xBGP: When You Can't Wait for the IETF and Vendors

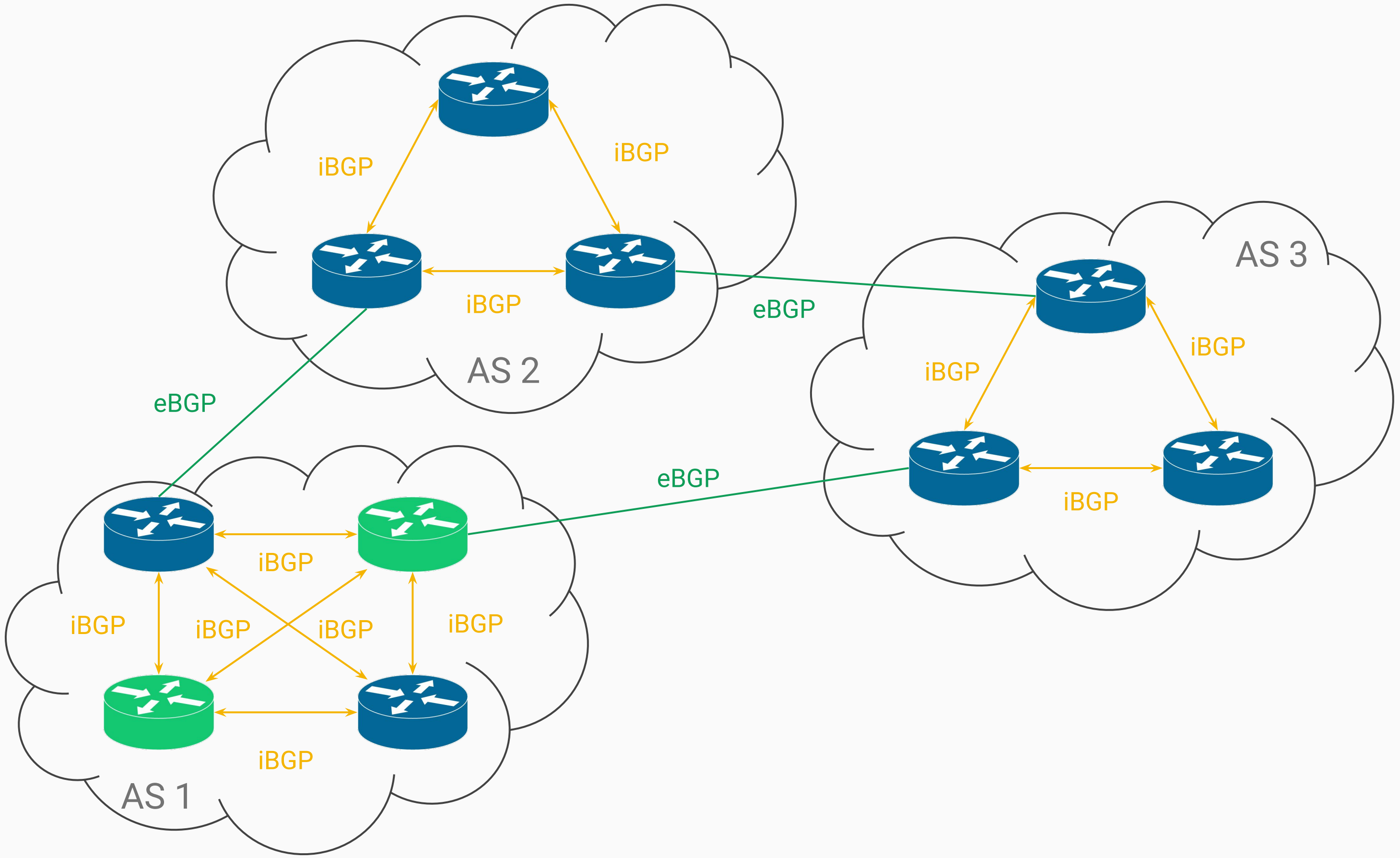
Thomas Wirtgen, Quentin De Coninck, Randy Bush, Laurent Vanbever and Olivier Bonaventure

UCLouvain icteam

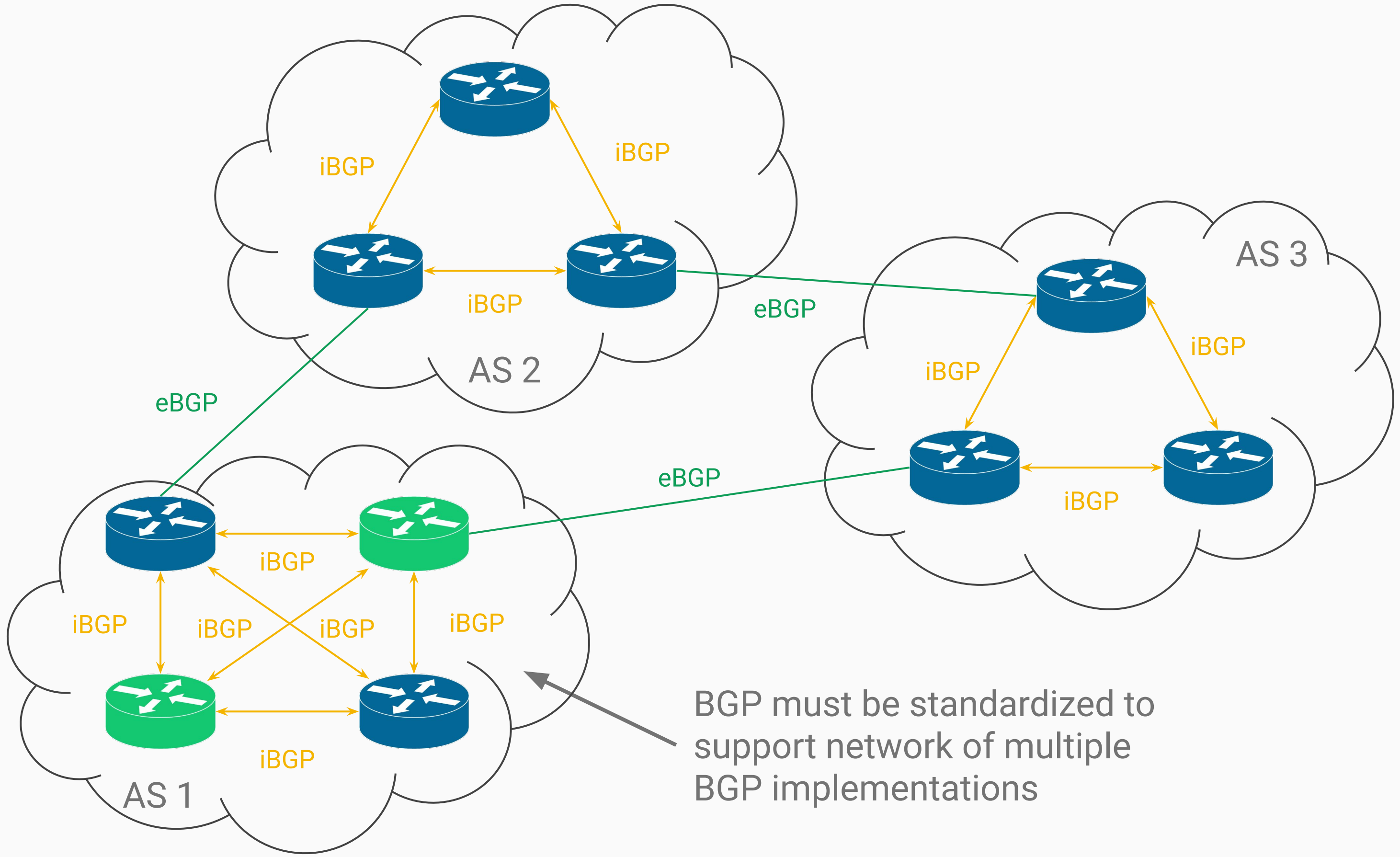
ETH zürich Networked Systems ETH Zürich — seit 2015

