Source Address Validation: Problem of Existing Solutions

draft-li-sava-intra-domain-use-cases-00

Dan Li (Tsinghua)

Jianping Wu (Tsinghua)

Yunan Gu (Huawei)

Lancheng Qin (Tsinghua)

Tao Lin (H3C)

Source Address Validation (SAV)

□ The traditional Internet architecture lacks the validation of a packet's source address

□ Source address spoofing is dangerous

- ✓ Well documented in RFC 6959
- ✓ Single-packet attack, flood-based DoS, poisoning attack, spoof-based worm/malware propagation, reflective attack, accounting subversion, man-in-the-middle attack, third-party recon

□ SAV is important to prevent source address spoofing

Existing SAV Solutions

□ Host-level SAV

✓ SAVI [RFC 7039]

◆ Problem: Requires all the access networks (sub-nets) to deploy simultaneously

□ Network-level SAV

✓ Ingress ACL [RFC 2827]

◆ Problem: Requires manual configuration to update

✓ uRPF

◆ Strict uRPF [RFC 3704]

◆ Loose uRPF [RFC 3704]

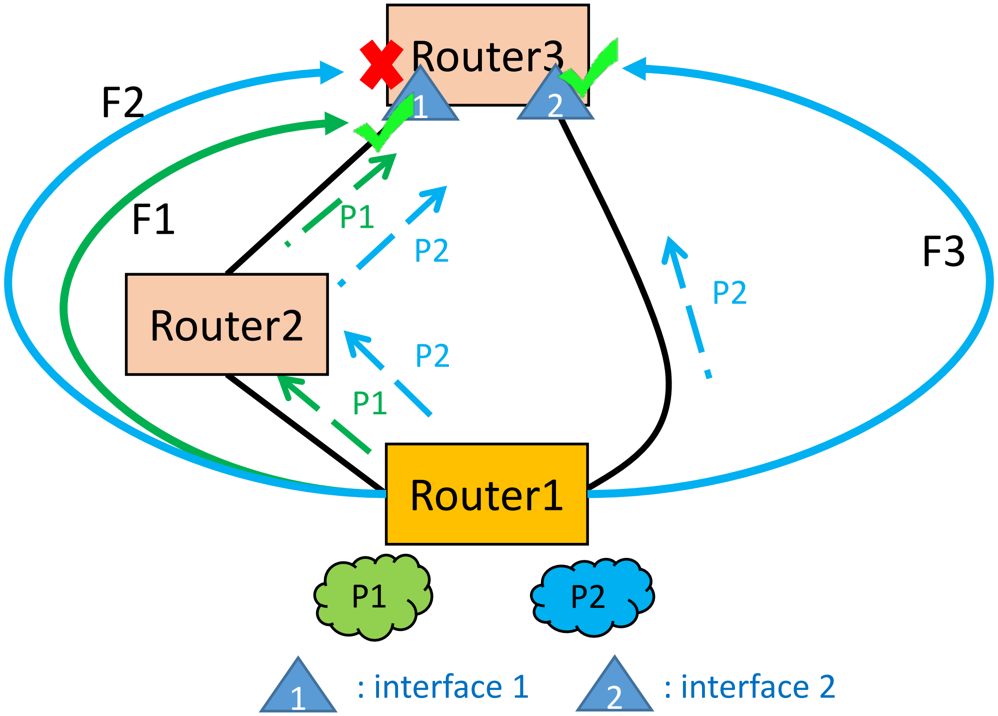
◆ Feasible-Path uRPF (FP-uRPF) [RFC 3704]

◆ Enhanced Feasible-Path uRPF (EFP-uRPF) [RFC 8704]

Strict uRPF and the Problem

- ✓ Take the source address as a destination address to lookup the FIB.
- ✓ If the outgoing interface of the FIB matches the incoming interface of the packet, then pass

