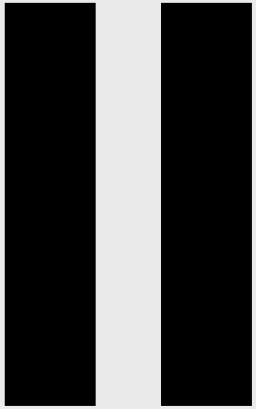


3

Markus Weber

Senior (46) Designer

Network Operations

2

KPN Eurorings Germany B.V.

KPN, KPN Eurorings, KPN International

L2-/L3-VPN services, VAS

wave

IP transit

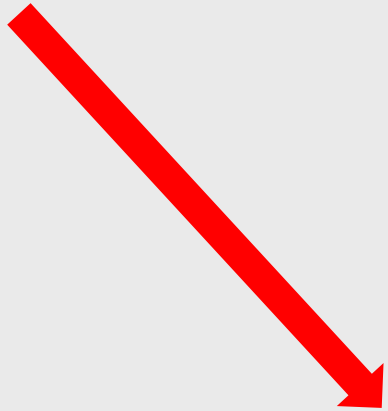
1

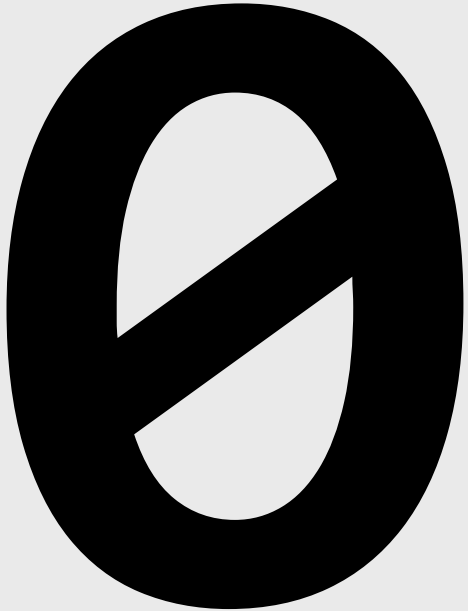
AS286

Europe / North America / Asia
transit free

mostly J, some legacy C

in case you forget the ASN / KomPaNy





rtBH – rtsBH - rtsdCoS

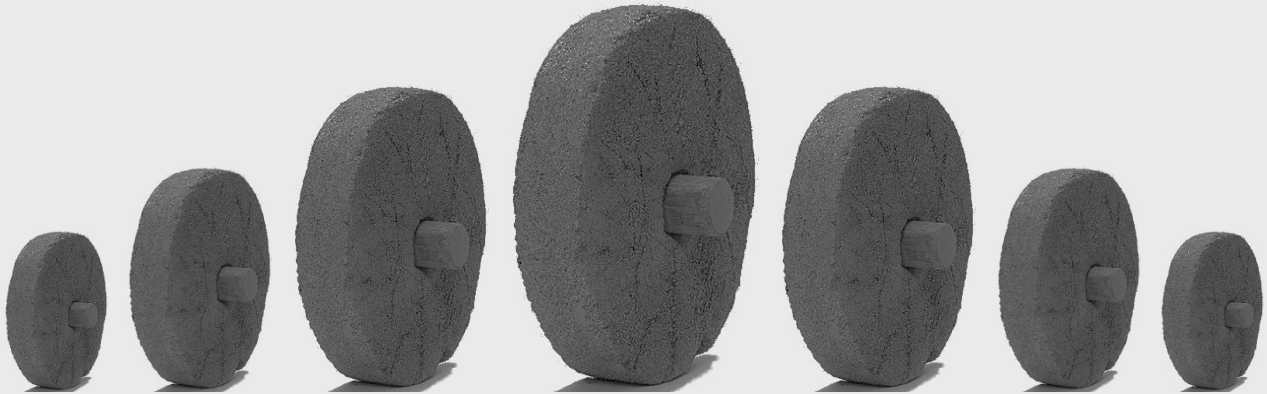
Markus Weber

https://as286.net

KPN Eurorings Germany B.V.

reinventing the wheel

over and over again



- this is about (very) old things – sorry!

- this is about (very) old things – sorry!
- it‘s really nothing new – sorry again for talking about old things!

- **this is about (very) old things – sorry!**
- **it‘s really nothing new – sorry again for talking about old things!**

- **with rtsdCoS (2016) I‘ve just reinvented (honestly) something – but the basic idea was already mentioned / presented @ Black Hat USA-2002 by Nicolas Fischbach** (as I found out a week ago) **[1]**

- **this is about (very) old things – sorry!**
- **it‘s really nothing new – sorry again for talking about old things!**

- **with rtsdCoS (2016) I‘ve just reinvented (honestly) something – but the basic idea was already mentioned / presented @ Black Hat USA-2002 by Nicolas Fischbach** (as I found out a week ago) **[1]**

- **I‘m in good company (e.g.):**
  - selective BH was brought to a wider audience by Job Snijders in 2014 (NANOG post, RIPE68 and other talks) – goes back (or even before) to RFC3882 (2002) and a Cisco paper on BH (2005) [2]
  - ABH (not covered here), RIPE73 (2016) by François Contat applies RFC5635 on RFC3882’s sinkhole device

- **this is about (very) old things – sorry!**
- **it‘s really nothing new – sorry again for talking about old things!**

- **with rtsdCoS (2016) I‘ve just reinvented (honestly) something – but the basic idea was already mentioned / presented @ Black Hat USA-2002 by Nicolas Fischbach** (as I found out a week ago) **[1]**

- **I‘m in good company (e.g.):**
  - selective BH was brought to a wider audience by Job Snijders in 2014 (NANOG post, RIPE68 and other talks) – goes back (or even before) to RFC3882 (2002) and a Cisco paper on BH (2005) [2]
  - ABH (not covered here), RIPE73 (2016) by François Contat applies RFC5635 on RFC3882’s sinkhole device

- **don’t get me wrong: don’t want to make anyone look bad!**

- **this is about (very) old things – sorry!**
- **it‘s really nothing new – sorry again for talking about old things!**

- **with rtsdCoS (2016) I‘ve just reinvented (honestly) something – but the basic idea was already mentioned / presented @ Black Hat USA-2002 by Nicolas Fischbach** (as I found out a week ago) **[1]**

- **I‘m in good company (e.g.):**
  - selective BH was brought to a wider audience by Job Snijders in 2014 (NANOG post, RIPE68 and other talks) – goes back (or even before) to RFC3882 (2002) and a Cisco paper on BH (2005) [2]
  - ABH (not covered here), RIPE73 (2016) by François Contat applies RFC5635 on RFC3882’s sinkhole device

- **don’t get me wrong: don’t want to make anyone look bad!**
  - on the contrary: we must cheer them (as long as we listen to them and are surprised)
  - often it requires real life examples, code snippets, promotion, people implementing it and thus forcing others to follow, repeating things, sharing knowledge, train people on using “old” things, …
  - everyone always adds another aspect or summarize nicely or combines ideas

- **this is about (very) old things – sorry!**
- **it‘s really nothing new – sorry again for talking about old things!**

- **with rtsdCoS (2016) I‘ve just reinvented (honestly) something – but the basic idea was already mentioned / presented @ Black Hat USA-2002 by Nicolas Fischbach** (as I found out a week ago) **[1]**

- **I‘m in good company (e.g.):**
  - selective BH was brought to a wider audience by Job Snijders in 2014 (NANOG post, RIPE68 and other talks) – goes back (or even before) to RFC3882 (2002) and a Cisco paper on BH (2005) [2]
  - ABH (not covered here), RIPE73 (2016) by François Contat applies RFC5635 on RFC3882’s sinkhole device

- **don’t get me wrong: don’t want to make anyone look bad!**
  - on the contrary: we must cheer them (as long as we listen to them and are surprised)
  - often it requires real life examples, code snippets, promotion, people implementing it and thus forcing others to follow, repeating things, sharing knowledge, train people on using “old” things, …
  - everyone always adds another aspect or summarize nicely or combines ideas

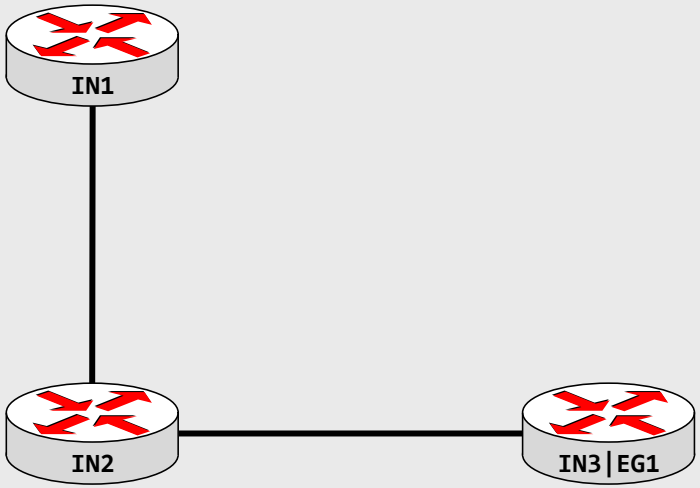
- **so let‘s talk about old things … hopefully still interesting …**

[1] https://www.blackhat.com/html/bh-media-archives/bh-archives-2002.html#USA-2002
http://www.securite.org/presentations/secip/

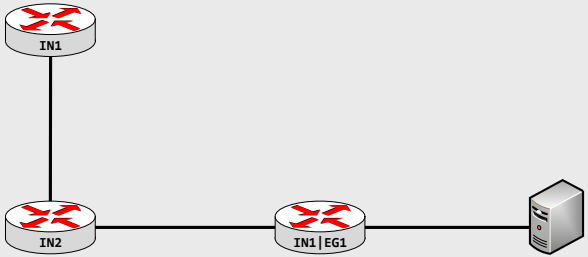
[2] http://www.cisco.com/c/dam/en_us/about/security/intelligence/blackhole.pdf

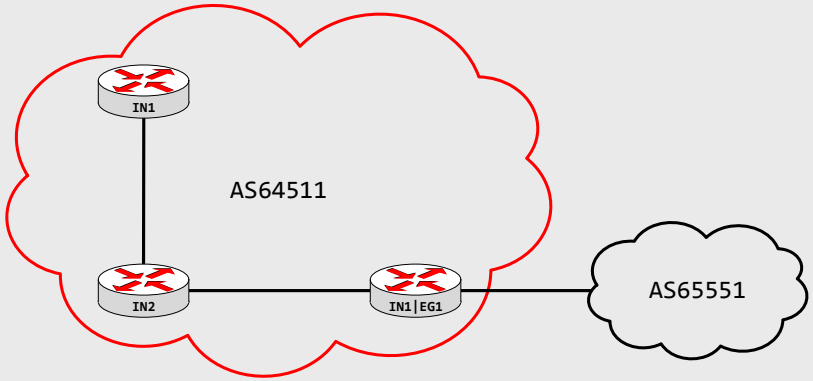
[3] https://tools.ietf.org/html/rfc3882
https://tools.ietf.org/html/rfc5635

a small network



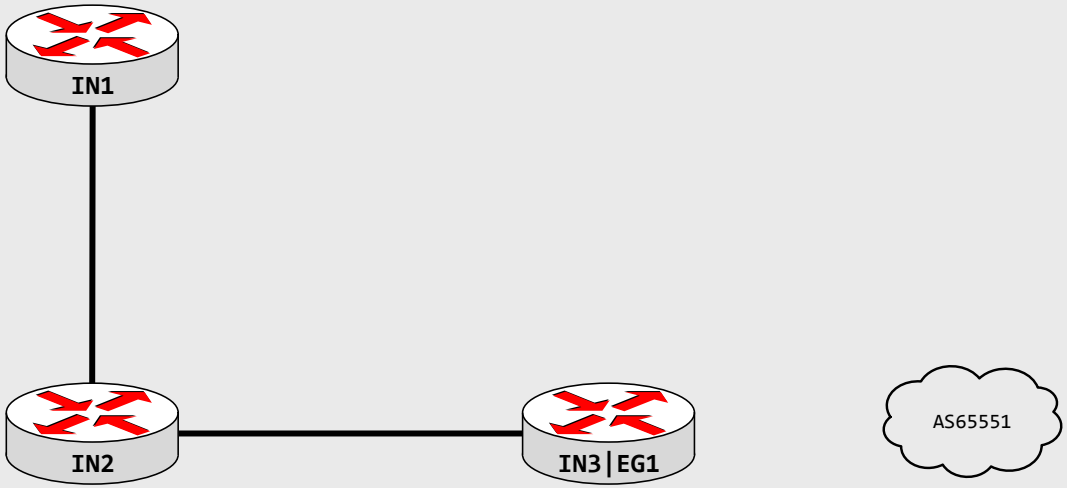
rtBH – rtsBH - rtsdCoS



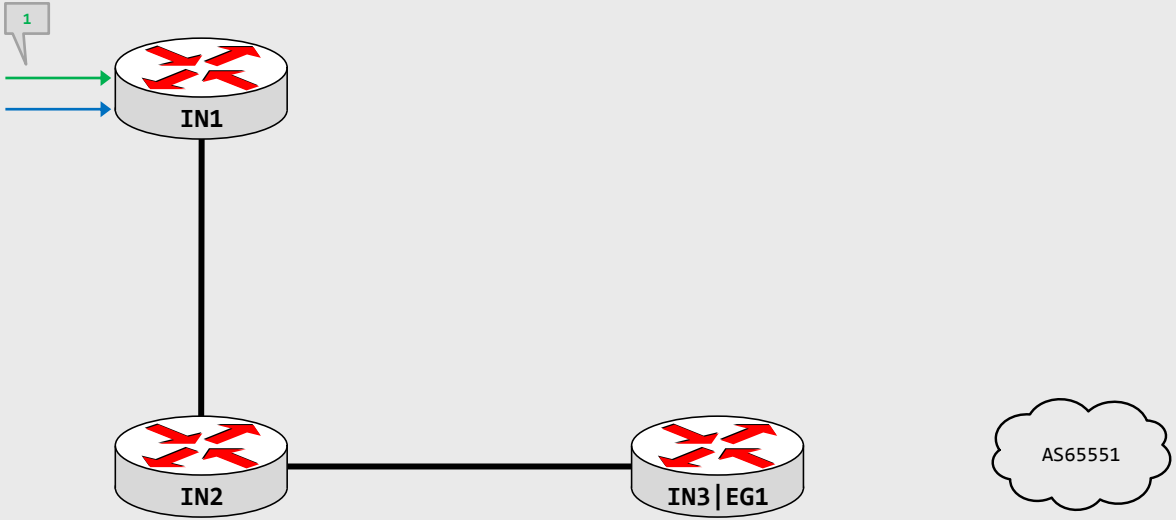


rtBH – rtsBH - rtsdCoS

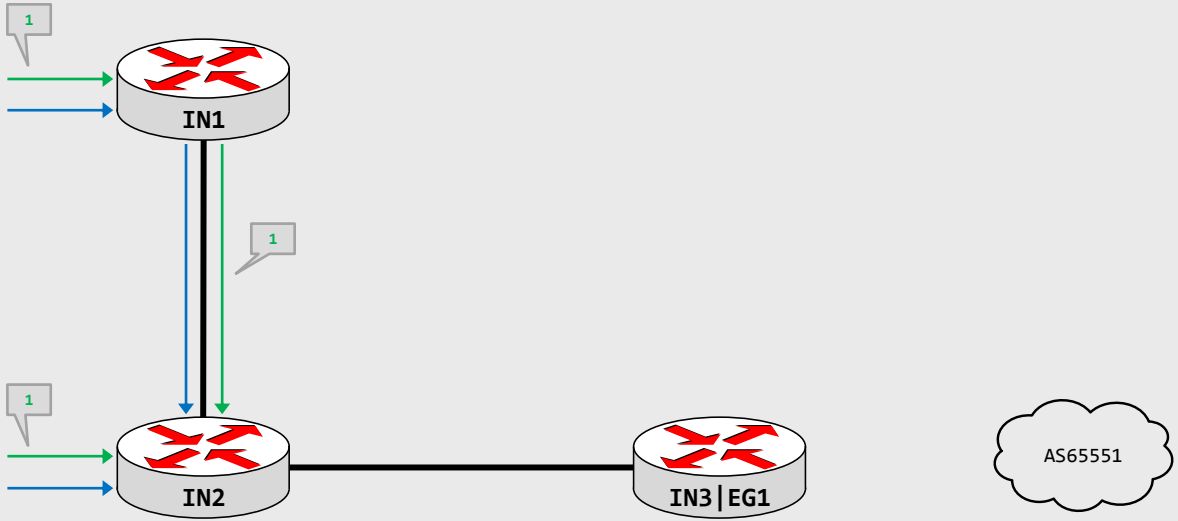
some traffic please



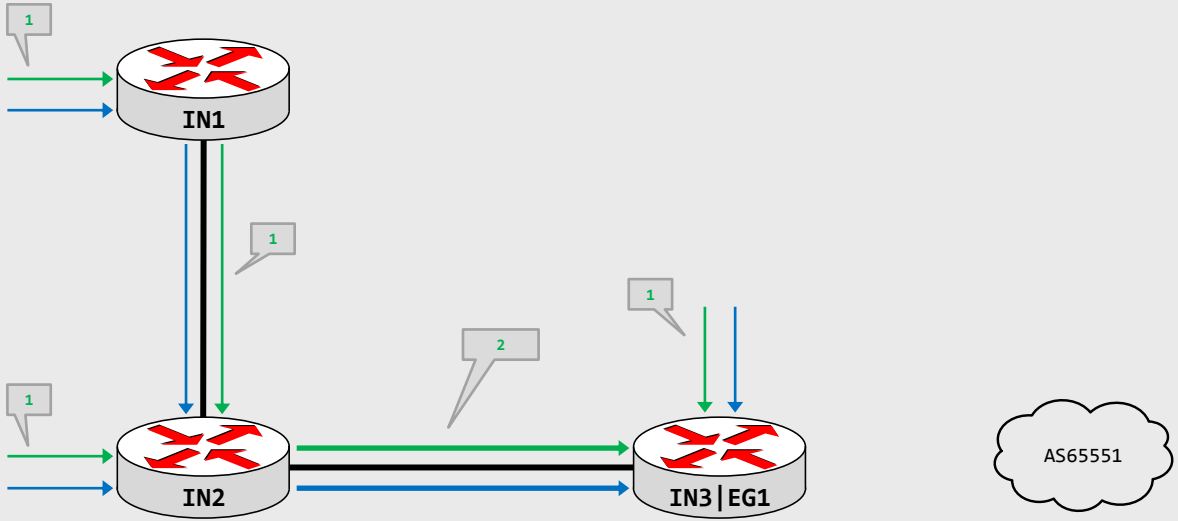
rtBH – rtsBH - rtsdCoS



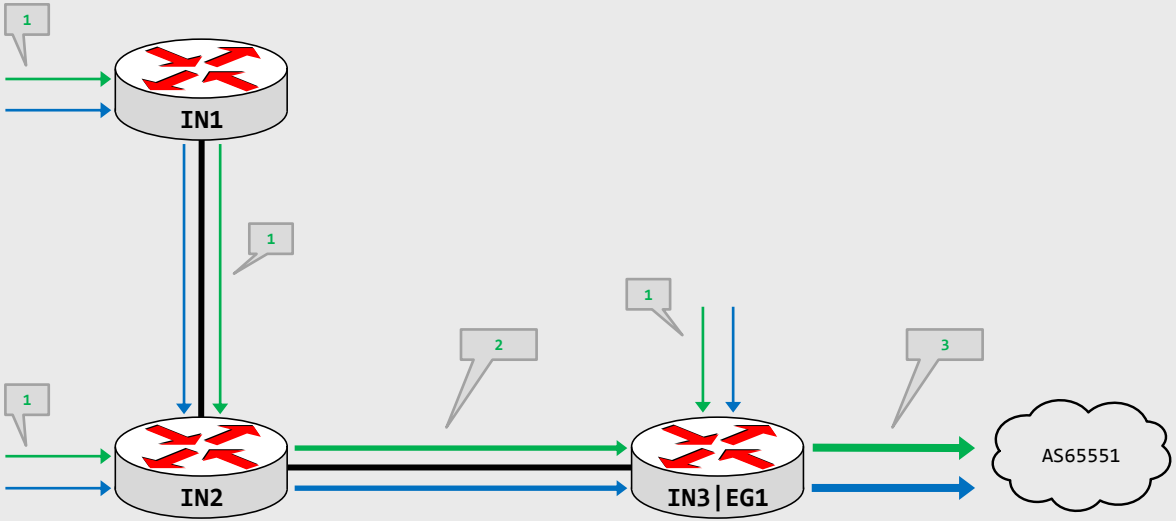
rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