IIJ Internet Initiative Japan

BGP enables routing on the Internet



BGP enables routing on the Internet

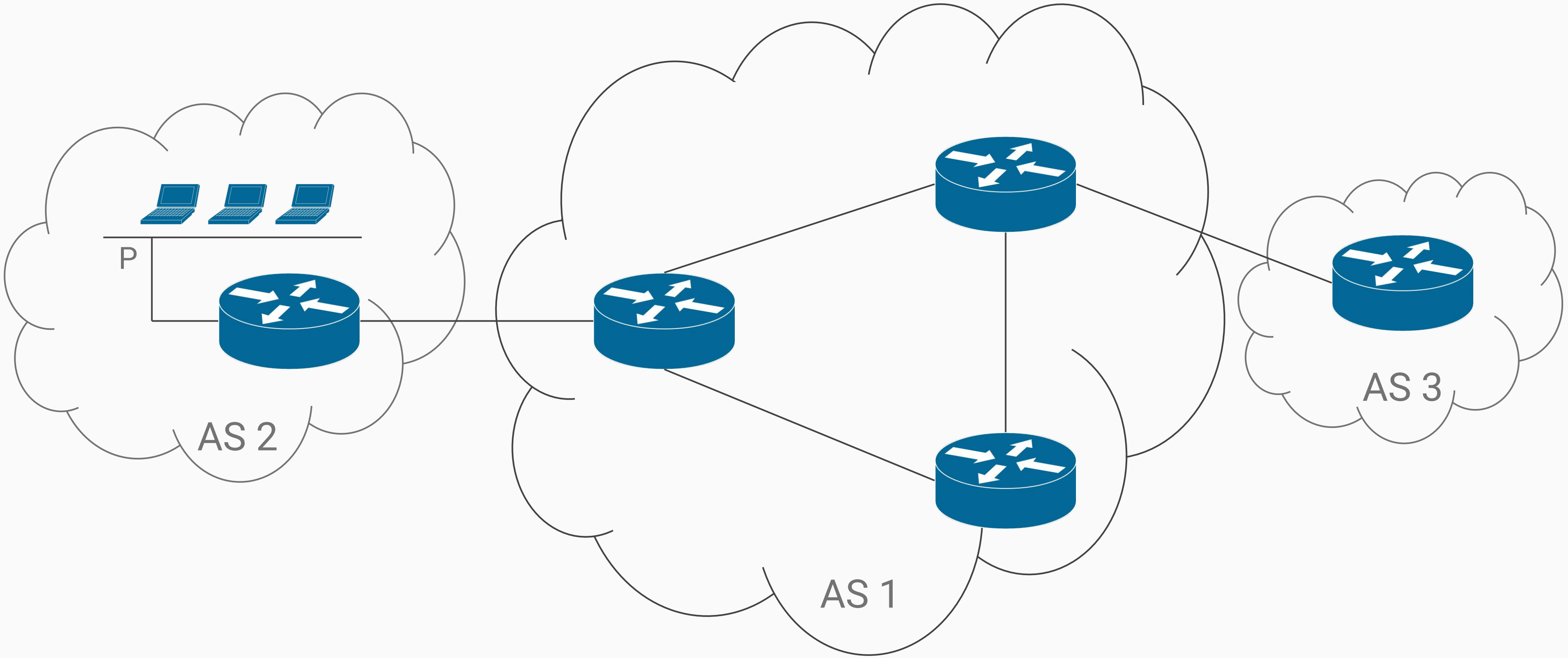


BGP must be standardized to support network of multiple BGP implementations

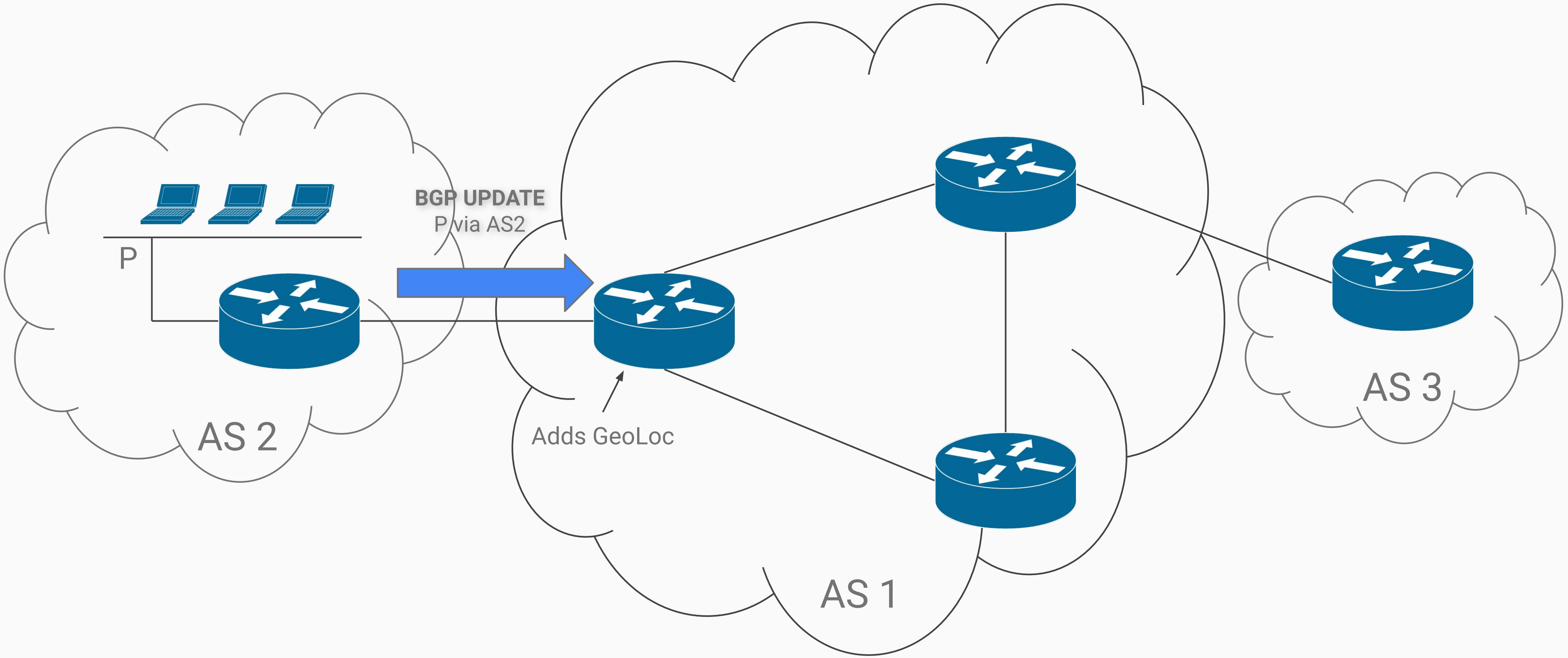
Agenda

- **The Weaknesses of the Current Routing Paradigm**
- xBGP: a Paradigm Shift
- Adding a new feature with xBGP
- Uses Cases

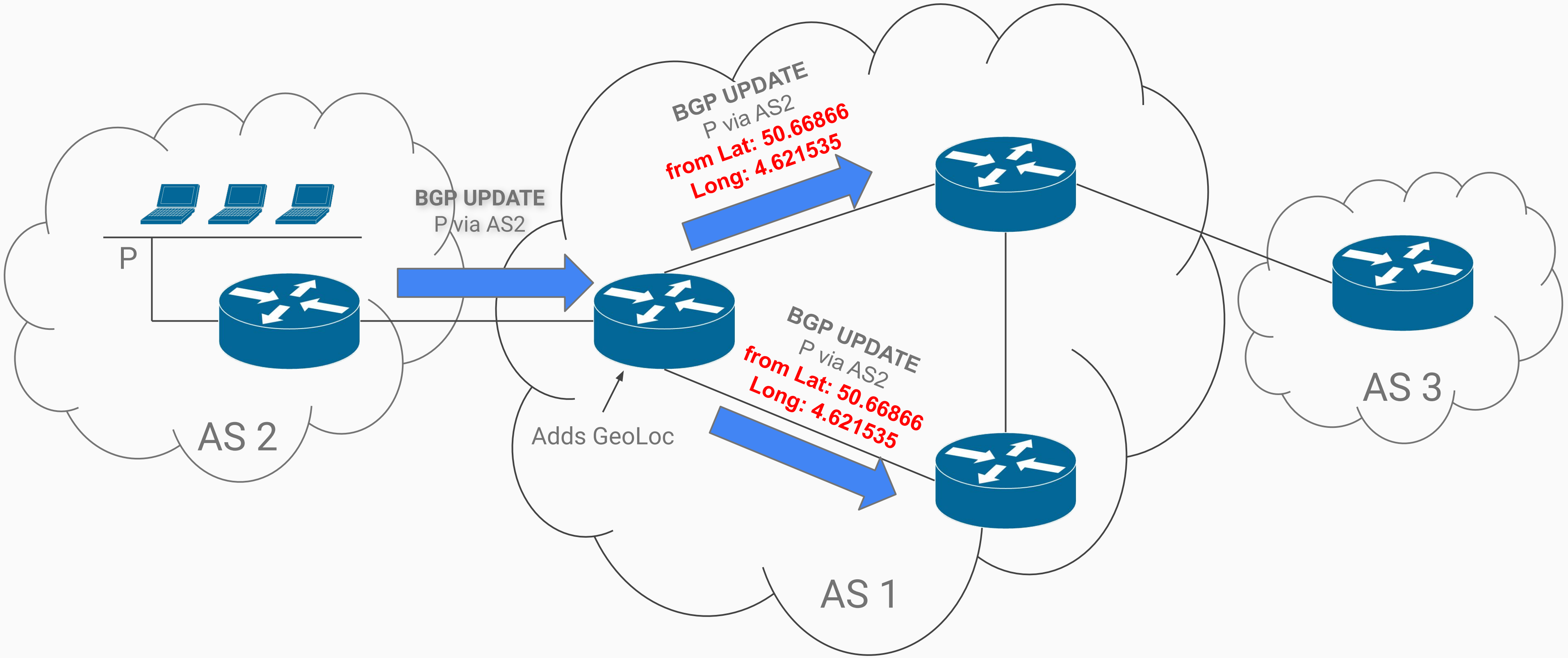
Example of rejected feature: Geo-location TLV



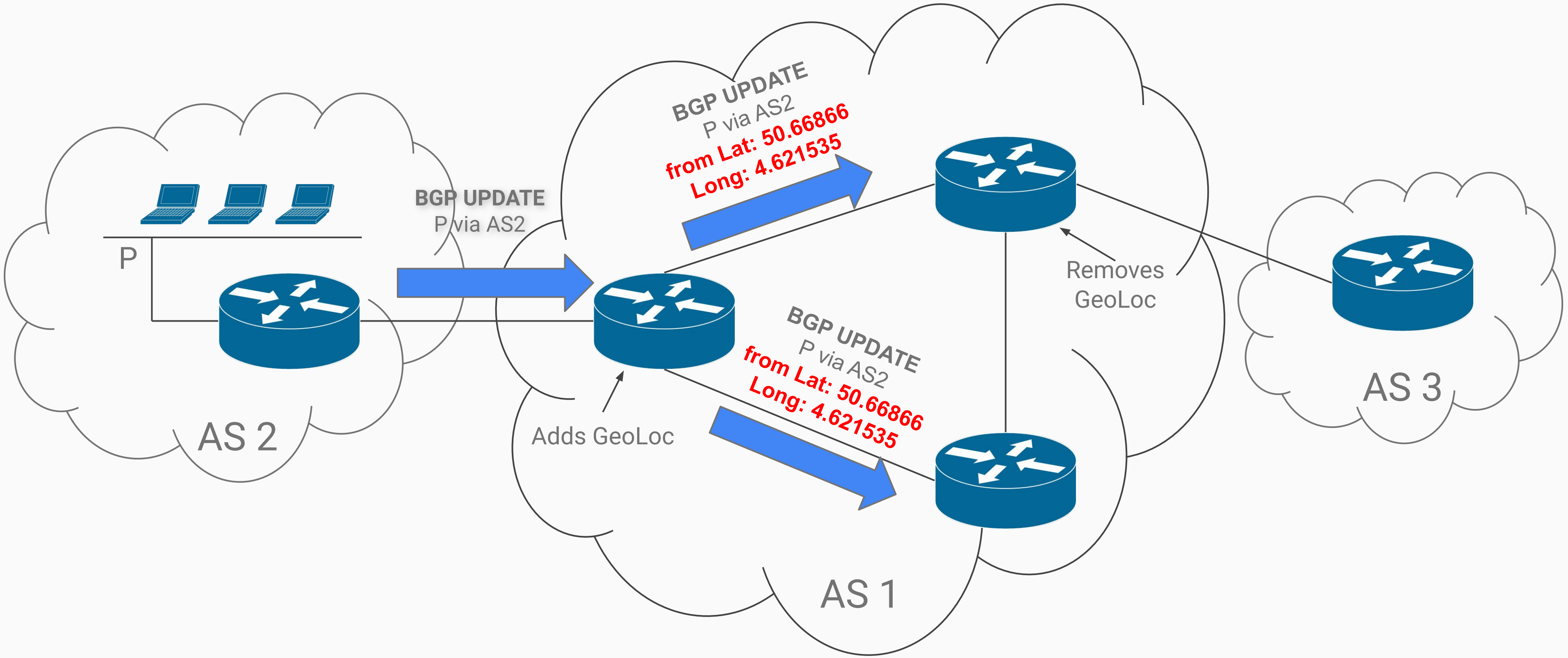
Example of rejected feature: Geo-location TLV



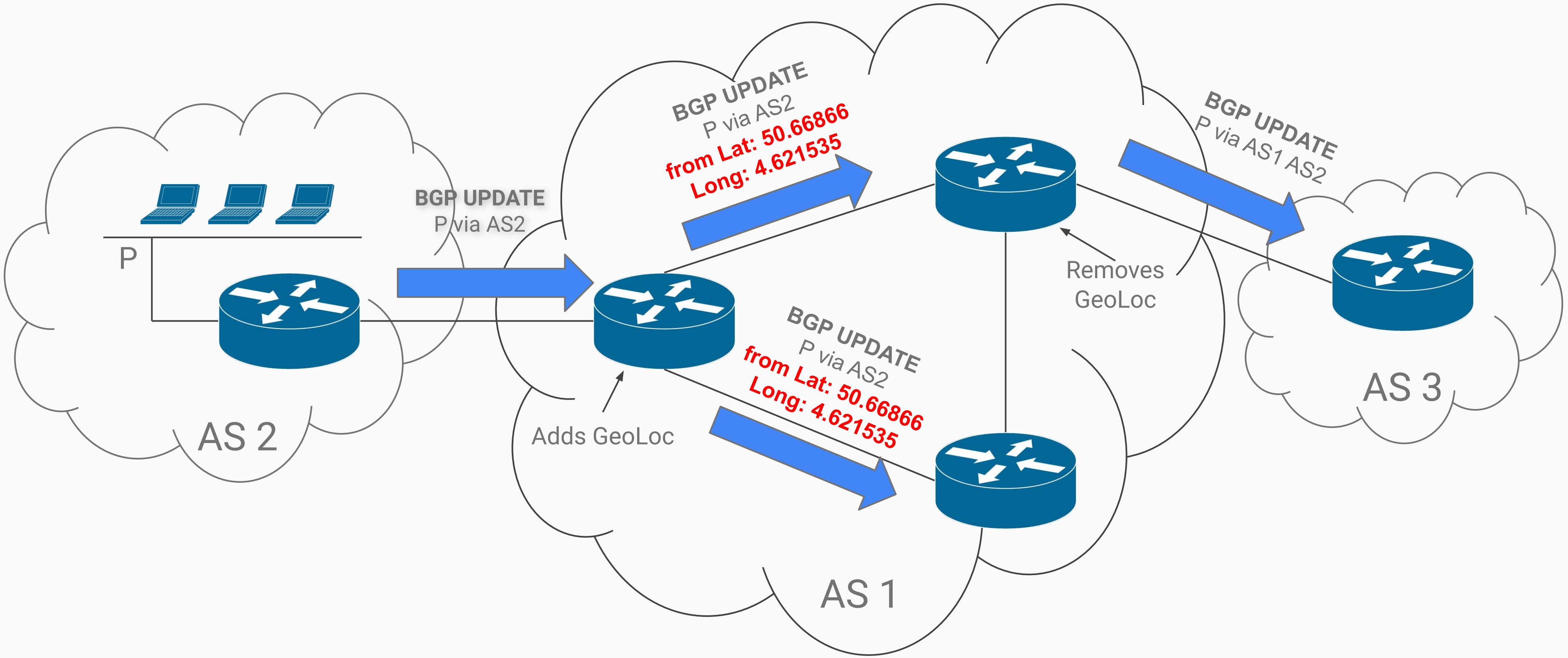
Example of rejected feature: Geo-location TLV



