DEVELOPING AND EVOLVING YOUR OWN CONTROL PLANE

David Barroso
Requirements
Requirements

Multiple transits
each with different full routing tables
required for redundancy, performance
Requirements

- **Multiple transits**
  - each with different full routing tables
  - required for redundancy, performance

- **Multiple hosts**
  - provide horizontal scalability
  - heavily optimized for application
Multiple transits
each with different full routing tables
required for redundancy, performance

reduce cost of traffic between hosts
must be cost-effective in order to scale
number of POPs

Multiple hosts
provide horizontal scalability
heavily optimized for application
Requirements

**Multiple transits**
each with different full routing tables required for redundancy, performance

**Routers**
lots of bells and whistles
limited to “best-path” forwarding or PBR
port density not very good
power hungry
expensive

**Multiple hosts**
provide horizontal scalability
heavily optimized for application
Requirements

**Multiple transits**
each with different full routing tables
required for redundancy, performance

**Switches**
limited FIB
only care about IP and ethernet
linux and programmable!
great port density
cheap

**Multiple hosts**
provide horizontal scalability
heavily optimized for application
The Team

~ 2 network engineers
   responsible for entire infrastructure
   too busy on-call to care about dealing with vendors

~ 2 software engineers
   one was CEO, so doesn’t count
   open source background, hate network appliances
   determined to push control to application
The Epiphany

**Routers are expensive**
- and don’t quite do what we want anyway
- port density is terrible in terms of size, $$$ and power consumption

**Switches are cheap**
- port density is great in terms of size, $$$ and power consumption
- don’t do what we want and have plenty of hardware limitations

**Programmable switches running linux!**
- and we have software engineers in the team
LET’S BUILD OUR OWN CONTROL PLANE
(how hard can it be\(^1\)?)

\(^1\) https://www.fastly.com/blog/building-and-scaling-fastly-network-part-1-fighting-fib
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier.
Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier.
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier.
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier
The architecture of the network is shown with a switch connecting two hosts. The network is divided into an L2 domain.

The FIB table includes:
- Connected: 192.168.1.0/24
- Connected: 10.0.0.0/31
- Connected: 10.0.1.0/31
- BGP 172.20.0.0/24 via 10.0.0.1
- BGP 172.20.0.0/24 via 10.0.1.1

The ARP table includes:
- 10.0.0.1 AA:BB:CC:DD:EE:00
- 10.0.1.0 AA:BB:CC:DD:EE:11

The destination IP is 172.20.0.1, and the destination MAC is AA:BB:CC:DD:EE:00.

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier.
Architecture

* Section of the network, full architecture usually consists of 4-way ECMP and may have more than one tier
st-ping

<table>
<thead>
<tr>
<th>Upstream</th>
<th>Intr</th>
<th>Nexthop</th>
<th>Sent</th>
<th>Loss</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>cogent</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.017</td>
<td>13.051</td>
<td>13.164</td>
<td>0.044</td>
<td></td>
</tr>
<tr>
<td>cogent</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.011</td>
<td>13.035</td>
<td>13.091</td>
<td>0.024</td>
<td></td>
</tr>
<tr>
<td>cogent</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.008</td>
<td>13.021</td>
<td>13.046</td>
<td>0.052</td>
<td></td>
</tr>
<tr>
<td>cogent</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.018</td>
<td>13.037</td>
<td>13.083</td>
<td>0.136</td>
<td></td>
</tr>
<tr>
<td>l3</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>10.52</td>
<td>10.546</td>
<td>10.621</td>
<td>0.133</td>
<td></td>
</tr>
<tr>
<td>l3</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>10.523</td>
<td>10.533</td>
<td>10.571</td>
<td>0.138</td>
<td></td>
</tr>
<tr>
<td>ntt</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.952</td>
<td>13.973</td>
<td>14.024</td>
<td>0.131</td>
<td></td>
</tr>
<tr>
<td>ntt</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.949</td>
<td>13.974</td>
<td>14.148</td>
<td>0.131</td>
<td></td>
</tr>
<tr>
<td>ntt</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.95</td>
<td>13.961</td>
<td>13.984</td>
<td>0.149</td>
<td></td>
</tr>
<tr>
<td>ntt</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>13.946</td>
<td>13.964</td>
<td>14.0</td>
<td>0.159</td>
<td></td>
</tr>
<tr>
<td>telia</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>14.198</td>
<td>14.326</td>
<td>14.963</td>
<td>0.234</td>
<td></td>
</tr>
<tr>
<td>telia</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>14.184</td>
<td>14.406</td>
<td>15.468</td>
<td>0.393</td>
<td></td>
</tr>
<tr>
<td>telia</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>14.379</td>
<td>14.417</td>
<td>14.558</td>
<td>0.152</td>
<td></td>
</tr>
<tr>
<td>telia</td>
<td>10</td>
<td></td>
<td>0.0%</td>
<td>14.383</td>
<td>14.405</td>
<td>14.444</td>
<td>0.056</td>
<td></td>
</tr>
</tbody>
</table>
New Requirements