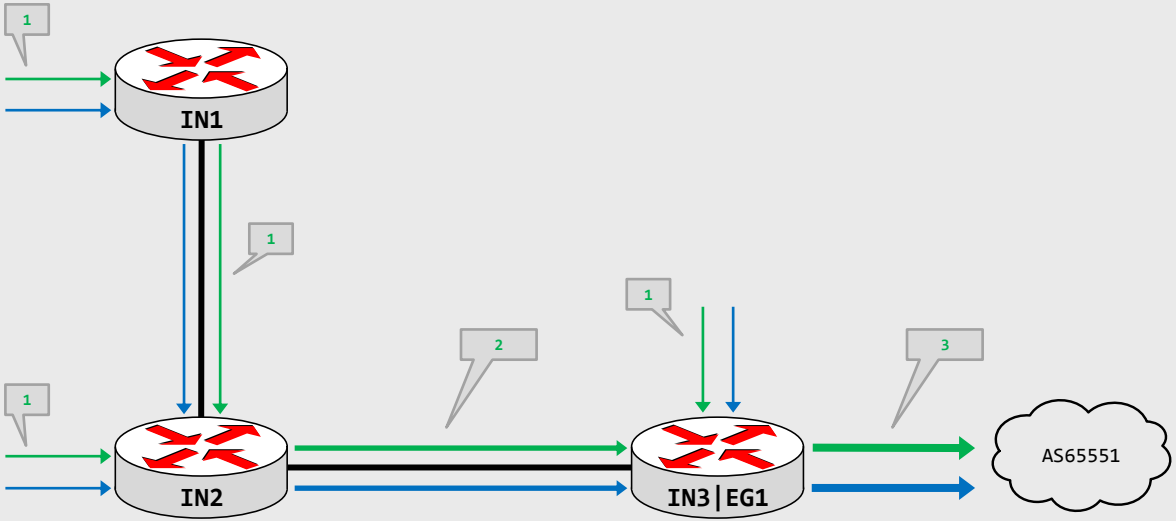


rtBH – rtsBH - rtsdCoS

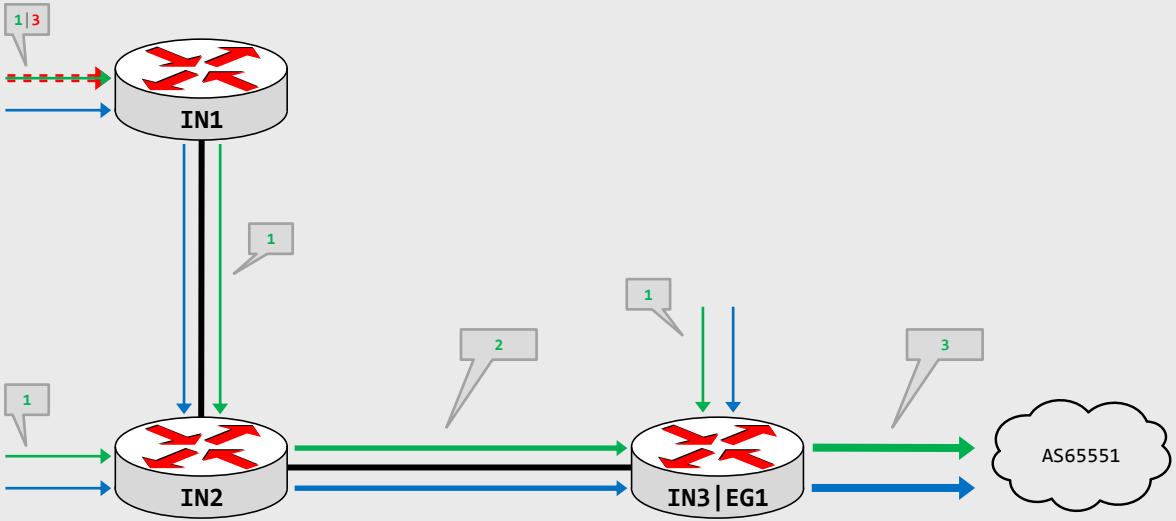


rtBH – rtsBH - rtsdCoS

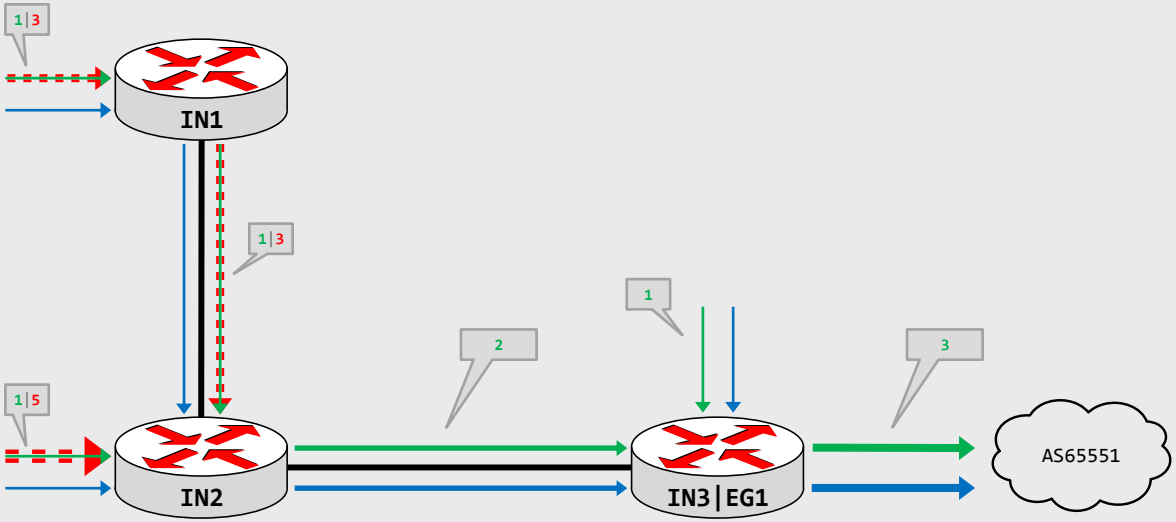
oh no, flooding …



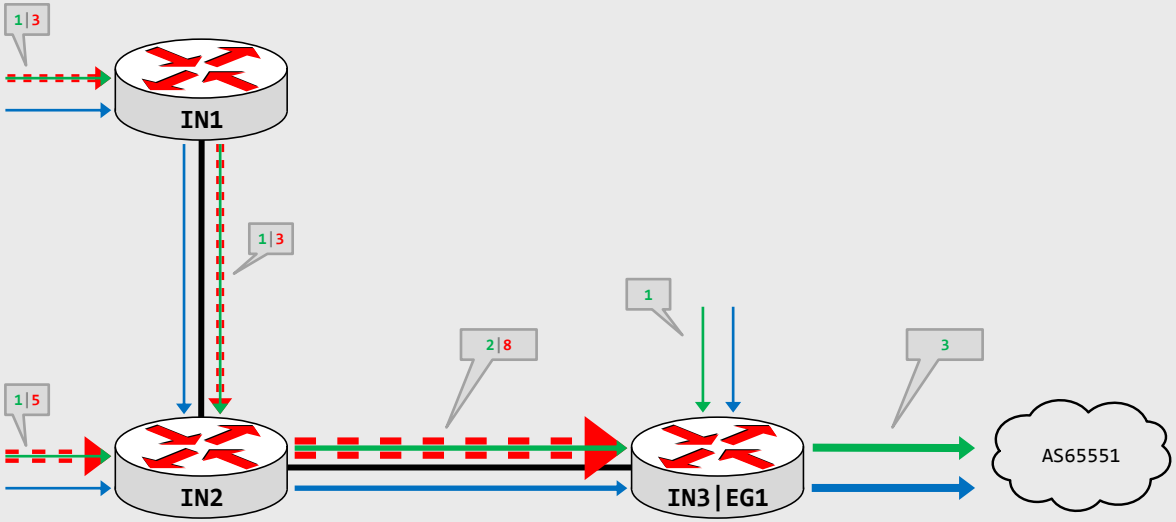
rtBH – rtsBH - rtsdCoS



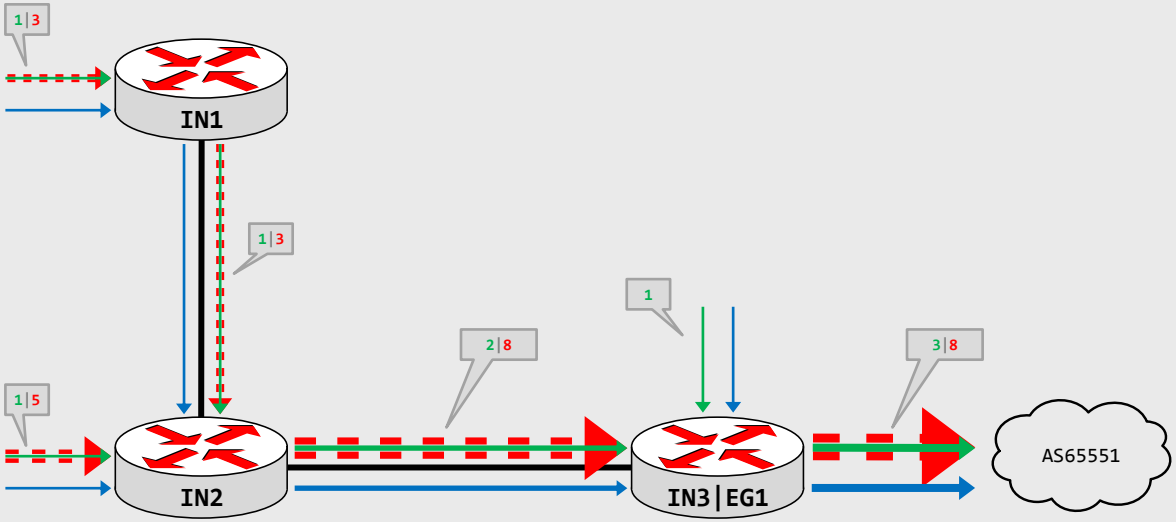
rtBH – rtsBH - rtsdCoS



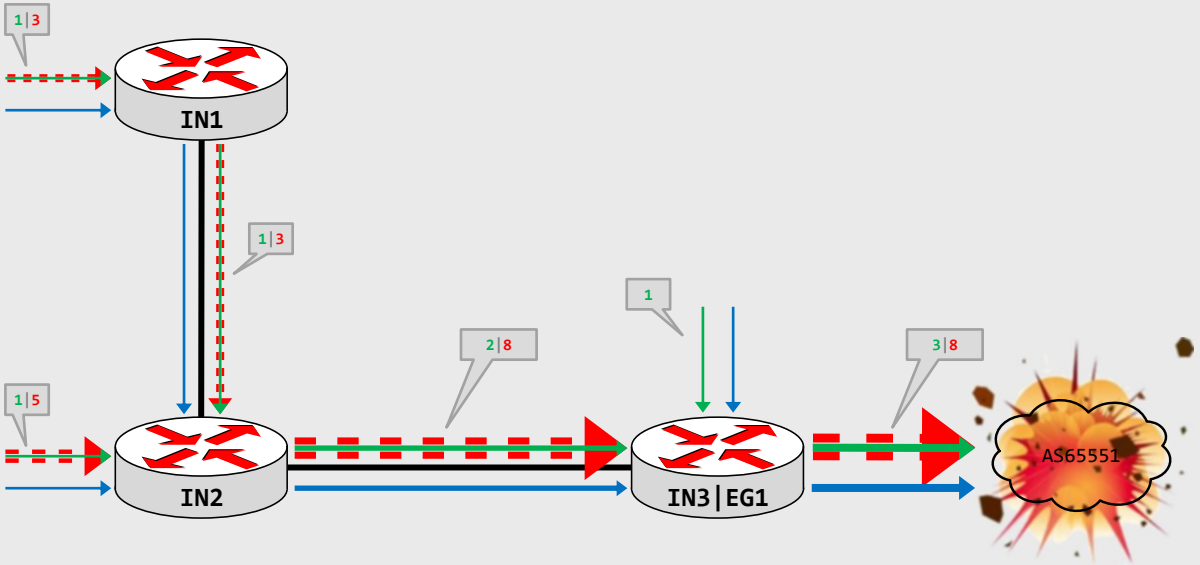
rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

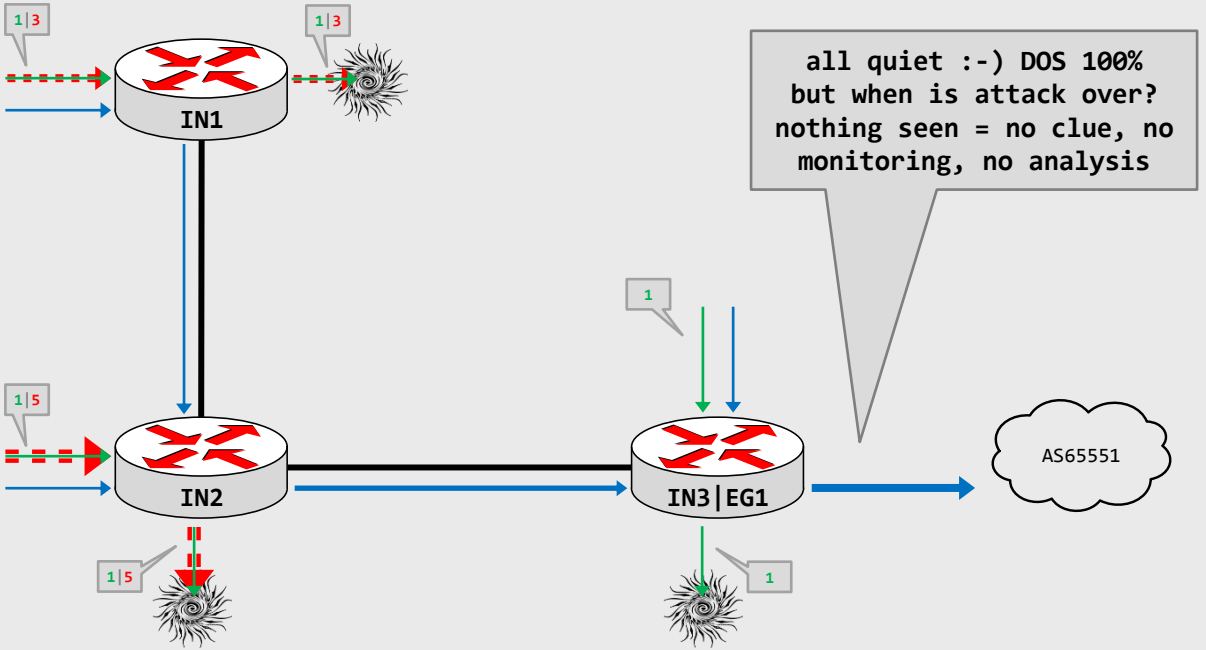


rtBH – rtsBH - rtsdCoS

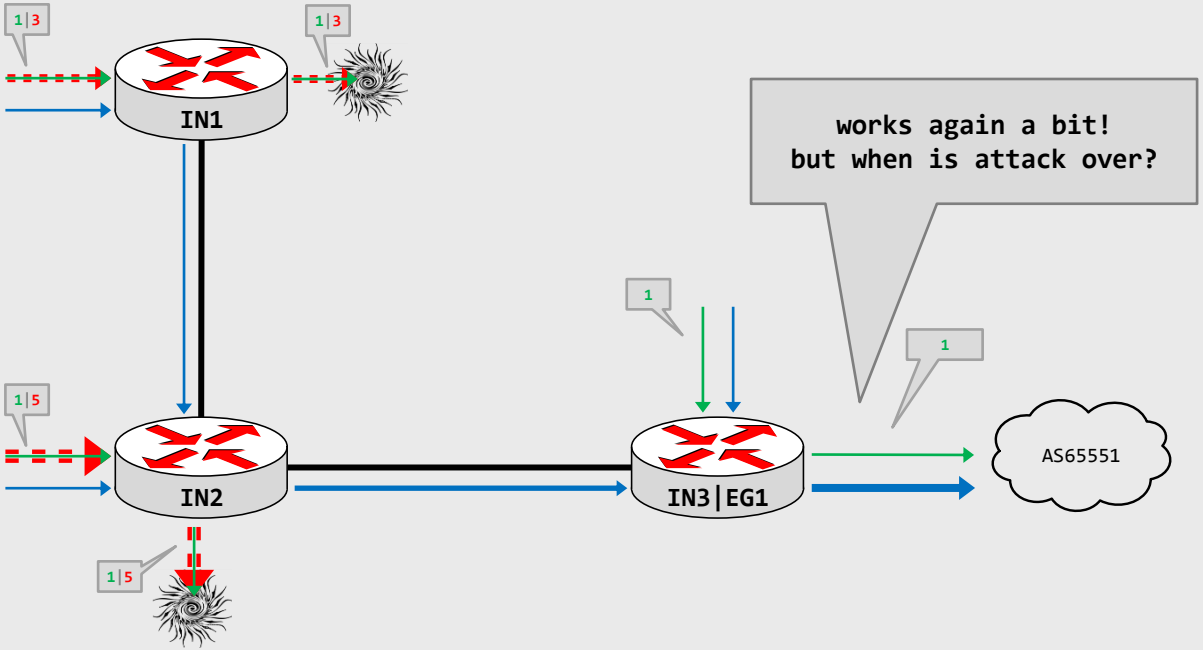


rtBH – rtsBH - rtsdCoS

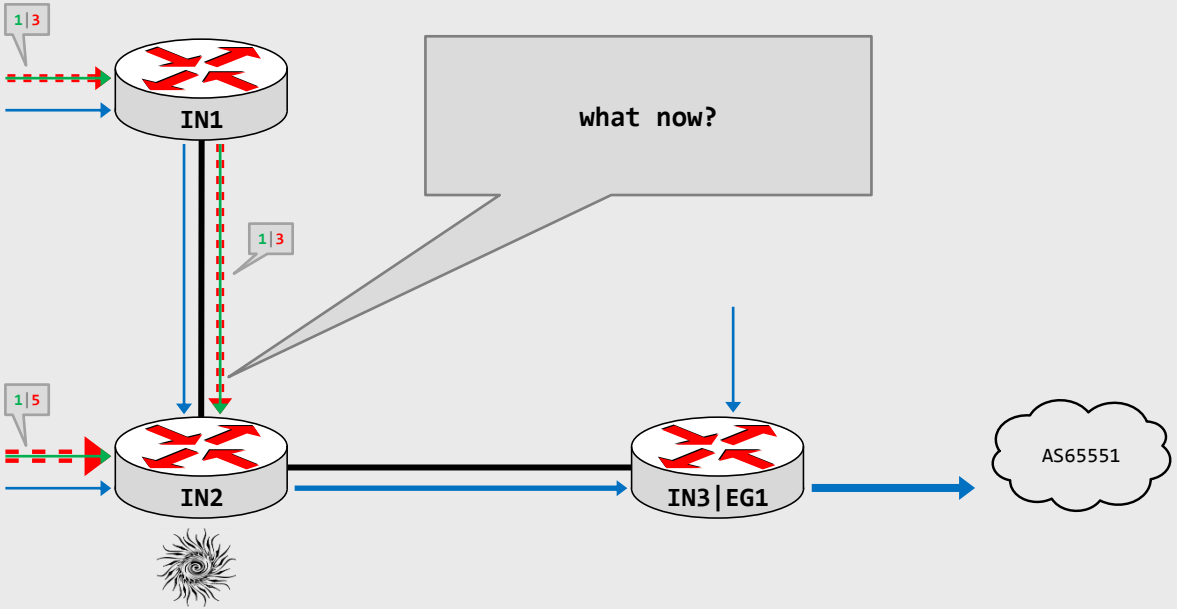
blackhole


1|3
1|3
IN1
all quiet :-) DOS 100%
but when is attack over?
nothing seen = no clue, no
monitoring, no analysis
1
1|5
IN2
IN3|EG1
AS65551
1|5
1

