

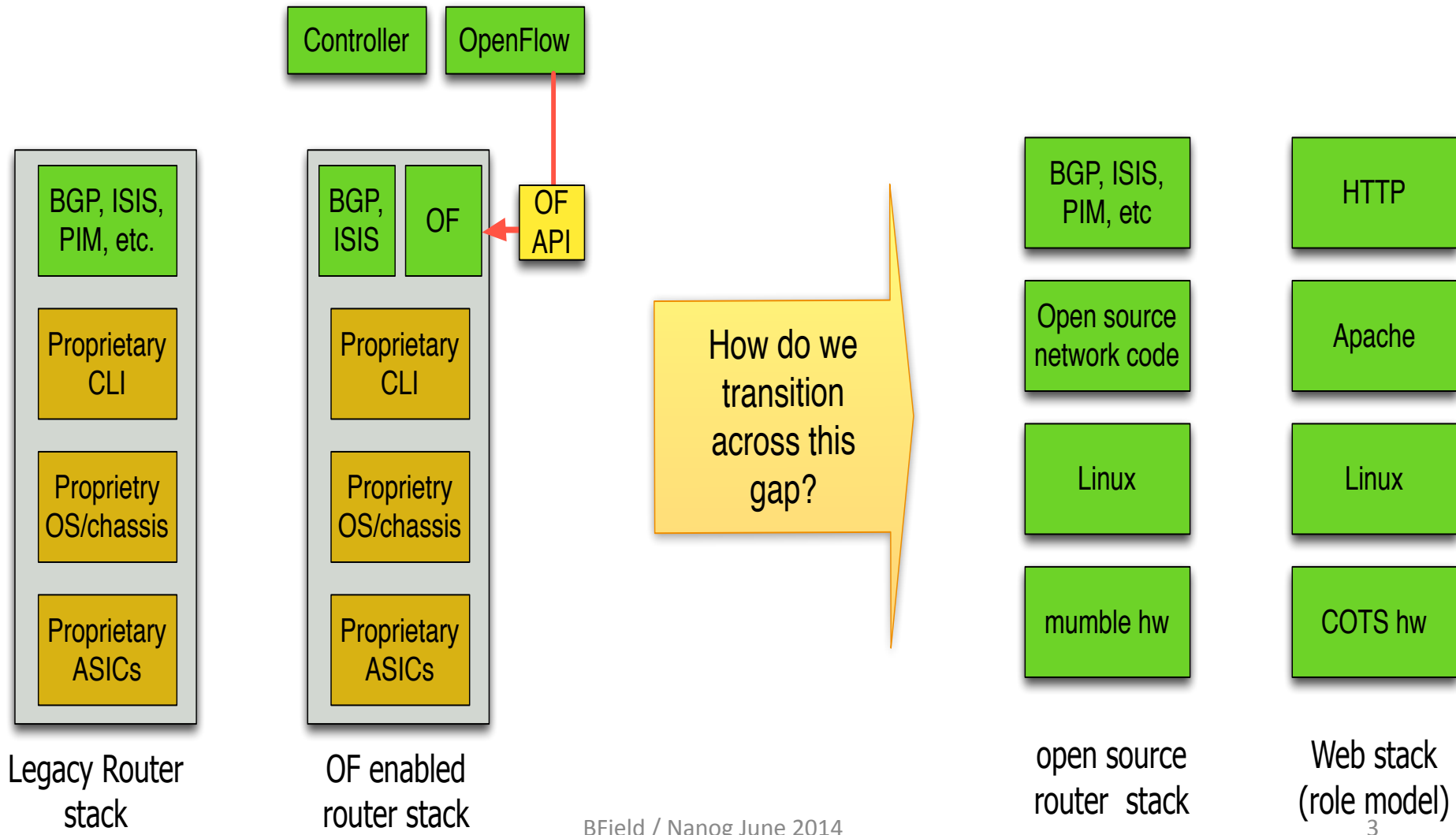
The Hybrid-Open (“HOpen”) router architecture