Address the shortcomings of the previous architecture

- L2 hacks don’t scale well
- Sharing L2 with transits means we had to filter broadcast traffic
- No clear separation between L2 and L3 tables confuses people

Support for IXPs

- sharing L2 with IXPs is a **strong NO**
- FIB limitations were starting to hurt
The State of Networking

**Routers**
- lots of bells and whistles
- limited to “best-path” forwarding or PBR
- early days of Segment Routing/BGP-LU
- expensive

**Switches**
- cheap
- limited FIB
- only care about IP and ethernet
- MPLS and other encapsulation protocols supported
The Team

~ 6 network engineers
   responsible for entire infrastructure
   too busy on-call to care about dealing with vendors

~ 6 software engineers
   focused on many different products; load balancing, distributed health checking, routing
   architecture, automation, kernel development, etc…
New Architecture

**P-t-p links**

**eBGP**

**An MPLS label per exit**
(transits, PNIs, peers...)

**MPLS starts on the host**
Host tags the packets with labels depending on the desired path
How does it work?

1. Provisioning system assigns unique label to each exit point
2. Silverton announces liveness of LSP down the network
3. Incoming prefixes from each transit are tagged with an extended community that identifies the LSP
4. On the host, BGP extracts the community and assigns labels to routes
The network couldn’t be simpler

```
switch-cmh8801# show mpls lfib route
Codes: S - Static MPLS Route, I A - ISIS-SR Adjacency Segment,
      I P - ISIS-SR Prefix Segment, L - LDP,
      I-L - ISIS-SR Segment to LDP, L-I - LDP to ISIS-SR Segment

<table>
<thead>
<tr>
<th>In-Label</th>
<th>Out-Label</th>
<th>Metric</th>
<th>Payload</th>
<th>NextHop</th>
<th>Egress-Acl</th>
<th>Source</th>
<th>FEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>pop</td>
<td>50</td>
<td>ipv4</td>
<td>38.104.22.37</td>
<td>Apply</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>1021</td>
<td>pop</td>
<td>50</td>
<td>ipv6</td>
<td>2001:550:2:12::c:1</td>
<td>Apply</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>1022</td>
<td>pop</td>
<td>50</td>
<td>ipv4</td>
<td>4.14.76.49</td>
<td>Apply</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>1023</td>
<td>pop</td>
<td>50</td>
<td>ipv6</td>
<td>2001:1900:2100::4181</td>
<td>Apply</td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>
```
Extended communities map to labels