Example of rejected feature: Geo-location TLV



Example of rejected feature: Geo-location TLV

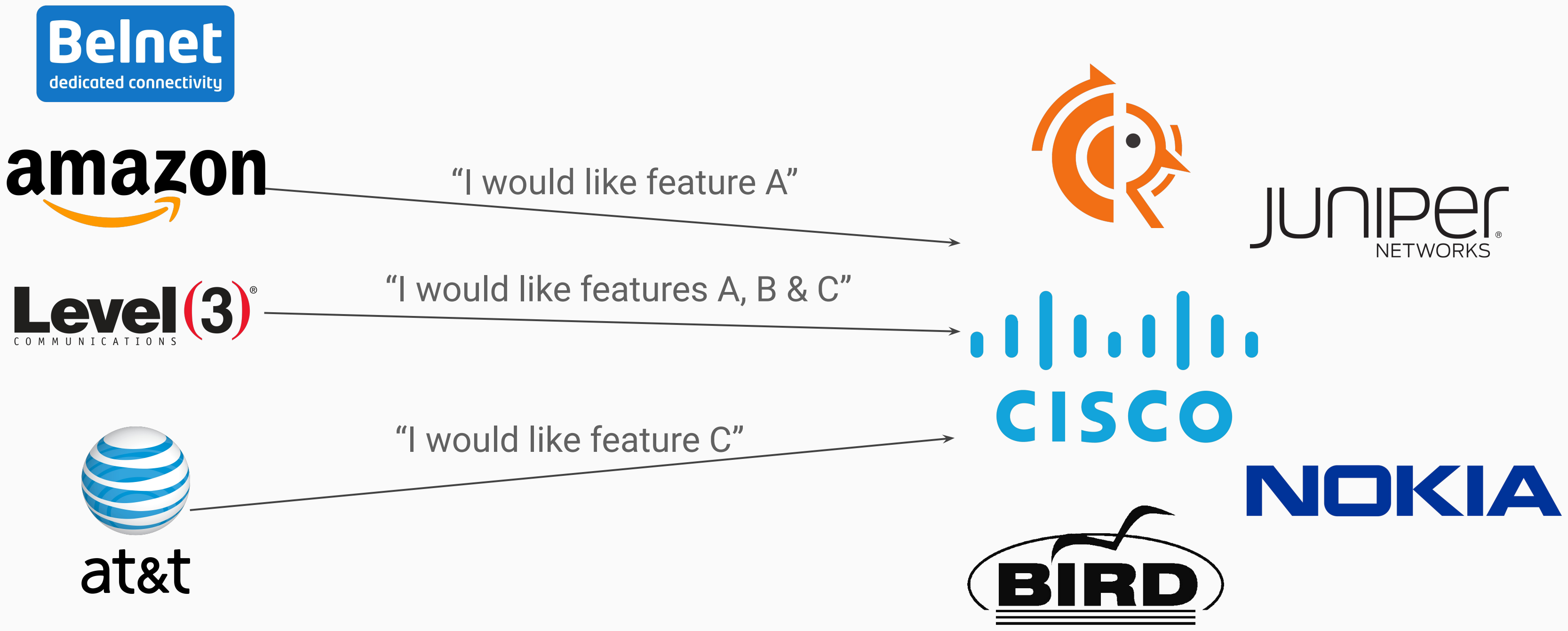


The Need of Programmable Routers

Routers vendors receive a lot of feature requests

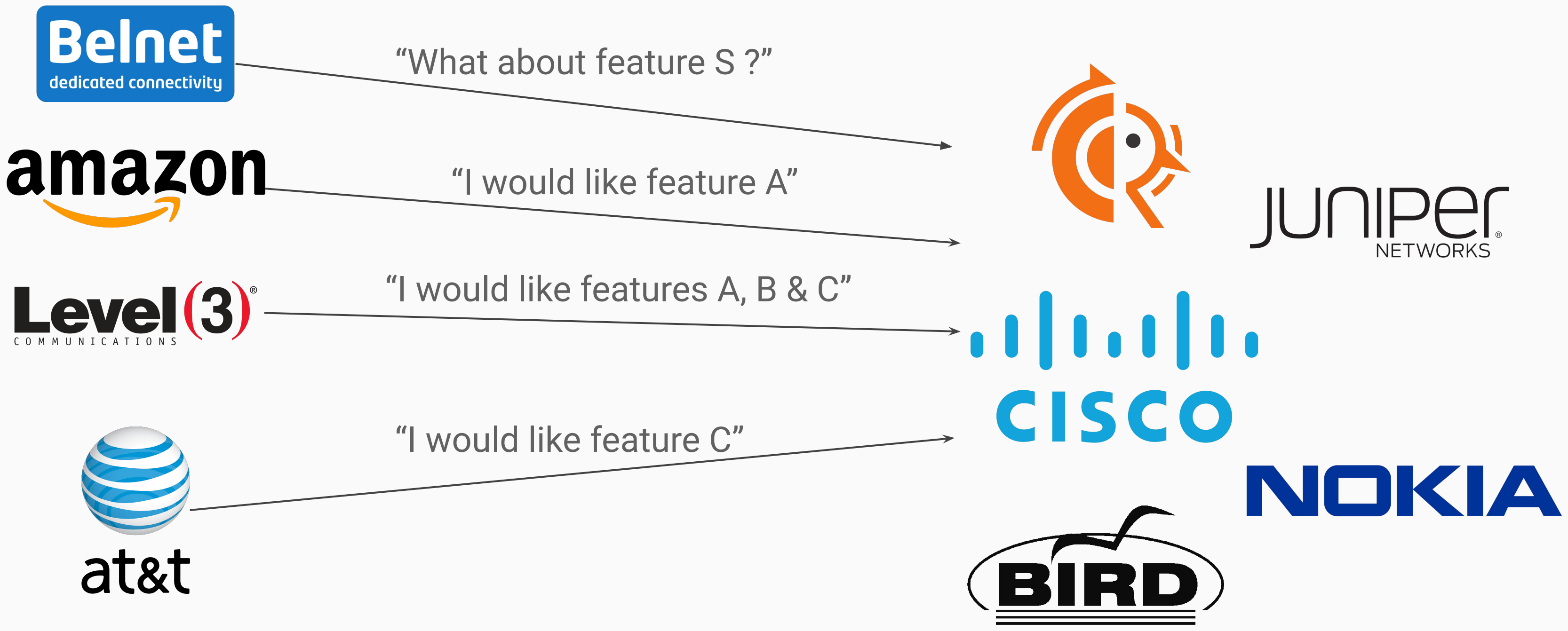
The Need of Programmable Routers

Routers vendors receive a lot of feature requests

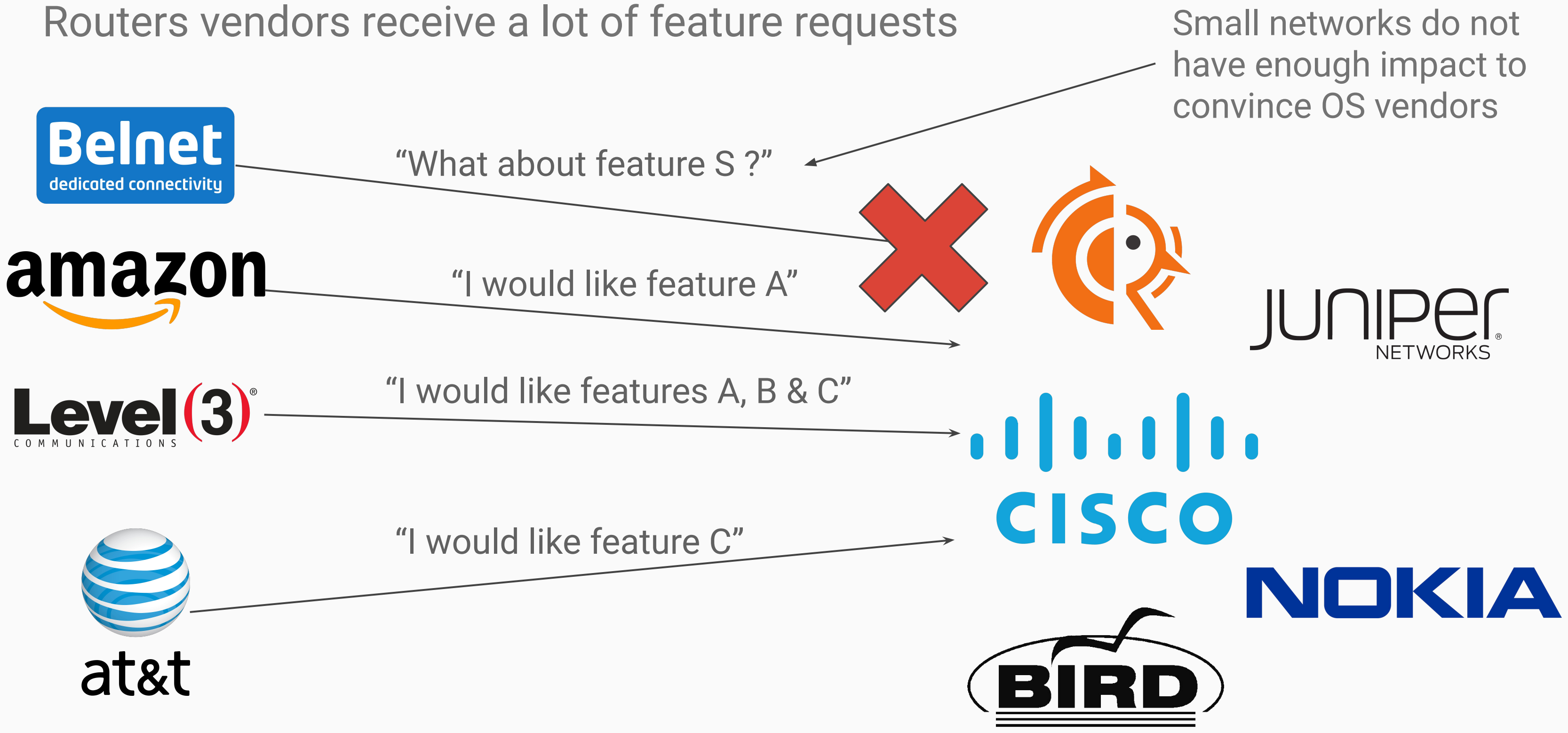


The Need of Programmable Routers

Routers vendors receive a lot of feature requests



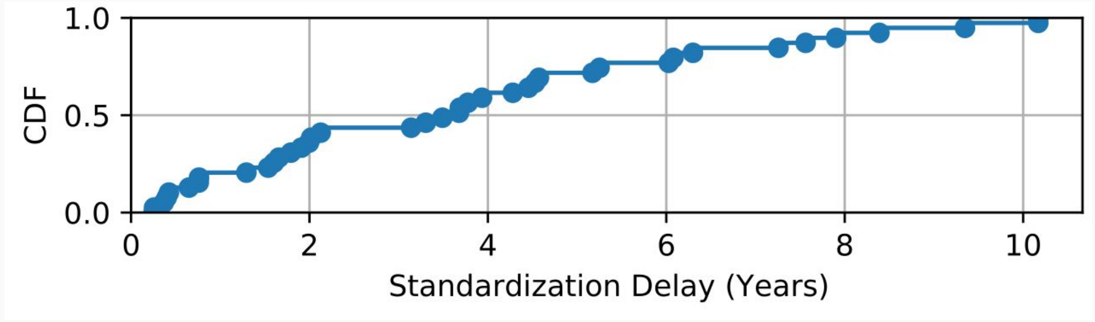
The Need of Programmable Routers


Routers vendors receive a lot of feature requests
Small networks do not have enough impact to convince OS vendors
"What about feature S ?"
"I would like feature A"
"I would like features A, B & C"
"I would like feature C"
Belnet dedicated connectivity
amazon
Level(3) COMMUNICATIONS
at&t
JUNIPER NETWORKS
CISCO
NOKIA
BIRD

Problem #1: Networks evolve, as do routing protocols

The evolution is complex:

1. Standardization by the IETF (3.5 years in average for BGP)
2. Implementation on the vendor OS
3. Update routers of networks



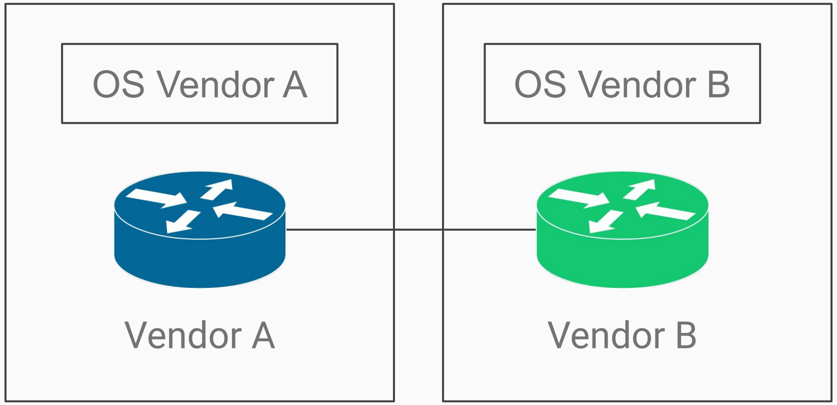
Problem #2: Large networks use diverse routers

Vendors do not propose the same set of extensions on their routers

The configuration of these routers differs as well

```
routing-options {
  router-id 1.1.1.1;
  autonomous-system 65001;
}

protocols {
  bgp {
    group Session-to-R1 {
      type external;
      neighbor 1.1.1.2 {
      peer-as 65002;
    }
  }
}
```

Simple Juniper configuration file



```
router bgp 65001
  bgp router-id 1.1.1.1
  neighbor 1.1.1.2 remote-as 65002
```

Simple Cisco configuration file

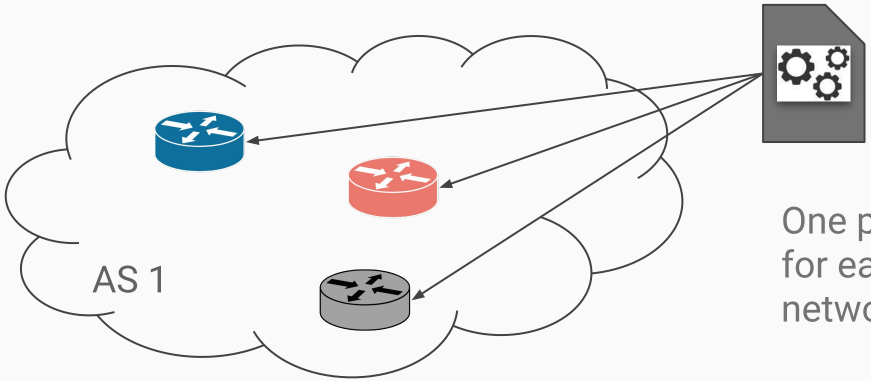
Agenda

- The Weaknesses of the Current Routing Paradigm
- xBGP: **a Paradigm Shift**
- Adding a new feature with xBGP
- Uses Cases

xBGP: toward a paradigm shift

xBGP proposes a common interface to dynamically update **any** BGP implementation.