FIB at Router 3:
Prefix: P1
NH: interface 1
Prefix: P2
NH: interface 2

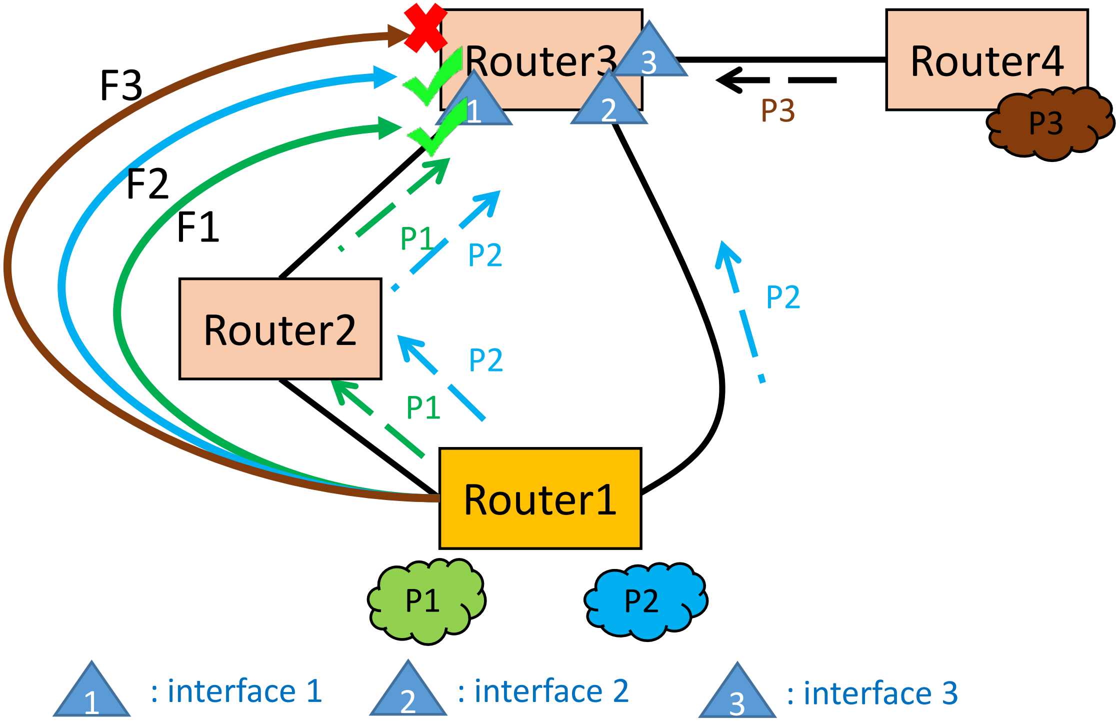


✓ Flow 1 with source address P1 is correctly accepted at interface 1
✗ Flow 2 with source address P2 is incorrectly denied at interface 1
✓ Flow 3 with source address P2 is correctly accepted at interface 2

Loose uRPF and the Problem

- ✓ Take the source address as a destination address to lookup the FIB
- ✓ If the address exists in the FIB, then pass