dbarroso@cache-cmh8820:~$ sudo birdc show route 223.255.251.0/24 all
BIRD 1.6.3 ready.
223.255.251.0/24 via 172.18.128.1 (MPLS label 1020) [switch_cmh8801] * (100) [AS63199i]
  Type: BGP unicast univ
  BGP.as_path: 4210088000 174 3491 63199
  BGP.next_hop: 172.18.128.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,21000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1020)
    via 172.18.130.1 (MPLS label 1024) [switch_cmh8802] (100) [AS63199i]
  Type: BGP unicast univ
  BGP.as_path: 4210088000 174 3491 63199
  BGP.next_hop: 172.18.130.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,21000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1024)
Extended communities map to labels

dbarroso@cache-cmh8820:~$ sudo birdc show route 223.255.251.0/24 all
BIRD 1.6.3 ready.

**223.255.251.0/24 via 172.18.128.1** (MPLS label 1020) [switch_cmh8801] * (100) [AS63199i]

- Type: BGP unicast univ
- BGP.as_path: 4210088000 174 3491 63199
- BGP.next_hop: 172.18.128.1
- BGP.med: 0
- BGP.local_pref: 50
- BGP.community: (174,21000) (174,22013)
- BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1020)

**via 172.18.130.1** (MPLS label 1024) [switch_cmh8802] (100) [AS63199i]

- Type: BGP unicast univ
- BGP.as_path: 4210088000 174 3491 63199
- BGP.next_hop: 172.18.130.1
- BGP.med: 0
- BGP.local_pref: 50
- BGP.community: (174,21000) (174,22013)
- BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1024)
Extended communities map to labels

dbarroso@cache-cmh8820:~$ sudo birdc show route 223.255.251.0/24 all
BIRD 1.6.3 ready.

223.255.251.0/24 via 172.18.128.1 (MPLS label 1020) [switch_cmh8801] * (100) [AS63199i]
  Type: BGP unicast univ
  BGP.as_path: 4210088000 174 3491 63199
  BGP.next_hop: 172.18.128.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,21000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1020)
  via 172.18.130.1 (MPLS label 1024) [switch_cmh8802] (100) [AS63199i]
  Type: BGP unicast univ
  BGP.as_path: 4210088000 174 3491 63199
  BGP.next_hop: 172.18.130.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,21000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1024)
Extended communities map to labels

dbarroso@cache-cmh8820:~$ sudo birdc show route 223.255.251.0/24 all
BIRD 1.6.3 ready.
223.255.251.0/24 via 172.18.128.1 (MPLS label 1020) [switch_cmh8801] * (100) [AS63199i]
  Type: BGP unicast univ
  BGP.as_path: 4210080000 174 3491 63199
  BGP.next_hop: 172.18.128.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,210000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1020)

via 172.18.130.1 (MPLS label 1024) [switch_cmh8802] (100) [AS63199i]

  Type: BGP unicast univ
  BGP.as_path: 4210080000 174 3491 63199
  BGP.next_hop: 172.18.130.1
  BGP.med: 0
  BGP.local_pref: 50
  BGP.community: (174,210000) (174,22013)
  BGP.ext_community: (ro, 2, 3) (ro, 3, 1) (generic, 0x80860000, 1024)
Routing decision is made on the host

dbarroso@cache-cmh8820$ ip rule
174: from all fwmark 0xae lookup 174
1299: from all fwmark 0x513 lookup 1299