Network operators can program their routers directly with plugins.



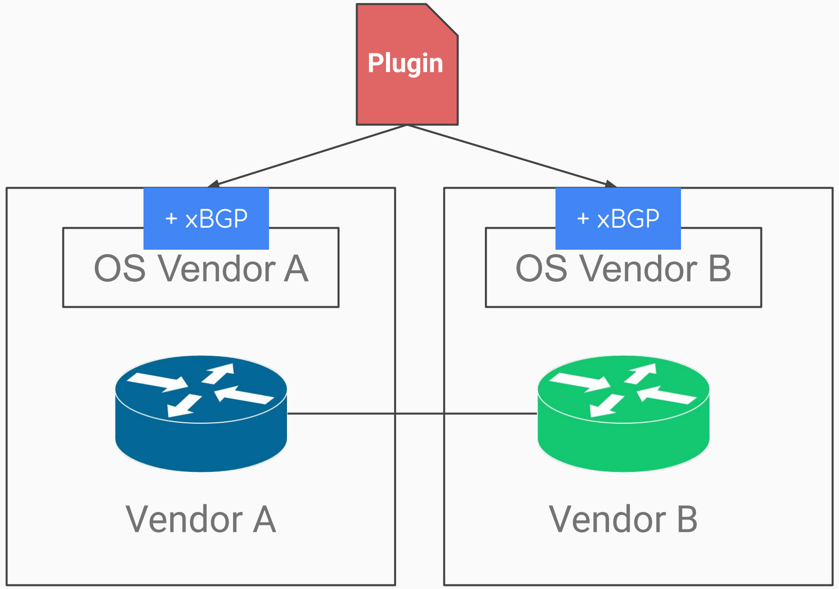
One plugin is injected for each router of the network

xBGP forces routers to follow the same rules

Each router adds xBGP on top of its implementation

With xBGP, routers expose a common API.

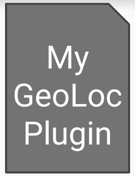
Can be seen as POSIX norms

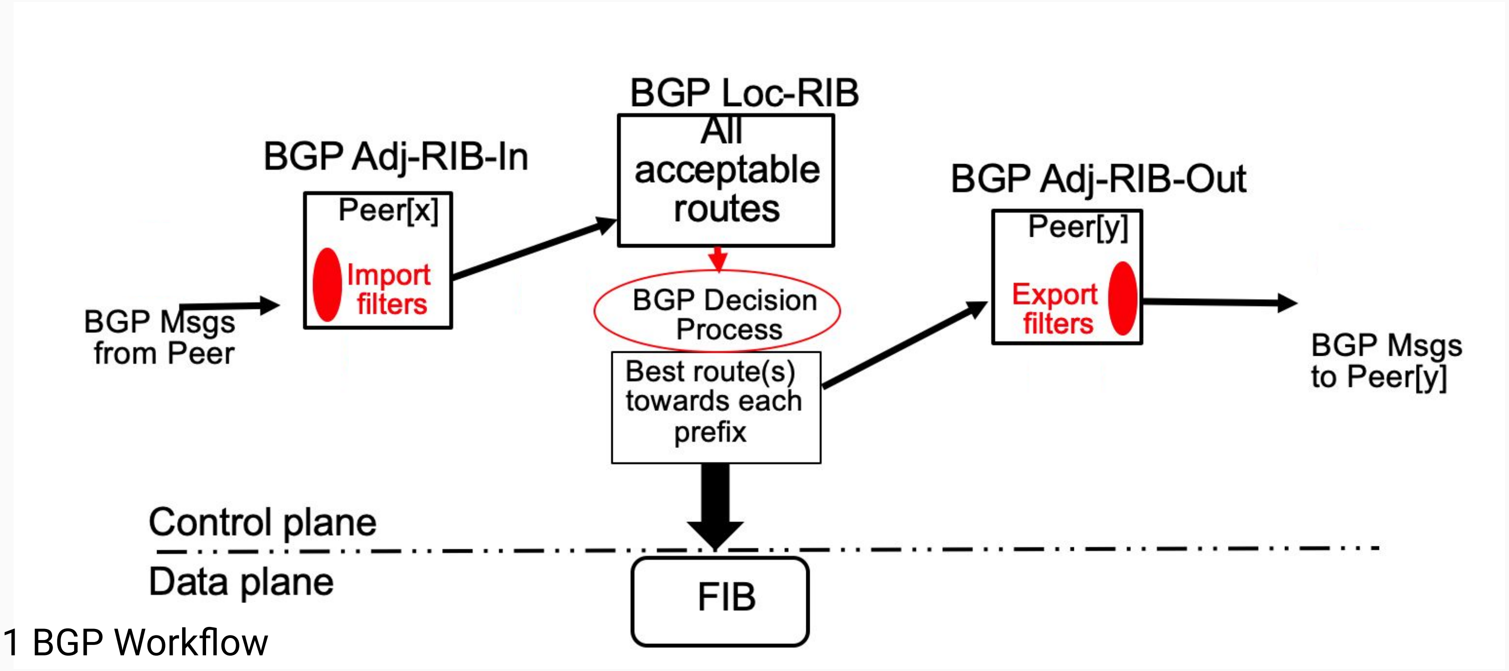


Agenda

- The Weaknesses of the Current Routing Paradigm
- xBGP: a Paradigm Shift
- **Adding a new feature with** xBGP
- Uses Cases

