

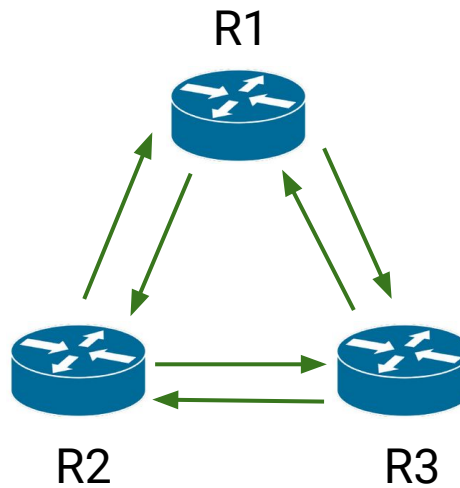
The Case For Pluginized Routing Protocols

Thomas Wirtgen, Cyril Dénos, Quentin De Coninck, Mathieu Jadin,
Olivier Bonaventure



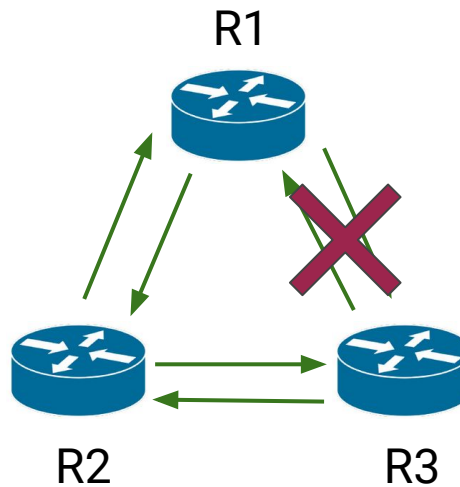
Routing Protocols

- Select a path to forward packets to their destination



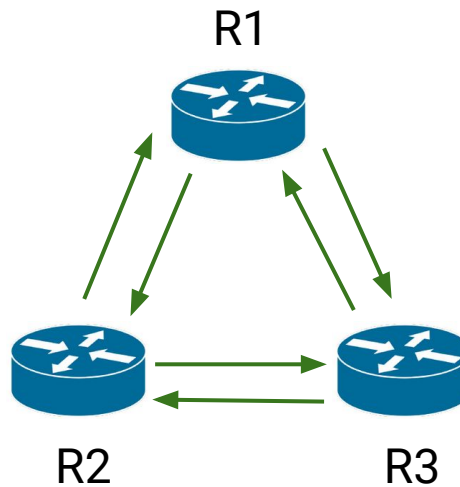
Routing Protocols

- Select a path to forward packets to their destination
- Fault tolerant technology → Automatic reconfiguration



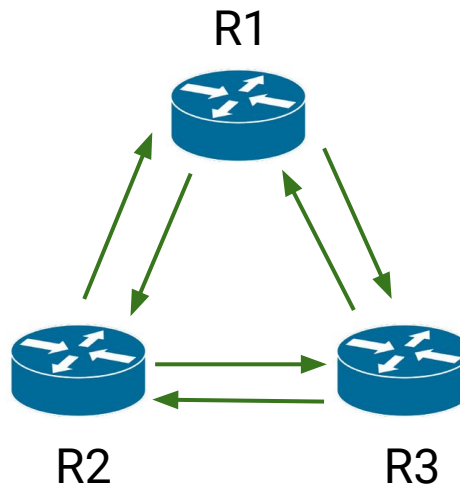
Routing Protocols

- Select a path to forward packets to their destination
- Fault tolerant technology → Automatic reconfiguration
- Run 24h/24, 7d/7



Routing Protocols

- Select a path to forward packets to their destination
- Fault tolerant technology → Automatic reconfiguration
- Run 24h/24, 7d/7
- Mature and scalable



Routing Protocols

BGP

Border Gateway Protocol

- Path Vector Routing
- External Gateway Protocol

OSPF

Open Shortest Path First

- Link State Routing
- Internal Gateway Protocol

Agenda

- **Objective**
- Deployment of new features
- Use cases and evaluation
- Conclusion

Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

- Used to tag routes during their advertisements on the network

Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

- Used to tag routes during their advertisements on the network
- Classical BGP communities are represented with 32 bits :
 - 16 bits ASn : 16 informational bits (e.g. '5698:111')

Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

- Used to tag routes during their advertisements on the network
- Classical BGP communities are represented with 32 bits :
 - 16 bits ASn : 16 informational bits (e.g. '5698:111')
- **Large BGP Communities = 32 bits AS : 16 bits (48 bits)**
 - e.g. '1234:5678:111'

Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

December 2002:

[draft-lange-flexible-bgp-communities-00](#)



Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities

December 2002:

[draft-lange-flexible-bgp-communities-00](#)

2009:

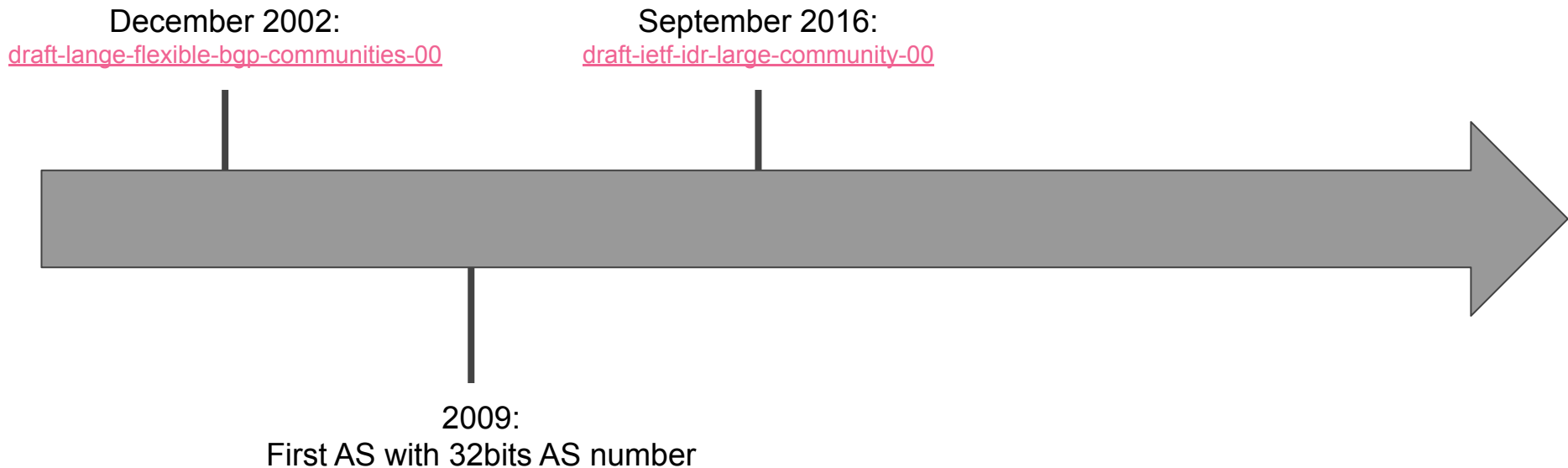
First AS with 32bits AS number



Difficult to innovate in BGP

Current situation : large delay for new propositions

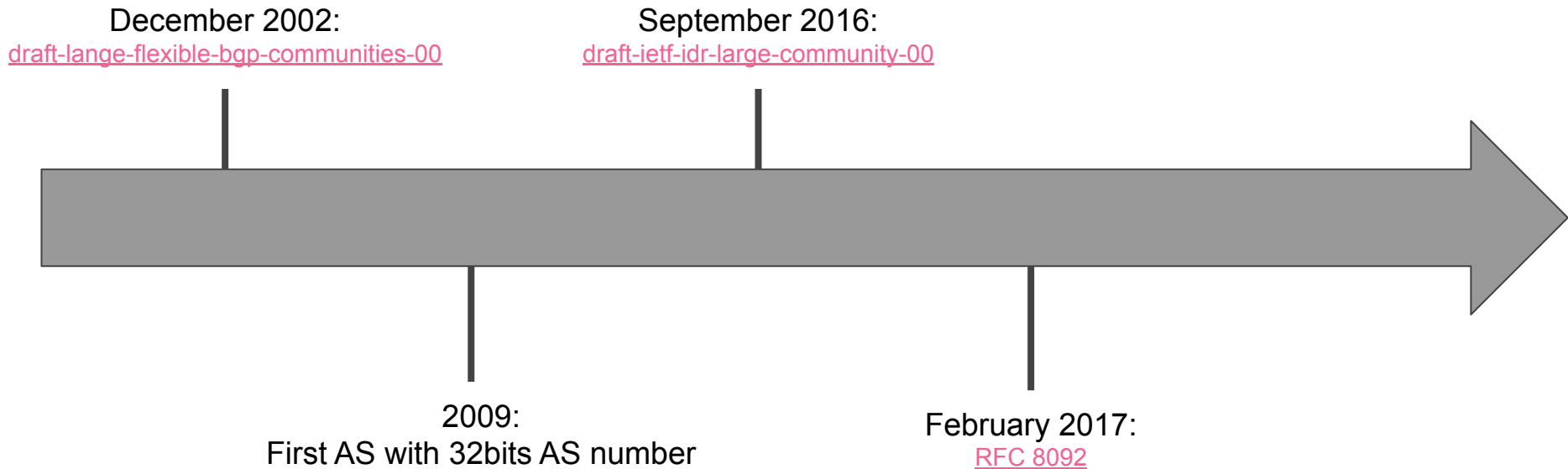
Example : Adding BGP Large Communities



Difficult to innovate in BGP

Current situation : large delay for new propositions

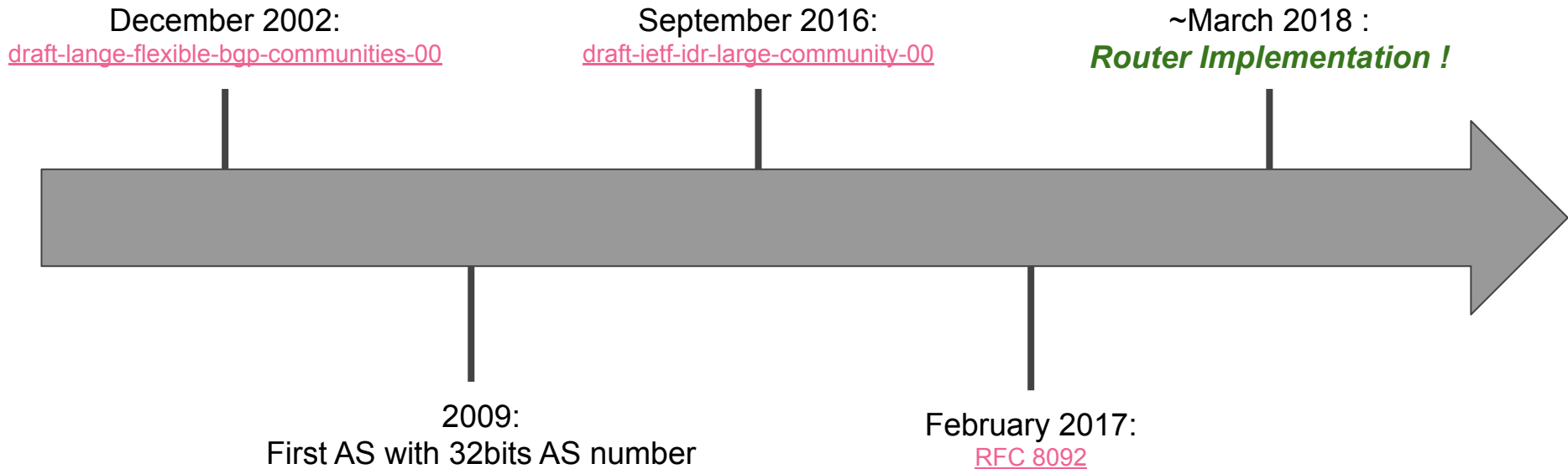
Example : Adding BGP Large Communities



Difficult to innovate in BGP

Current situation : large delay for new propositions

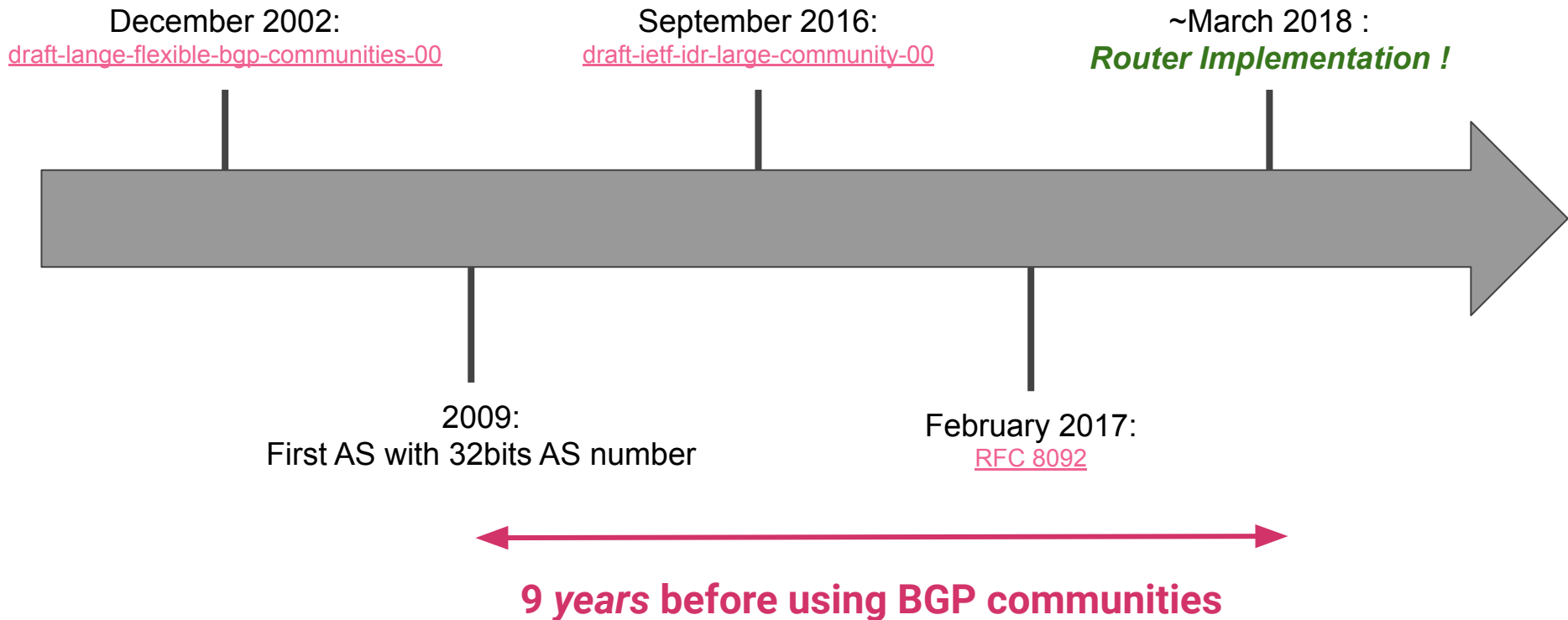
Example : Adding BGP Large Communities



Difficult to innovate in BGP

Current situation : large delay for new propositions

Example : Adding BGP Large Communities



Difficult to innovate in protocols

Current situation : large delay for new propositions

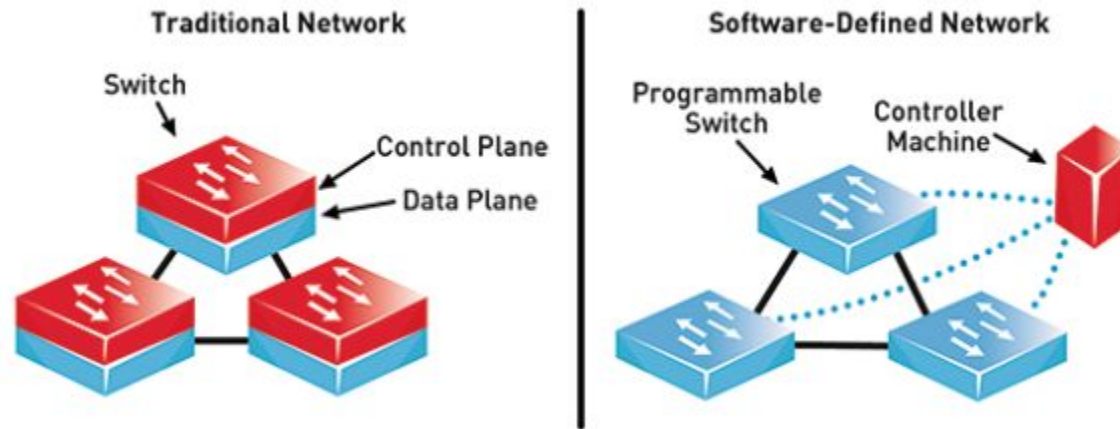
Due to the current procedures of standardisation:

1. Discuss with router vendors for a new feature
2. Convince standardisation team
3. Convince router vendors
4. Wait for standardisation
5. Wait for implementation
6. Upgrade routers OS
7. New feature

Objective

- Reducing innovation delay
- Speeding up the deployment of new features
- Making customisation of protocol easier
- Reconfiguring protocol at runtime

Innovating with Software Defined Networking (SDN)



An SDN Perspective to Mitigate the Energy Consumption of Core Networks – GEANT2 - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Traditional-Network-versus-SDN_fig1_319876305

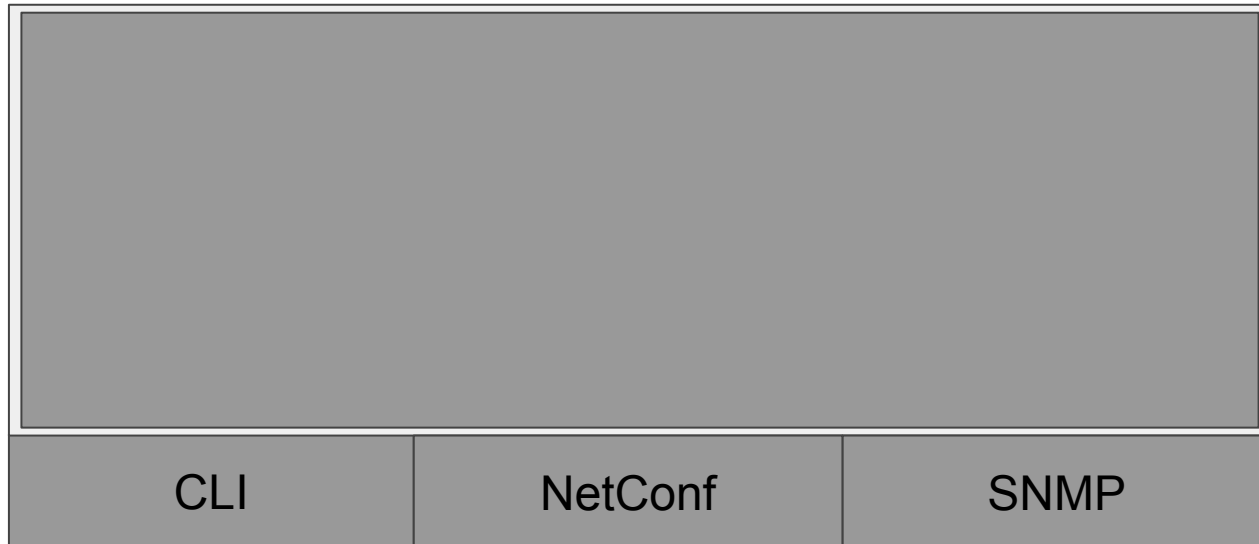
- Centralised routing
- Scalability and performance issues