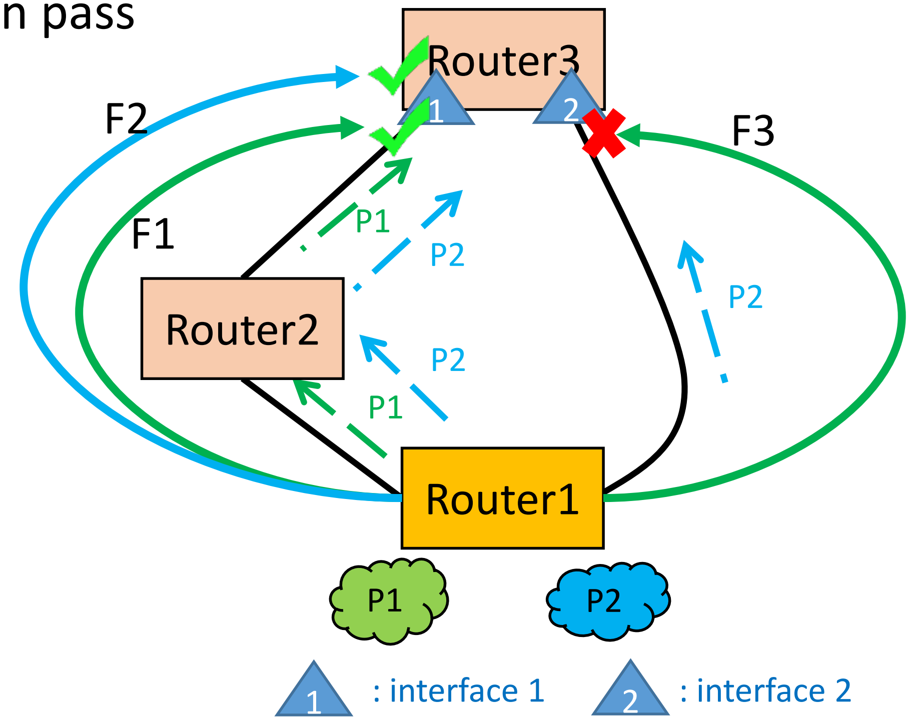
FIB:
Prefix: P1
NH: interface 1
Prefix: P2
NH: interface 2
Prefix: P3
NH: interface 3



✓ Flow 1 with source address P1 is correctly accepted at interface 1
✓ Flow 2 with source address P2 is correctly accepted at interface 1
✗ Flow 3 with source address P3 is incorrectly accepted at interface 1

FP-uRPF and the Problem

- ✓ Take the source address as a destination address to lookup the RIB (including other routing information besides FIB)
- ✓ If the outgoing interface of the RIB matches the incoming interface of the packet, then pass

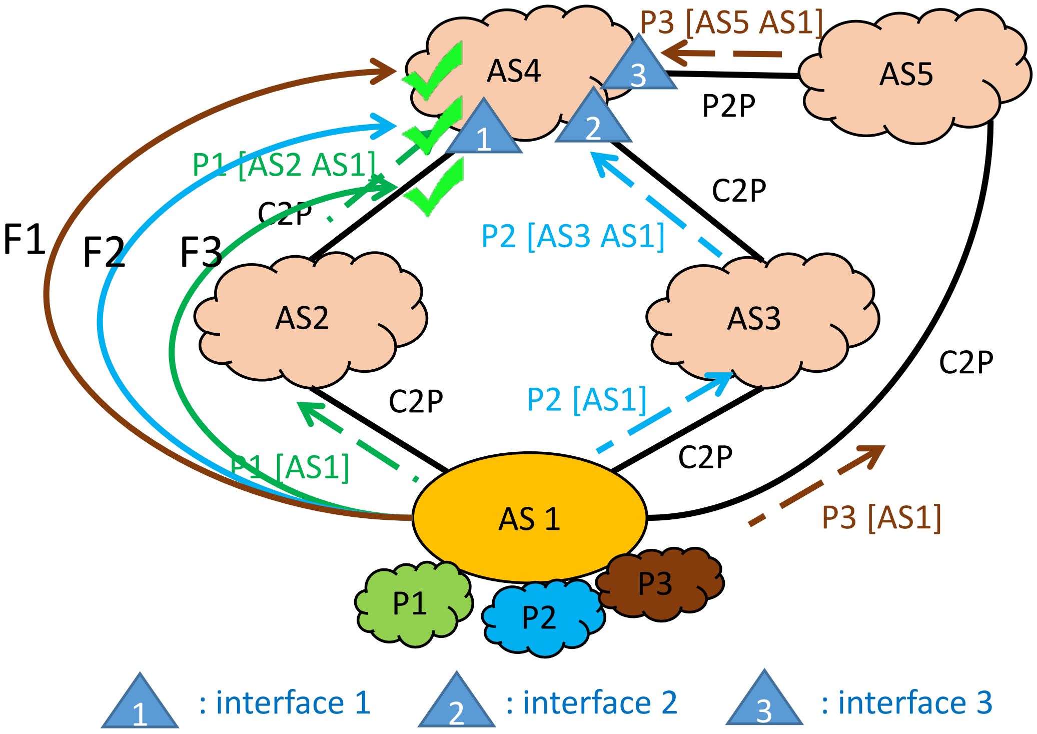


RIB:
Prefix: P1
NH: interface 1
Prefix: P2
NH: interface 1
Prefix: P2
NH: interface 2