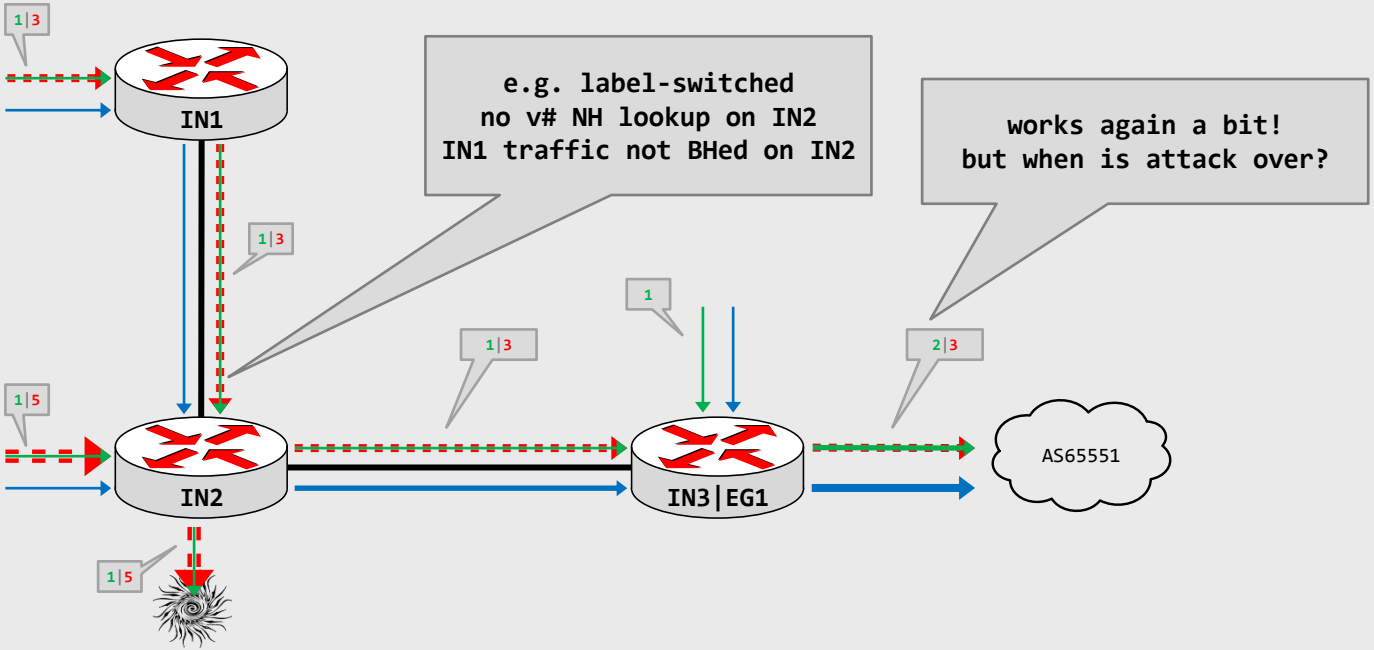
blackholing in some places



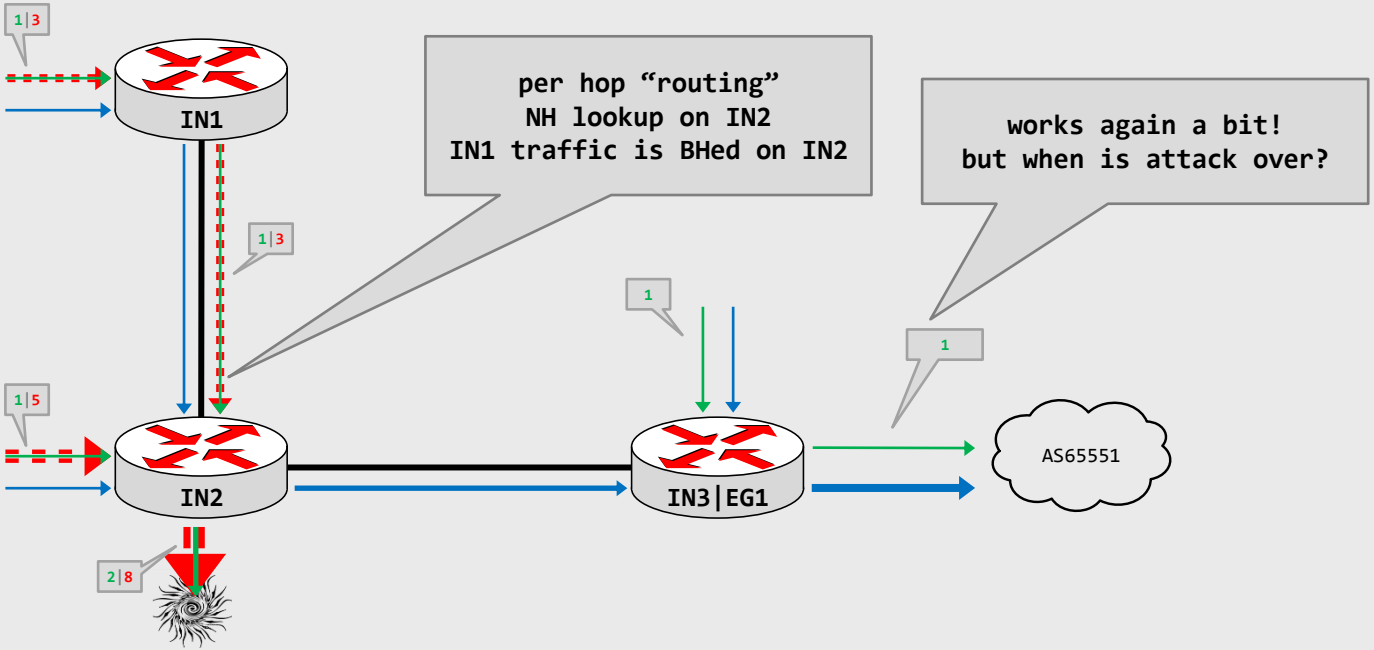
rtBH – rtsBH - rtsdCoS



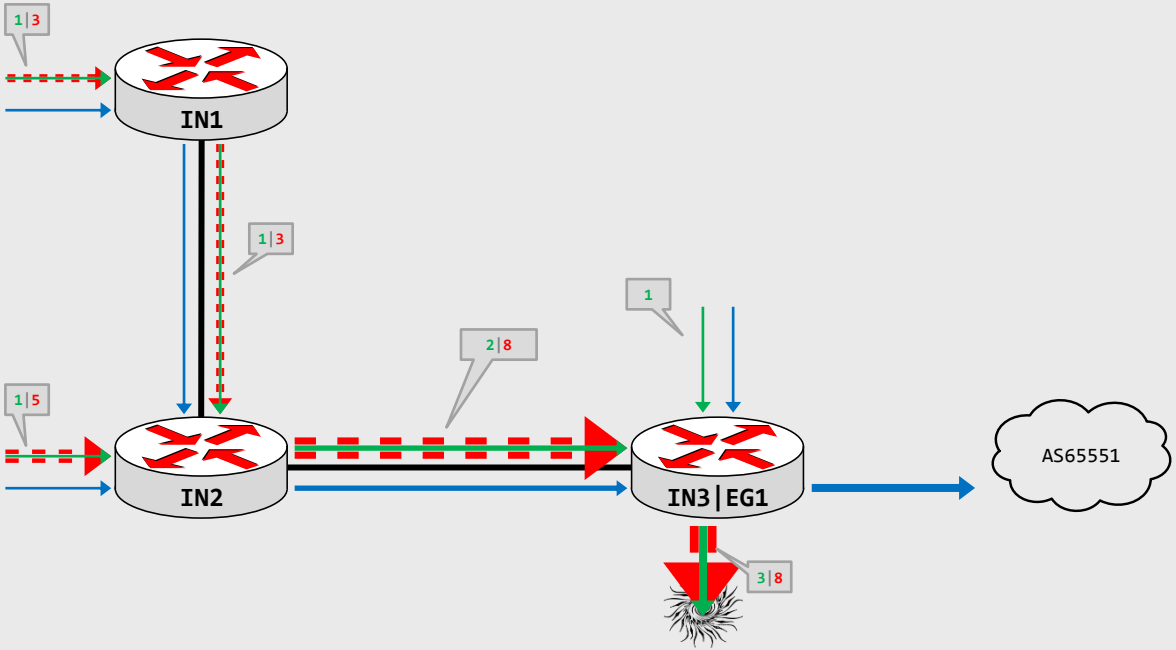
rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

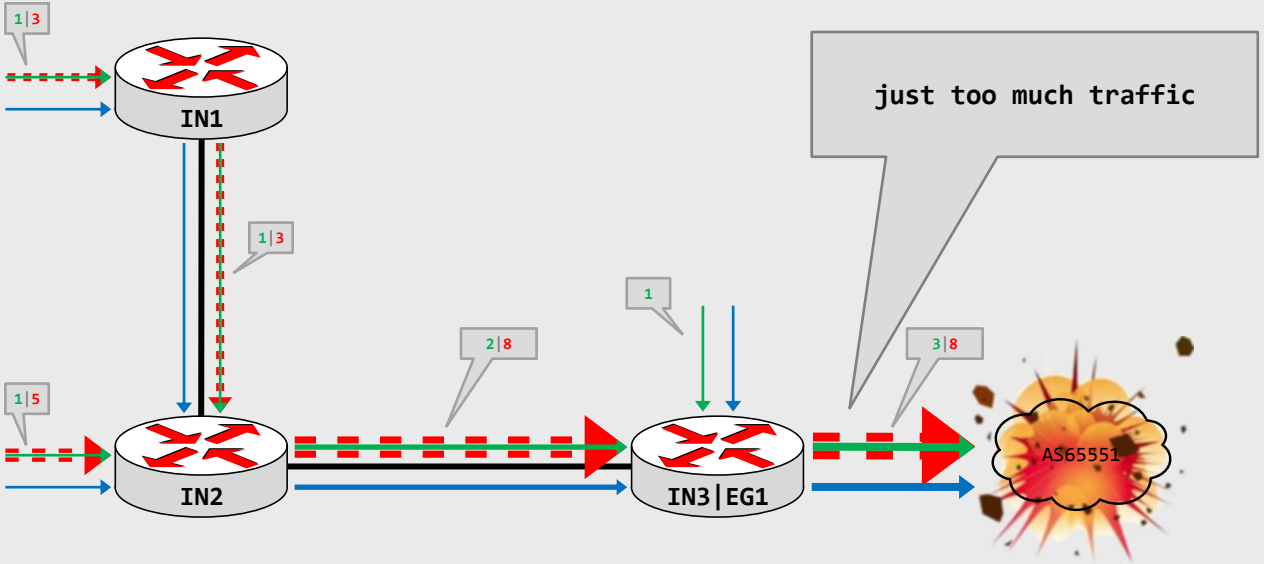


rtBH – rtsBH - rtsdCoS

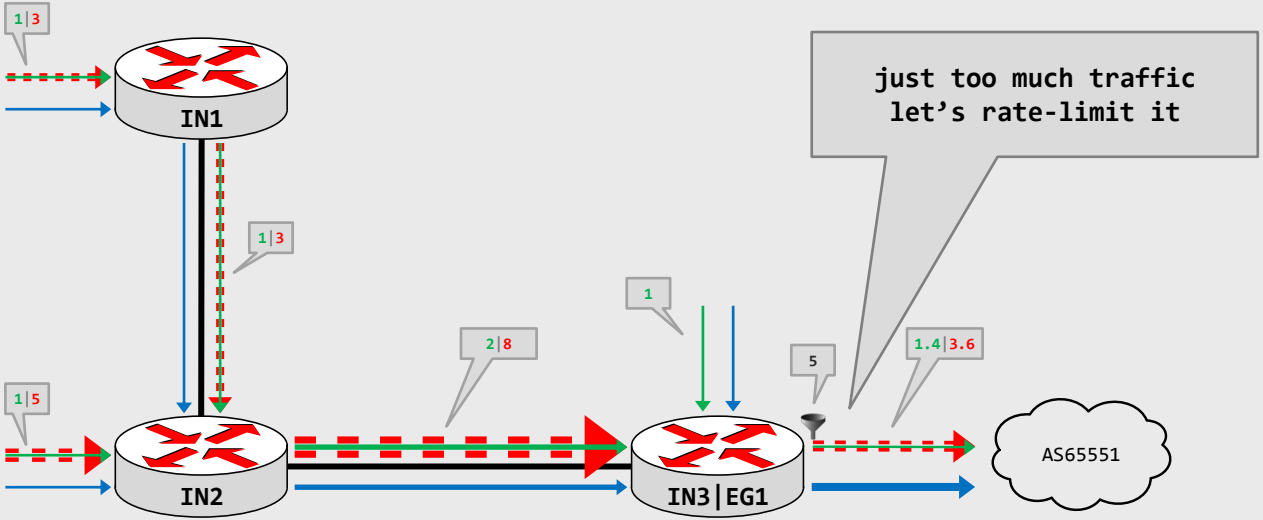


rtBH – rtsBH - rtsdCoS

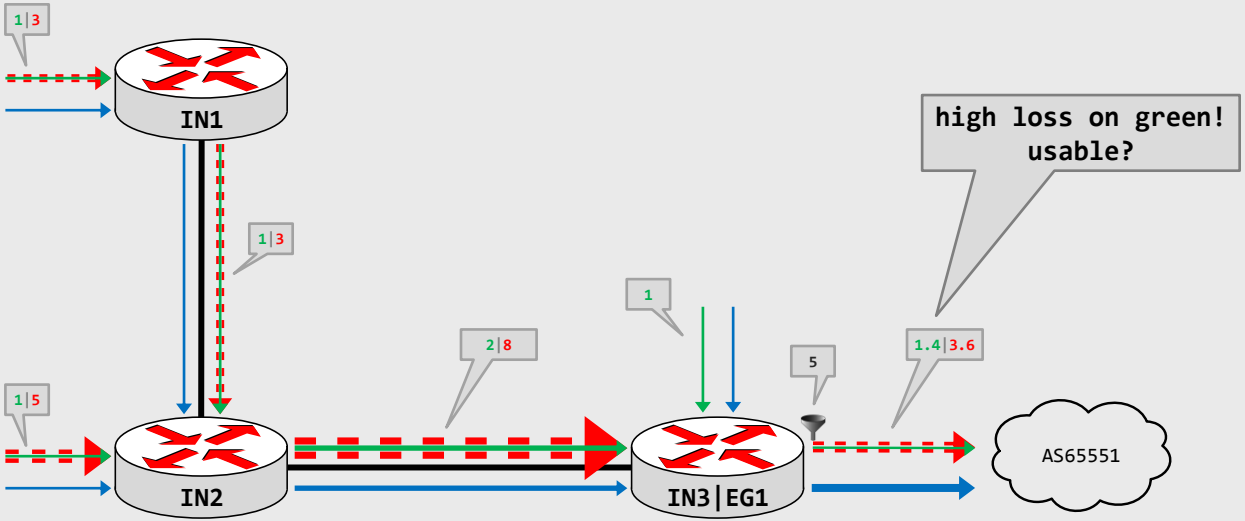
is less better?



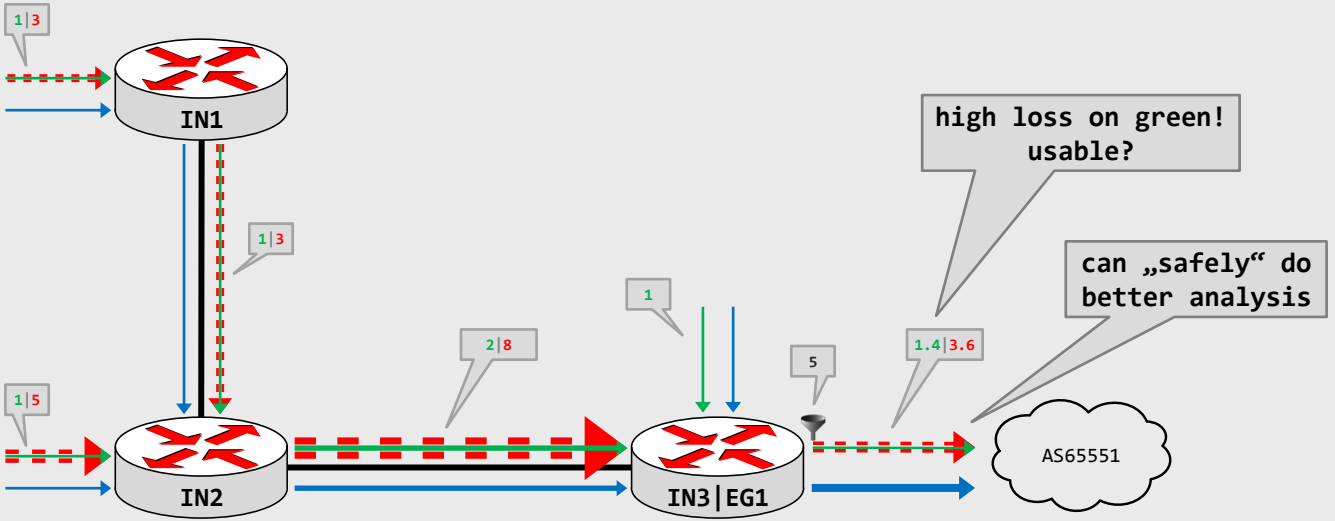
rtBH – rtsBH - rtsdCoS



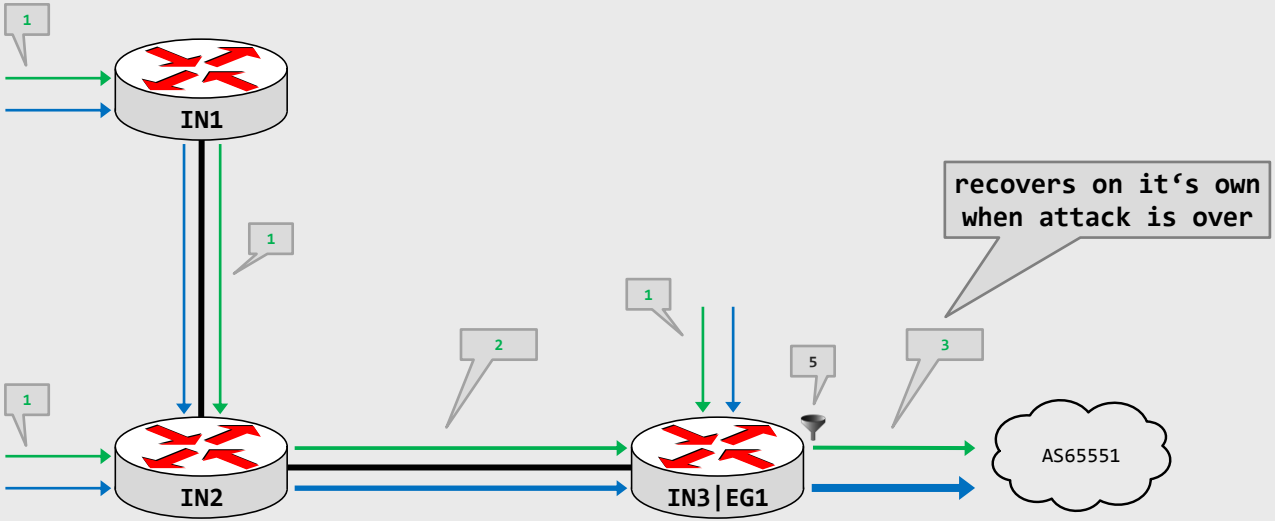
rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS



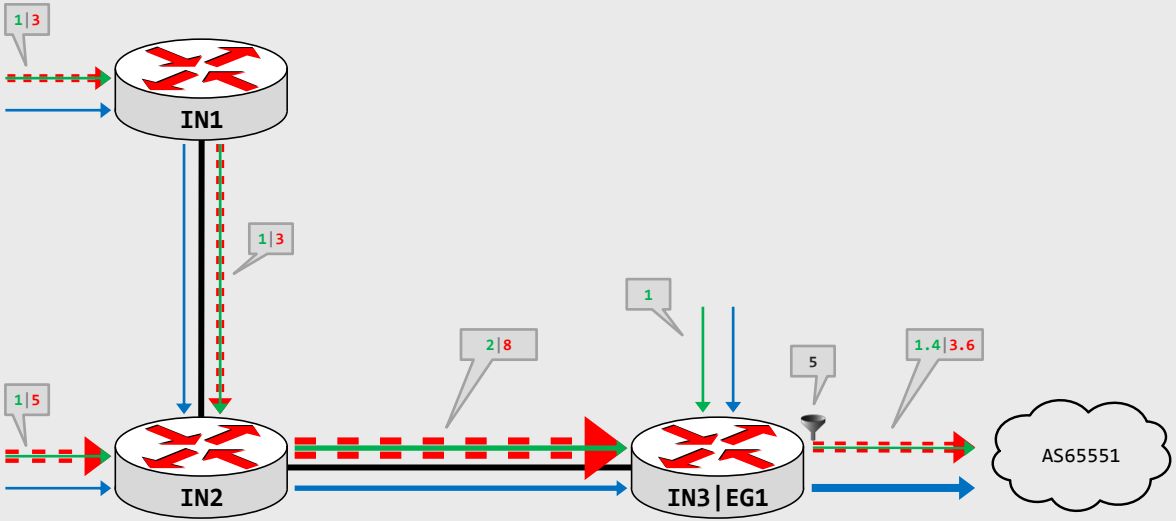
rtBH – rtsBH - rtsdCoS



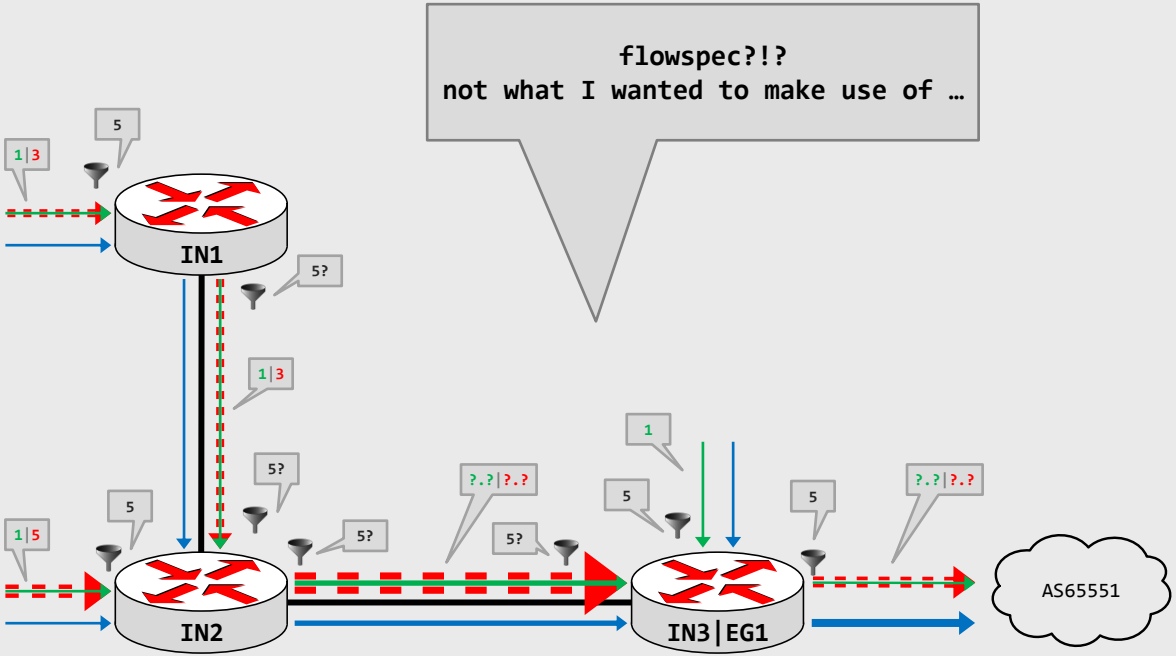
rtBH – rtsBH - rtsdCoS

reusing the information

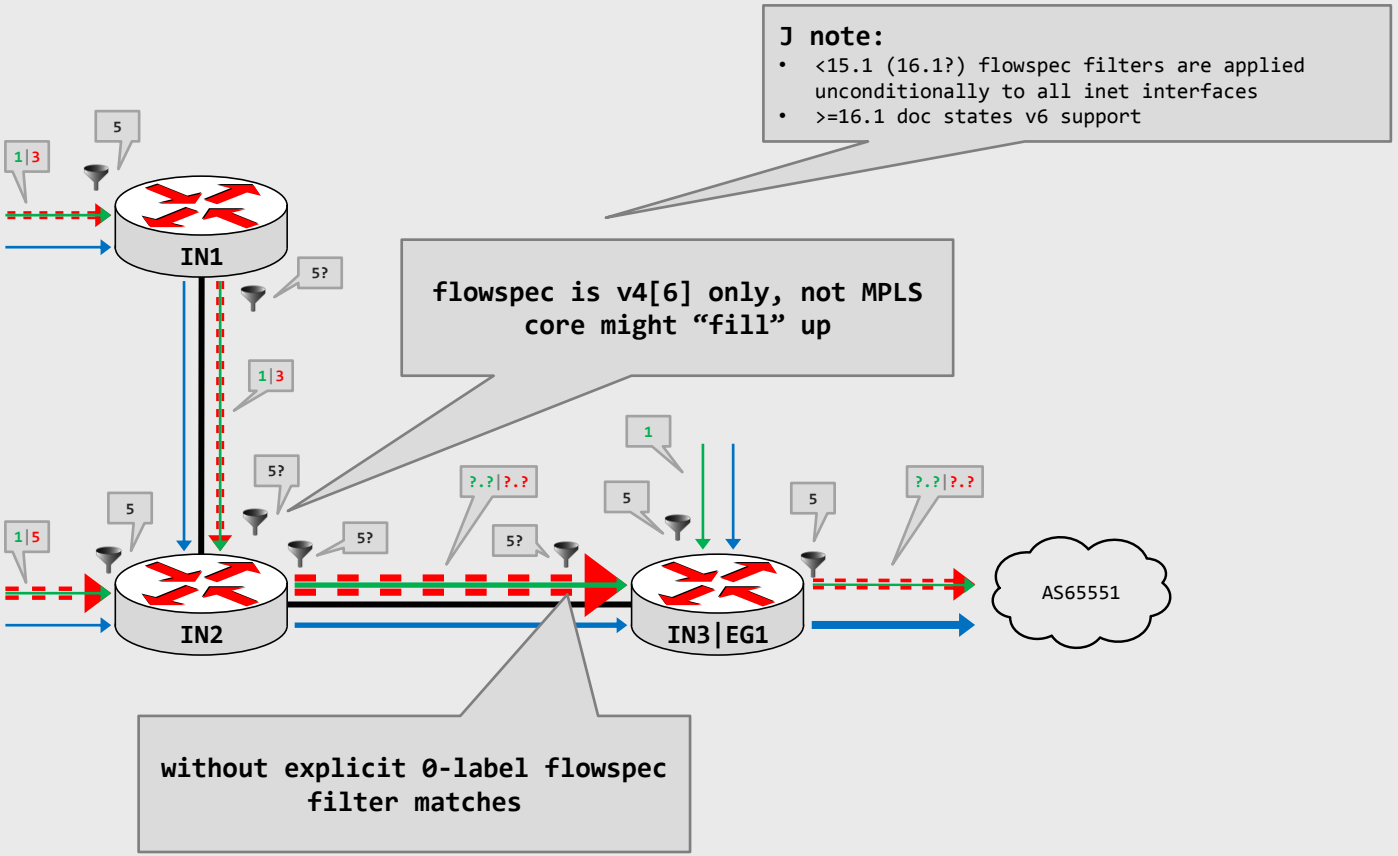
used on egress for rate-limiting on ingress



rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

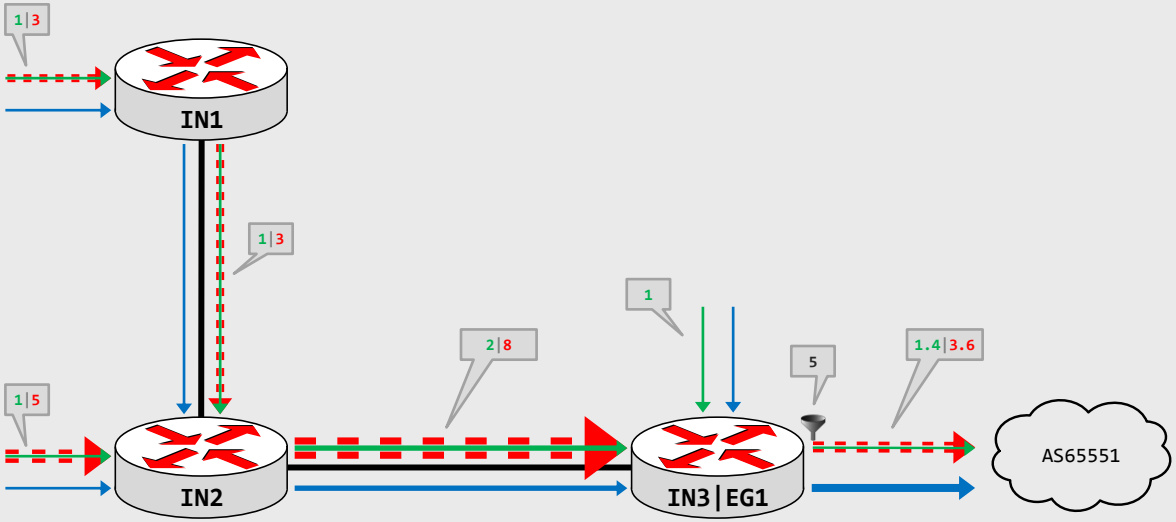


rtBH – rtsBH - rtsdCoS

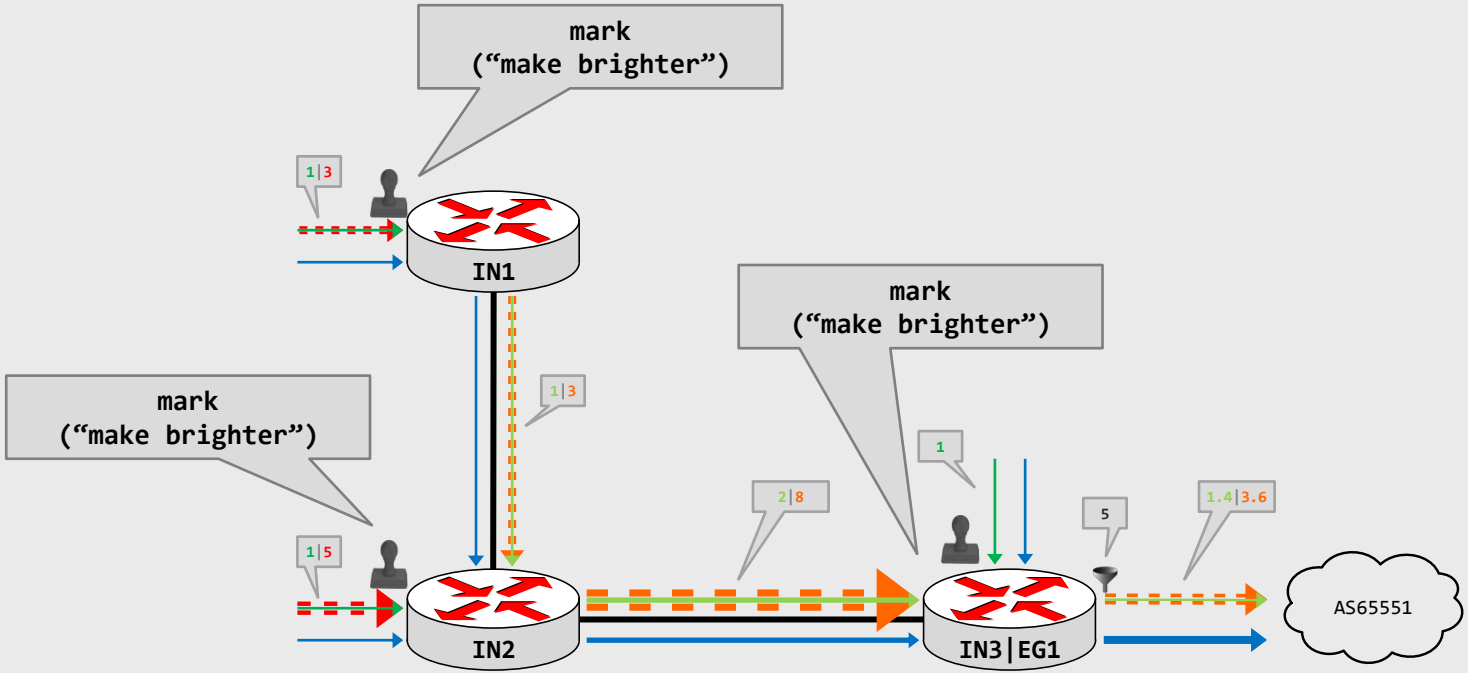
reusing the information

used on egress for rate-limiting on ingress

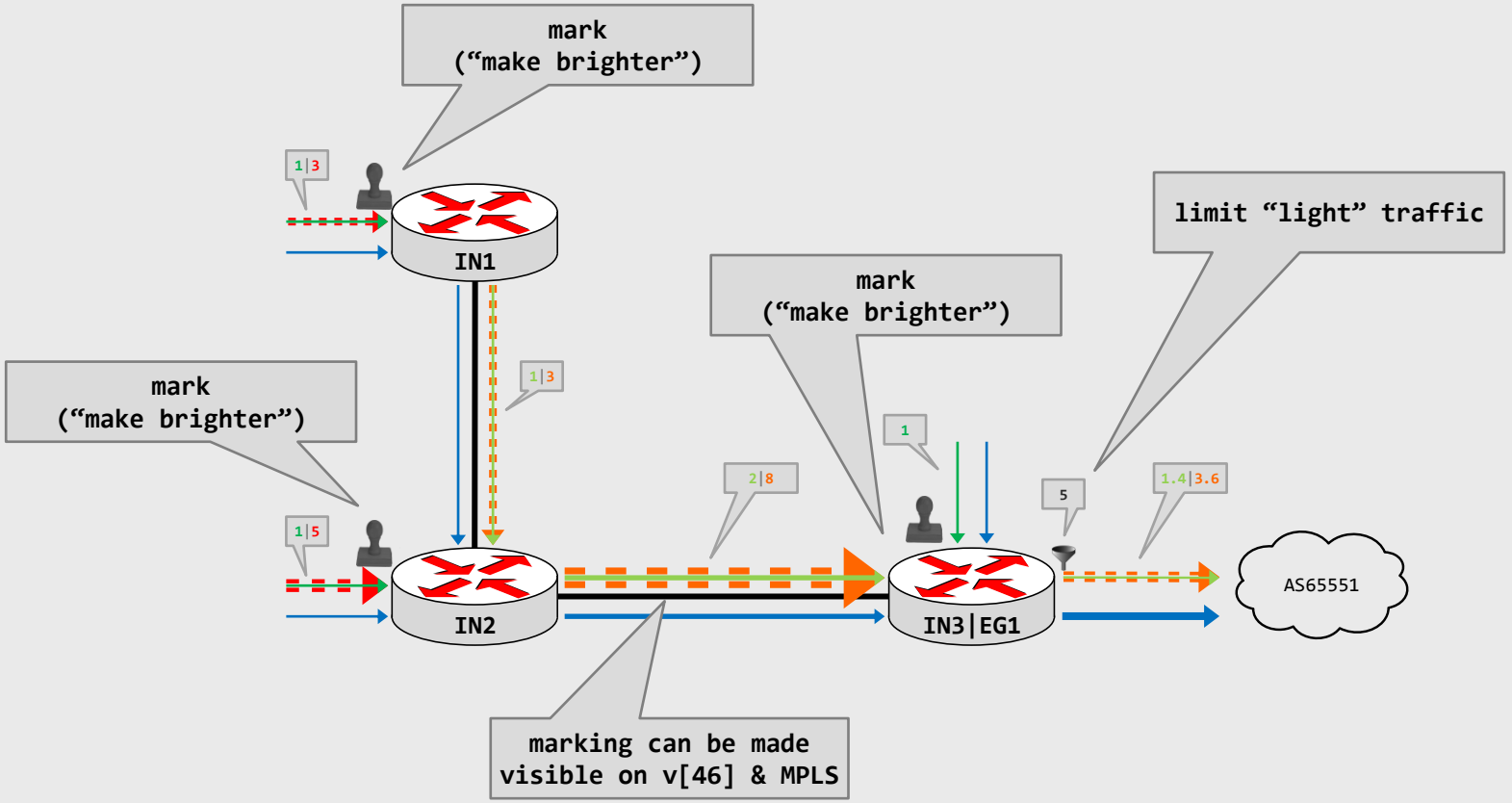
no flowspec



rtBH – rtsBH - rtsdCoS

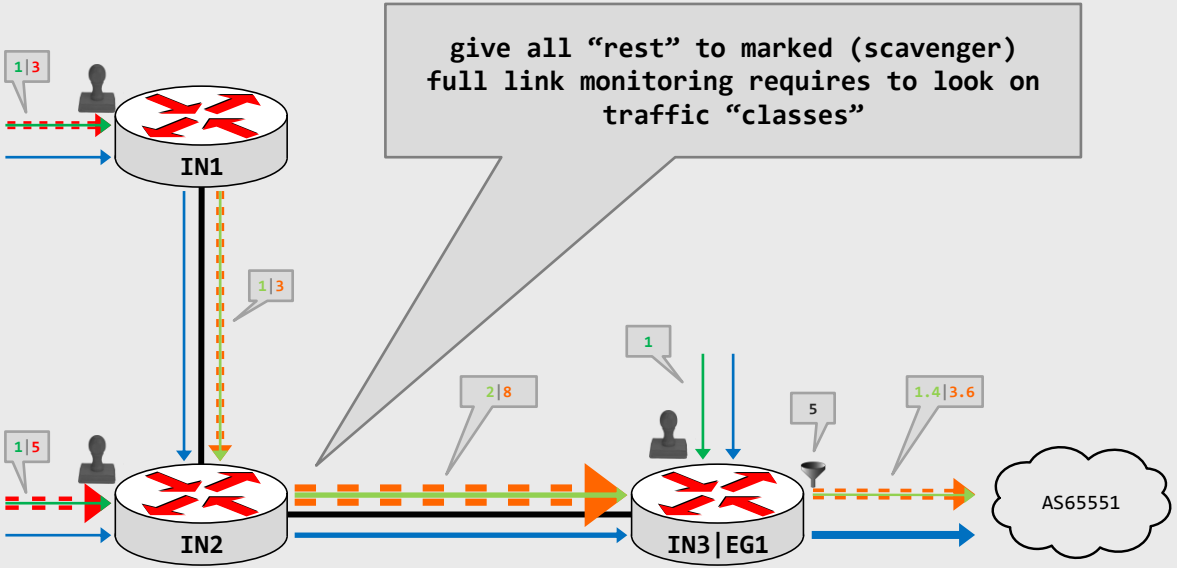


rtBH – rtsBH - rtsdCoS

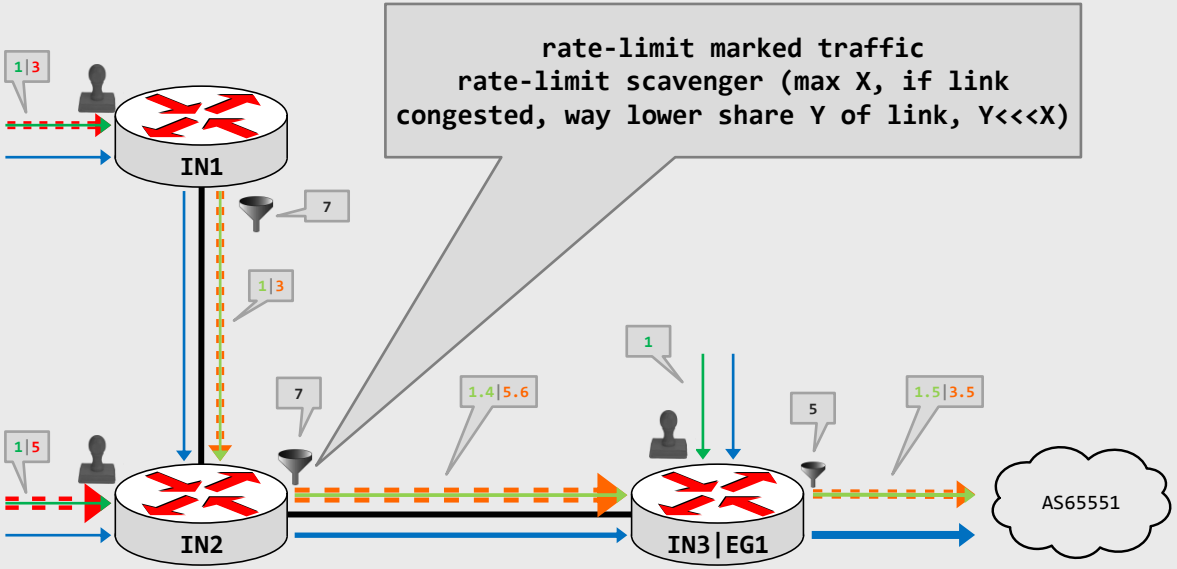


rtBH – rtsBH - rtsdCoS

make use of it in the core

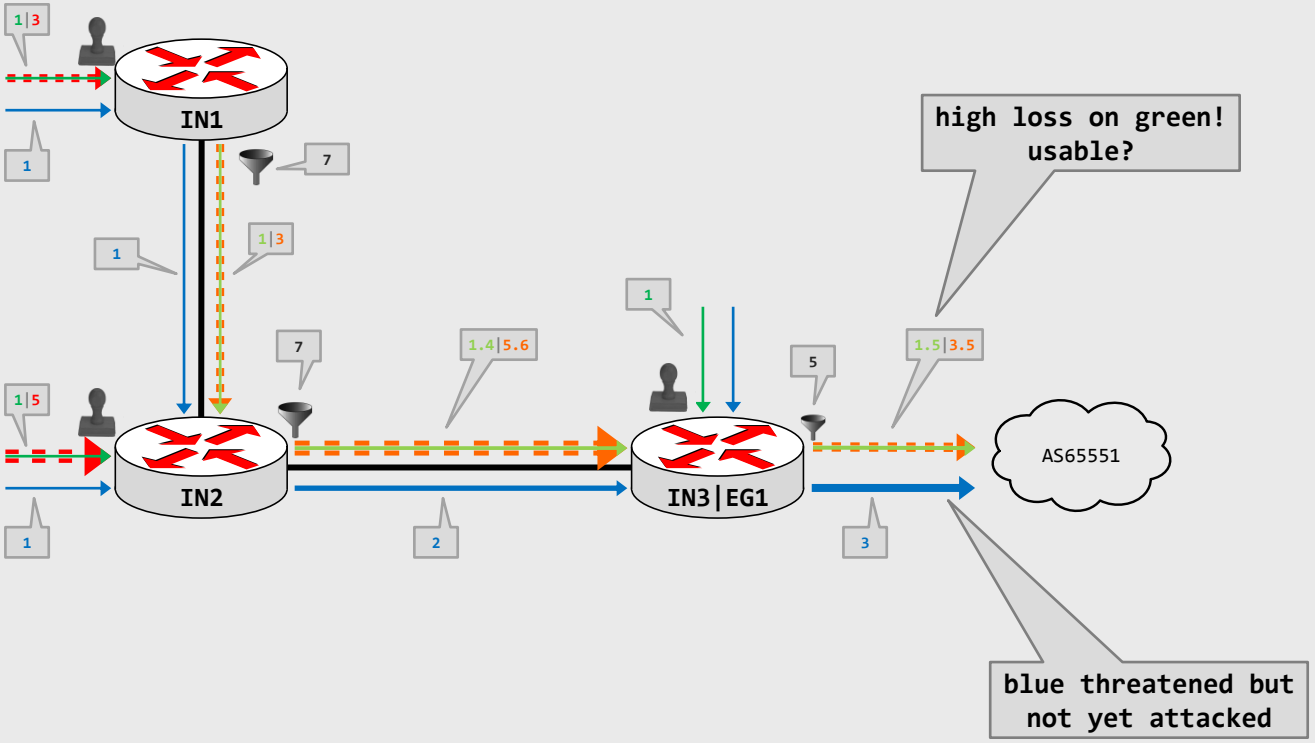


rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

don’t fool yourself
don’t get fooled by others

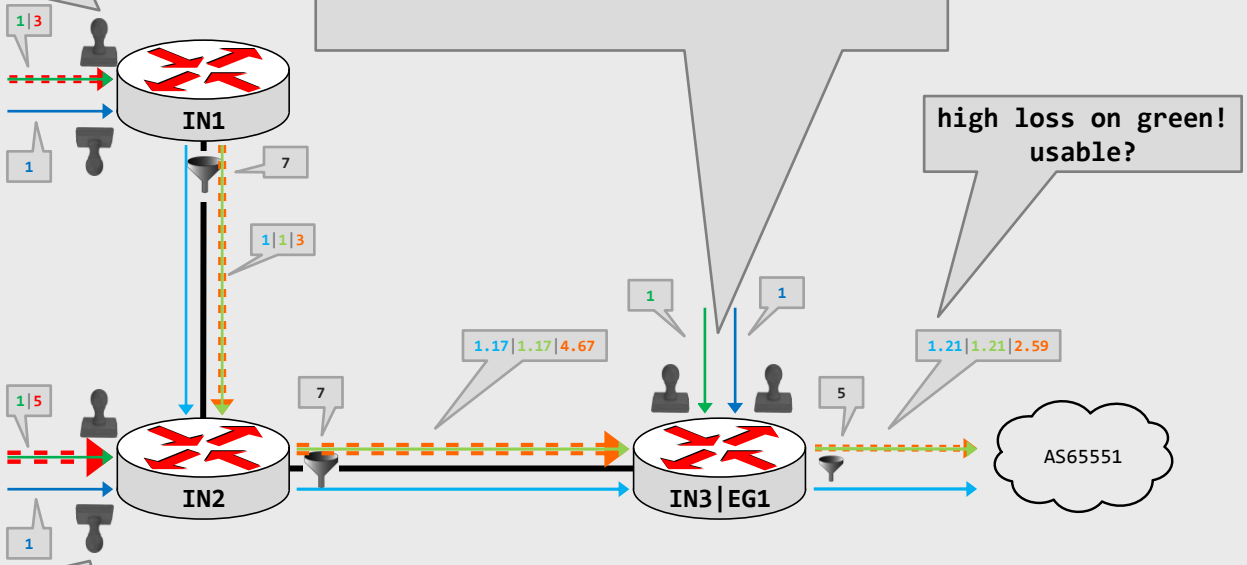


rtBH – rtsBH - rtsdCoS

marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

high loss on green! usable?



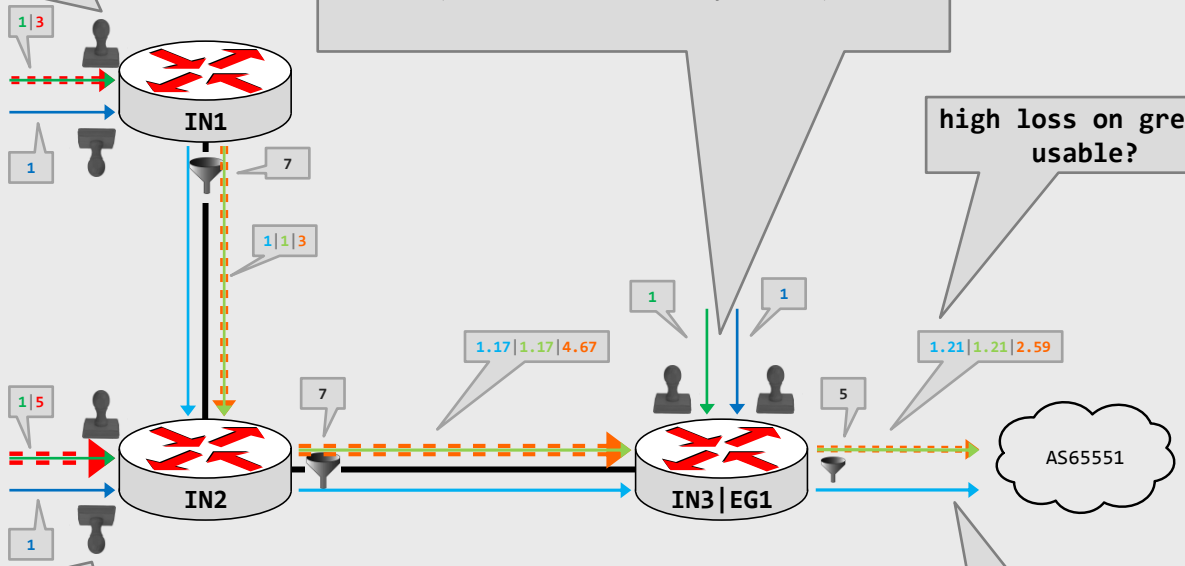
marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

rtBH – rtsBH - rtsdCoS

marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

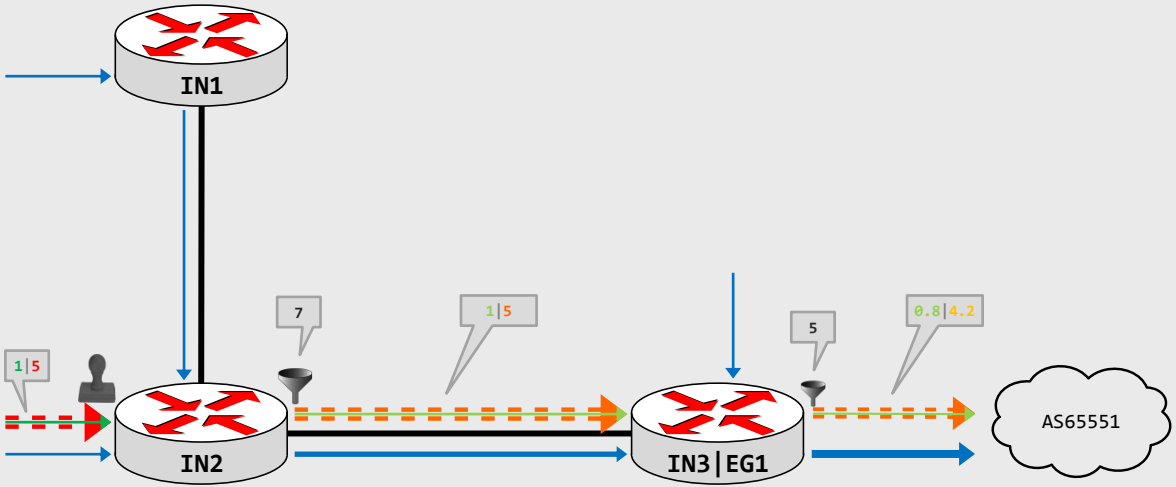
high loss on green! usable?



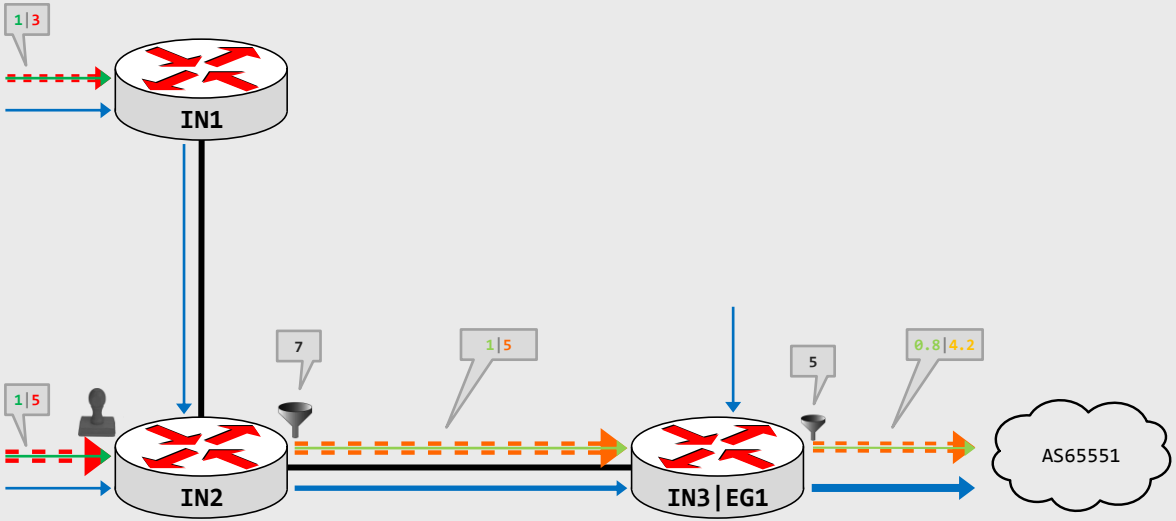
marking puts ALL traffic (towards green AND blue) most likely in ONE class
(could use more, but very limit #)

now loss on blue as well even not attacked (yet)

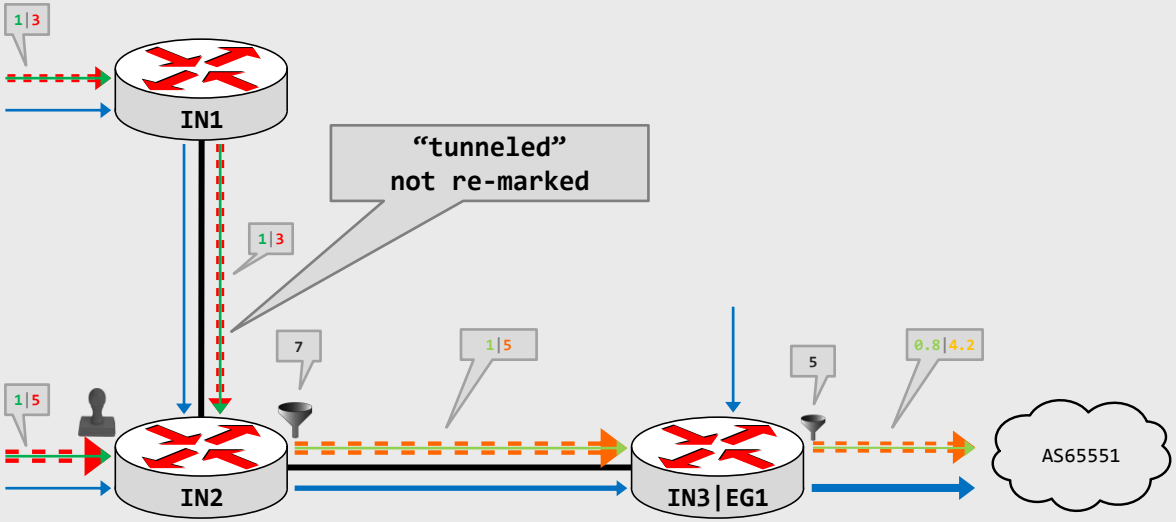
works selective as well



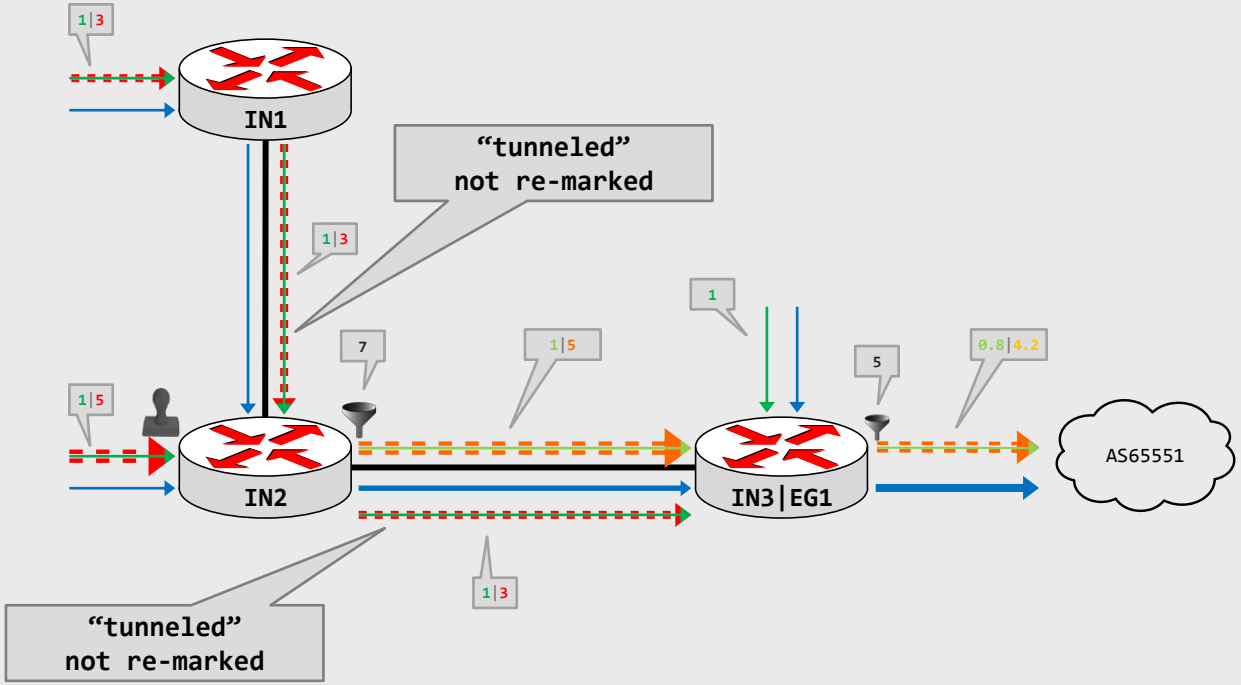
rtBH – rtsBH - rtsdCoS



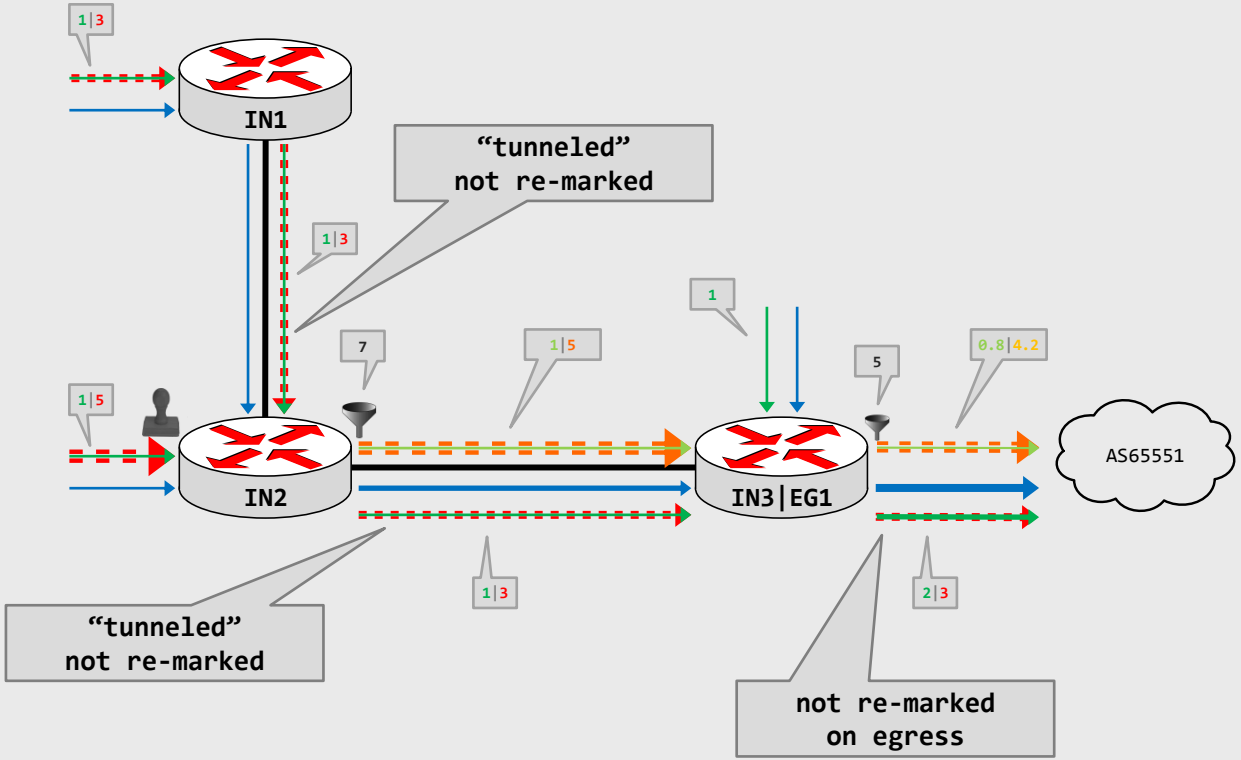
rtBH – rtsBH - rtsdCoS



rtBH - rtsBH - rtsdCoS

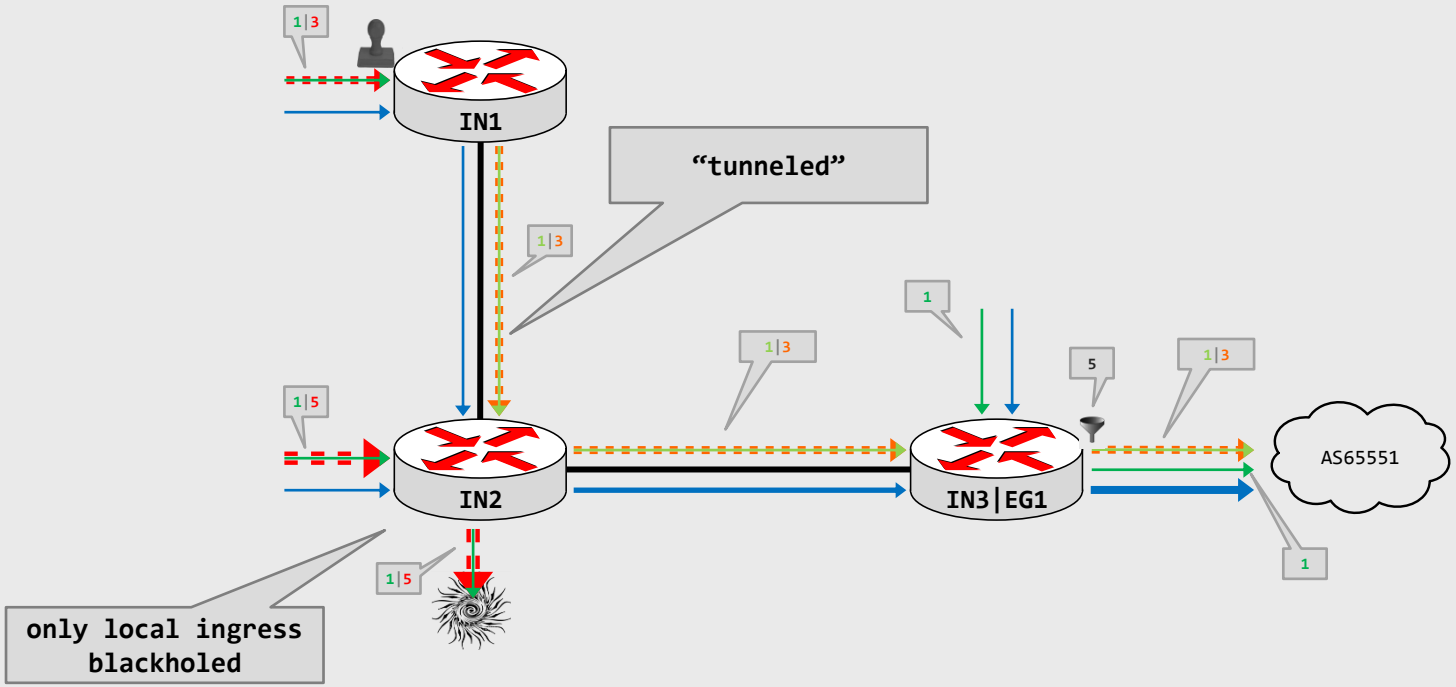


rtBH – rtsBH - rtsdCoS



rtBH – rtsBH - rtsdCoS

combined selective methods



rtBH - rtsBH - rtsdCoS

tech notes

rt

remote trigger

(signaling)

- rt = don‘t log in on every router on your own

- in your own network it could be even scripting, NETCONF, …

- BGP mostly used to inject into your / other’s network
- communities, next-hop, …
- „normal“ BGP session or dedicated BGP session (right NH?)

rtBH

remote triggered
blackholing

- modify the next hop (NH) of the prefix in iBGP to a NH pointing to /dev/null (Null, discard, dsc, …)
- multiple different BH NHs in own network (or per customer) could be useful e.g. to distinguish parallel attacks and see “end-of-attack” in flow data
- pretty straight forward to implement (still room for fun)

rtsBH

remote triggered
selective
blackholing

- **Job's theorem:**

  Most prefixes/content have a geographical significance which decreases as distance between the sender and receiver increases.