Agenda

- Objective
- **Deployment of new features**
- Use cases and evaluation
- Conclusion

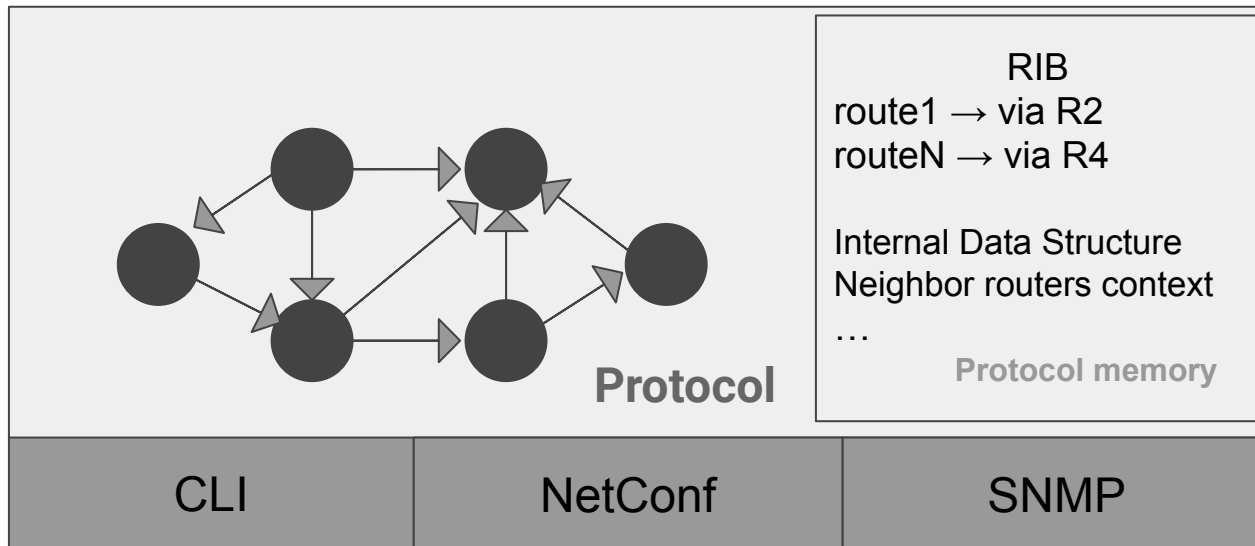
Deployment of new features

Protocols representation



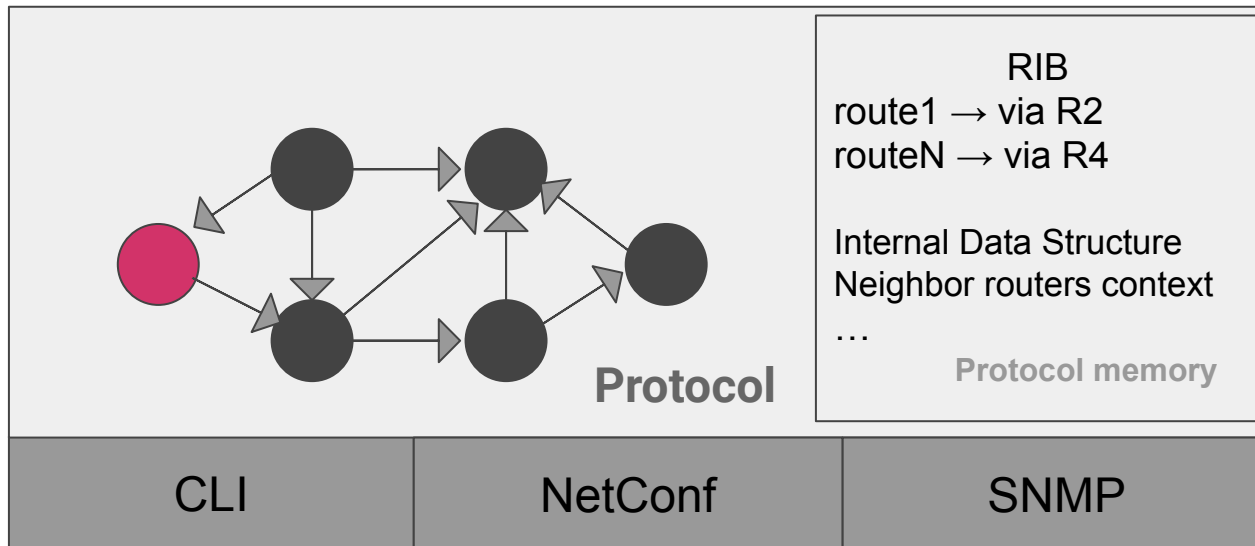
Deployment of new features

Protocols can be seen as a finite state machine



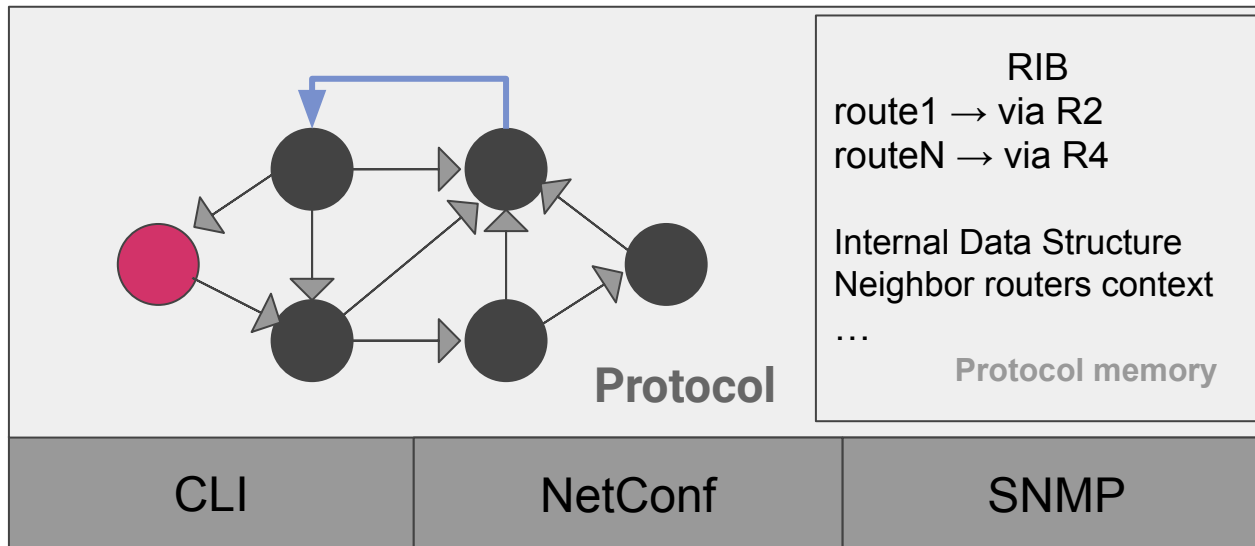
Deployment of new features

Modify in a fine grained way some protocol states



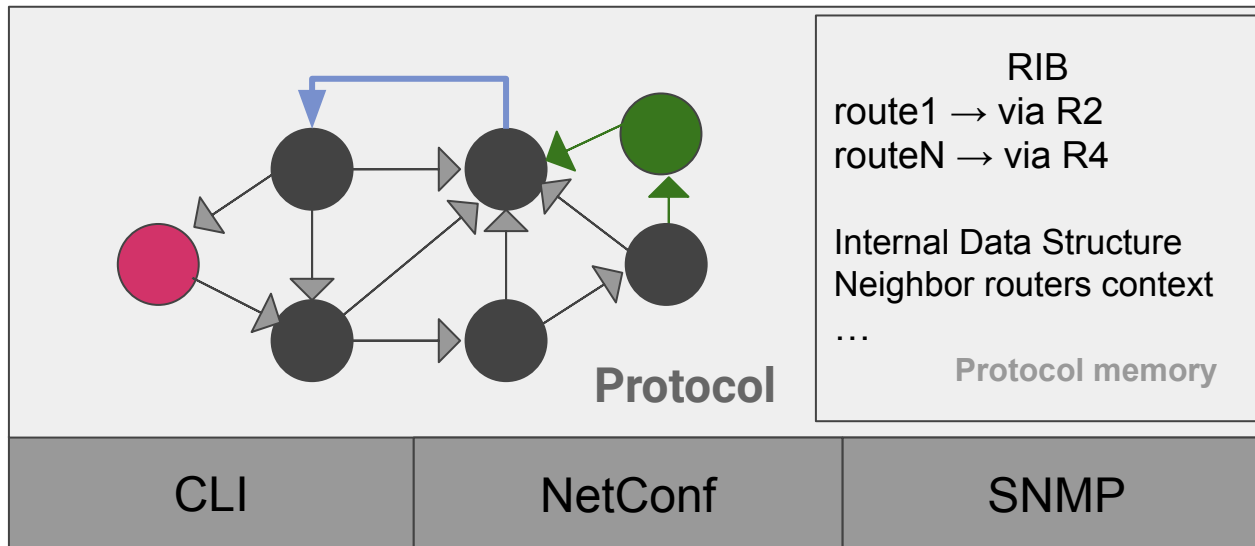
Deployment of new features

Add in a fine grained way **new transitions**

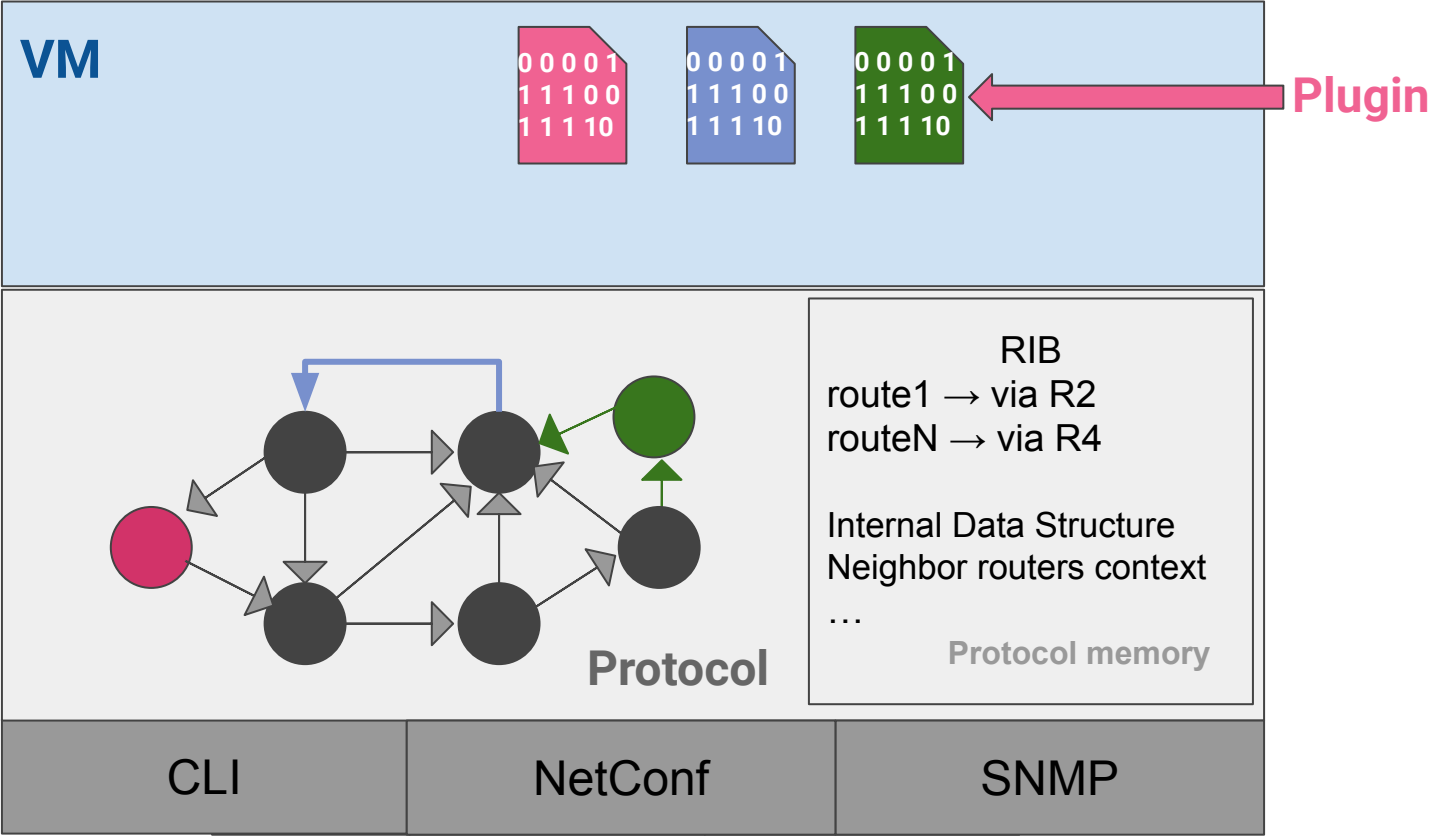


Deployment of new features

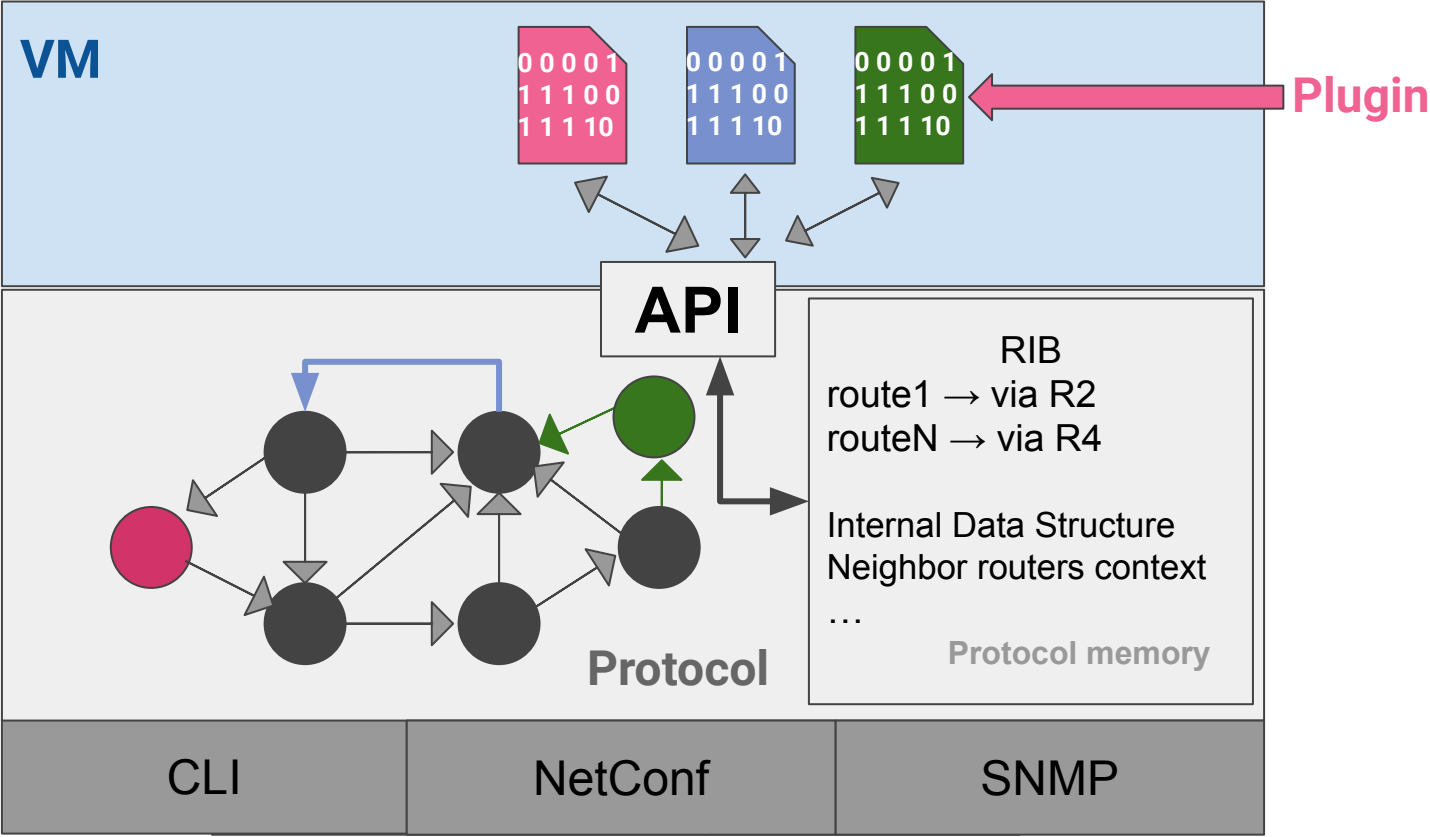
Add in a fine grained way **new** protocol states



Deployment of new features



Deployment of new features



Plugin execution environment

eBPF based Virtual Machine :

- Lightweight and mature
 - eBPF is used in the Linux Kernel
- Compiled to native CPU architecture
 - eBPF → x86_64
- Platform Independent

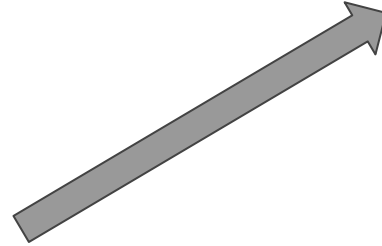
Example: adding *Monitoring*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```

```
int decision_process(args)
{
    // code
    // ...
    // ...
    // ...
    return something;
}
```

Example: adding *Monitoring*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```



```
int decision_process(args)
{
    // code
    // ...
    // ...
    // ...
    return something;
}
```