- ✓ Flow 1 with source address P1 is correctly accepted at interface 1
- ✓ Flow 2 with source address P2 is correctly accepted at interface 1
- ✗ Flow 3 with source address P1 is incorrectly denied at interface 2

EFP-uRPF Algorithm A

- ✓ EFP-uRPF is designed for Inter-AS case
- ✓ Set all the prefixes received for an AS on each customer interface that received an update

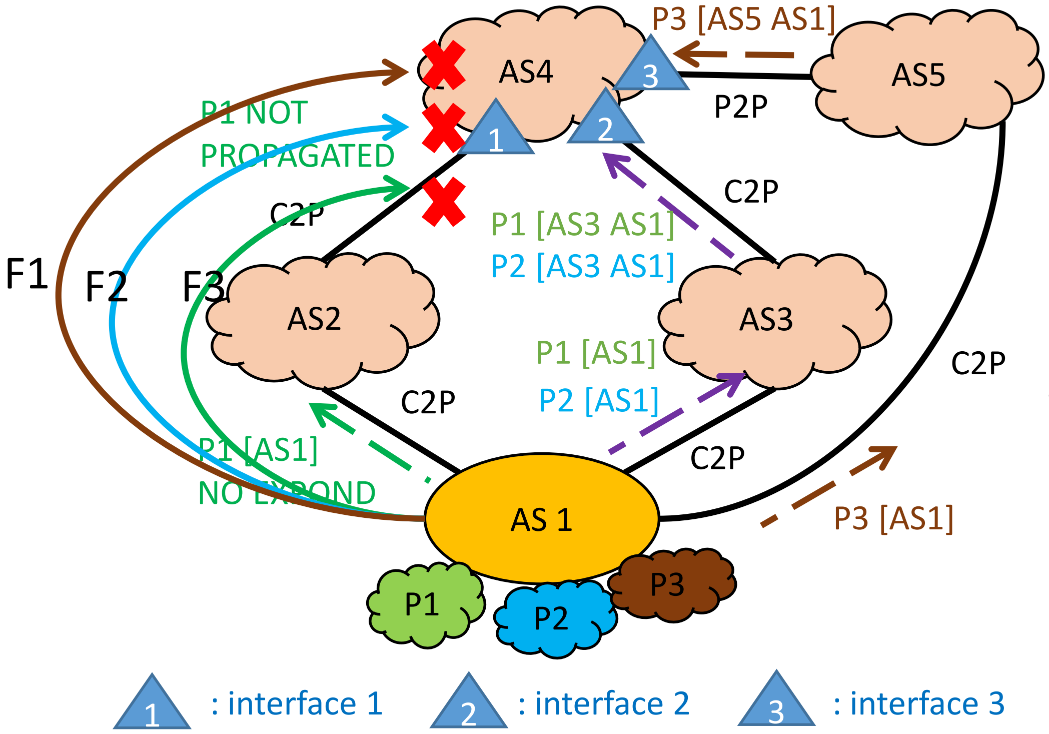


EFP-uRPF Algorithm A:
1. Set A = {AS1……}
2. X1 = {P1,P2,P3}
3. Include X1 in RPF list on interface 1 and interface 2

- ✓ Flow 1 with source address P1 is correctly accepted at interface 1
- ✓ Flow 2 with source address P2 is correctly accepted at interface 1
- ✓ Flow 3 with source address P3 is correctly accepted at interface 1

The Problem of EFP-uRPF Algorithm A

- ✓ EFP-uRPF is designed for Inter-AS case
- ✓ Set all the prefixes received for an AS on each customer interface that received an update