Brian Field / Comcast

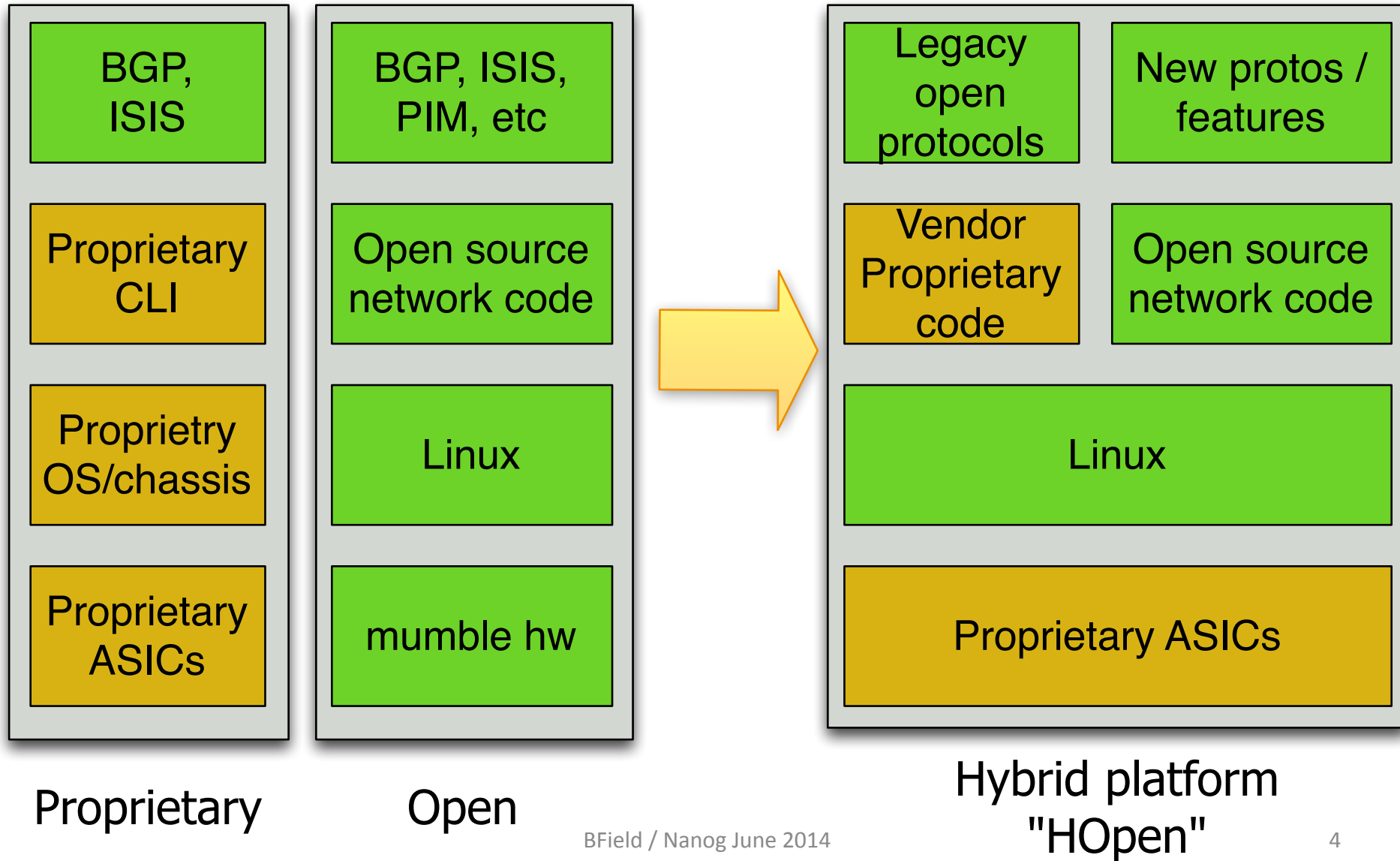
Background

- Lots of “excitement” in the SDN space
 - Programmability (config)
 - Network virtualization (existing features)
- What if I need a new feature on a router-- today?
- Does the existing SDN paradigm help this?
 - No [not really]
- This talk presents a new “SDN” paradigm:
 - Something we can Do Now.