`time_t start = time(NULL);`

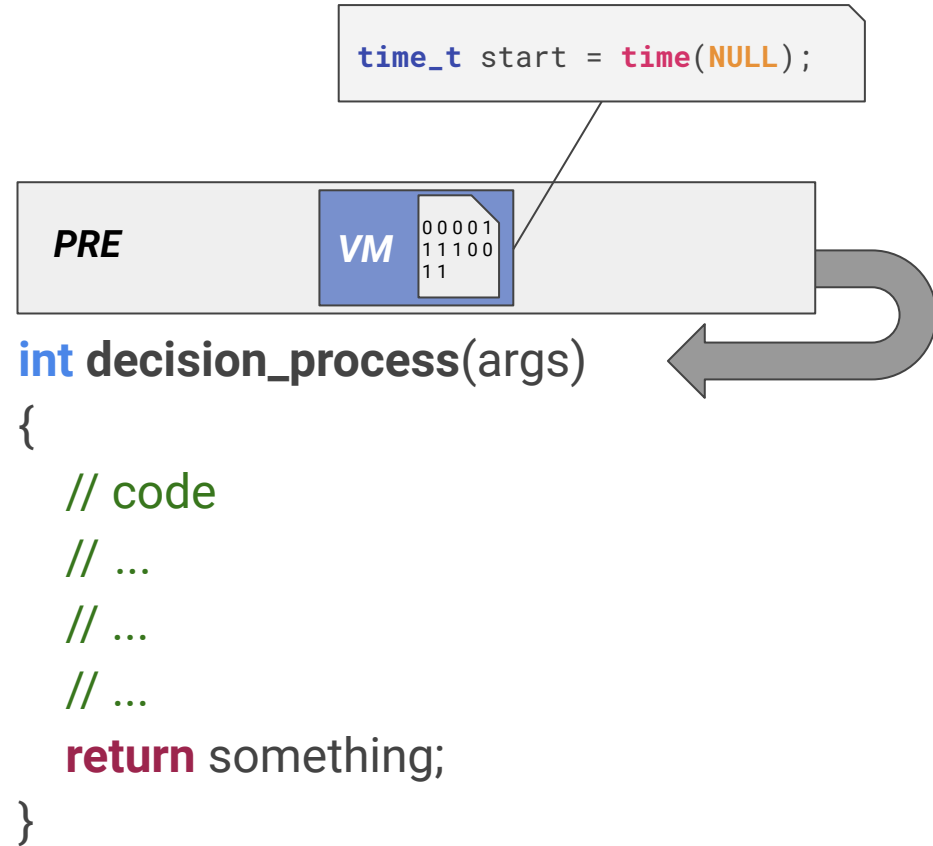
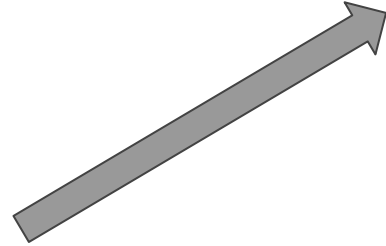
PRE

VM

00001
11100
11

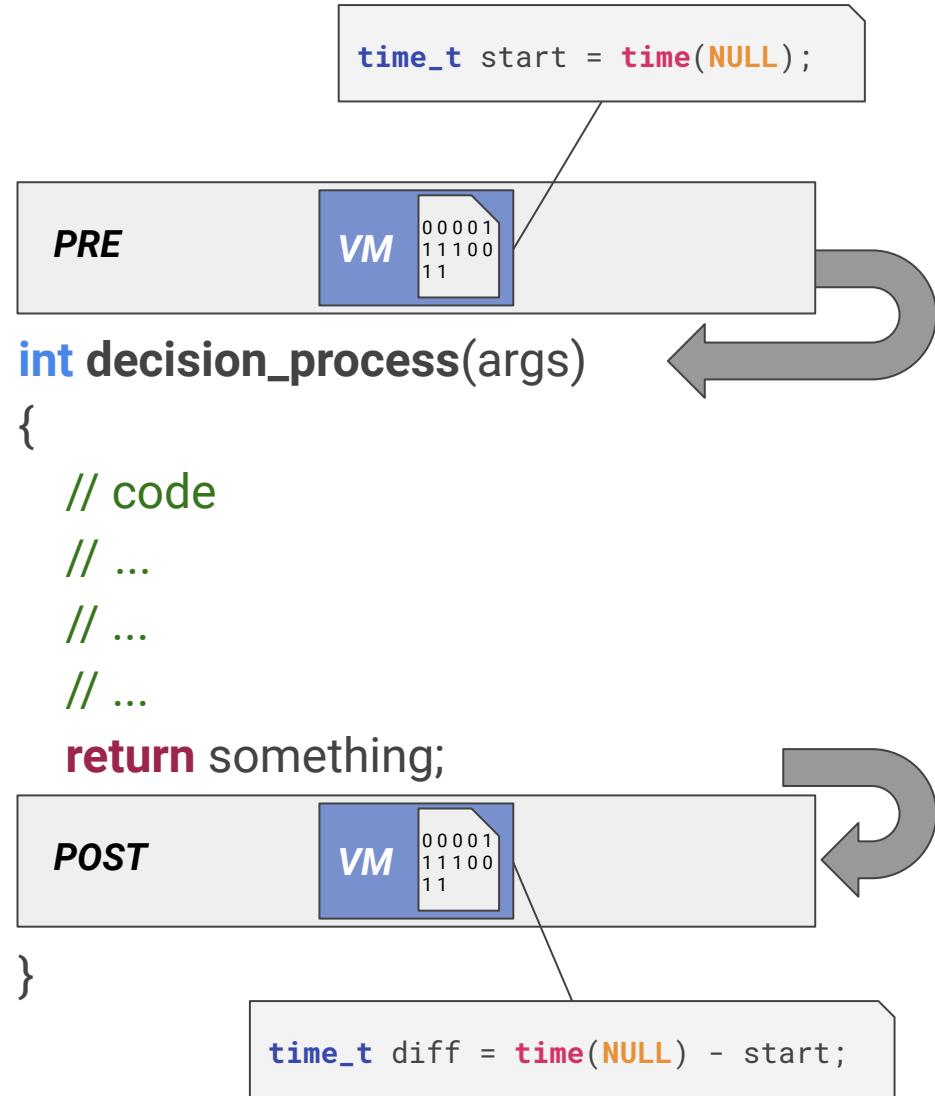
Example: adding *Monitoring*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```



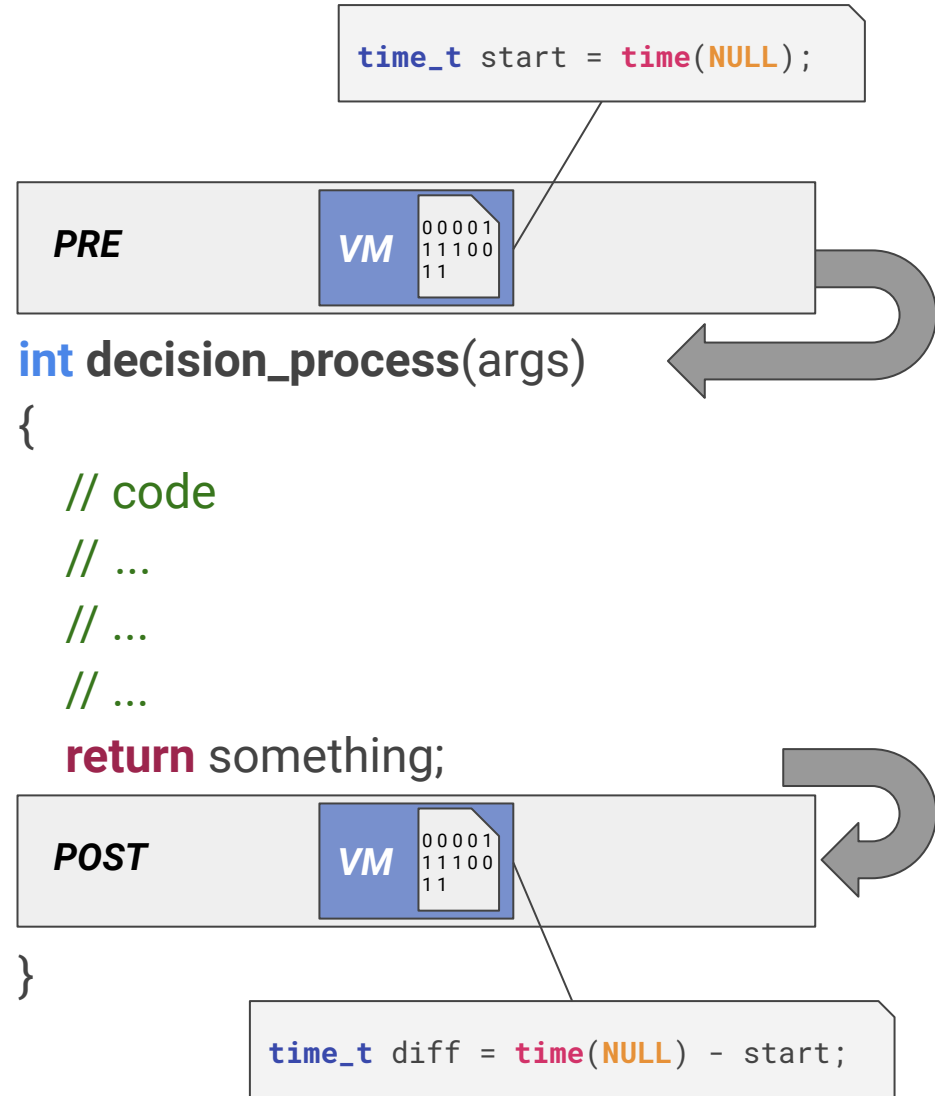
Example: adding *Monitoring*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```



Example: adding *Monitoring*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```



Example: *protocol function replacement*


```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```

```
int decision_process(args)
{
    // code
    // ...
    // ...
    // ...
    return something;
}
```

Example: *protocol function replacement*