EFP-uRPF Algorithm A:

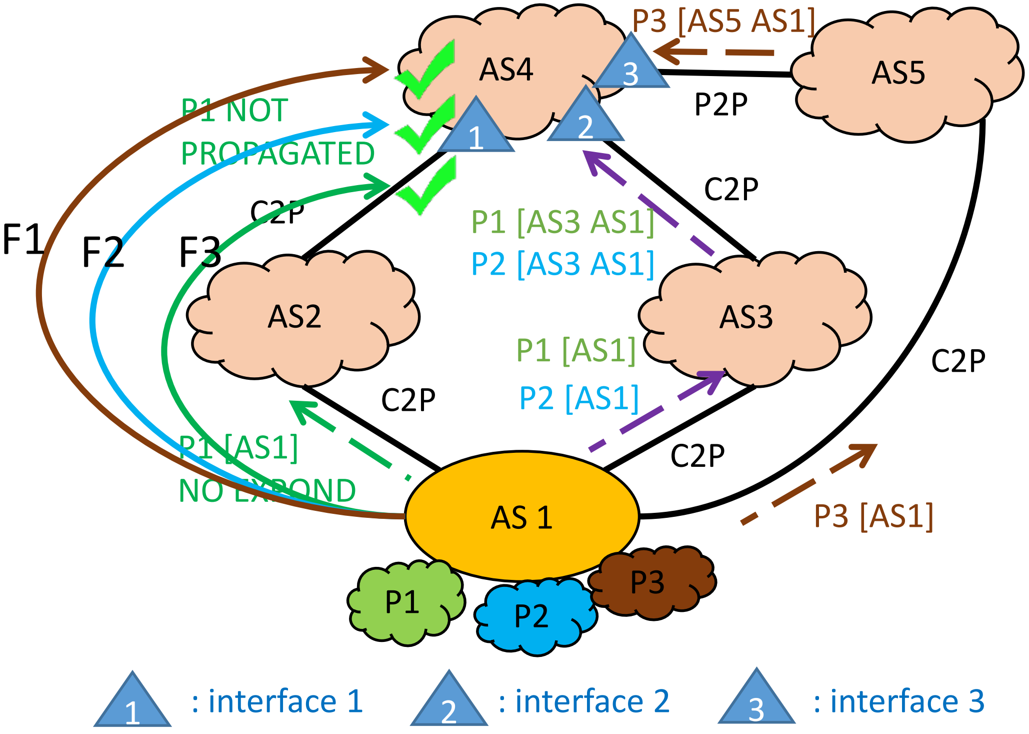
1. Set A = {AS1……}
2. X1 = {P1,P2,P3}
3. RPF List on interface 2: X1;

RPF List on interface 1: {Ø}

- ✘ Flow 1 with source address P1 is incorrectly denied at interface 1
- ✘ Flow 2 with source address P2 is incorrectly denied at interface 1
- ✘ Flow 3 with source address P3 is incorrectly denied at interface 1

EFP-uRPF Algorithm B

- ✓ Set Z on all the customer interfaces
- ✓ Z is composed of both prefixes learned from customer interfaces and prefixes learned from peer/provider interfaces for an AS learned from customer interfaces