dbarroso@cache-cmh8820$ ip route show table 174
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820$ ip route show table 1299
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1022 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1026 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820$ ip route show table main | head
1.0.4.0/24 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
1.0.4.0/22 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
Routing decision is made on the host

```
dbarroso@cache-cmh8820:~$ ip rule
174: from all fwmark 0xae lookup 174
1299: from all fwmark 0x513 lookup 1299
```

```
dbarroso@cache-cmh8820:~$ ip route show table 174
default  proto static  src 199.27.79.20  mtu 1500 advmss 1460
   nexthop encap mpls  1020  via 172.18.128.1  dev vlan100 weight 1
   nexthop encap mpls  1024  via 172.18.130.1  dev vlan200 weight 1
```

```
dbarroso@cache-cmh8820:~$ ip route show table 1299
default  proto static  src 199.27.79.20  mtu 1500 advmss 1460
   nexthop encap mpls  1022  via 172.18.128.1  dev vlan100 weight 1
   nexthop encap mpls  1026  via 172.18.130.1  dev vlan200 weight 1
```

```
dbarroso@cache-cmh8820:~$ ip route show table main | head
1.0.4.0/24  proto bird  src 199.27.79.20  mtu 1500 advmss 1460
   nexthop encap mpls  1020  via 172.18.128.1  dev vlan100 weight 1
   nexthop encap mpls  1024  via 172.18.130.1  dev vlan200 weight 1
1.0.4.0/22  proto bird  src 199.27.79.20  mtu 1500 advmss 1460
   nexthop encap mpls  1020  via 172.18.128.1  dev vlan100 weight 1
   nexthop encap mpls  1024  via 172.18.130.1  dev vlan200 weight 1
```

We have a routing table per transit/peer
Routing decision is made on the host

dbarroso@cache-cmh8820:~$ ip rule
174: from all fwmark 0xae lookup 174
1299: from all fwmark 0x513 lookup 1299

dbarroso@cache-cmh8820:~$ ip route show table 174
default proto static src 199.27.79.20 mtu 1500 advmss 1460
  nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
  nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:~$ ip route show table 1299
default proto static src 199.27.79.20 mtu 1500 advmss 1460
  nexthop encap mpls 1022 via 172.18.128.1 dev vlan100 weight 1
  nexthop encap mpls 1026 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:~$ ip route show table main | head
1.0.4.0/24 proto bird src 199.27.79.20 mtu 1500 advmss 1460
  nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
  nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
1.0.4.0/22 proto bird src 199.27.79.20 mtu 1500 advmss 1460
  nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
  nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

Per peer/transit routing tables only contain a default route indicating the MPLS label required to use
Routing decision is made on the host

```
dbarroso@cache-cmh8820:~$ ip rule
174: from all fwmark 0xae lookup 174
1299: from all fwmark 0x513 lookup 1299
```

Rules allow the application to force traffic into specific routing tables

```
dbarroso@cache-cmh8820:~$ ip rule show table 174
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
```

```
dbarroso@cache-cmh8820:~$ ip rule show table 1299
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1022 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1026 via 172.18.130.1 dev vlan200 weight 1
```

```
dbarroso@cache-cmh8820:~$ ip route show table main | head
1.0.4.0/24 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
1.0.4.0/22 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
```
Routing decision is made on the host

dbarroso@cache-cmh8820:~$ ip rule
174: from all fwmark 0xae lookup 174
1299: from all fwmark 0x513 lookup 1299

dbarroso@cache-cmh8820:~$ ip route show table 174
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:~$ ip route show table 1299
default proto static src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1022 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1026 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:~$ ip route show table main | head
1.0.4.0/24 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
1.0.4.0/22 proto bird src 199.27.79.20 mtu 1500 advmss 1460
   nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
   nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

If application sets the fwmark “ae” traffic is forced into the table 174
Routing decision is made on the host

dbarroso@cache-cmh8820:$ ip rule
174: from all fwmark 0x2ae lookup 174
1299: from all fwmark 0x513 lookup 1299

dbarroso@cache-cmh8820:$ ip route show table 174
default proto static src 199.27.79.20 mtu 1500 advmss 1460
    nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
    nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:$ ip route show table 1299
default proto static src 199.27.79.20 mtu 1500 advmss 1460
    nexthop encap mpls 1022 via 172.18.128.1 dev vlan100 weight 1
    nexthop encap mpls 1026 via 172.18.130.1 dev vlan200 weight 1

dbarroso@cache-cmh8820:$ ip route show table main | head
1.0.4.0/24 proto bird src 199.27.79.20 mtu 1500 advmss 1460
    nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
    nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
1.0.4.0/22 proto bird src 199.27.79.20 mtu 1500 advmss 1460
    nexthop encap mpls 1020 via 172.18.128.1 dev vlan100 weight 1
    nexthop encap mpls 1024 via 172.18.130.1 dev vlan200 weight 1
**Retrospective**