```
int bgp_update(args)
{
    // code
    r = decision_process(args);
    // ...
    // end of function
}
```

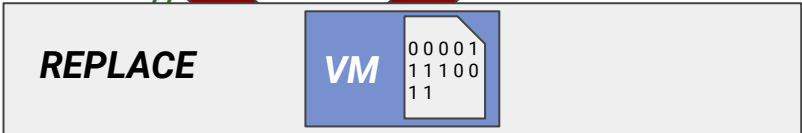
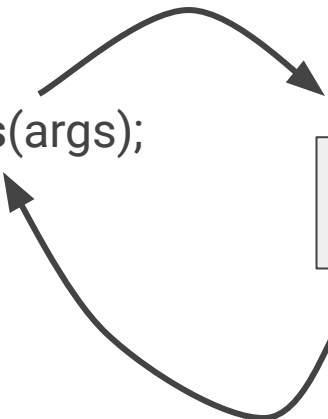
```
int decision_process(args)
{
    // code
    //
    // ...
    // ...
    return something;
}
```



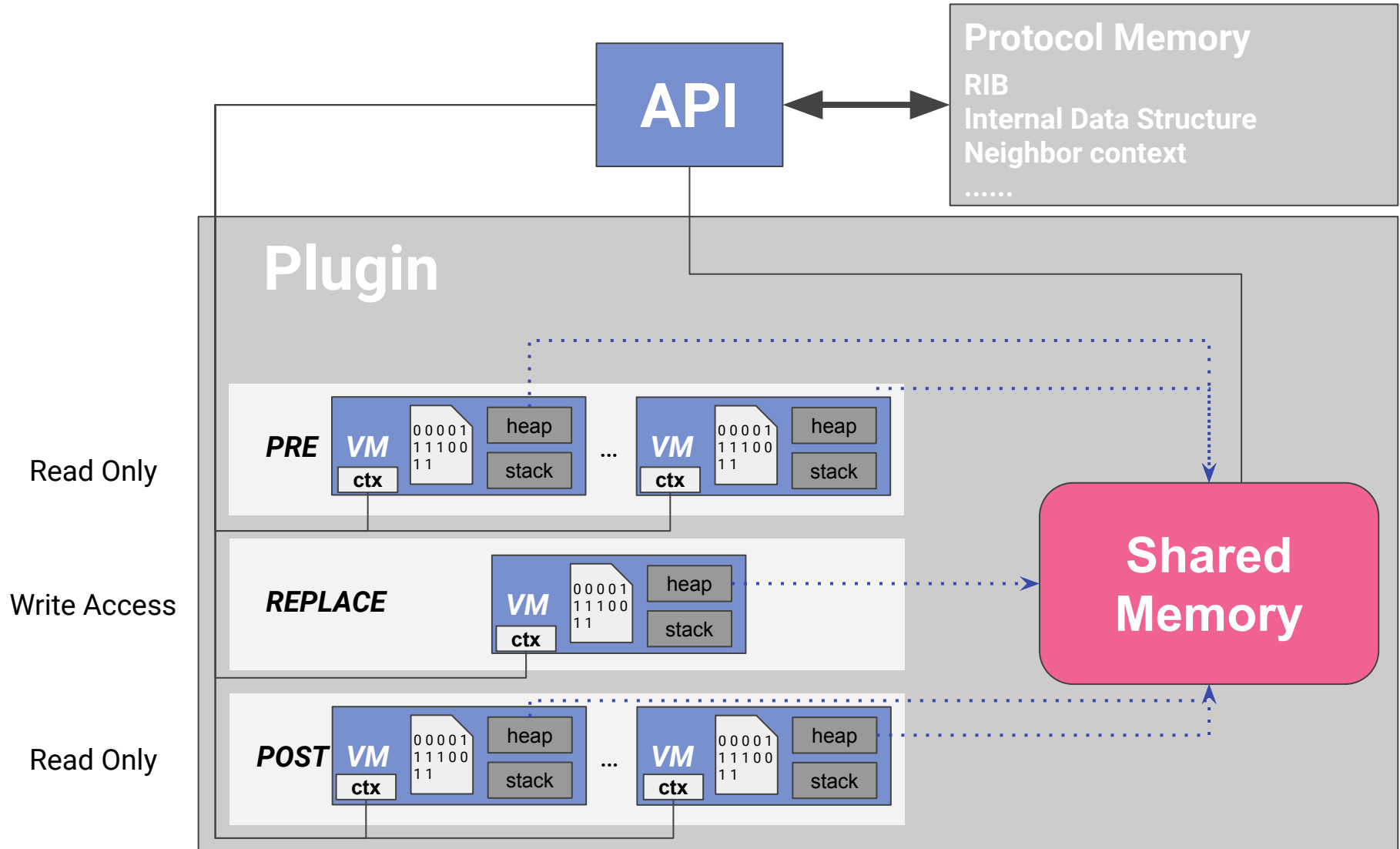
Example: *protocol function replacement*

```
int bgp_update(args)
{
  // code
  r = decision_process(args);
  // ...
  // end of function
}
```