- **doesn't need to be regional … might even make sense with two routers!**

- **Job's theorem:**

  Most prefixes/content have a geographical significance which decreases as distance between the sender and receiver increases.

- **doesn't need to be regional … might even make sense with two routers!**

- **Job's implementation uses distance as criteria and normal iBGP mesh for signaling**

- **we implemented a slightly different logic**
  - BH of ingress traffic in selected PoPs
  - BH of ingress traffic on any other PE (the egress PE must never BH), outside the PoP, outside the city, outside the country or outside the continent and whitelisting of selected PoP, city, country or continent
  - ➢ complex, but allows tweaking

- **but we had our challenge(s) with it:**
  - we have route reflectors in the forwarding path serving different regions
  - fix the design :-(or (try to) do NH rewrite in the FIB and keep it untouched in the RIB … a bit ugly
  - we don’t use “continue” in route-maps ;-)
  - just make longer and more complex route-maps or use continue

- **but we had our challenge(s) with it:**
  - we have route reflectors in the forwarding path serving different regions
  - fix the design :-(or (try to) do NH rewrite in the FIB and keep it untouched in the RIB … a bit ugly
  - we don’t use “continue” in route-maps ;-)
  - just make longer and more complex route-maps or use continue

- **we ended with dedicated route-reflectors just for sBH**
  - simple, separated from normal iBGP policies, flexible, easy to extend/change logic, everything in once place, log, inject, …
  - logical-systems on J, any reasonable BGP implementation should be ok

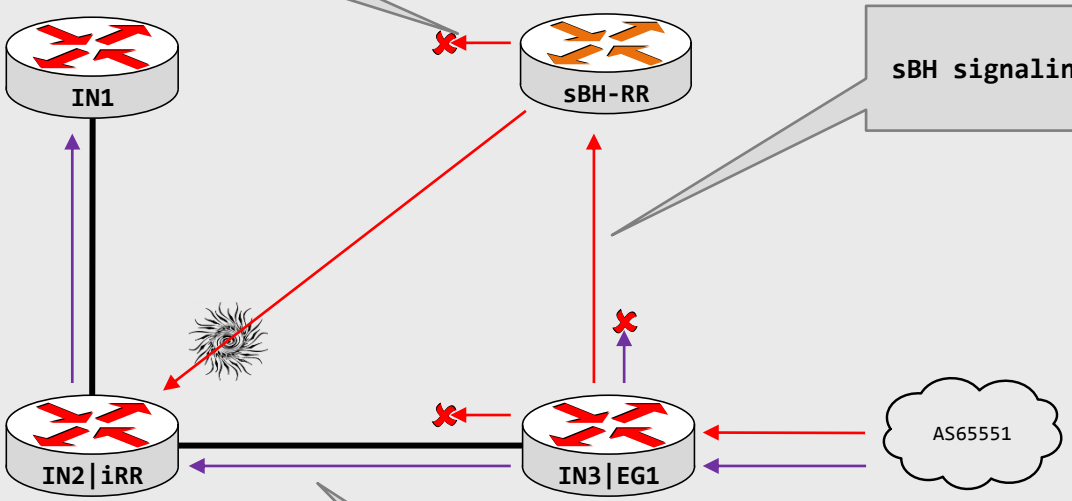
- **but we had our challenge(s) with it:**
  - we have route reflectors in the forwarding path serving different regions
  - fix the design :-(or (try to) do NH rewrite in the FIB and keep it untouched in the RIB … a bit ugly
  - we don't use "continue" in route-maps ;-)
  - just make longer and more complex route-maps or use continue

- **we ended with dedicated route-reflectors just for sBH**
  - simple, separated from normal iBGP policies, flexible, easy to extend/change logic, everything in once place, log, inject, …
  - logical-systems on J, any reasonable BGP implementation should be ok

- **validated sBH announcements from customers are**
  - never propagated through the normal iBGP mesh; only to dedicated sBH RR

- **the special sBH RRs do all the magic**
  - know where the announcement comes from, know what communities
  - know what outside PoP / city / country / continent means for each client
  - signal (no-adv) sBH prefix only to PEs which should see it or signal (no-adv) sBH prefix to all PEs and just set BH-NH to PEs to do BH

don‘t signal (more specific) sBH prefix to PEs not supposed to BH (req. less specific)
or
signal original NH



rtBH – rtsBH - rtsdCoS

some configuration snippets

blackhole everywhere except: on the PE you are connected to
blackhole everywhere except: PE connected and all other PEs on/in this site/PoP
blackhole everywhere except: PE connected and all other PEs in the city
blackhole everywhere except: PE connected and all other PEs in this country
blackhole everywhere except: PE connected and all other PEs on this continent
blackhole only in the explicitly (286:62xx) listed PoPs - whitelisting will still be considered (but hardly makes sense ;-)

WL continent	WL country	WL city	WL PoP	BL PoP
286:6991 North America	286:6924 United States	286:6747 Ashburn	286:6069 ahbn-s1	286:6269 ahbn-s1
		286:6748 Chicago	286:6070 chg-s1	286:6270 chg-s1
		286:6749 Dallas	286:6071 dlls-s1	286:6271 dlls-s1
		286:6750 Los Angeles	286:6072 lags-s1	286:6272 lags-s1
		286:6751 Miami	286:6073 miaf-s1	286:6273 miaf-s1
		286:6752 New York	286:6074 nyk-s1	286:6274 nyk-s1
			286:6075 nyk-s2	286:6275 nyk-s2
		286:6753 San Jose	286:6076 sjca-s1	286:6276 sjca-s1

```
community sBH-marked members 286:28667;
```

```
/*
   PE->normal RR iBGP export
   export [ … r-sBH-routes … ]
*/

policy-statement r-sBH-routes {
    term marked {
        from community sBH-marked;
        then discard;
    }
}
```

```
/*
   PE->sBH RR iBGP export
   if you want to signal prefix from there with “original” to all non-BHing
   PEs, then eventually add here “then next-hop self” (YMMV)
*/

policy-statement ra-only-sBH-routes {
    term marked {
        from community sBH-marked;
        then accept;
    }
    then discard;
}
```

286:28667 is set internally if it‘s a validated (e.g. prefix + as-origin + as-path filter and rPKI validated ;-) announcement from a customer with one of the sBH communities set

```
/* import on sBH RR from client
*/


policy-statement m-sBH-from-miaf-s1 {
    term all-not-PE {
        from community sBH-BL-all-not-PE;
        then {
            community add sBH-BL-ALL;
        }
    }
    term all-not-POP {
        from community sBH-BL-all-not-POP;
        then {
            community add sBH-BL-ALL;
            community add sBH-WL-pop-MIAF-S1;
        }
    }
    term all-not-CITY {
        from community sBH-BL-all-not-CITY;
        then {
            community add sBH-BL-ALL;
            community add sBH-WL-city-MIAF;
        }
    }
    term all-not-COUNTRY {
        from community sBH-BL-all-not-COUNTRY;
        then {
            community add sBH-BL-ALL;
            community add sBH-WL-country-US;
        }
    }
    term all-not-CONTINENT {
        from community sBH-BL-all-not-CONTINENT;
        then {
            community add sBH-BL-ALL;
            community add sBH-WL-continent-NORTH-AMERICA;
        }
    }
}
```

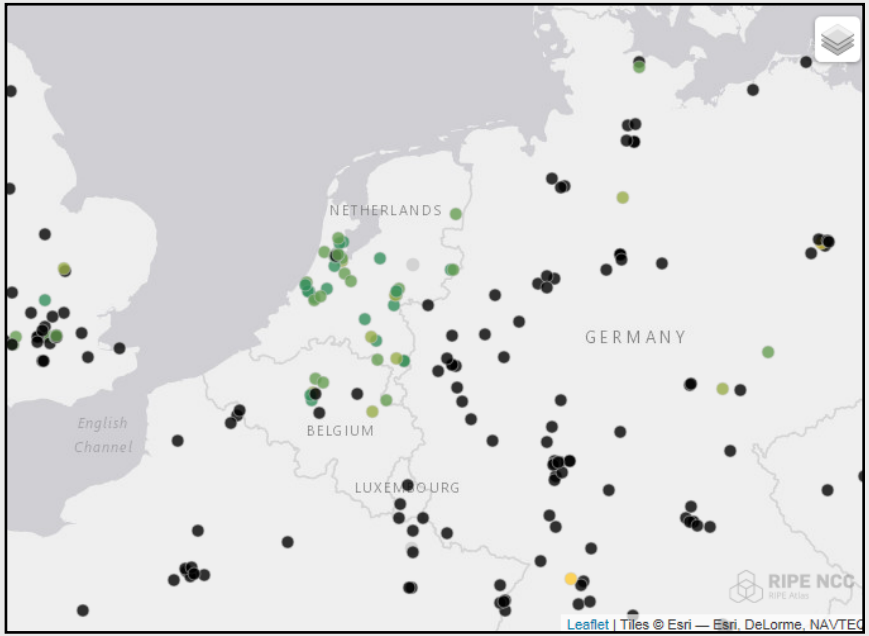
```
/* export on sBH RR to client
   signal only to PEs which should get it – for signal with original NH
   then add replace reject with accept and add no-advertise in policy
*/
policy-statement ra-sBH-to-miaf-s1 {
    term honor-whitelist {
        from community [ sBH-WL-pop-MIAF-S1
                         sBH-WL-country-US
                         sBH-WL-continent-NORTH-AMERICA
                         sBH-WL-city-MIAF ];
        then reject;
    }
    term accept-BL-v4 {
        from {
            family inet;
            protocol [ bgp static ];
            community [ sBH-BL-ALL sBH-BL-pop-MIAF-S1 ];
        }
        then {
            next-hop 134.222.87.254;
            community add no-advertise;
            local-preference 665;
            accept;
        }
    }
    term accept-BL-v6 {
         from {
             family inet6;
             protocol [ bgp static ];
             community [ sBH-BL-ALL sBH-BL-pop-MIAF-S1 ];
         }
         then {
             next-hop ::ffff:134.222.87.254;
             community add no-advertise;
             local-preference 665;
             accept;
         }
    }
    then reject;
}
```

regional different understanding

geo-location vs. geographical ingress / interconnect

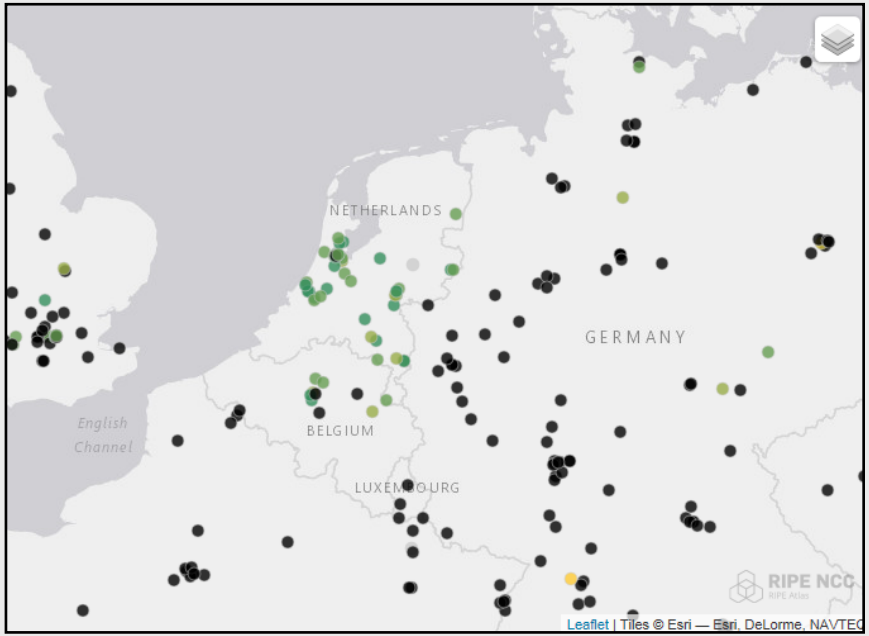
NL originated prefix

BH out-side country

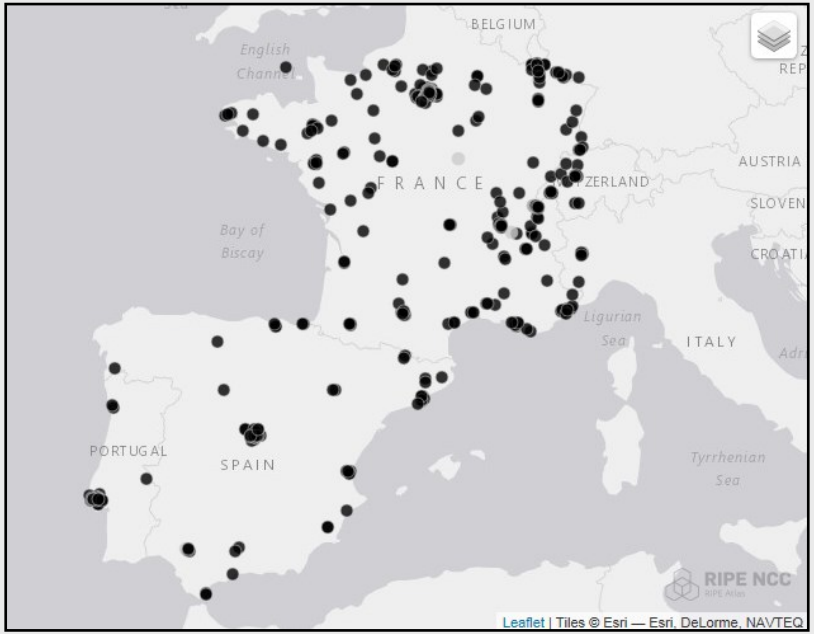


RIPE Atlas, ~500 probes around capital of country in 1000km radius; #6931736 (NL)
test IPs are no longer announced (and if, then for something different)

NL originated prefix
BH out-side country



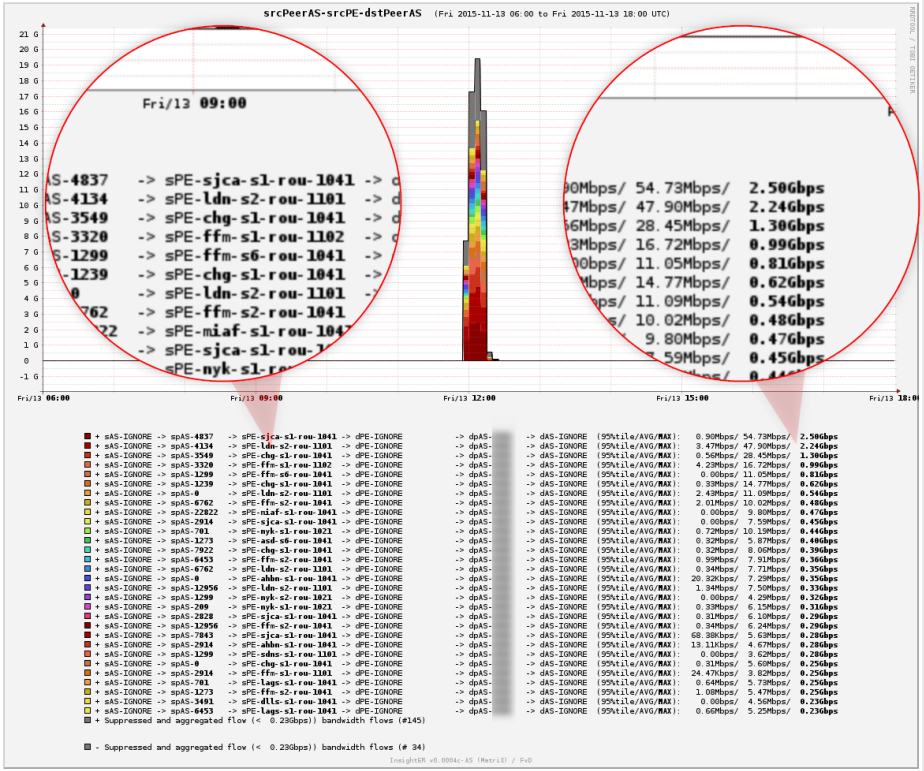
ES originated prefix
BH out-side country



RIPE Atlas, ~500 probes around capital of country in 1000km radius; #6931736 (NL), #6931735 (ES)
test IPs are no longer announced (and if, then for something different)

“you” don’t know what “I” do

(unless “I” share)



- without having insight where the flood came in, it‘s hard to make use of the „s“
- pure guessing if it’s not your network unless you get access to this information

rtsdCoS

remote triggered

selective

destination / dummy

Class of Service (QoS)

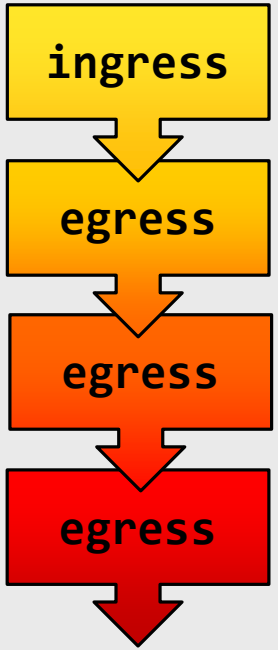
- CoS or QoS
- I’m not a friend … a lot of people aren’t a friend of it

- CoS or QoS
- I’m not a friend … a lot of people aren’t a friend of it
- but it’s there, used, requested and even though bigger pipes might be considered as the one and only and best solution, CoS does make sense (I’ve been told / I even used it to “hide” something for most user)

- CoS or QoS
- I’m not a friend … a lot of people aren’t a friend of it
- but it’s there, used, requested and even though bigger pipes might be considered as the one and only and best solution, CoS does make sense (I’ve been told / I even used it to “hide” something for most user)

- doing CoS is more than just NH rewriting (HW requirements, HW limitations, available queues, …)
- global scope requires “clean” CoS domain (proper classification on ingress everywhere, keep classification through the core, …) – but can be used as well just “locally”

simplified J CoS



classification (BA, MF, fixed, …) & policing
assign traffic to **forwarding-class** based on ingress interface, codepoints, header fields, … & policing