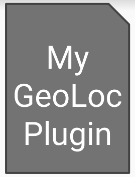
Structure of xBGP

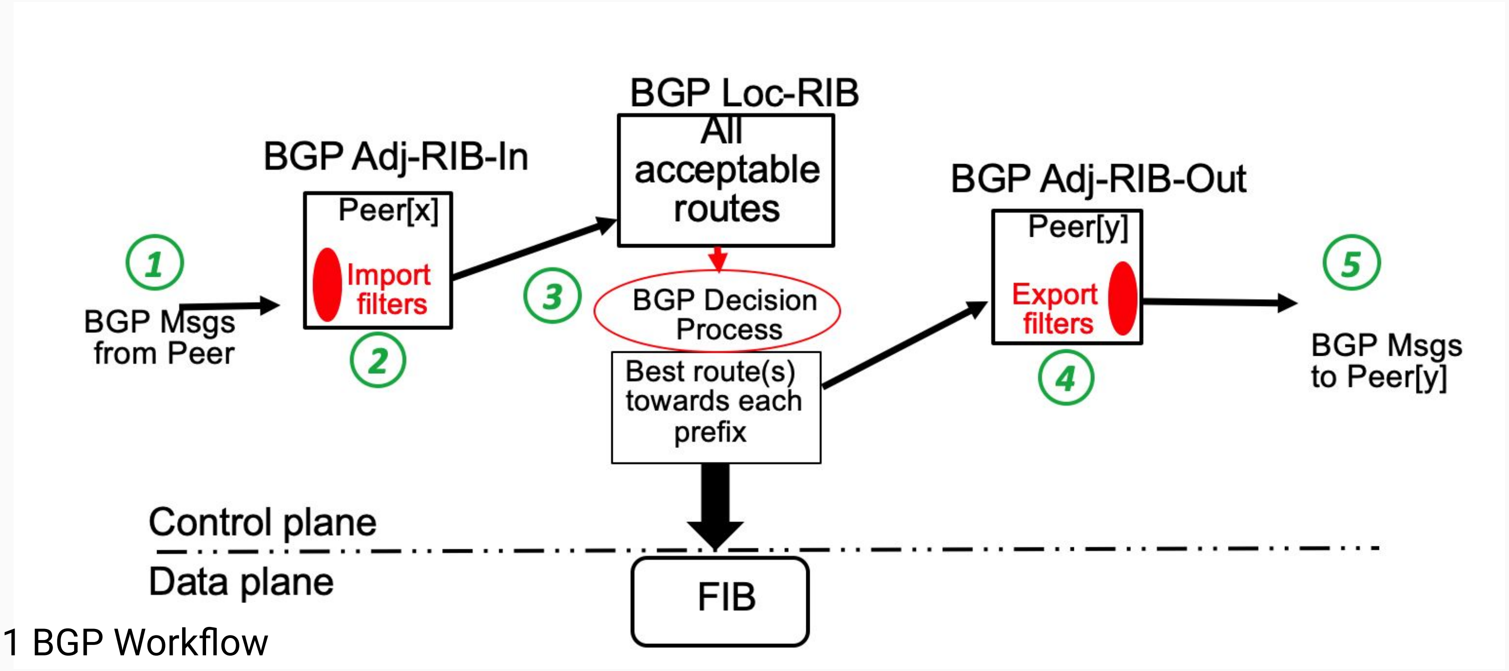




RFC 4271 BGP Workflow

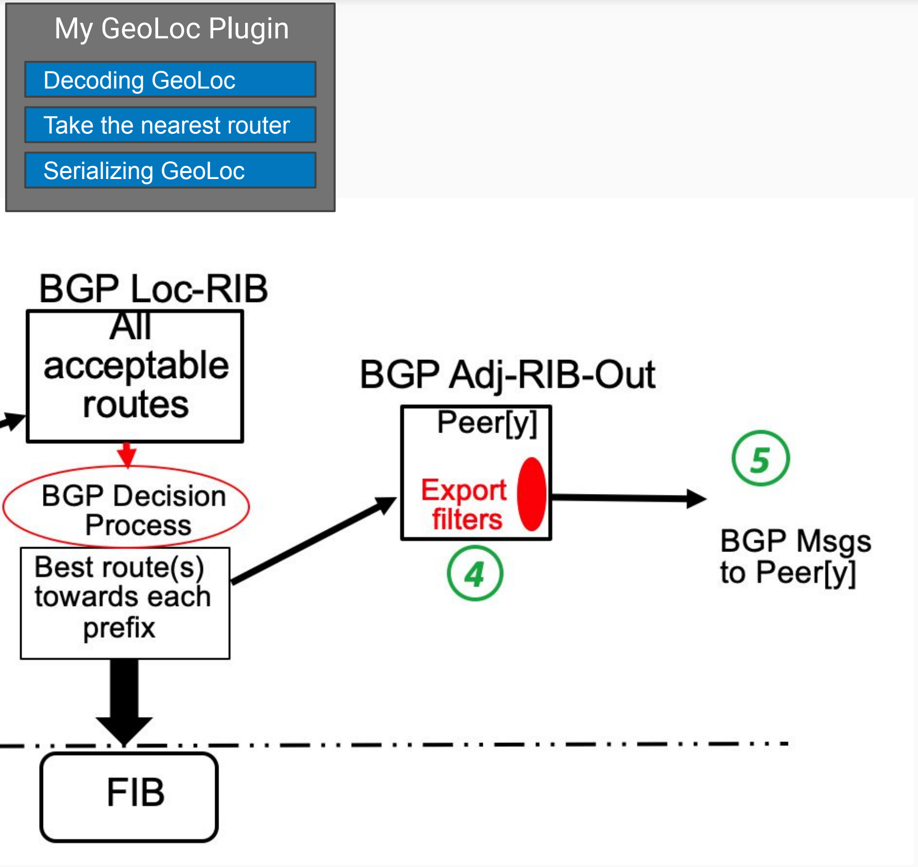
Structure of xBGP





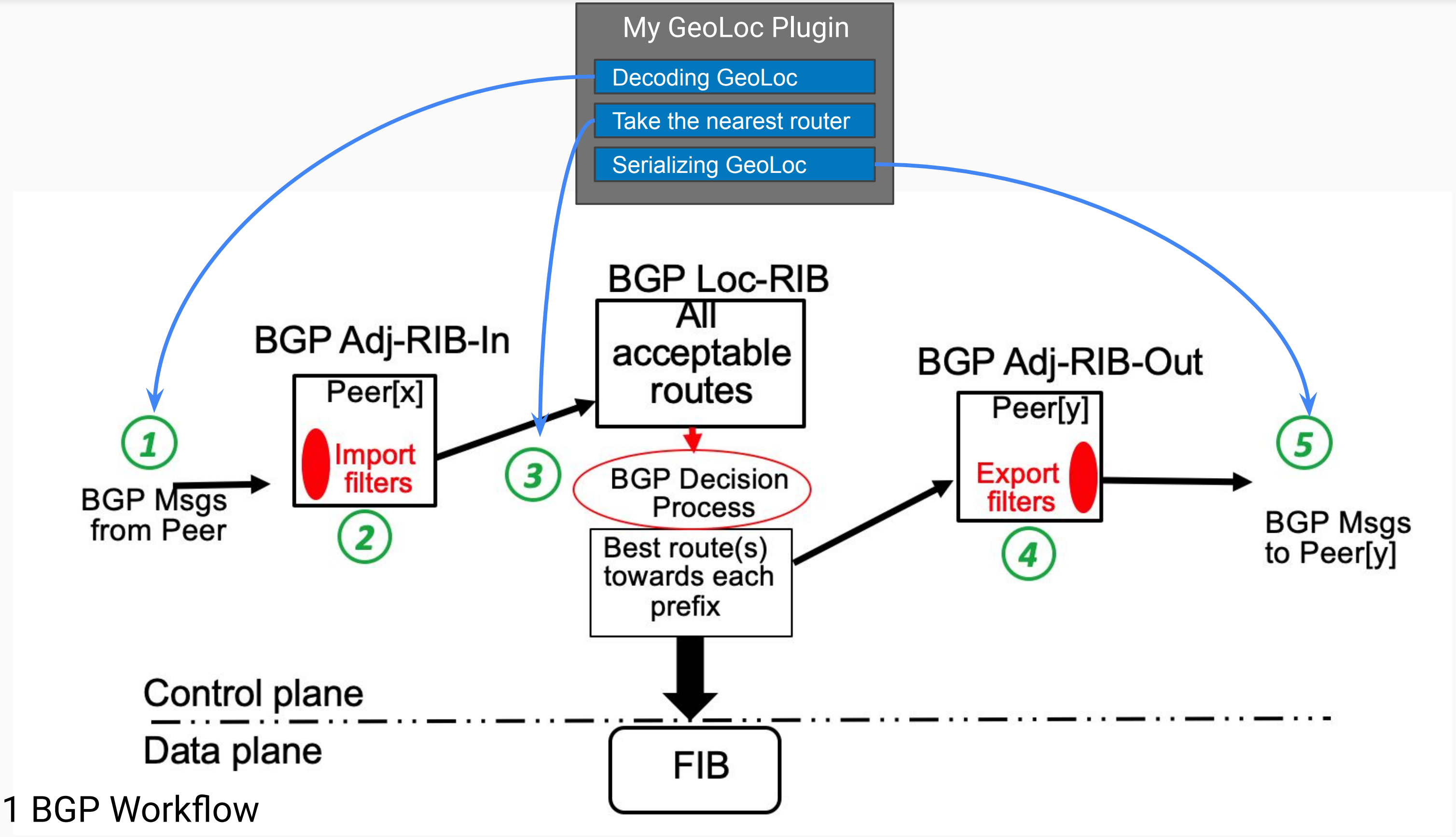
RFC 4271 BGP Workflow

Structure of xBGP



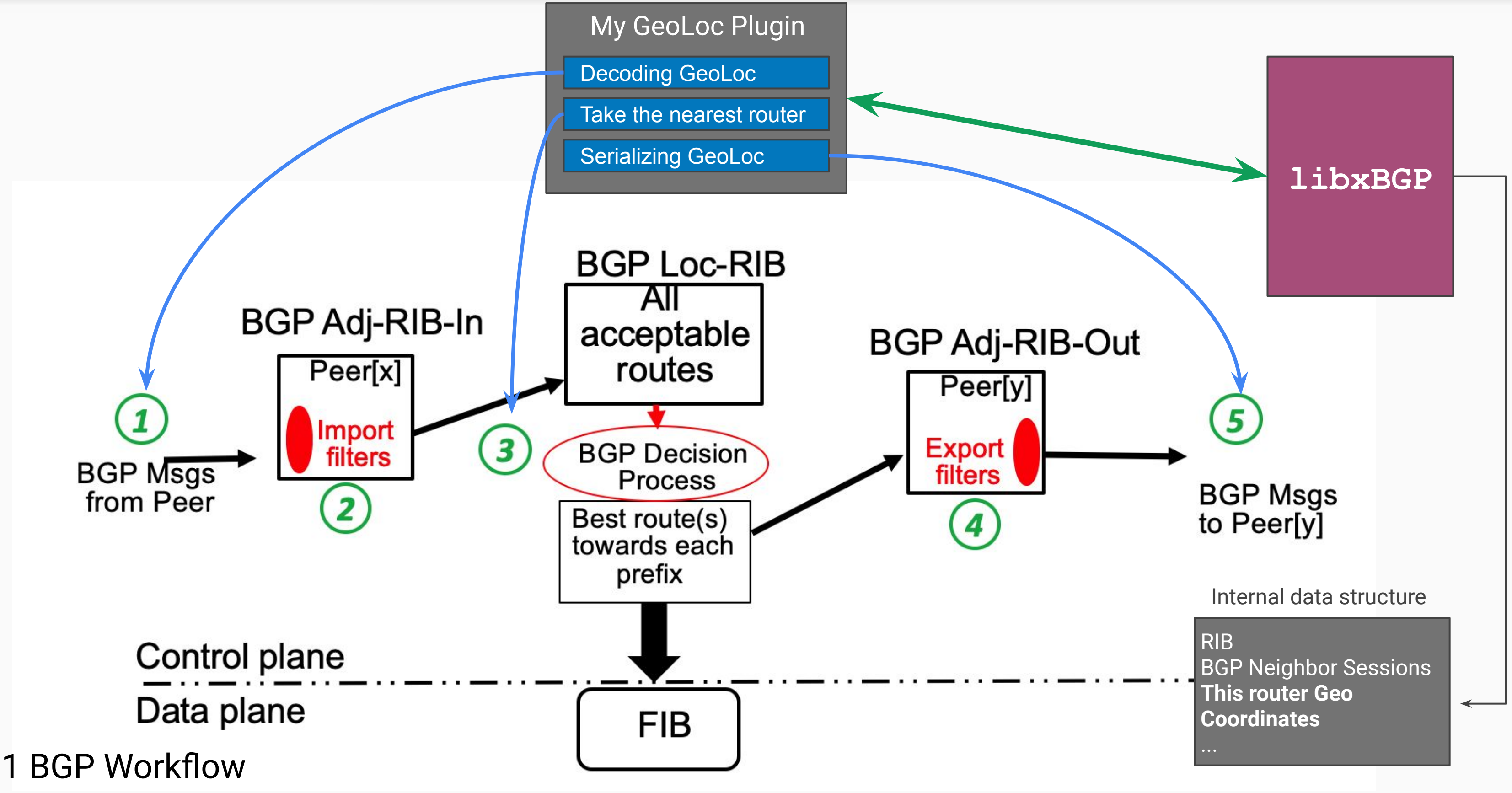
RFC 4271 BGP Workflow

Structure of xBGP



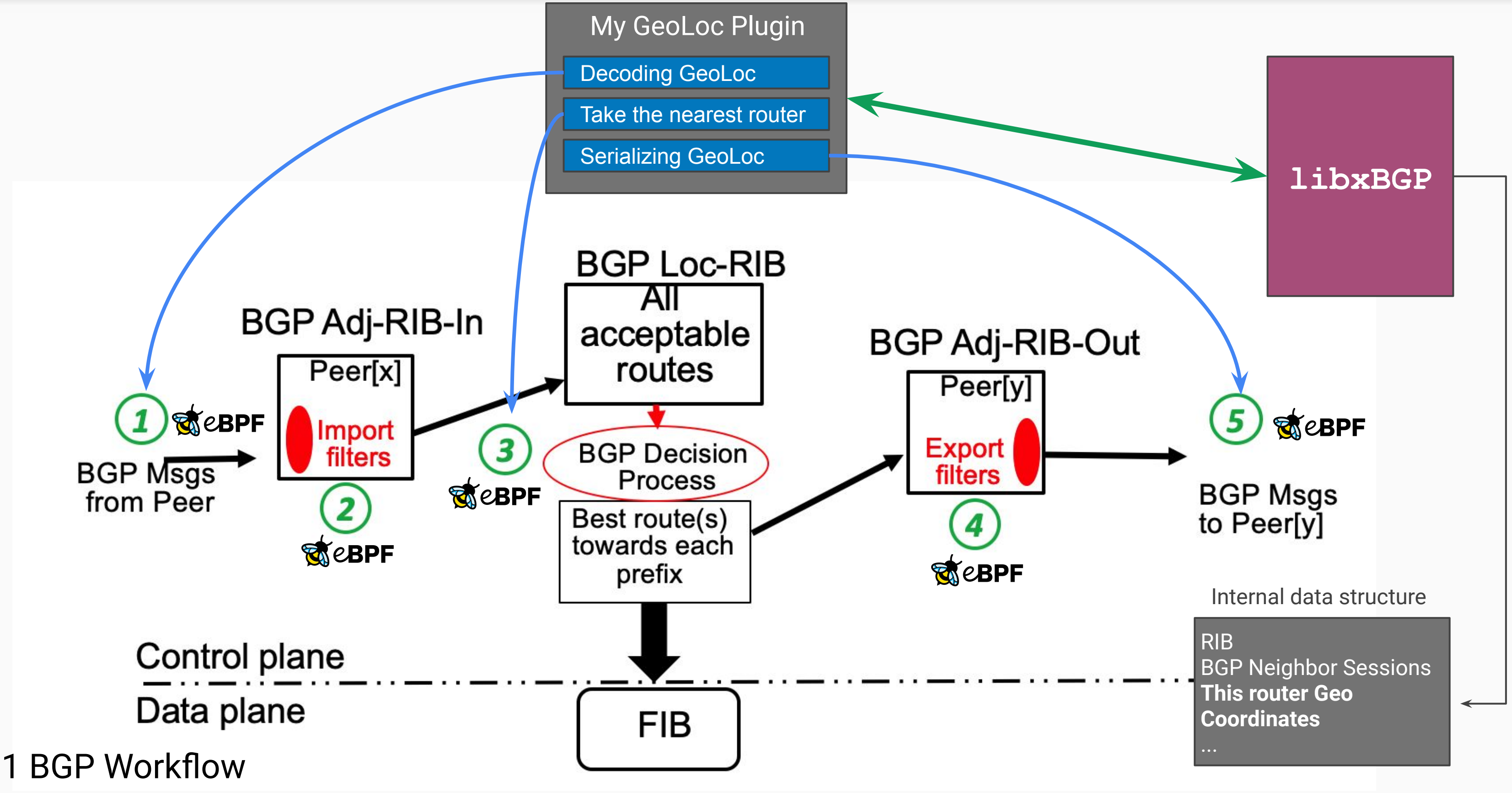
RFC 4271 BGP Workflow

Structure of xBGP



RFC 4271 BGP Workflow

Structure of xBGP



RFC 4271 BGP Workflow

Agenda

- The Weaknesses of the Current Routing Paradigm
- xBGP: a Paradigm Shift
- Adding a new feature with xBGP
- **Uses Cases**

Demonstrating the programmability of xBGP

xBGP requires a little adaptation on the host BGP implementation

We have adapted both FRRouting and BIRD to be xBGP compliant

	FRRouting (LoC)	BIRD Routing (LoC)
Modification to the codebase	30	10
Insertion Points	73	66
Plugin API	624	415
`libxbgp`	3004 + dependencies	
User Space eBPF VM	2776	

