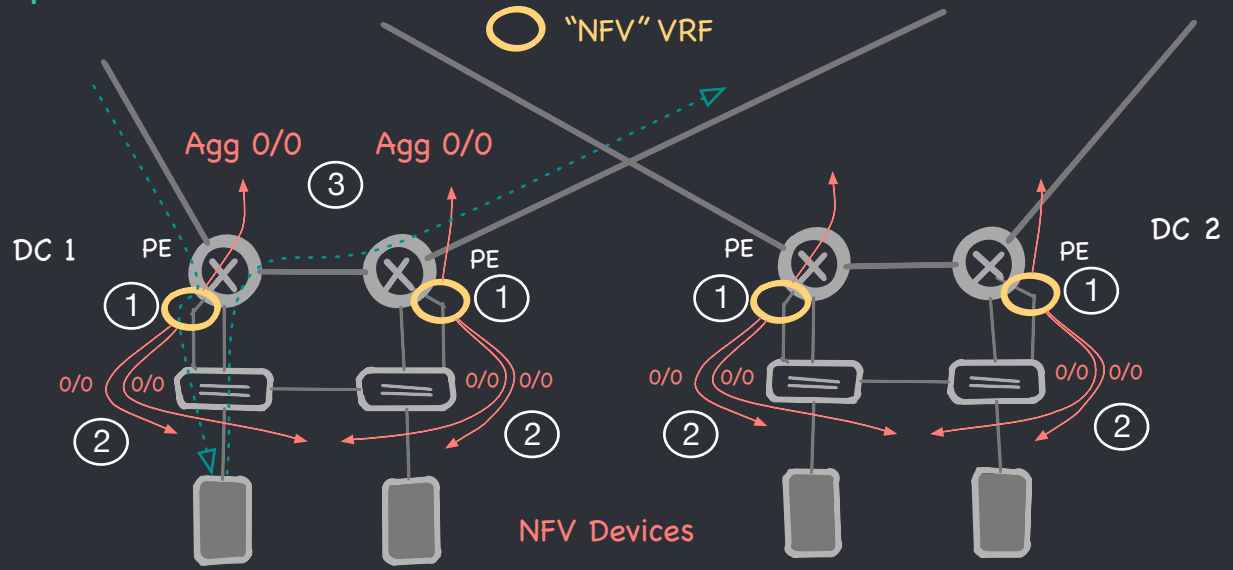
# NFV with BGP FlowSpec – traffic flow




Traffic Flow

- FlowSpec policy divert (1) upstream traffic
- Traffic exit from NFV vrf (2) on PE dedicated interface and it's distributed trough NFV devices
- Selected device receive traffic (3) and perform DNat for «catch all» services
- Return traffic and sessions to real destinations uses PE interface (4) in Global Routing Table

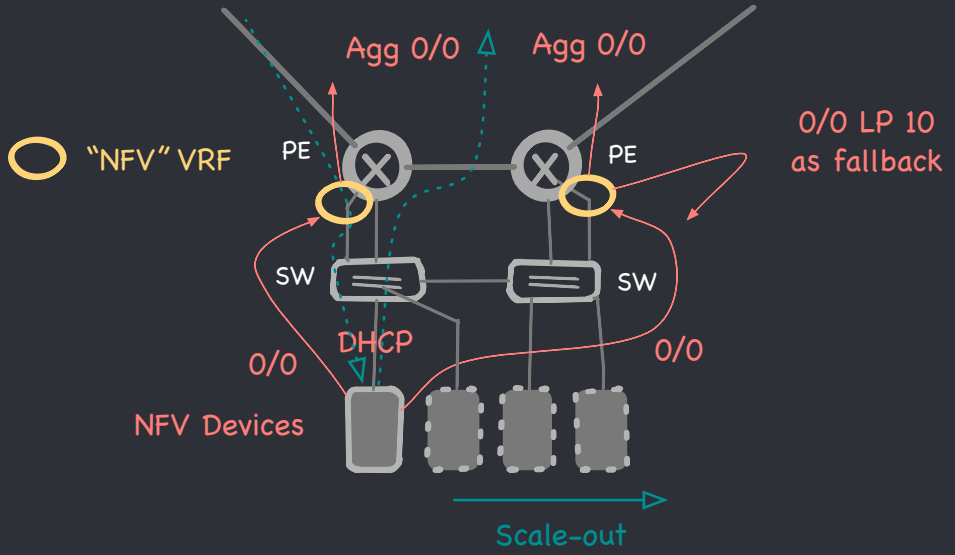
# NFV with BGP FlowSpec




NFV VRF exit points:

- Dedicated [sub] interface (1) in vrf NFV on at least 2 PE routers per DC
- Multiple default-route (2) pointing to each NFV device
- Multipath & consistent hash for local load balancing to NFV devices
- Only an «aggregate» default-route (3) advertised from each PE
- Remote PE will select the closer exit-point using IGP cost
- multipath / consistent hash it's not required on remote PE

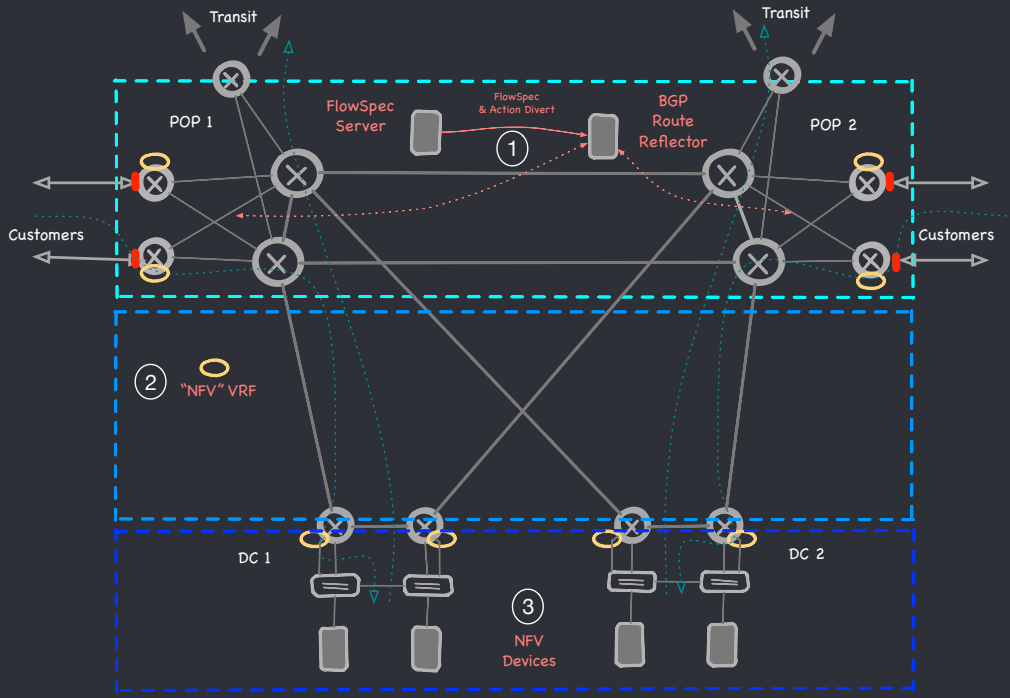
# NFV with BGP FlowSpec




Scale-out NFV solution with BGP:

- NFV as VM using dynamic IP via DHCP
- Setup 2 BGP session with PE interfaces in VRF NFV (hint: ExaBGP)
- Advertise default-route to PE in NFV vrf pointing to the NFV device
- NFV uses default-gw in GRT and traffic is asymmetric
- ready to migrate to container and K8S

# NFV with BGP FlowSpec




The solution is divided into 3 layer:

1 - Traffic diversion (BGP FlowSpec)
2 - Optimal traffic distribution & fallback (MPLS L3VPN)
3 - High Availability, Load Balancing and Scale-Out (BGP Session & Multipath)
Each layer it's **independent** and consistent
The common thread is BGP but used in three different ways

## Summary

- BGP FlowSpec it’s a powerful toolset
- Very often not considered and used just for DDOS mitigation
- just few lines of configuration on existing infrastructure (BGP & MPLS)
- NFV with Flowspec it’s more flexible & controllable than plain anycast

CONS

- it’s still PBR -> does not scale
- HW dependent -> check support & limits on each platform
- use with care, traffic loops are lurking

- Is this enough SDN ? ☺

0

# THANK YOU

Questions ?

a special thanks to:
Ivan Pepelnjak for invaluable input

Check for latest version of this presentation at https://github.com/nmodena/blog