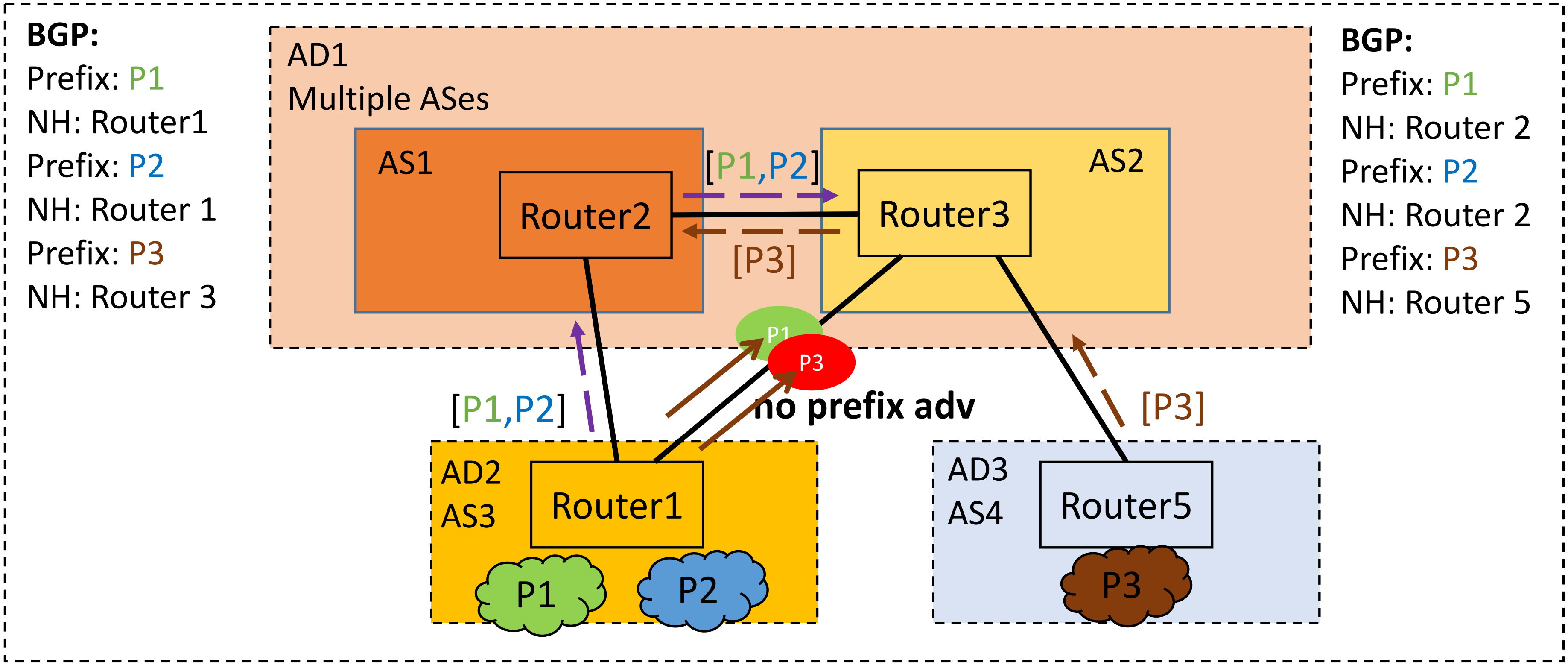
EFP-uRPF Algorithm B

1. Set I = {interface 1,interface 2}
2. P = {P1,P2}
3. A = {AS1}
4. Q = {P3}
5. Z = {P1,P2,P3} for interface 1 and interface2

- ✓ Flow with source address in P1 is correctly accepted at interface 1
- ✓ Flow with source address in P2 is correctly accepted at interface 1
- ✓ Flow with source address in P3 is correctly accepted at interface 1