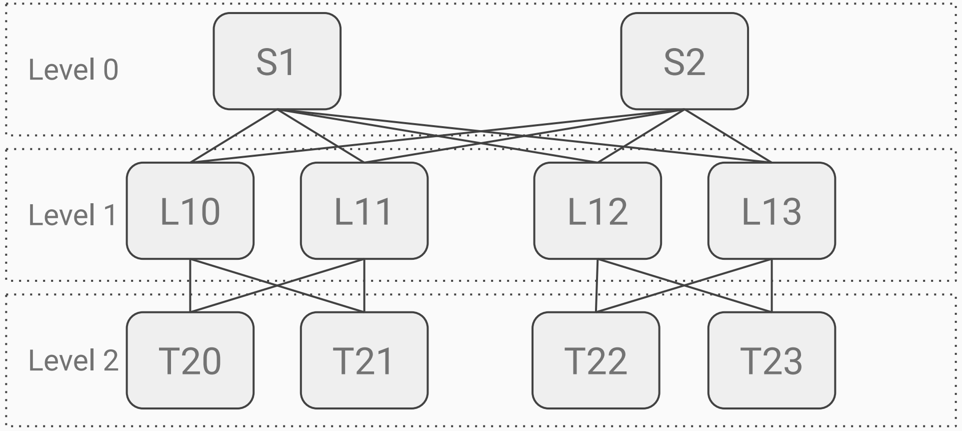
Use Cases

1. Re-implementation of route reflectors (295 LoC)
2. Expressive filters
   - Route Origin Validation (126 LoC)
   - Valley Free path check (81 LoC)
3. GeoTags attribute as MED alternative (261 LoC)

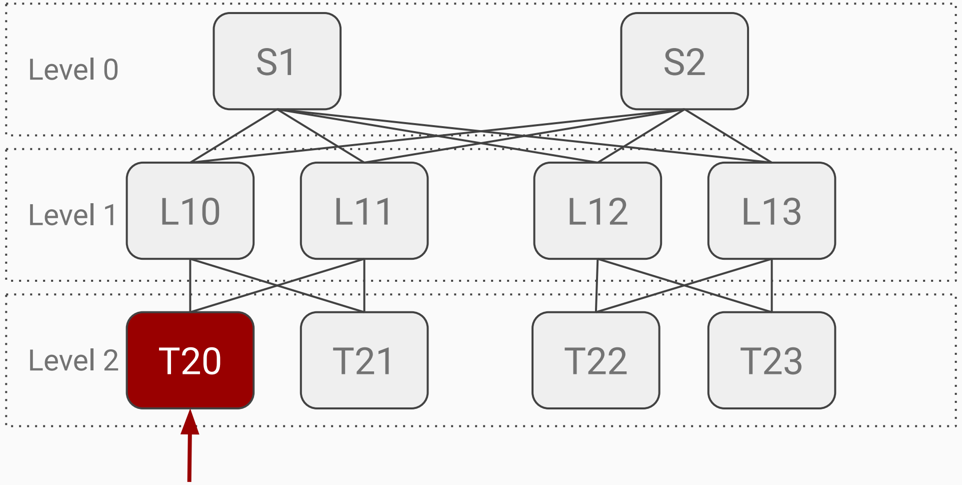
Use Cases

1. Re-implementation of route reflectors (295 LoC)
2. Expressive filters
    - Route Origin Validation (126 LoC)
    - **Valley Free path check (81 LoC)**
3. GeoTags attribute as MED alternative (261 LoC)

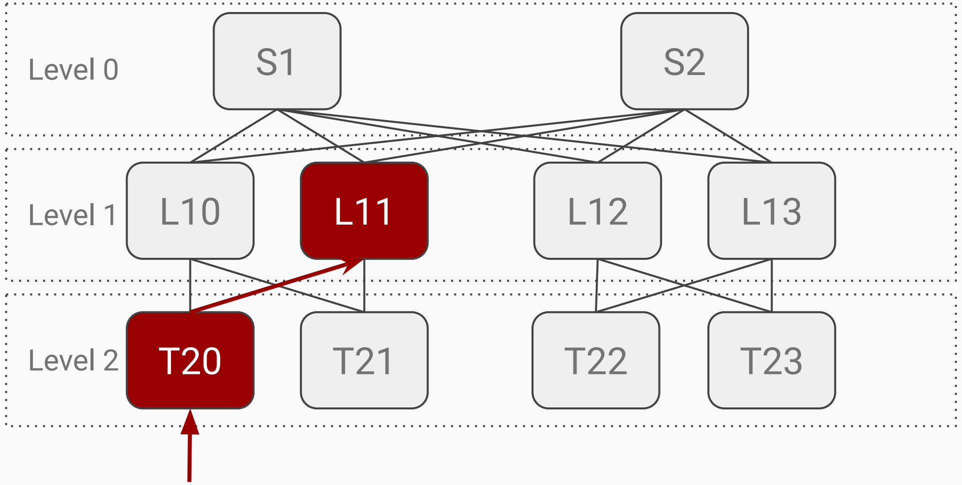
Valley Free path check



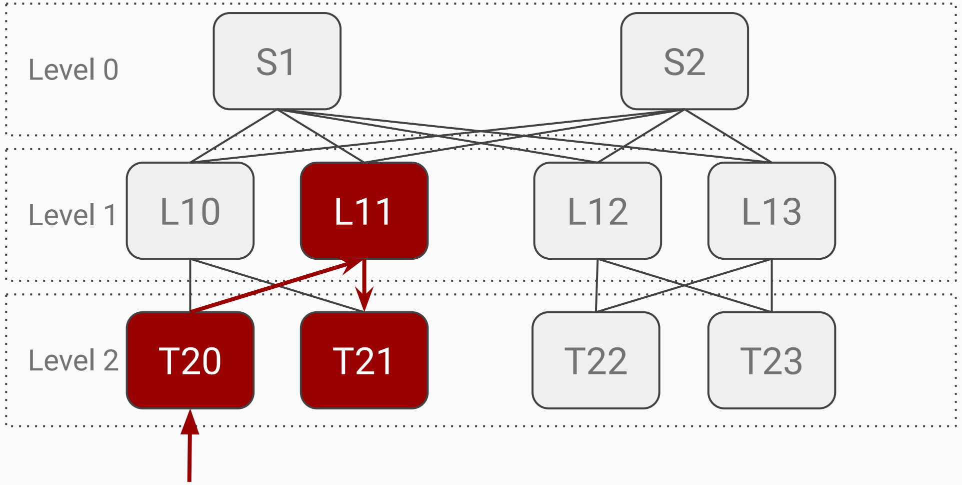
Valley Free path check



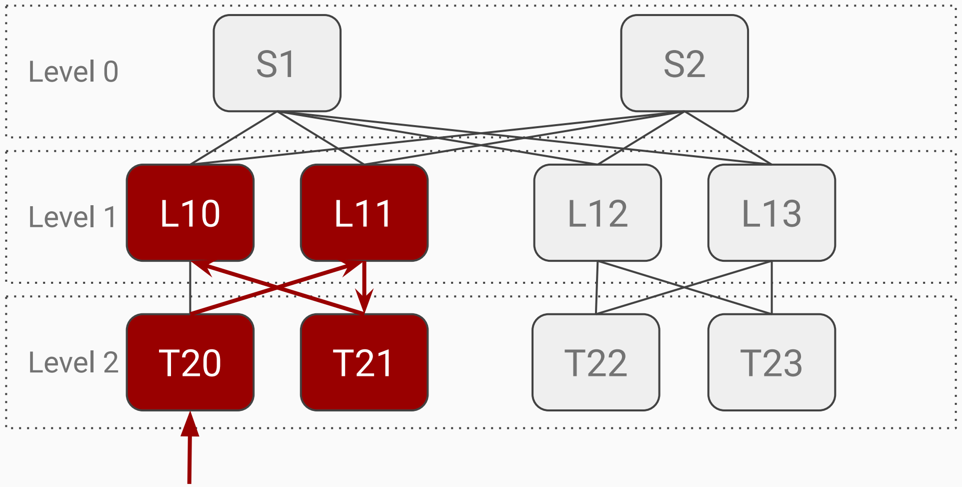
Valley Free path check



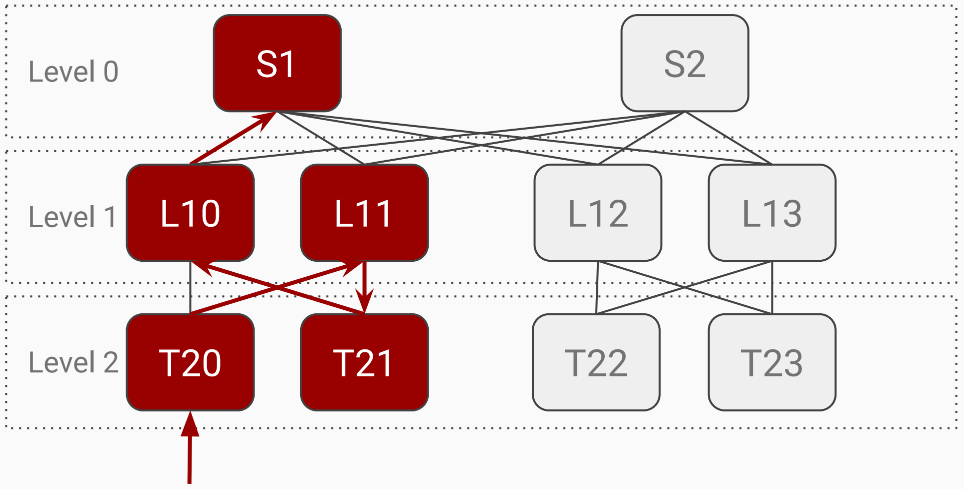
Valley Free path check



Valley Free path check



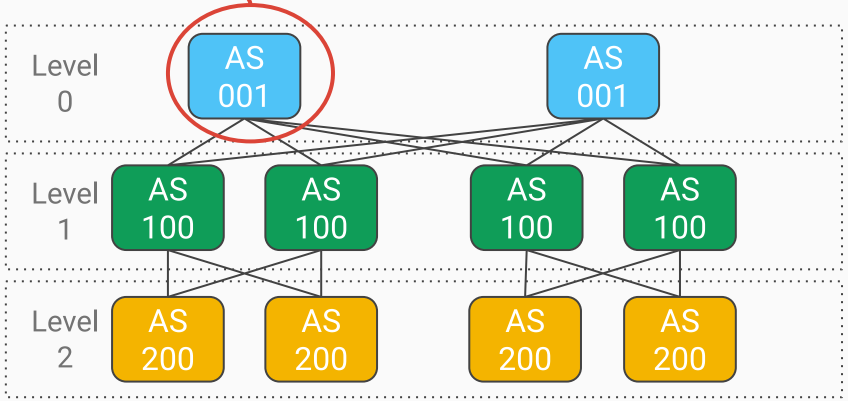
Valley Free path check



Valley Free path check

RFC7938 Use of BGP for Routing in Large-Scale Data Centers

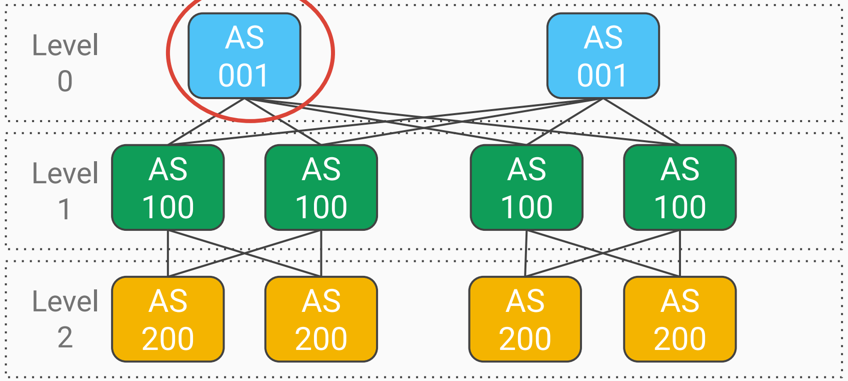
```
MyRouterCli > show ip bgp

BGP Routing table information for VRF default
Router identifier 192.168.254.5, local AS number 1

        Network           Next Hop         Metric   LocPref  Weight   Path
* >Ec 192.168.10.0/24    192.168.255.20   0        100      0        100 200 i
*  ec  192.168.10.0/24   192.168.255.4    0        100      0        100 200 i
* >Ec 192.168.254.3/32   192.168.255.4    1        100      0        100 200 i
*  ec  192.168.254.3/32  192.168.255.20   0        100      0        100 200 i
* >Ec 192.168.254.4/32   192.168.255.20   0        100      0        100 200 i
```