```
int decision_process(args)
{
  // code
  // ...
  return something;
}
```



Summary : plugin structure

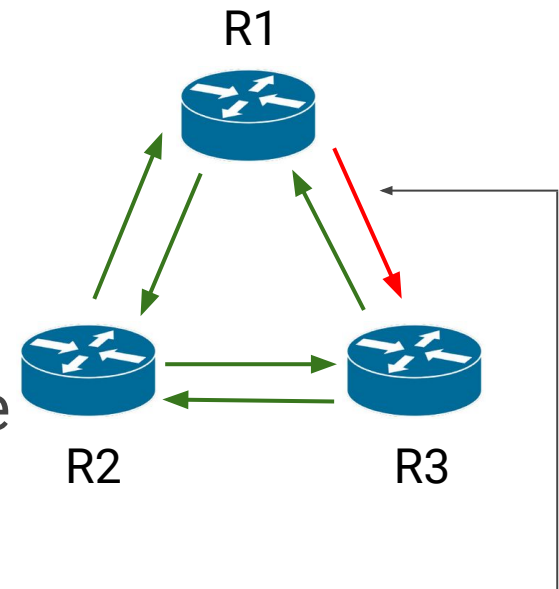


Agenda

- Objective
- Deployment of new features
- **Use cases and evaluation**
- Conclusion

Use Case 1 : Flexible SPF Algorithm

- Possible flexibilisation by adding a new OSPF LSA
 - Assign a color to each link
 - “Force” a router to use a determined path
 - New LSA with color metric
- Easy interface to add new LSAs
... and to compute the SPF
- 160 LoC to implement the Use Case

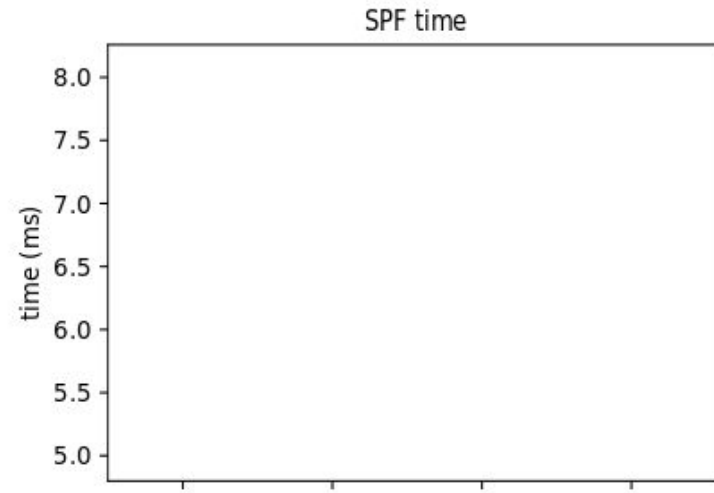
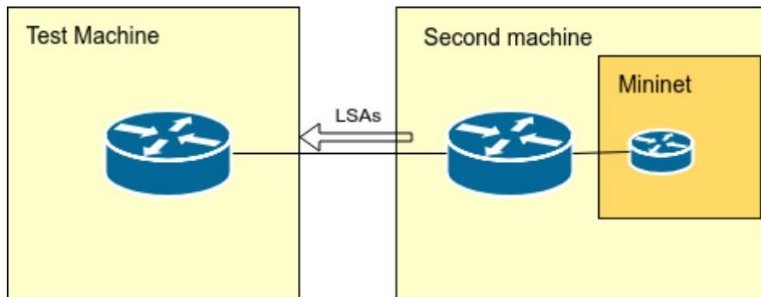


Do not use the **red** direction

Use Case 1 : Flexible SPF Algorithm

Experiment with : GTS Central Europe topology → 150 Routers

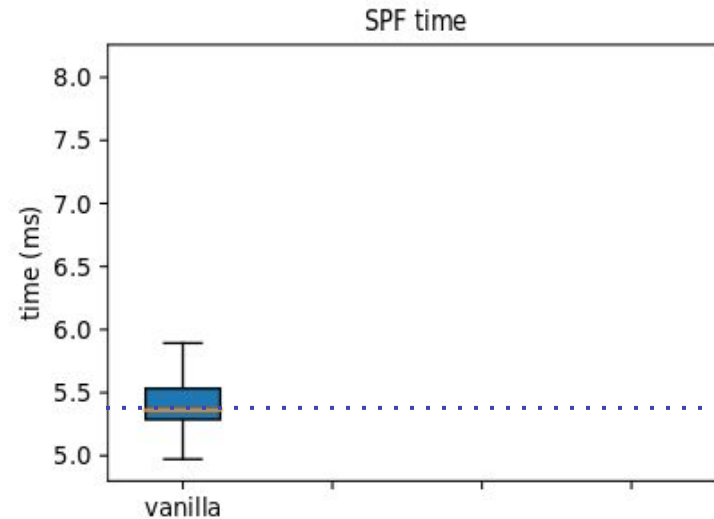
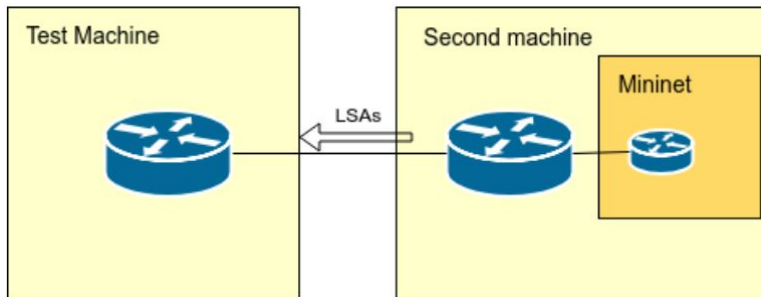
- ~300 LSAs
- The largest in the topology zoo data-set



Use Case 1 : Flexible SPF Algorithm

Experiment with : GTS Central Europe topology → 150 Routers

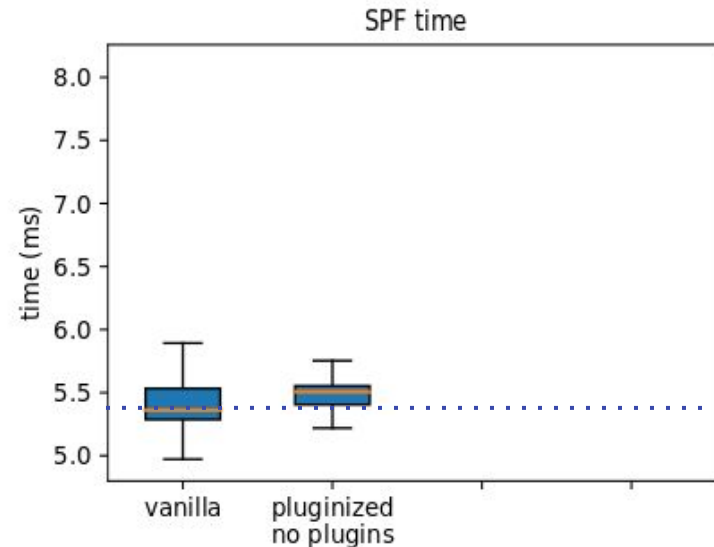
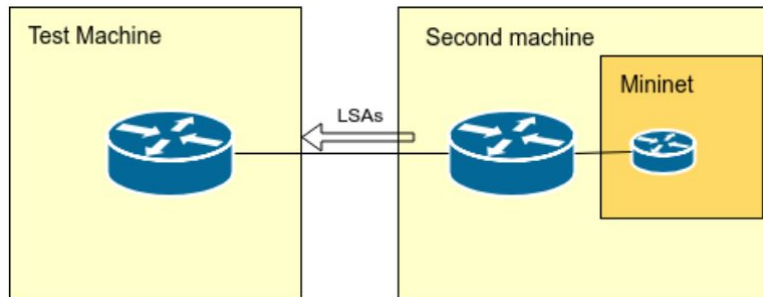
- ~300 LSAs
- The largest in the topology zoo data-set



Use Case 1 : Flexible SPF Algorithm

Experiment with : GTS Central Europe topology → 150 Routers

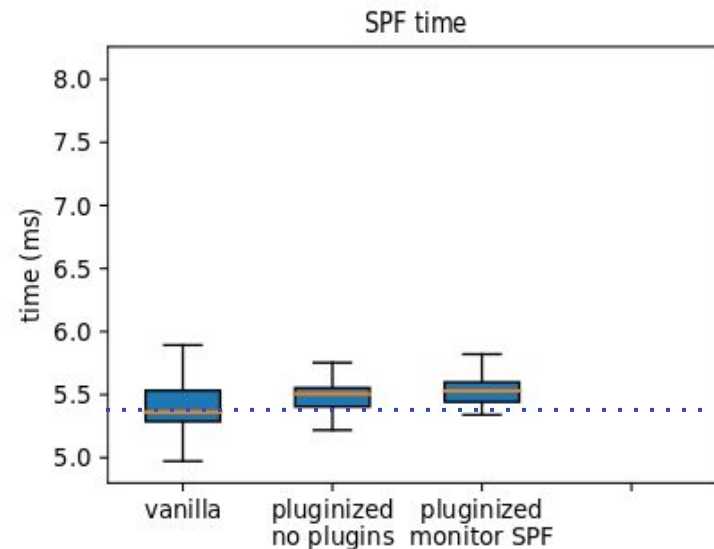
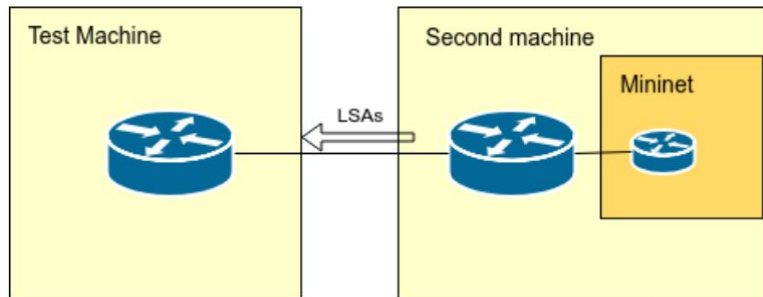
- ~300 LSAs
- The largest in the topology zoo data-set



Use Case 1 : Flexible SPF Algorithm

Experiment with : GTS Central Europe topology → 150 Routers

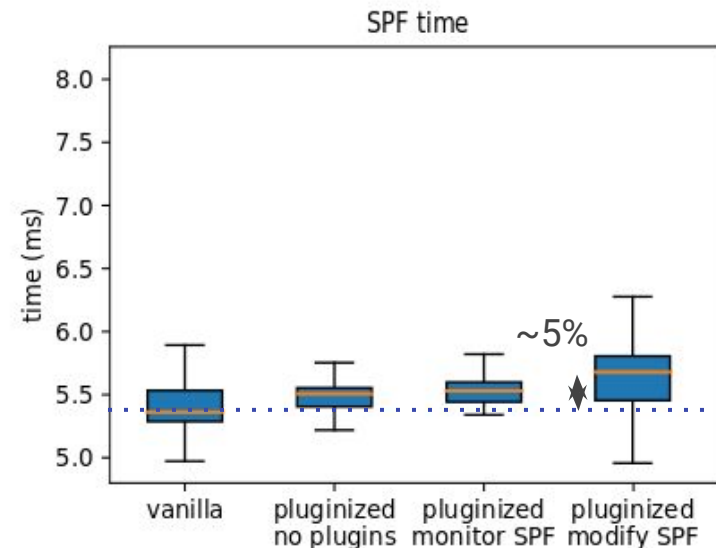
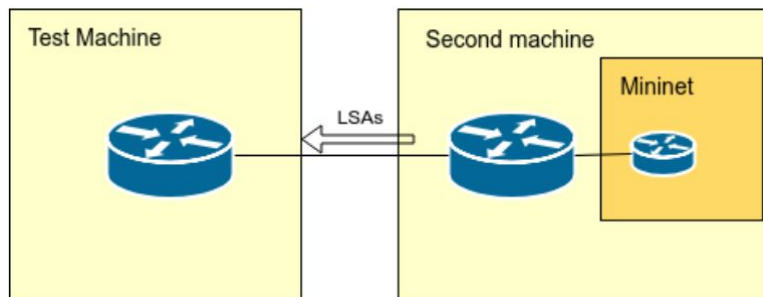
- ~300 LSAs
- The largest in the topology zoo data-set



Use Case 1 : Flexible SPF Algorithm

Experiment with : GTS Central Europe topology → 150 Routers

- ~300 LSAs
- The largest in the topology zoo data-set



Use Case 2 : New Interface for BGP Filters

- C code plugins instead of router DSL
 - New alternatives to create complex filters
 - Clearer, easier to debug

```
router bgp 64512
  bgp router-id 10.236.87.1
  neighbor 10.0.0.1 remote-as 64515
  neighbor 10.0.0.1 filter-list IN in
  !
  ! IN list accepts routes originated from odd AS only
  as-path access-list IN permit ^(.+)_+*(.*)1$
  as-path access-list IN permit ^(.+)_+*(.*)3$
  as-path access-list IN permit ^(.+)_+*(.*)5$
  as-path access-list IN permit ^(.+)_+*(.*)7$
  as-path access-list IN permit ^(.+)_+*(.*)9$
  as-path access-list IN deny any
```

DSL solution in FRRouting/CISCO conf

Use Case 2 : New Interface for BGP Filters

- C code plugins instead of router DSL
 - New alternatives to create complex filters
 - Clearer, easier to debug