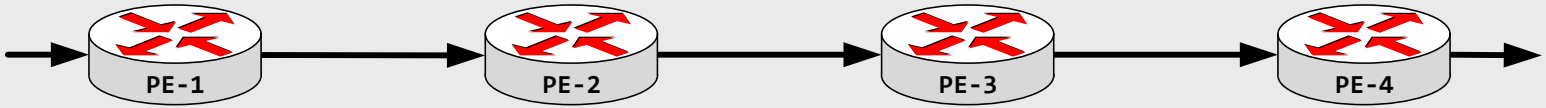
policing

queuing / shaping / scheduling
service queues associated with forwarding-classes

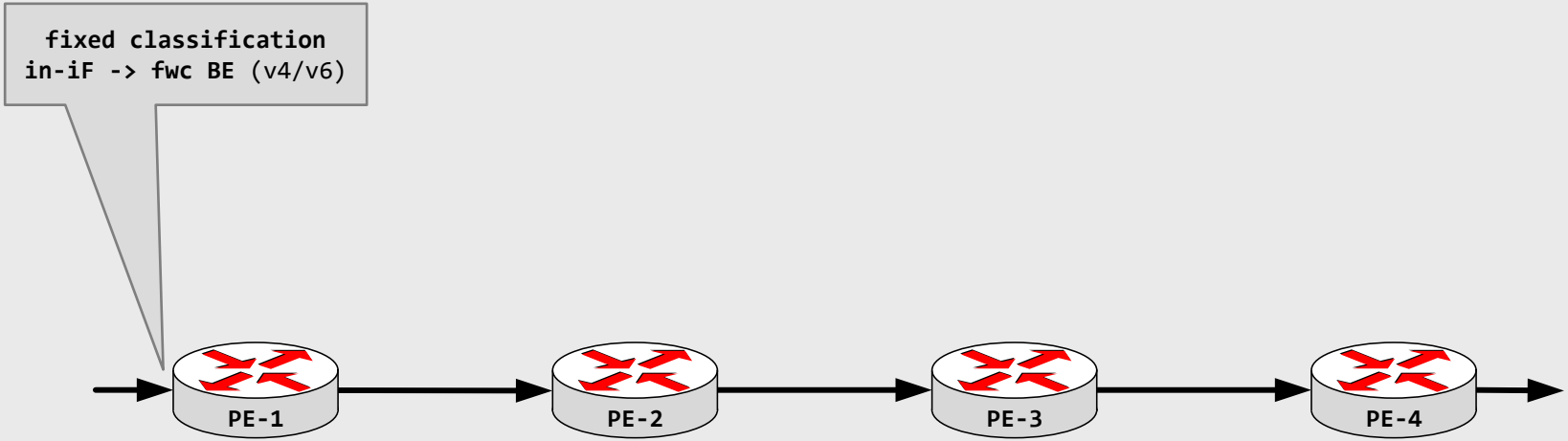
rewrite
rewrite of CoS fields in egress packet

simplified – check J's manuals on your hardware

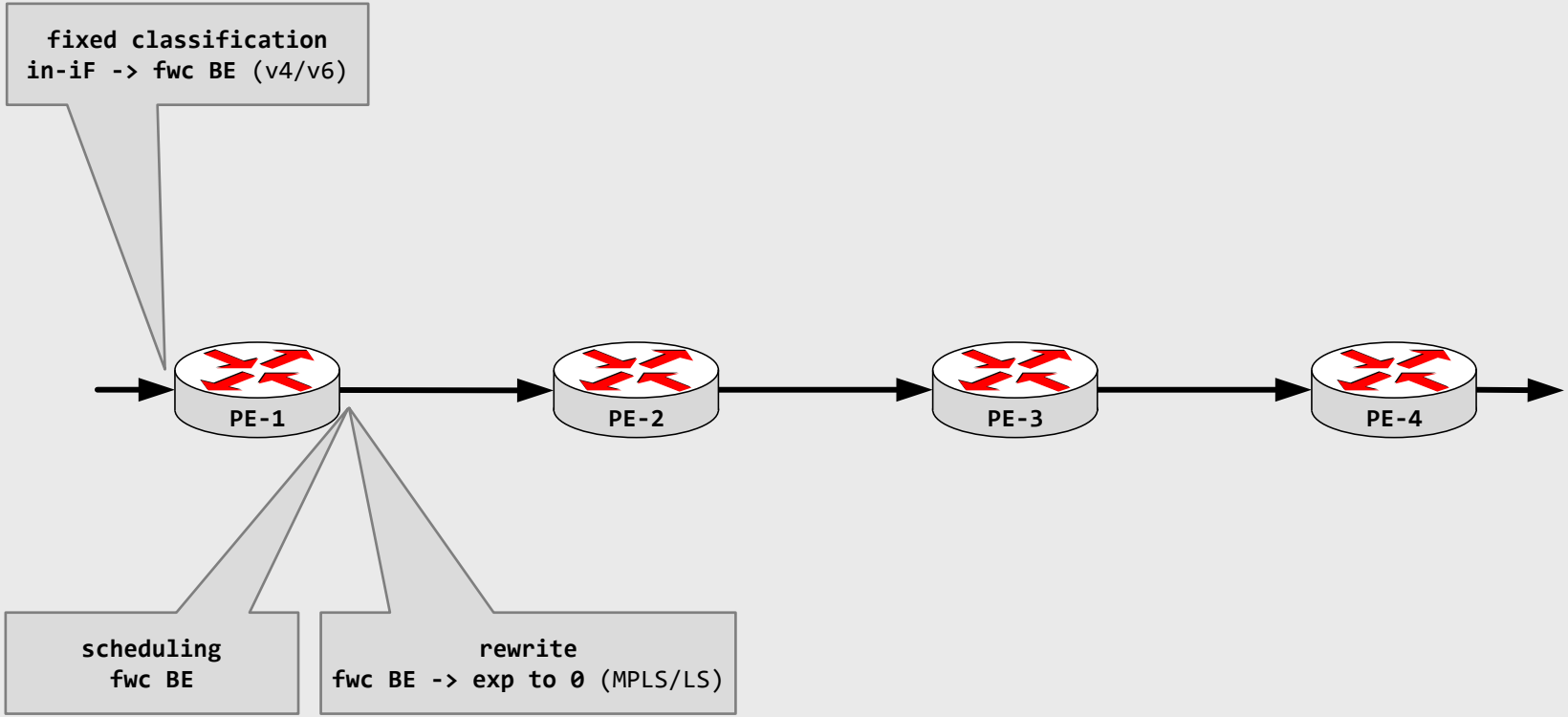
a normal transit packet’s CoS life



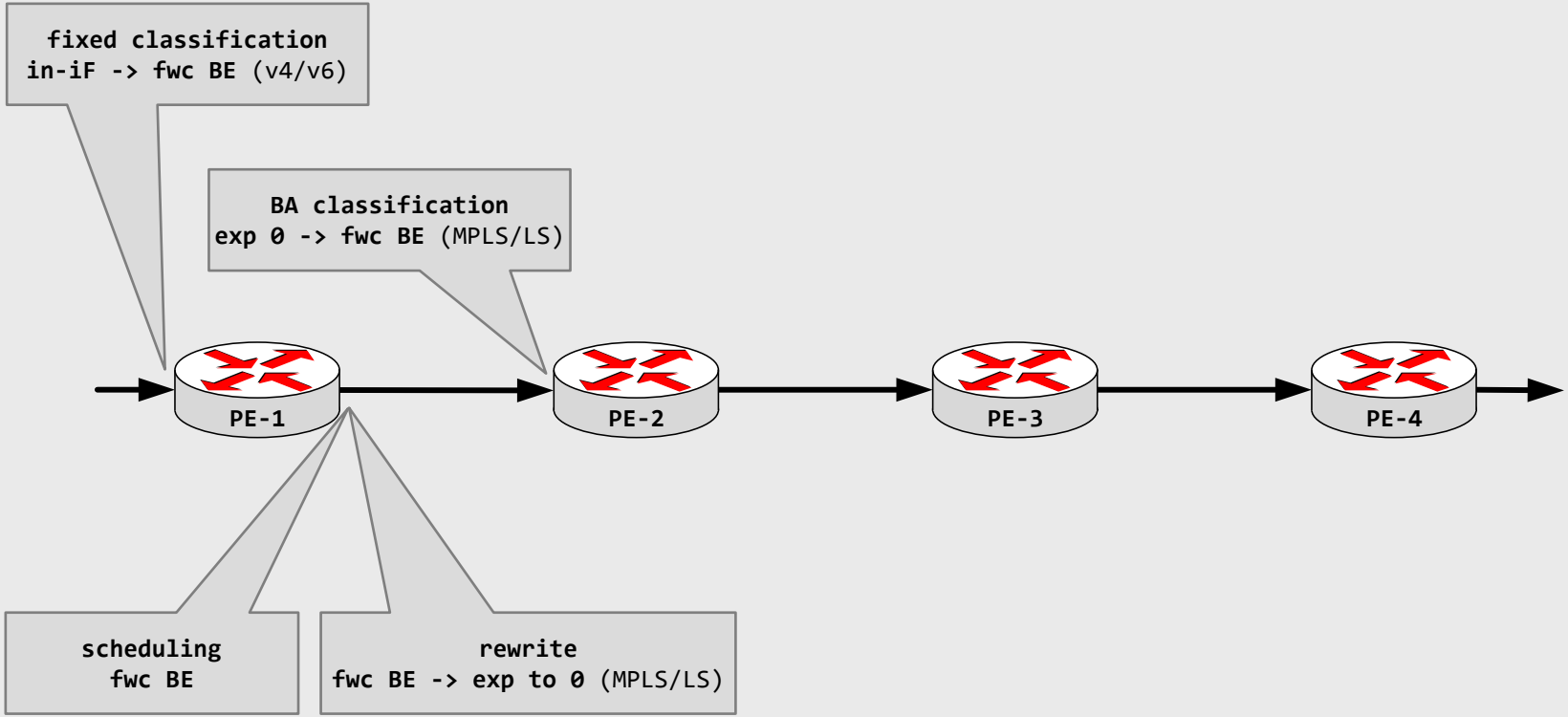
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



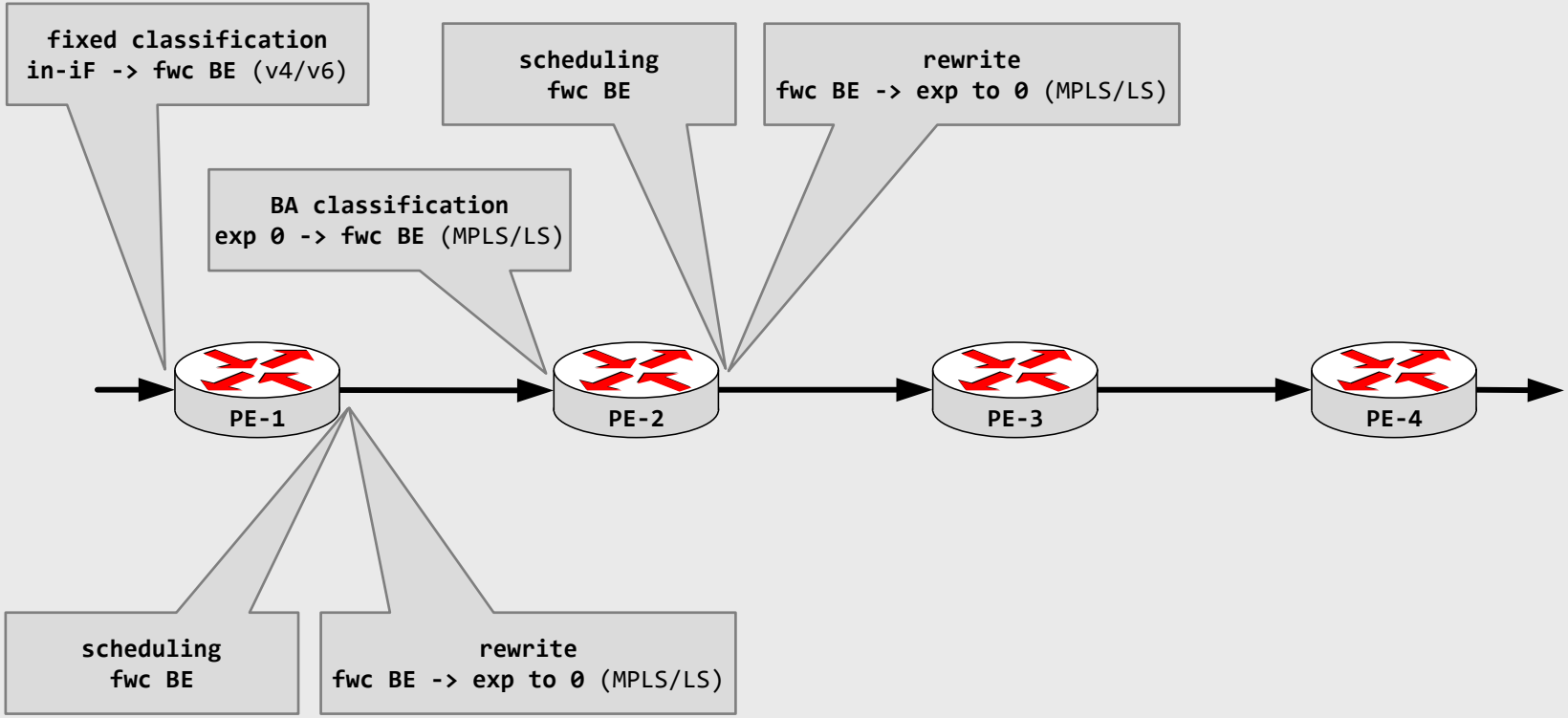
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



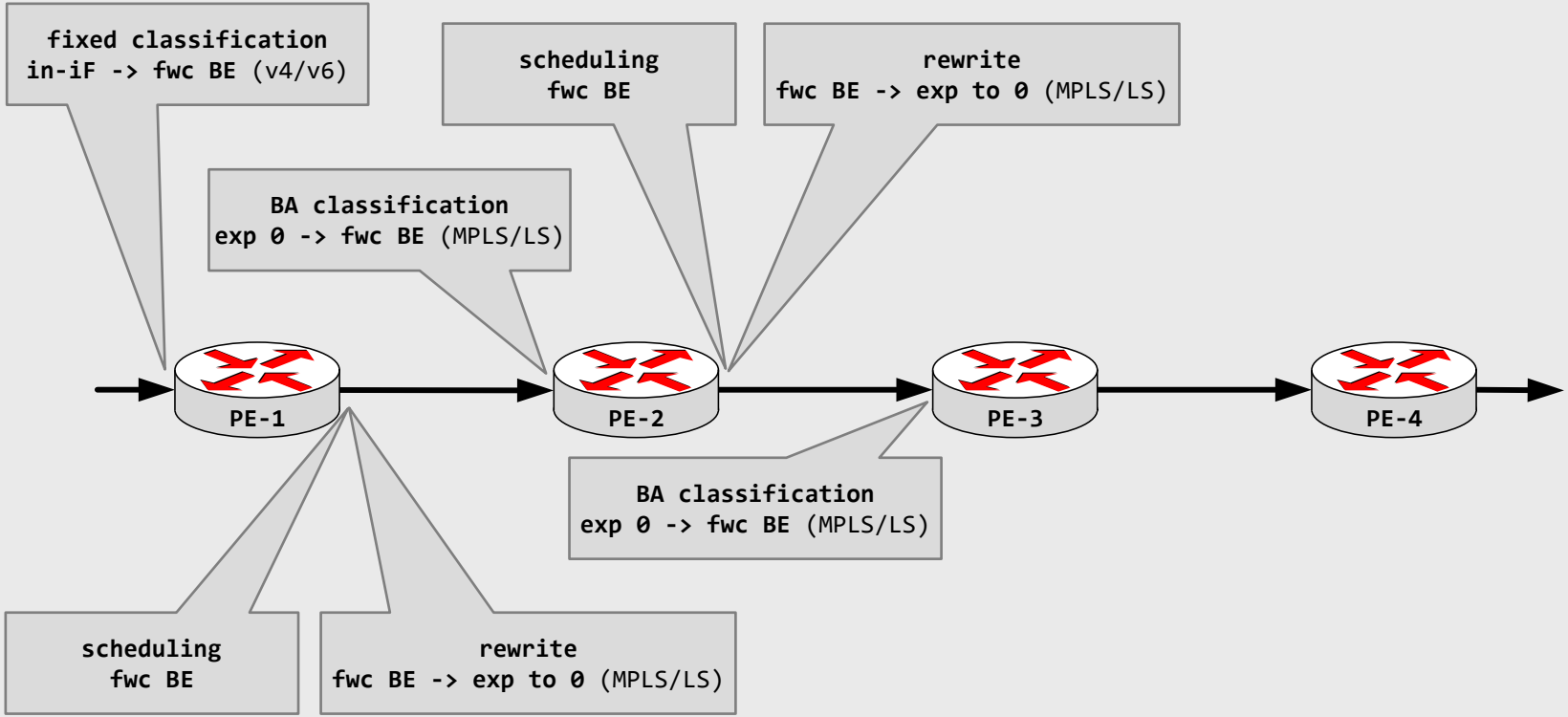
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



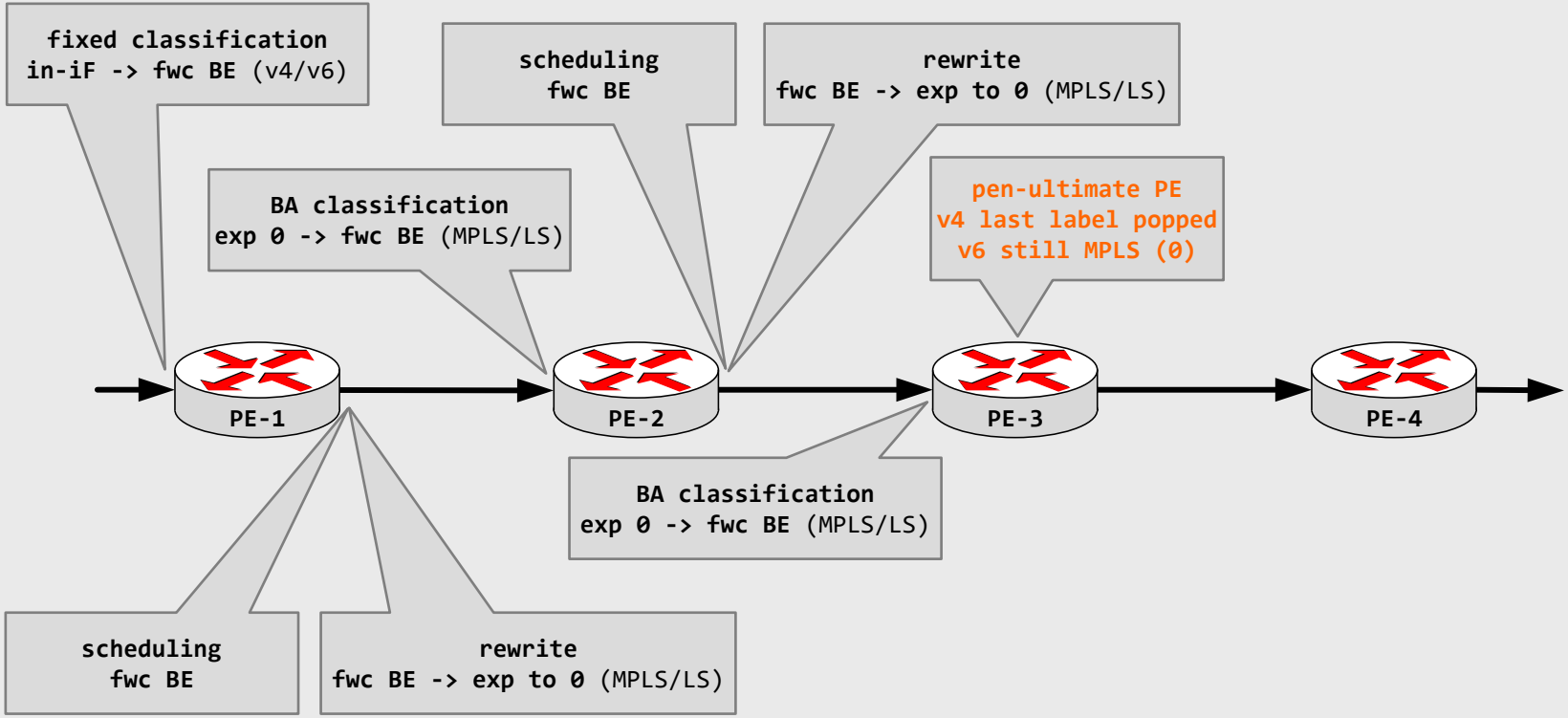
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