Valley Free path check

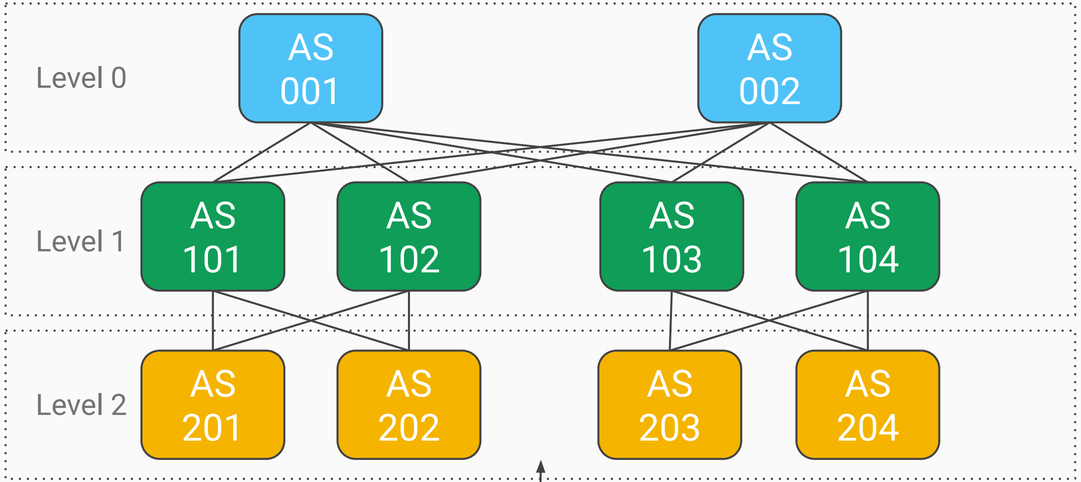
RFC7938 Use of BGP for Routing in Large-Scale Data Centers

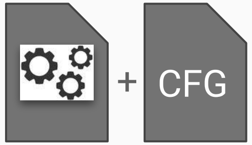
```
MyRouterCli > show ip bgp

BGP Routing table information for VRF default
Router identifier 192.168.254.5, local AS number 1

       Network              Next Hop          Metric   LocPref  Weight   Path
* >Ec 192.168.10.0/24    192.168.255.20   0        100      0        100 200 i
*  ec  192.168.10.0/24   192.168.255.4    0        100      0        100 200 i
* >Ec 192.168.254.3/32   192.168.255.4    1        100      0        100 200 i
*  ec  192.168.254.3/32  192.168.255.20   0        100      0        100 200 i
* >Ec 192.168.254.4/32   192.168.255.20   0        100      0        100 200 i
```



Where are these routes sourced from ?

Valley Free path check with xBGP



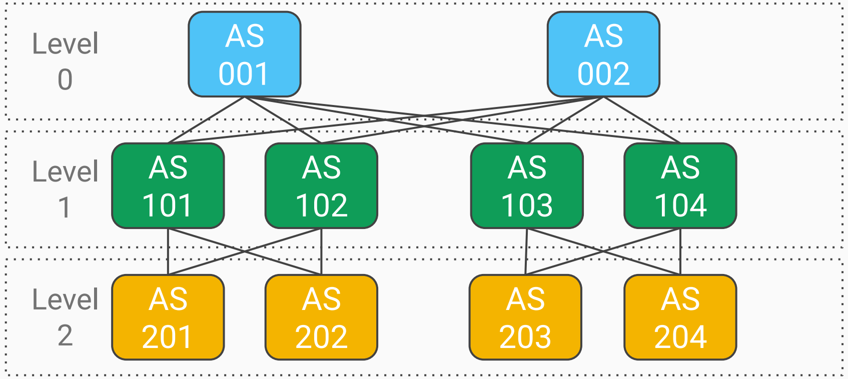


(81 LoC)

One plugin + one topology manifest
for all routers !

Valley Free path check with xBGP

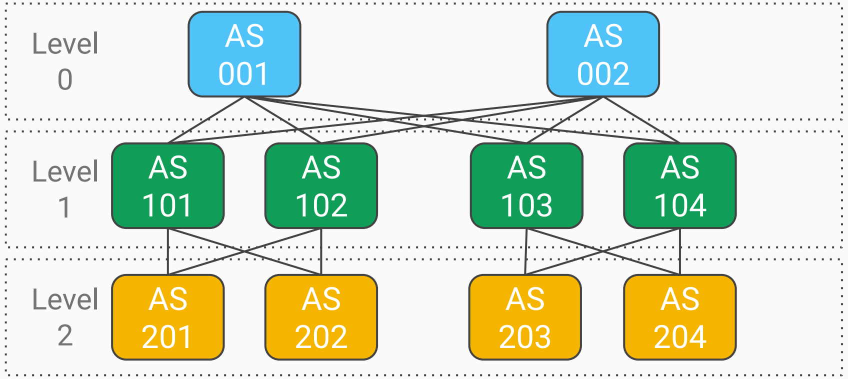
```
uint64_t valley_free_check(args_t *args UNUSED) {
 /* variable declaration omitted  */
 attr = get_attr_from_code(AS_PATH_ATTR_CODE);
 peer = get_src_peer_info();
 if (!attr || !peer) return FAIL;

 my_as = peer->local_bgp_session->as;
 as_path = attr->data;
 as_path_len = attr->len;

 while (i < as_path_len) {
   i++; /* omit segment type  */
   segment_length = as_path[i++];
   for (j = 0; j < segment_length - 1; j++) {
     curr_as = get_u32(as_path + i);
     i += 4;
     if (!valley_check(next_as, curr_as)) return PLUGIN_FILTER_REJECT;
   }
 }
 next();
 return FAIL;
}
```



Valley Free path check with xBGP

```
uint64_t valley_free_check(args_t *args UNUSED) {
 /* variable declaration omitted  */
 attr = get_attr_from_code(AS_PATH_ATTR_CODE);
 peer = get_src_peer_info();
 if (!attr || !peer) return FAIL;

 my_as = peer->local_bgp_session->as;
 as_path = attr->data;
 as_path_len = attr->len;

 while (i < as_path_len) {
   i++; /* omit segment type  */
   segment_length = as_path[i++];
   for (j = 0; j < segment_length - 1; j++) {
     curr_as = get_u32(as_path + i);
     i += 4;
     if (!valley_check(next_as, curr_as)) return PLUGIN_FILTER_REJECT;
   }
 }
 next();
 return FAIL;
}
```

Retrieve data from the host implementation

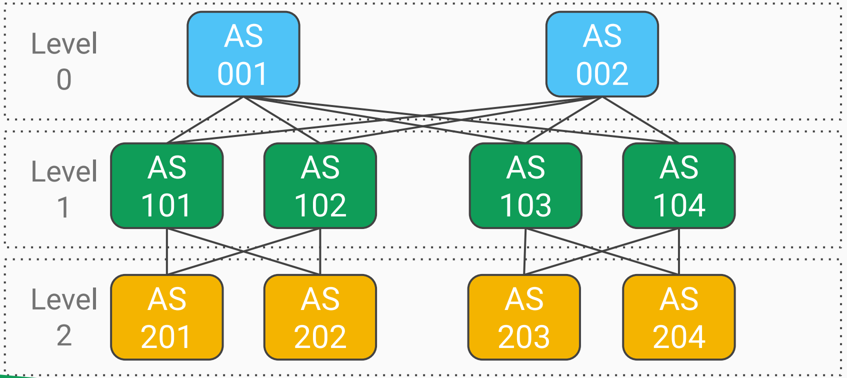


Valley Free path check with xBGP

```
uint64_t valley_free_check(args_t *args UNUSED) {
 /* variable declaration omitted  */
 attr = get_attr_from_code(AS_PATH_ATTR_CODE);
 peer = get_src_peer_info();
 if (!attr || !peer) return FAIL;

 my_as = peer->local_bgp_session->as;
 as_path = attr->data;
 as_path_len = attr->len;

 while (i < as_path_len) {
   i++; /* omit segment type  */
   segment_length = as_path[i++];
   for (j = 0; j < segment_length - 1; j++) {
     curr_as = get_u32(as_path + i);
     i += 4;
     if (!valley_check(next_as, curr_as)) return PLUGIN_FILTER_REJECT;
   }
 }
 next();
 return FAIL;
}
```

Retrieve data from the host implementation

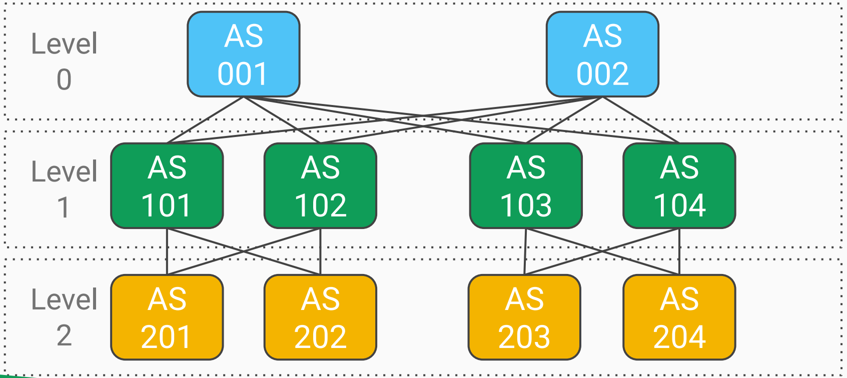
Main processing of the plugin



Valley Free path check with xBGP

```
uint64_t valley_free_check(args_t *args UNUSED) {
 /* variable declaration omitted  */
 attr = get_attr_from_code(AS_PATH_ATTR_CODE);
 peer = get_src_peer_info();
 if (!attr || !peer) return FAIL;

 my_as = peer->local_bgp_session->as;
 as_path = attr->data;
 as_path_len = attr->len;

 while (i < as_path_len) {
   i++; /* omit segment type  */
   segment_length = as_path[i++];
   for (j = 0; j < segment_length - 1; j++) {
     curr_as = get_u32(as_path + i);
     i += 4;
     if (!valley_check(next_as, curr_as)) return PLUGIN_FILTER_REJECT;
   }
 }
 next();
 return FAIL;
}
```

Retrieve data from the host implementation



Main processing of the plugin

The route is rejected if such a pair exists

Conclusion

xBGP proposes a new methodology to upgrade routing protocols

xBGP provides new opportunities with other routing protocols

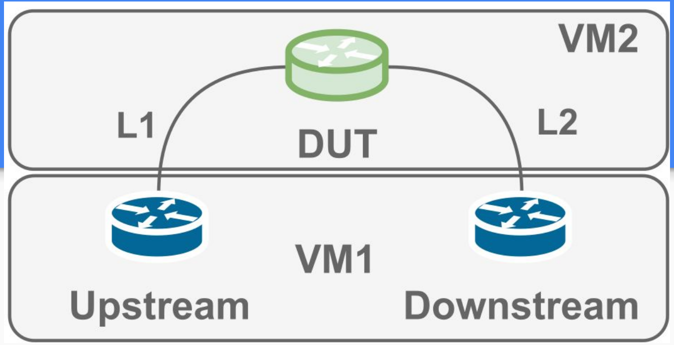
From a **monolithic** to a **modular** approach

The next steps:
- Standardizing the API + the VM
- New use cases

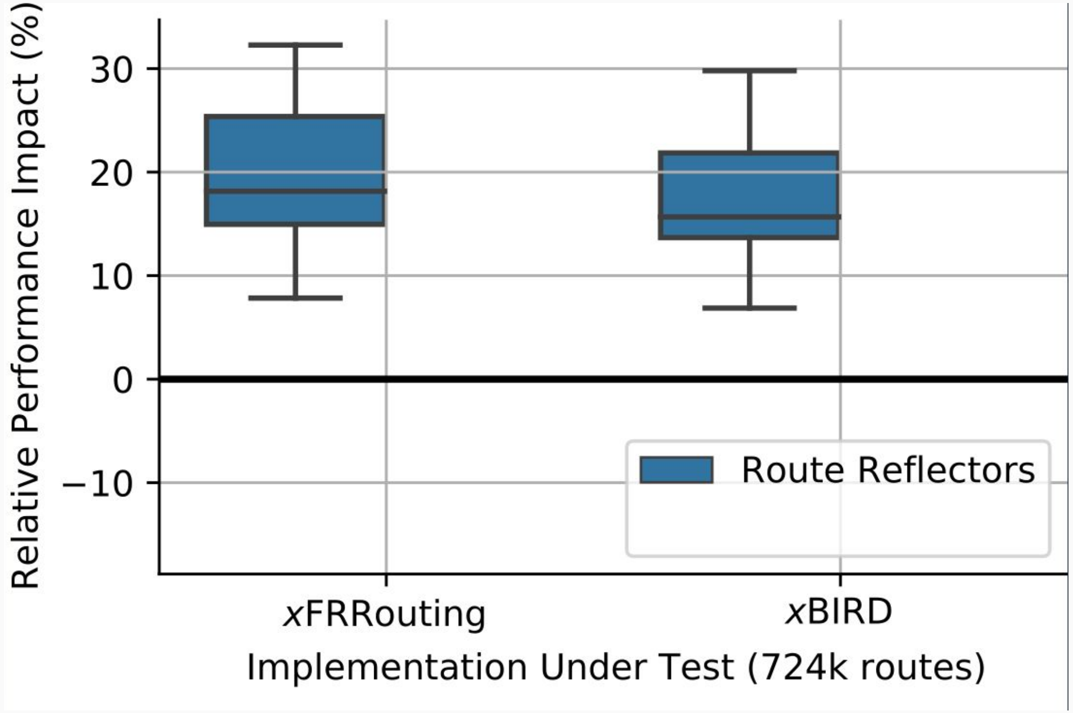
See https://www.pluginized-protocols.org/xbgp for the latest updates and the source code