Cases When All uRPF Solutions cannot Work

Case 1: Inter-AS

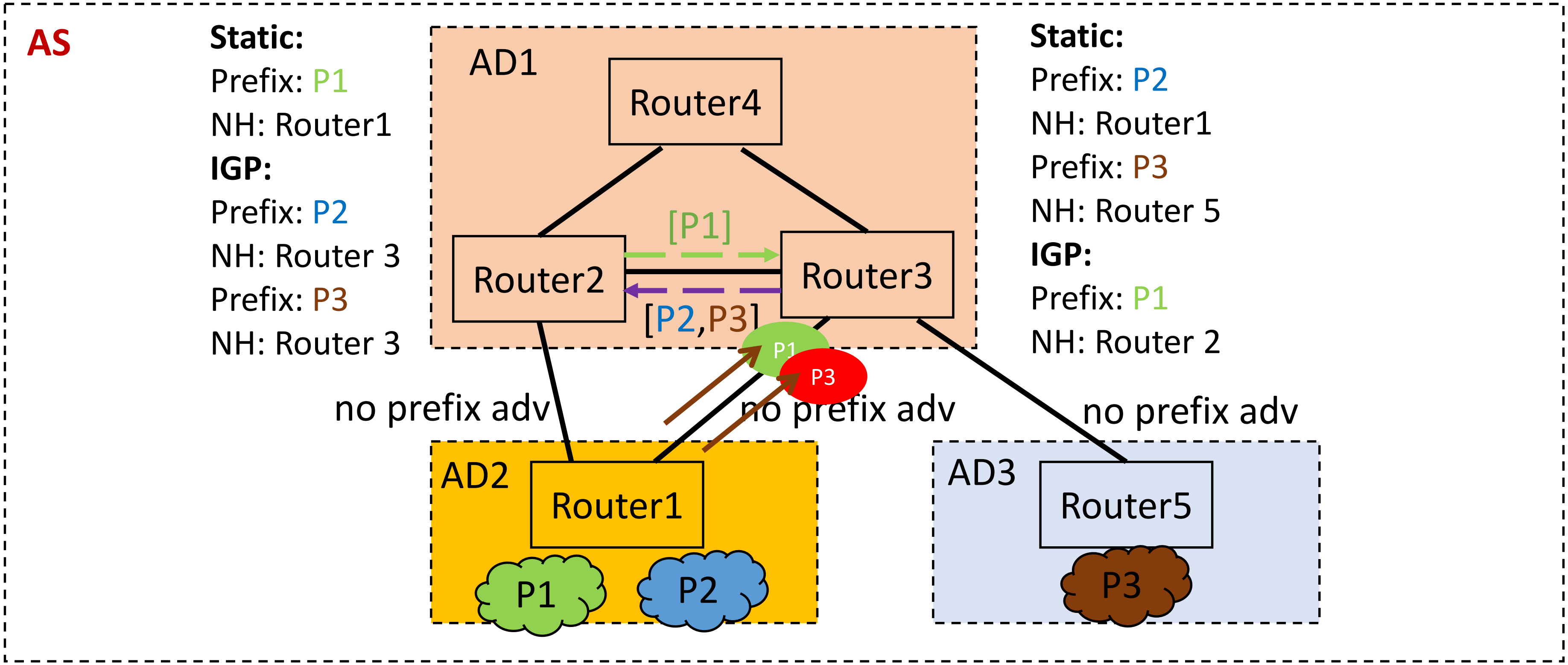


BGP:
Prefix: P1
NH: Router1
Prefix: P2
NH: Router 1
Prefix: P3
NH: Router 3

BGP:
Prefix: P1
NH: Router 2
Prefix: P2
NH: Router 2
Prefix: P3
NH: Router 5

Strict uRPF drops the legitimate packet	✖	drops the forged packet	✓
Loose uRPF accepts the legitimate packet	✓	accepts the forged packet	✖
Feasible- path uRPF drops the legitimate packet	✖	drops the forged packet	✓
EFP-uRPF A drops the legitimate packet	✖	drops the forged packet	✓
EFP-uRPF B accepts the legitimate packet	✓	accepts the forged packet	✖

Case 2: Intra-AS



Static:
Prefix: P1
NH: Router1
IGP:
Prefix: P2
NH: Router 3
Prefix: P3
NH: Router 3

Static:
Prefix: P2
NH: Router1
Prefix: P3
NH: Router 5
IGP:
Prefix: P1
NH: Router 2