*Same nice features as the previous iteration*
  - per flow routing

*Shortcomings were addressed*
  - no more broken assumptions on how networks work or are operated
  - no more L2 hacks

*New architectural changes worked as expected*
Origins 2013
Evolution 2015
Migration 2017
Origins 2013
Evolution 2015
Migration 2017
Objectives

Change of protocol
Old architecture used iBGP while new used eBGP

Must minimize concurrent architectures
  twice the code
  twice the tooling
  twice the knowledge
  twice the things that can go wrong

Must minimize migration period
  traditional migrations can take years for large infrastructure
  scheduling, customer notification, executing the work must be reduced to bare minimum
1. Changing of ASN requires synch’ing changes in multiple devices which comes with some automation challenges
iBGP to eBGP challenges

2. ECMP

While we have the same protocol everywhere all paths are eligible.
iBGP to eBGP challenges

2. ECMP
   While we have the same protocol everywhere all paths are eligible.

eBGP wins over iBGP
   now all traffic is going via a single link potentially congesting the links
How do we solve those problems?

“That’s impossible and I could cite 5 RFCs and reference 20 $vendor white-papers explaining why”

Jimmy, Network Architect, CCIE, JNCIE, Naysayer

“I have no respect for my elders”

Lisa, Software Developer, Nihilist, Hacker
ASN migration

Starting config

```
protocol bgp host-1 {
    local as 54113;
    neighbor host_ip as 54113;
}

protocol bgp switch-1 {
    local as 54113;
    neighbor switch1_ip as 54113;
}
```
We allow peers to connect with either ASN.
ASN migration

protocol bgp host-1 {
  local as 65000;
  neighbor host_ip as 54113 as 65100;
}

protocol bgp switch-1 {
  local as 54113;
  neighbor switch1_ip as 54113 as 65000;
}

Now we can change the local ASN independently on each device
iBGP/eBGP prefix compatibility

How to make "eBGP prefixes" compatible with "iBGP prefixes"?

switch-1
AS65000

switch-2
AS54113

protocol: eBGP
AS PATH: 65000 $AS_PATH
local-pref: 0

protocol: iBGP
AS PATH: $AS_PATH
local-pref: 100
iBGP/eBGP prefix compatibility

eBGP > iBGP

switch-1
AS65000

switch-2
AS54113

host

protocol: eBGP
AS PATH: 65000 $AS_PATH
local-pref: 0

protocol: iBGP
AS PATH: $AS_PATH
local-pref: 100
**iBGP/eBGP prefix compatibility**

**compare_as_ibgp**

applied to an eBGP session, its prefixes will be compared against other candidates as if they were coming from an iBGP session.

---

**Diagram:**

- **switch-1**
  - AS: AS65000
  - Protocol: eBGP
  - AS PATH: 65000 $AS_PATH
  - local-pref: 0

- **switch-2**
  - AS: AS54113
  - Protocol: iBGP
  - AS PATH: $AS_PATH
  - local-pref: 100

- **host**
iBGP/eBGP prefix compatibility

\[
\text{len}(65000 \ AS\ PATH) > \text{len}($\text{AS\ PATH}$)
\]
iBGP/eBGP prefix compatibility

**skip_private_as_path_prefix**
do not count leading private ASNs
when computing length of AS path

```
protocol: eBGP
AS PATH: 65000 $AS_PATH
local-pref: 0
```

```
protocol: iBGP
AS PATH: $AS_PATH
local-pref: 100
```
iBGP/eBGP prefix compatibility