Backup slides

Comparison with native code

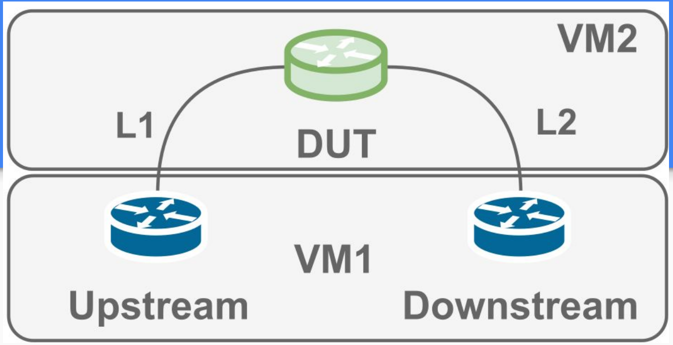


$$\frac{\text{Time xBGP - Time Native}}{\text{Time Native}} \times 100$$

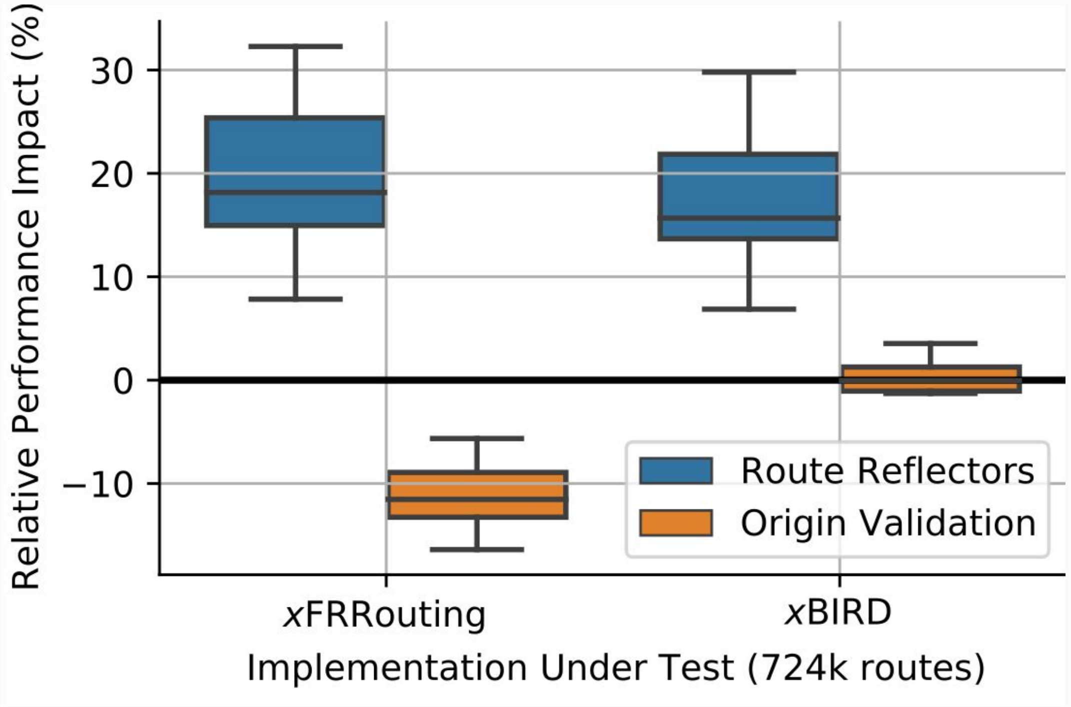


Lower is better

Comparison with native code

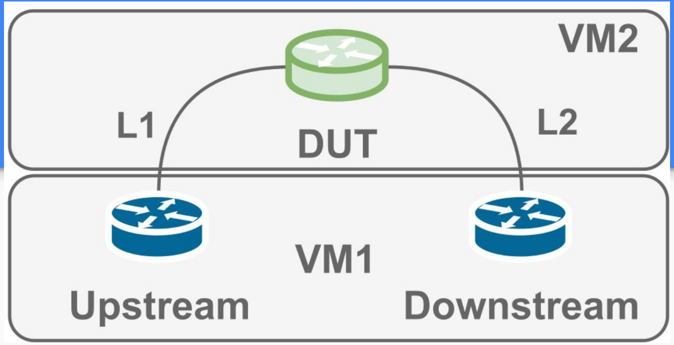


$$\frac{\text{Time xBGP - Time Native}}{\text{Time Native}} \times 100$$

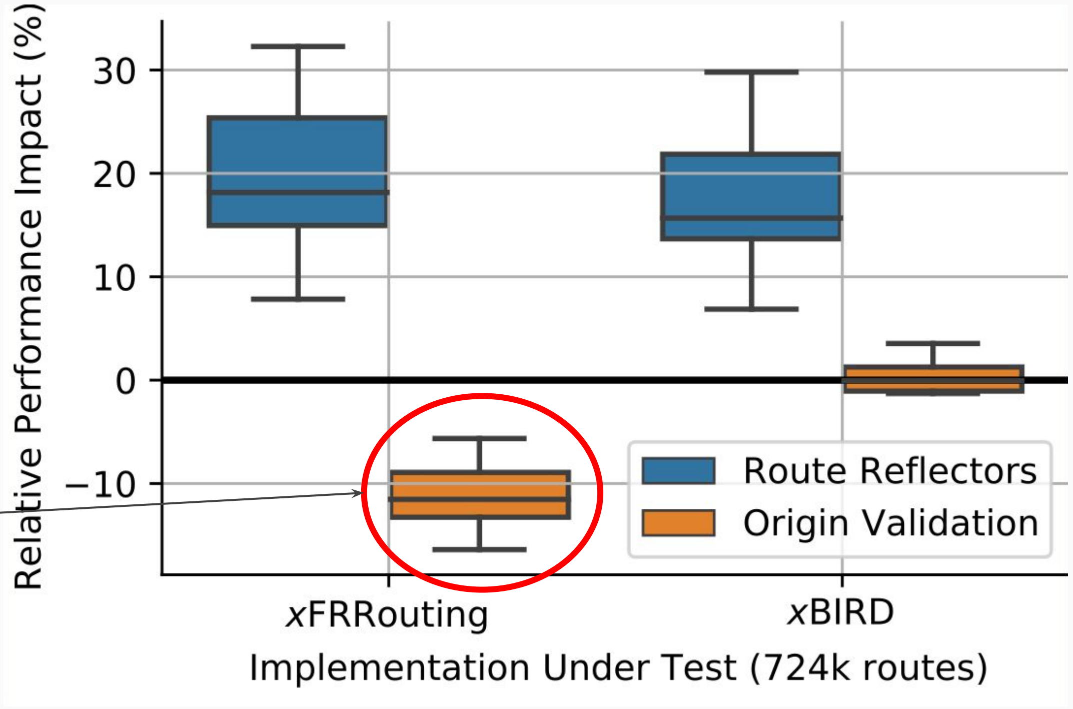


Lower is better

Comparison with native code



$$\frac{\text{Time xBGP - Time Native}}{\text{Time Native}} \times 100$$

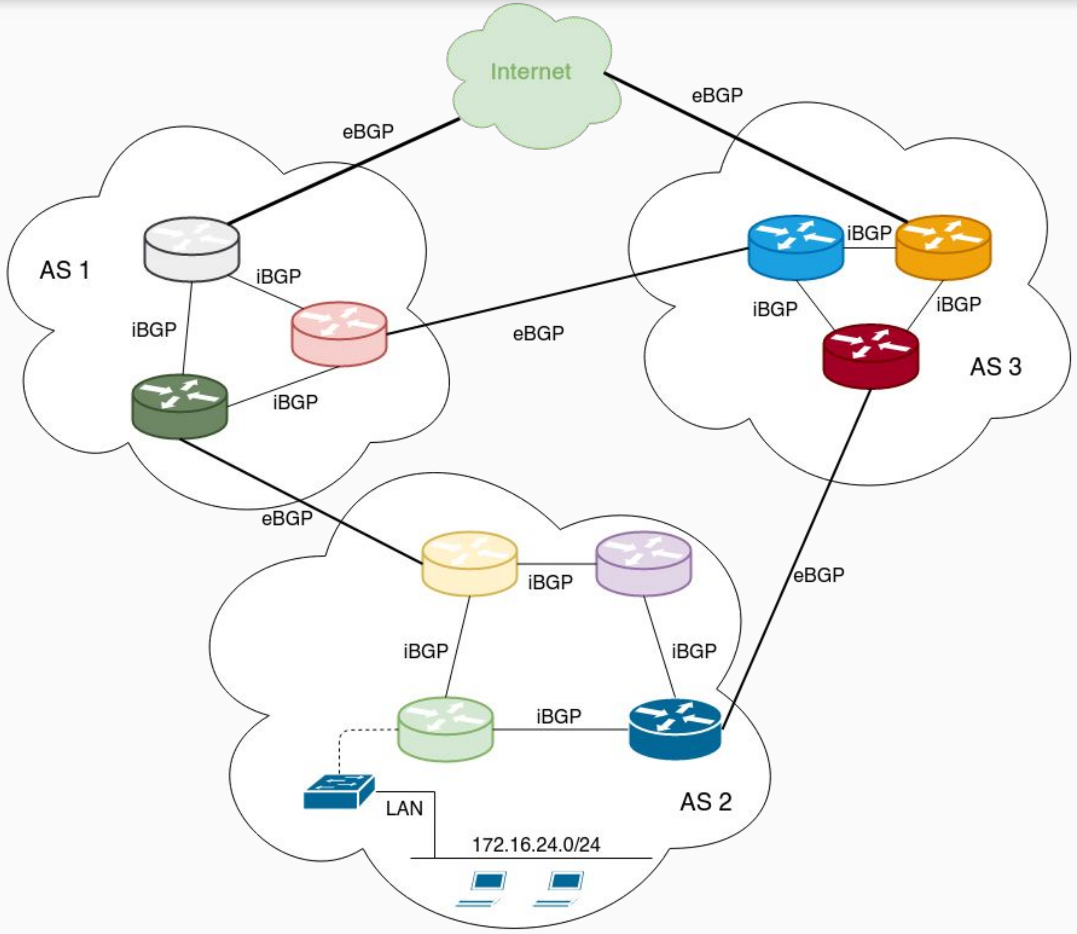


Native code uses a slower data structure

Lower is better

Old slides

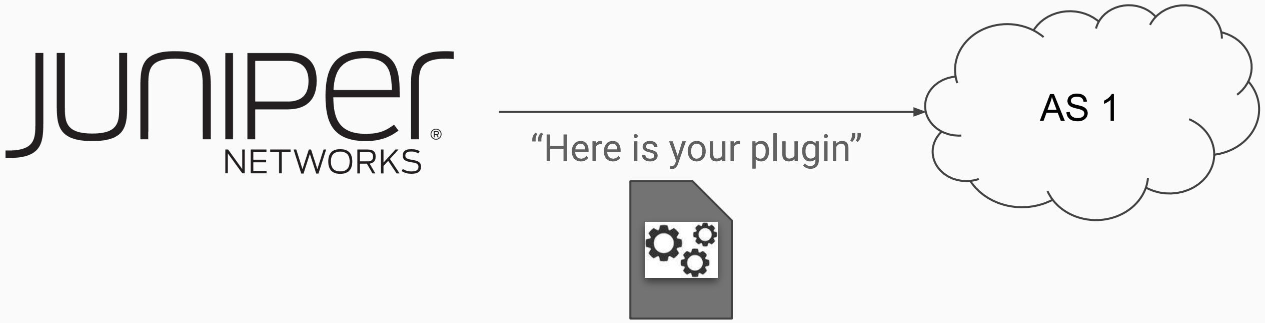
BGP enables routing on the Internet



xBGP: toward a paradigm shift

Extending a protocol is complex.

Why not offer operators the opportunity to program/update their own extensions?


JUNIPER NETWORKS
"Here is your plugin"
AS 1