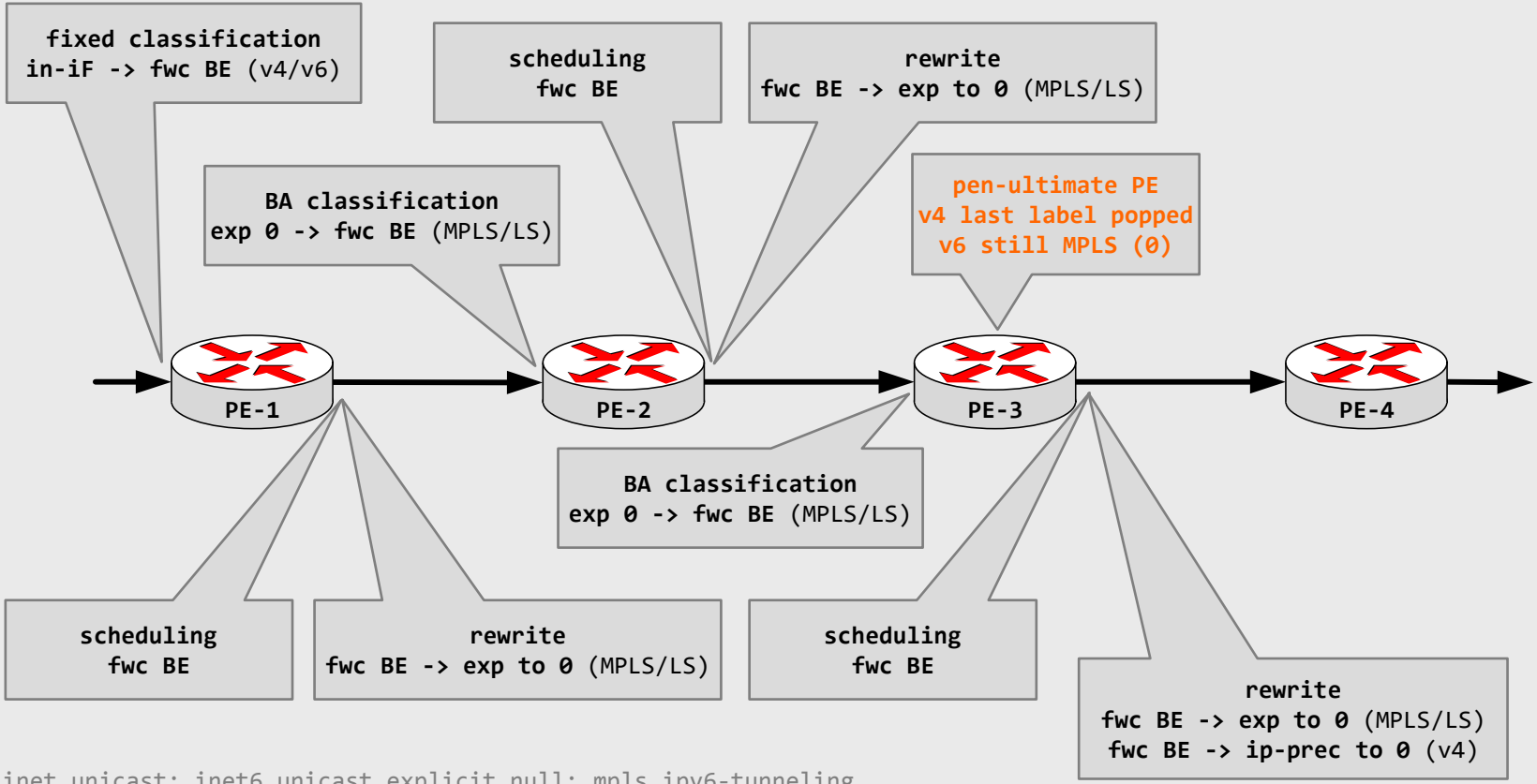
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



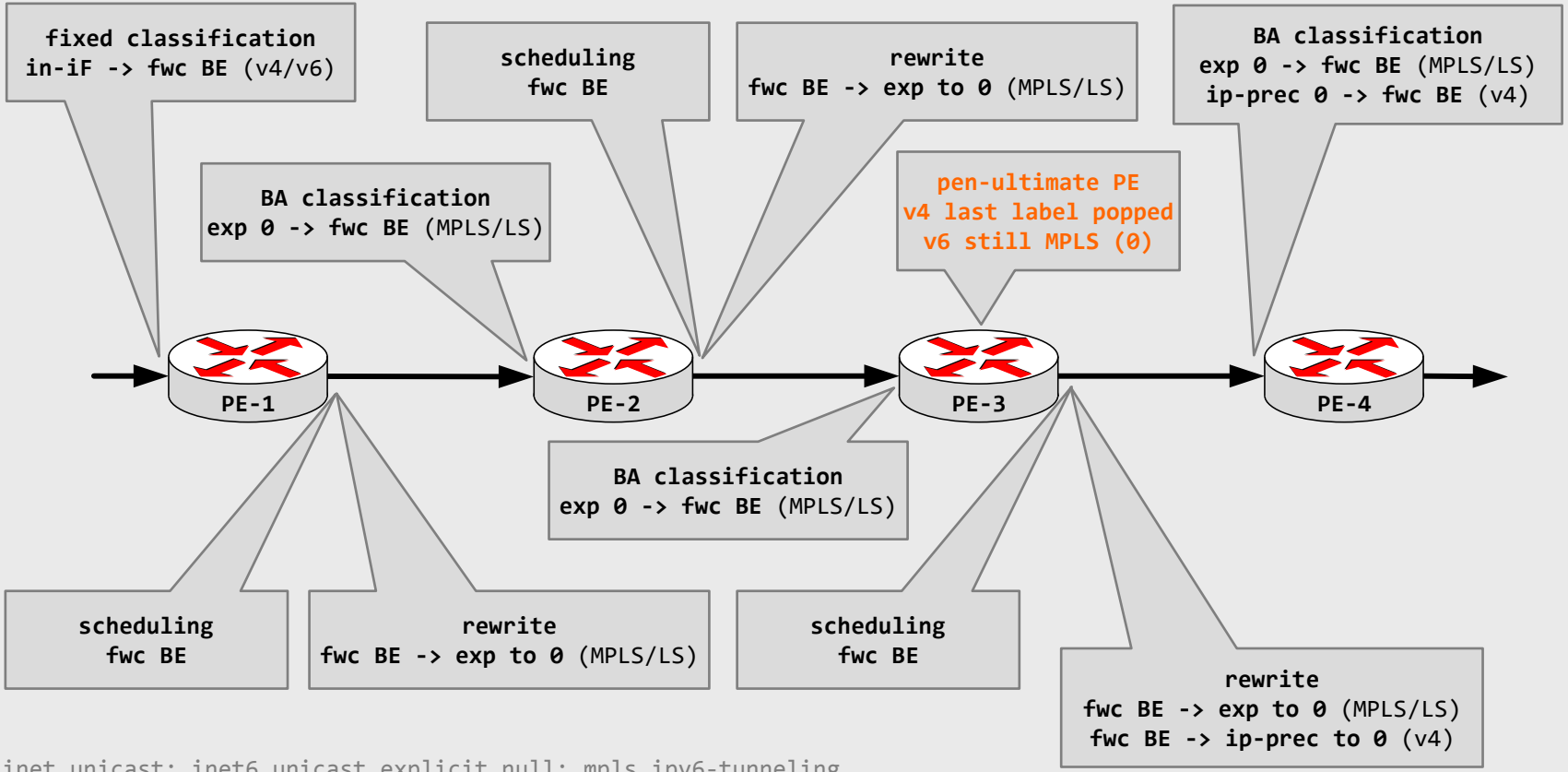
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



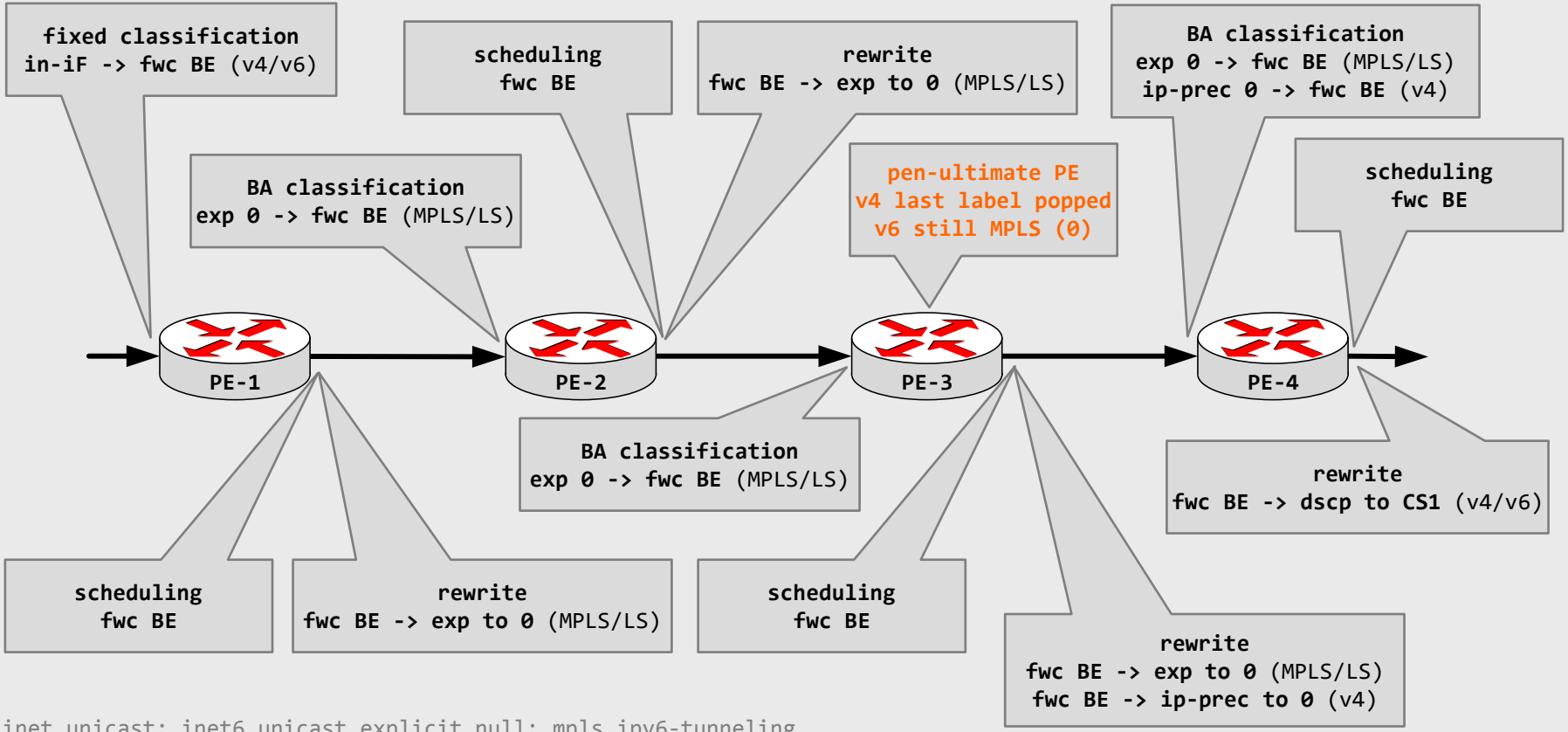
inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”



inet unicast; inet6 unicast explicit null; mpls ipv6-tunneling
yes, simplified (in-/egress policer, forward policy options)
Note: linked to later examples, BE in core is 0, BE egress is “1”

the idea of rtsdCoS

- RFC3882 mentions the use of sinkhole devices to redirect (rather than BH) attack traffic through tunnels to sinkhole(s) & then rate-limit & apply QoS/CoS policies/… on the sinkhole (simple scrubbing centers)

- RFC3882 mentions the use of sinkhole devices to redirect (rather than BH) attack traffic through tunnels to sinkhole(s) & then rate-limit & apply QoS/CoS policies/… on the sinkhole (simple scrubbing centers)

- no sinkhole device (nor tunnel) needed for simple destination IP class/group rate-limit or CoS/QoS (… limitations)

- RFC3882 mentions the use of sinkhole devices to redirect (rather than BH) attack traffic through tunnels to sinkhole(s) & then rate-limit & apply QoS/CoS policies/… on the sinkhole (simple scrubbing centers)

- no sinkhole device (nor tunnel) needed for simple destination IP class/group rate-limit or CoS/QoS (… limitations)

- “normal” traffic is marked and handled as BE (best effort)
- “special” (based on destination IP) traffic is marked and handled as NE (no effort)

- RFC3882 mentions the use of sinkhole devices to redirect (rather than BH) attack traffic through tunnels to sinkhole(s) & then rate-limit & apply QoS/CoS policies/… on the sinkhole (simple scrubbing centers)

- no sinkhole device (nor tunnel) needed for simple destination IP class/group rate-limit or CoS/QoS (… limitations)

- “normal” traffic is marked and handled as BE (best effort)
- “special” (based on destination IP) traffic is marked and handled as NE (no effort)

- rate-limit NE traffic or put into (rl) scavenger class (core/egress)

- RFC3882 mentions the use of sinkhole devices to redirect (rather than BH) attack traffic through tunnels to sinkhole(s) & then rate-limit & apply QoS/CoS policies/… on the sinkhole (simple scrubbing centers)

- no sinkhole device (nor tunnel) needed for simple destination IP class/group rate-limit or CoS/QoS (… limitations)

- “normal” traffic is marked and handled as BE (best effort)
- “special” (based on destination IP) traffic is marked and handled as NE (no effort)

- rate-limit NE traffic or put into (rl) scavenger class (core/egress)

- use same signaling mechanisms as with sBH (sBH overrules rtsdCoS)

some configuration snippets

for marking and handling BE traffic &
handling NE traffic

- customer facing port
- marking on ingress interface (default BE)
- be nice and “signal” used class to neighbor by setting DSCP bits on egress (BE to CS1, NE to CS)

```
[class-of-service interfaces <#Customer-IF#> unit *]
set forwarding-class BE
set rewrite-rules dscp rewrite-ipt-peer-port-dscp
set rewrite-rules dscp-ipv6 rewrite-ipt-peer-port-dscp-ipv6

[class-of-service rewrite-rules dscp rewrite-ipt-peer-port-dscp]
set forwarding-class NE loss-priority high code-point 000000
set forwarding-class BE loss-priority low code-point 001000
set […  don’t forget your other fwd-classes / priorities … don’t forget NC …]

[class-of-service rewrite-rules dscp-ipv6 rewrite-ipt-peer-port-dscp-ipv6]
set forwarding-class NE loss-priority high code-point 000000
set forwarding-class BE loss-priority low code-point 001000
set […  don’t forget your other fwd-classes / priorities … don’t forget NC …]
```

- maintaining the marking through the core
- used ip-prec, different to egress BE is 000, NE is 111
- watch out when using MPLS (exp/traffic class)
  - non-labeled unicast will have it’s last labeled popped and is “native” again; 6PE traffic will arrive with null label on egress PE

```
[class-of-service interfaces <#BB-IF#> unit *]
set rewrite-rules exp rewrite-bb-link-exp
set rewrite-rules inet-precedence rewrite-bb-link-ipprec
set classifiers exp classifier-bb-link-exp
set classifiers inet-precedence classifier-bb-link-ipprec

[class-of-service rewrite-rules exp rewrite-bb-link-exp]
set forwarding-class BE loss-priority low code-point 000
set forwarding-class NE loss-priority high code-point 111
set […  don’t forget your other fwd-classes / priorities …]
[class-of-service rewrite-rules inet-precedence rewrite-bb-link-ipprec]
set forwarding-class BE loss-priority low code-point 000
set forwarding-class NE loss-priority high code-point 111
set […  don’t forget your other fwd-classes / priorities …]

[class-of-service classifiers exp classifier-bb-link-exp]
set forwarding-class BE loss-priority low code-points 000
set forwarding-class NE loss-priority high code-points 111
set […  don’t forget your other fwd-classes …]
[class-of-service classifiers inet-precedence classifier-bb-link-inet-precedence]
set forwarding-class BE loss-priority low code-points 000
set forwarding-class NE loss-priority high code-points 111
set […  don’t forget your other fwd-classes …]
```

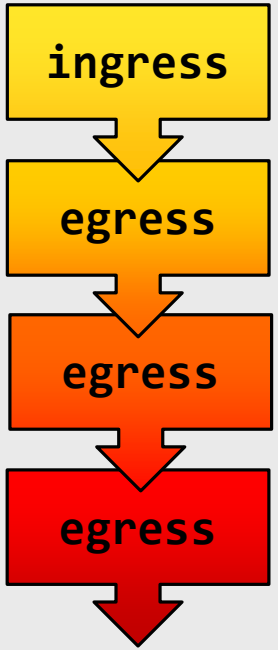
- core scheduling (smarter is better)
- here limited to 8%, exact vs rate-limit
- scavenger class better option (?)
- better drop-profiles, CoS-tuning …

```
[class-of-service interfaces <#BB-IF#>]
set scheduler-map schedmap-bb-link-default

[class-of-service scheduler-maps schedmap-bb-link-default]
set forwarding-class BE scheduler sched-bb-link-default-BE
set forwarding-class NE scheduler sched-bb-link-default-NE
set [… don’t forget your other fwd-classes …]

[class-of-service schedulers sched-bb-link-default-BE]
set transmit-rate remainder
set buffer-size remainder
set priority low

[class-of-service schedulers sched-bb-link-default-NE]
set transmit-rate percent 8
set transmit-rate exact
set buffer-size percent 8
set priority low

[… don’t forget other schedulers …]
```

- customer scheduling (smarter is better)
- here simple limit to 400m (exact vs. rate-limit)
- better drop-profiles, CoS-tuning

```
[class-of-service scheduler-maps schedmap-ipt-default]
set forwarding-class NC scheduler sched-ipt-default-NC
set forwarding-class BE scheduler sched-ipt-default-BE
set forwarding-class NE scheduler sched-ipt-default-NE
[…]

[class-of-service schedulers sched-ipt-default-NC]
set transmit-rate percent 5
set buffer-size percent 5
set priority medium-high

[class-of-service schedulers sched-ipt-default-BE]
set transmit-rate remainder
set buffer-size remainder
set priority low

[class-of-service schedulers sched-ipt-default-NE]
set transmit-rate 400m exact
set buffer-size percent 1
set priority low

[…]
```

using information in BGP for (overwriting) BE classification



classification (BA, MF, fixed, forwarding policy) & policing
assign traffic to **forwarding-class** based on ingress interface, codepoints, header fields, forwarding policy & policing

policing

queuing / shaping / scheduling
service queues associated with forwarding-classes

rewrite
rewrite of CoS fields in egress packet

simplified – check J's manuals on your hardware

```
/* all validated prefixes sdCoS prefixes signaled from RR */
policy-options {
    community rtsdCoS-active members 286:28655;

    policy-statement m-rtsdCoS {
        term overwrite-fixed-CA {
            from community rtsdCoS-active;
            then class rtsdCoS-overwrite;
    }
}
class-of-service {
    forwarding-policy {
        class rtsdCoS-overwrite {
            classification-override {
                forwarding-class NE;
        }
    }
routing-options {
    forwarding-table {
        export [ … load-sharing m-rtsdCoS … ];
    }
}
```

causes classification to be overwritten from BE to NE for "marked" prefixes during route-lookup

J engineering: impact when overwriting forwarding-class in FIB is insignificant (worked with full table); impact on forwarding performance is insignificant; debugging / displaying overwrite is unclear so far (evt. >13.3??)

Cisco?

(don’t ask me about others)

- **what about Cisco? => QPPB (QoS Policy Propagation with/for/via BGP)**
- just “theoretical” looked at not even lab tested
- don’t ask for v4/v6 parity, limitations, supported platforms …
- but it has been around for a while … and it’s “way more” documented than for J

```
! IOS (XR different)
! Just the marking part (QoS group)
! other stuff left out here

interface <#Ingress/Egress-IF#>
  bgp-policy destination ip-qos-map
  service-policy output xxx

router bgp 517
  address-family ipv4
    table-map m-rtsdCoS
  address-family ipv6
    table-map m-rtsdCoS
```

```
ip community-list 1 permit 286:28655

route-map m-rtsdCoS permit 10
  match community 1
  set ip qos-group 55
route-map m-rtsdCoS permit 20
  set ip qos-group 0

! policy-map xxx e.g.
! - limiting qos-group 99 traffic to x Mbps
! – set DSCP bits
! - […]

! or use ip-prec-map / precedence
```

cool?
for sure fancy!

… and with being creative on CoS it could do even more

- statement: rt(s)dCoS is the better rt(s)BH

- statement: rt(s)dCoS is the better rt(s)BH
- protects destination/infrastructure/down-|up-link/core of an overload

- statement: rt(s)dCoS is the better rt(s)BH
- protects destination/infrastructure/down-|up-link/core of an overload
- still allows some traffic to be sent/come in (analysis/end detecting)
- helpful on non-automated mitigation (filter setup)

- statement: rt(s)dCoS is the better rt(s)BH

- protects destination/infrastructure/down-|up-link/core of an overload

- still allows some traffic to be sent/come in (analysis/end detecting)
- helpful on non-automated mitigation (filter setup)

- but as BHing it doesn’t help to restore the affected service
- in most cases service will remain unusable ‘till end of flood
- might be affected when multiple used at the same time – auto-recovery

- statement: rt(s)dCoS is the better rt(s)BH

- protects destination/infrastructure/down-|up-link/core of an overload

- still allows some traffic to be sent/come in (analysis/end detecting)
- helpful on non-automated mitigation (filter setup)

- but as BHing it doesn’t help to restore the affected service
- in most cases service will remain unusable ‘till end of flood
- might be affected when multiple used at the same time – auto-recovery

- don‘t underestimate CoS setup & maintenance, HW requirements

- statement: rt(s)dCoS is the better rt(s)BH
- protects destination/infrastructure/down-|up-link/core of an overload
- still allows some traffic to be sent/come in (analysis/end detecting)
- helpful on non-automated mitigation (filter setup)
- but as BHing it doesn’t help to restore the affected service
- in most cases service will remain unusable ‘till end of flood
- might be affected when multiple used at the same time – auto-recovery
- don‘t underestimate CoS setup & maintenance, HW requirements
- not competing with flowspec – different story (but if can only have the one or other, use flowspec as it gives you more – v4/v6?)

available?

- soon on https://AS286.net … most likely
- fine tuning schedulers & CoS domain check (legacy Cisco edge)
- (delivery & change process, training, monitoring, troubleshooting, …)
- customer documentation

rtsdCoS → flowspec?

this is the end