**eBGP local-pref**

- **switch-1**
  - **AS65000**
  - **protocol:** eBGP
  - **AS PATH:** 65000 $AS_PATH
  - **local-pref:** 0

- **switch-2**
  - **AS54113**
  - **protocol:** iBGP
  - **AS PATH:** $AS_PATH
  - **local-pref:** 100

**host**
allow bgp_local_pref
allow local-pref between eBGP speakers
Migration process (overview)

Step 1. Enable new options.

```plaintext
protocol bgp cache-cmh8820 from tpl_bgp {
  local as 54113;
  neighbor cache_cmh8820_ip as 54113;
  neighbor cache_cmh8820_ip as 54113 as 4210088001;
  compare_as_ibgp;
  skip_private_as_path_prefix;
  allow bgp_local_pref;
}
```

Impact: None

Step 2. Flip “local as”:

```plaintext
protocol bgp cache-cmh8820 from tpl_bgp {
  local as 54113;
  local as 4210088000;
}
```

Impact: Only a BGP session flap thanks to the BGP “hacks”. Progressive rollout is enough.
bird> show route 2.2.2.2/32 all
2.2.2.2/32 via 10.0.0.2 on eth0 [switch_1] *(110) [AS123i]

Type: BGP unicast univ
BGP.origin: IGP
BGP.as_path: 123
BGP.next_hop: 10.0.0.2
BGP.local_pref: 110
BGP.originator_id: 10.0.0.51
BGP.cluster_list: 10.0.0.101

via 10.0.0.1 on eth0 [switch_0] (110) [AS123i]

Type: BGP unicast univ
BGP.origin: IGP
BGP.as_path: 65100 65000 65000 65000 65000 65000 65000 123
BGP.next_hop: 10.0.0.1
BGP.local_pref: 110

bird> show route 2.2.2.2/32 export injector
2.2.2.2/32 multipath [switch_1 16:03:32 from 10.0.0.101] (100) [AS123i]

via 10.0.0.1 on eth0 weight 1
via 10.0.0.2 on eth0 weight 1
bird> show route 2.2.2.2/32 all

2.2.2.2/32 via 10.0.0.2 on eth0 [switch_1] * (110) [AS123i]

Type: BGP unicast univ  
BGP.origin: IGP  
BGP.as_path: 123  
BGP.next_hop: 10.0.0.2  
BGP.local_pref: 110  
BGP.originator_id: 10.0.0.51  
BGP.cluster_list: 10.0.0.101

via 10.0.0.1 on eth0 [switch_0] (110) [AS123i]

Type: BGP unicast univ  
BGP.origin: IGP  
BGP.as_path: 65100 65000 65000 65000 65000 65000 123  
BGP.next_hop: 10.0.0.1  
BGP.local_pref: 110

bird> show route 2.2.2.2/32 export injector

2.2.2.2/32 multipath [switch_1 16:03:32 from 10.0.0.101] (100) [AS123i]

via 10.0.0.1 on eth0 weight 1  
via 10.0.0.2 on eth0 weight 1
### Skip Private AS Path Prefix

```
bird> show route 2.2.2.2/32 all
2.2.2.2/32 via 10.0.0.2 on eth0 [switch_1] * (110) [AS123i]
  Type: BGP unicast univ
  BGP.origin: IGP
  **BGP.as_path:** 123
  BGP.next_hop: 10.0.0.2
  BGP.local_pref: 110
  BGP.originator_id: 10.0.0.51
  BGP.cluster_list: 10.0.0.101
  via 10.0.0.1 on eth0 [switch_0] (110) [AS123i]
```

```
bird> show route 2.2.2.2/32 export injector
2.2.2.2/32 multipath [switch_1 16:03:32 from 10.0.0.101] (100) [AS123i]
  via 10.0.0.1 on eth0 weight 1
  via 10.0.0.2 on eth0 weight 1
```

```sql
# BGP as_path

<table>
<thead>
<tr>
<th>Type</th>
<th>BGP unicast univ</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP.origin</td>
<td>IGP</td>
</tr>
<tr>
<td><strong>BGP.as_path</strong></td>
<td>123</td>
</tr>
</tbody>
</table>

**Private AS Path Prefix**