Strict uRPF drops the legitimate packet ✗ drops the forged packet ✓

Loose uRPF accepts the legitimate packet ✓ accepts the forged packet ✗

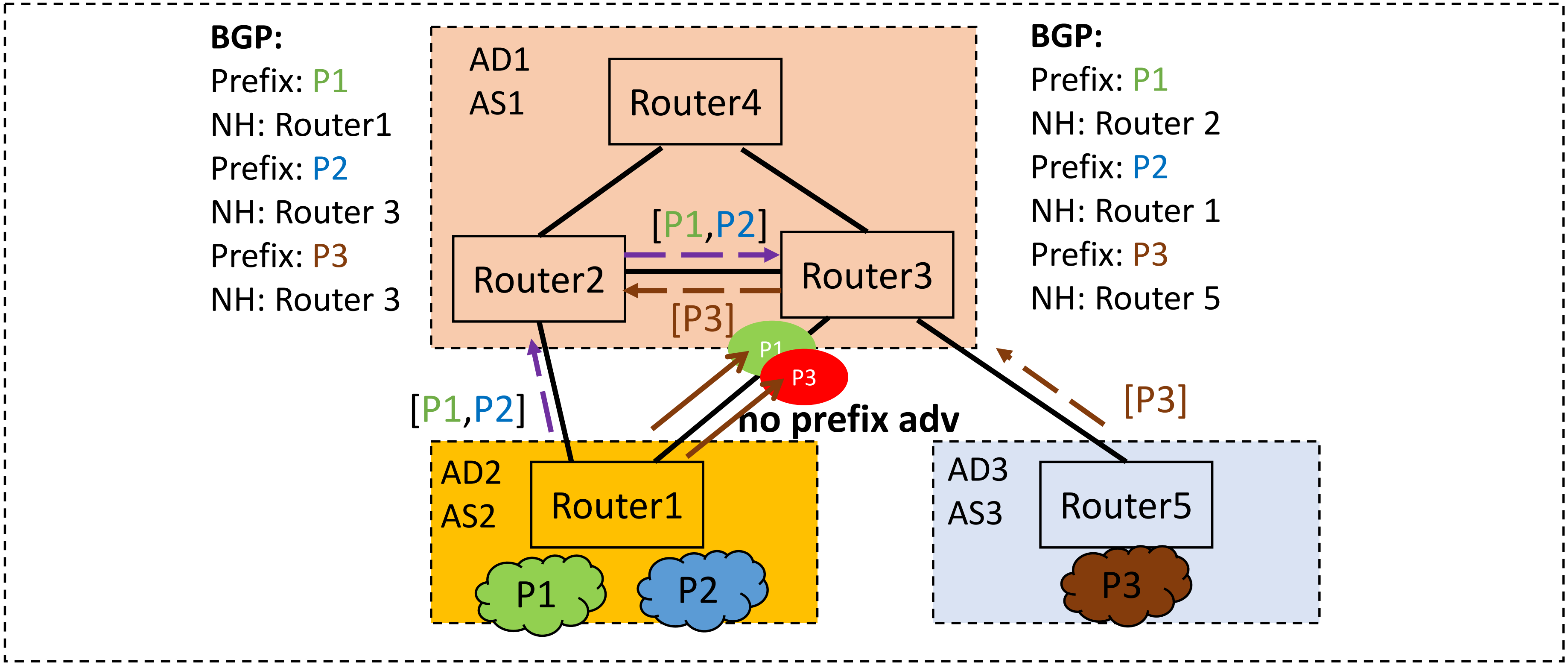
Feasible- path uRPF drops the legitimate packet ✗ drops the forged packet ✓

EFP-uRPF does not apply at the intra-AS case

Thanks!

Any comments?

Use Case 3: Inter-AS



BGP:
Prefix: P1
NH: Router1
Prefix: P2
NH: Router 3
Prefix: P3
NH: Router 3

BGP:
Prefix: P1
NH: Router 2
Prefix: P2
NH: Router 1
Prefix: P3
NH: Router 5

Strict uRPF drops the legitimate packet ✖ drops the forged packet ✓

Loose uRPF accepts the legitimate packet ✓ accepts the forged packet ✖

Feasible- path uRPF drops the legitimate packet ✖ drops the forged packet ✓

EFP-uRPF does not mention this case