```
router bgp 64512
  bgp router-id 10.236.87.1
  neighbor 10.0.0.1 remote-as 64515
  neighbor 10.0.0.1 filter-list IN in
!
! IN list accepts routes originated from odd AS only
as-path access-list IN permit ^(.+)*(.*)1$
as-path access-list IN permit ^(.+)*(.*)3$
as-path access-list IN permit ^(.+)*(.*)5$
as-path access-list IN permit ^(.+)*(.*)7$
as-path access-list IN permit ^(.+)*(.*)9$
as-path access-list IN deny any
```

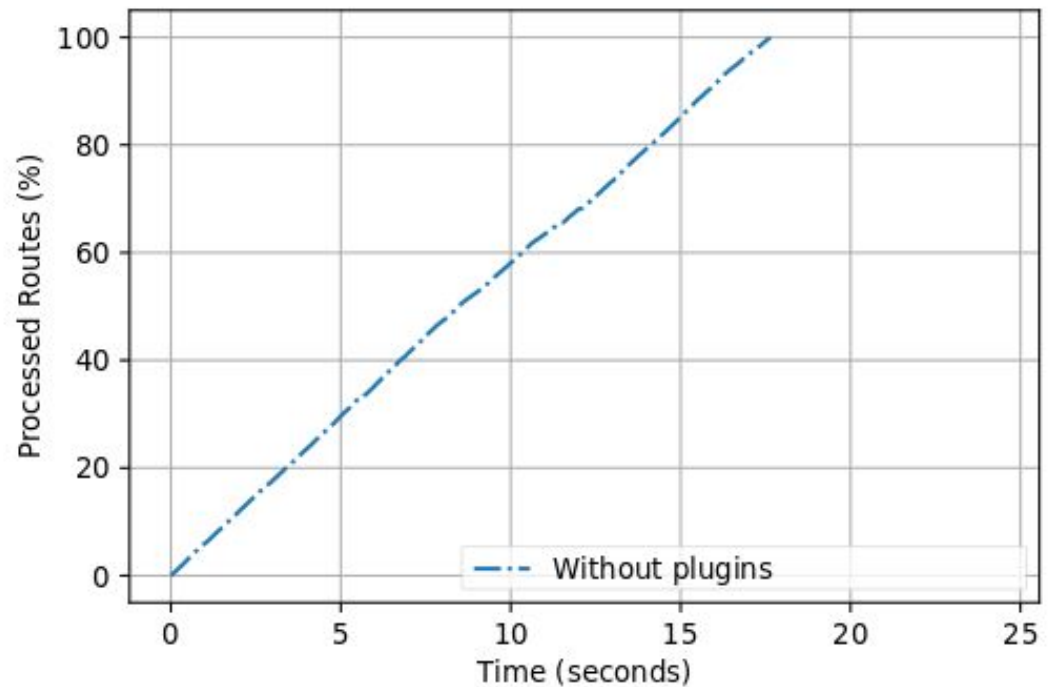
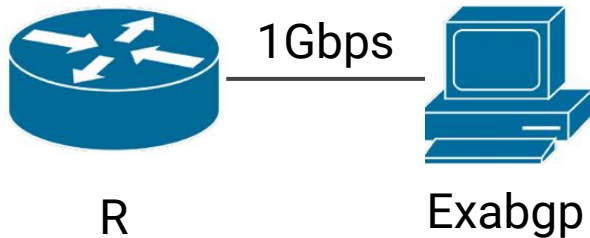
DSL solution in FRRouting/CISCO conf

```
uint64_t
filter_routes_from_even_as(bpf_full_args_t *args)
{
  as_t a = bpf_get_args(args, 2);
  if (a % 2 == 0) return FILTER_DENY; // from even AS → DENY
  return FILTER_PERMIT; // the route is originated from odd AS → ACCEPT
}
```

Our solution

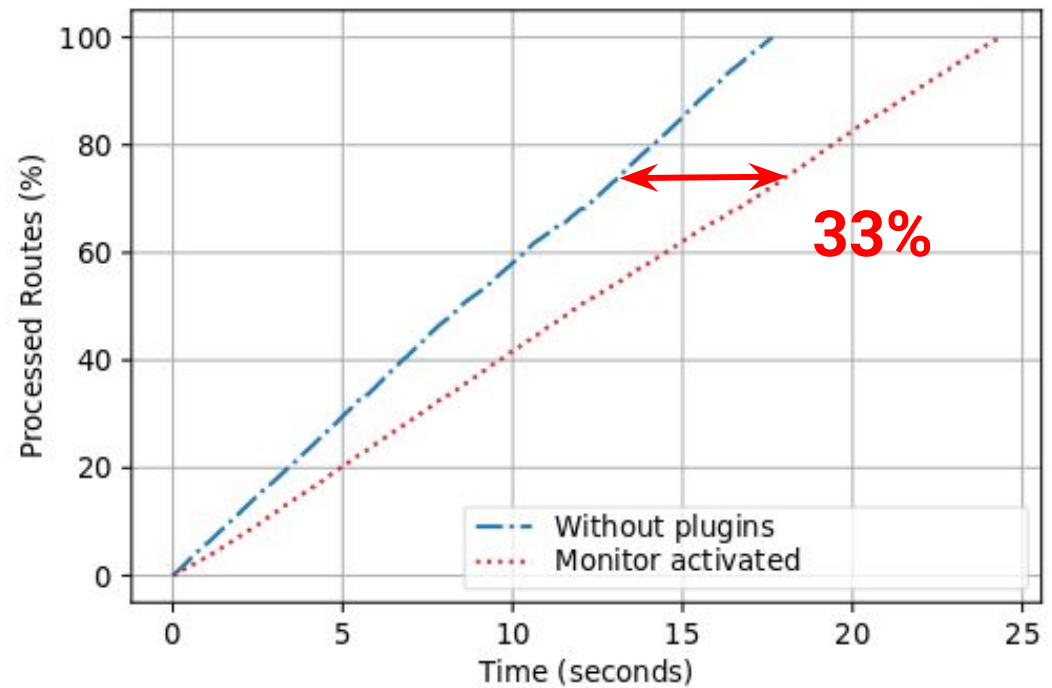
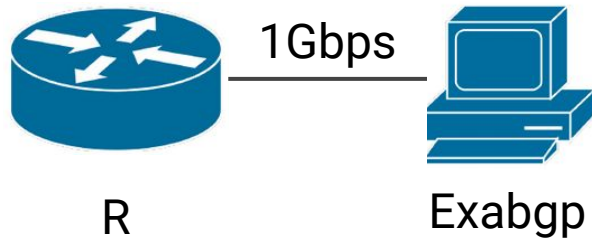
Use Case 2 : New Interface for BGP Filters

Experiment : Injection of 200K routes to router **R** via Exabgp



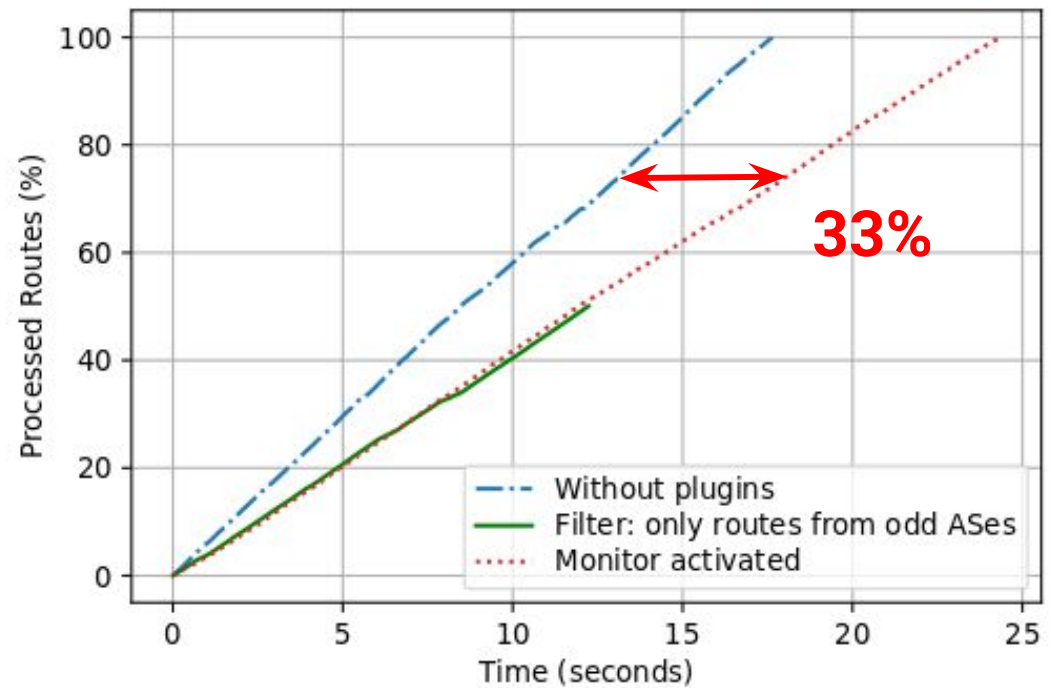
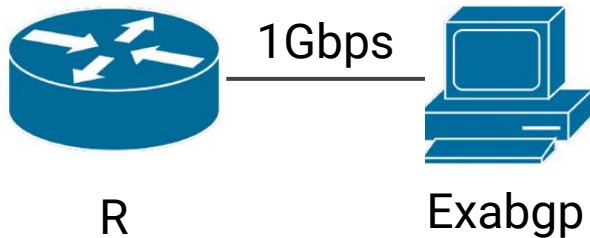
Use Case 2 : New Interface for BGP Filters

Experiment : Injection of 200K routes to router **R** via Exabgp



Use Case 2 : New Interface for BGP Filters

Experiment : Injection of 200K routes to router **R** via Exabgp



Agenda

- Objective
- Deployment of new features
- Use cases and evaluation
- **Conclusion**

Conclusion

- Reducing the innovation delay when adding new features
- Easier extension of protocol extension thanks to the plugins
- Next steps
 - Perform plugin validation to avoid malicious plugins or programming errors
 - Improve the performance of the eBPF virtual machine
 - Support multiple implementations of the same protocol on the long term
 - Specification of the API, VM

Code available at :

<https://github.com/twirtgen/pluginized-frrouting>



Questions ?



Backup slides

Is that all ?

- eBPF Stack limitation (512KB)
- Communication between the protocol and the eBPF VM
 - Via an API
- Memory safety guaranties

Plugin Insertion

```
void example_fun() {  
    printf("Hello everyone!");  
}
```