```
eBGP
BGP.as_path: 65100 65000 65000 65000 65000 123
```

```
iBGP
```

<table>
<thead>
<tr>
<th>Type</th>
<th>BGP unicast univ</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP.origin</td>
<td>IGP</td>
</tr>
<tr>
<td><strong>BGP.as_path</strong></td>
<td>65100 65000 65000 65000 65000 123</td>
</tr>
<tr>
<td>BGP.next_hop</td>
<td>10.0.0.1</td>
</tr>
<tr>
<td>BGP.local_pref</td>
<td>110</td>
</tr>
</tbody>
</table>
bird> show route 2.2.2.2/32 all

2.2.2.2/32 via 10.0.0.2 on eth0 [switch_1] *(110) [AS123i]

Type: BGP unicast univ
BGP.origin: IGP
BGP.as_path: 123
BGP.next_hop: 10.0.0.2
BGP.local_pref: 110
BGP.originator_id: 10.0.0.51
BGP.cluster_list: 10.0.0.101

via 10.0.0.1 on eth0 [switch_0] (110) [AS123i]

Type: BGP unicast univ
BGP.origin: IGP
BGP.as_path: 65100 65000 65000 65000 65000 123
BGP.next_hop: 10.0.0.1
BGP.local_pref: 110

bird> show route 2.2.2.2/32 export injector

2.2.2.2/32 multipath [switch_1 16:03:32 from 10.0.0.101] (100) [AS123i]

via 10.0.0.1 on eth0 weight 1
via 10.0.0.2 on eth0 weight 1
iBGP/eBGP “compatible” prefix

bird> show route 2.2.2.2/32 all
2.2.2.2/32  via 10.0.0.2 on eth0 [switch_1] *(110) [AS123i]
  Type: BGP unicast univ
  BGP.origin: IGP
  BGP.as_path: 123
  BGP.next_hop: 10.0.0.2
  BGP.local_pref: 110
  BGP.originator_id: 10.0.0.51
  BGP.cluster_list: 10.0.0.101

bird>

bird> show route 2.2.2.2/32 export injector
2.2.2.2/32  multipath [switch_1 16:03:32 from 10.0.0.101] (100) [AS123i]
  via 10.0.0.1 on eth0 weight 1
  via 10.0.0.2 on eth0 weight 1

bird>
Origins 2013
Evolution 2015
Migration 2017
Did you consider...
Did you consider... Openflow

No comments
No, seriously, no comments
Did you consider... Segment Routing

Core ideas are great
We borrowed some and we will probably borrow more

Complexity
Many protocols doing many things, complexity better in the host

Solves half the problem
How does it integrate with the application/host?

No open-source option
not even sure vendors support it yet
Did you consider... BGP-LU

Egress Peer Engineering using BGP-LU
draft-gredler-idr-bGPLU-epe-09
similar idea but different implementation

Solves half the problem
How does it integrate with the application/host?

No OSS implementation
requires new AF
requires dealing with race conditions

Using BGP-LU is probably the right thing to do
to revisit in the future
Building your own control plane

No viable alternatives in 2013
   existing solutions were expensive
   didn’t quite do what we wanted
   hardware constraints limited our options

Custom control plane solution was critical
   saved money and time
   attracted talent

Revisited problem space
   addressed issues in the previous architecture
   less hardware constraints
   hacked our way towards seamless migration
   open source was critical
Questions?
Appendix

https://www.mail-archive.com/netdev@vger.kernel.org/msg123533.html
https://www.mail-archive.com/netdev@vger.kernel.org/msg176877.html
https://patchwork.kernel.org/patch/9552935/