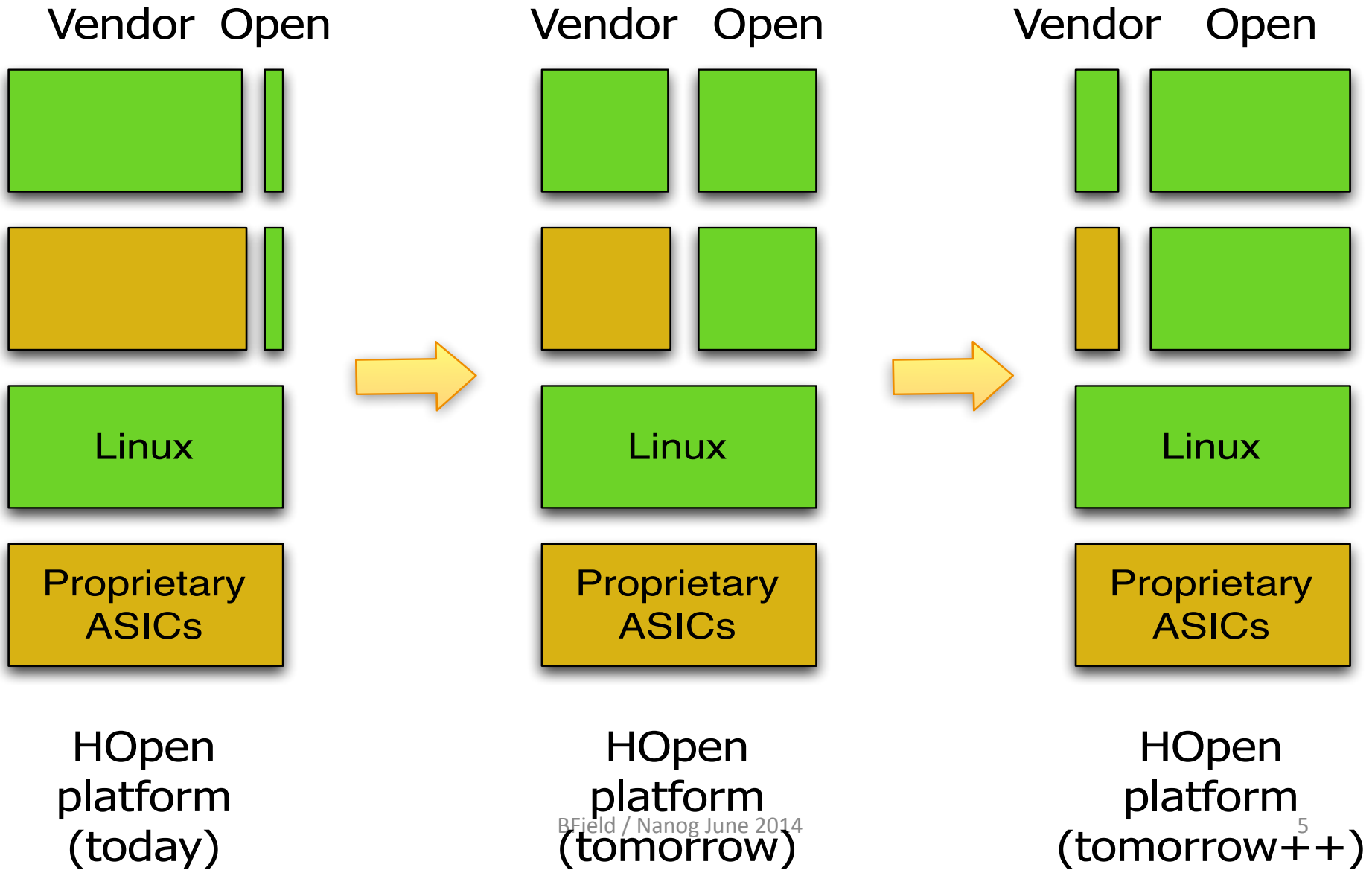
Router platform evolution



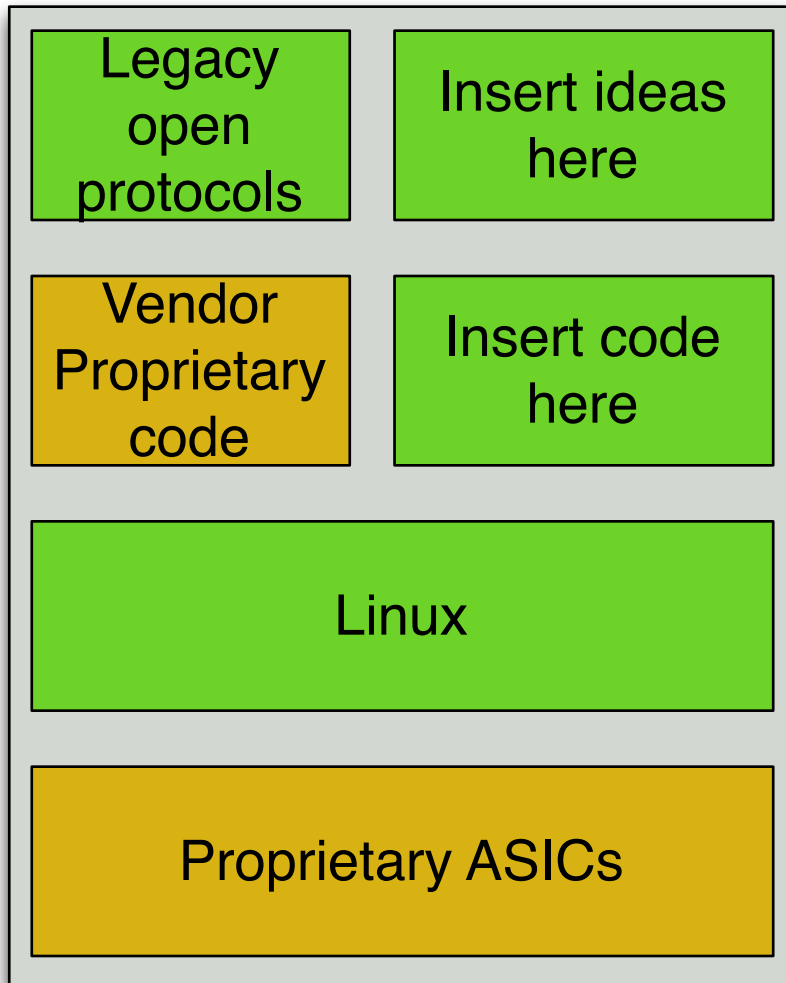
We glue them together... "HOpen"



In theory, over time....



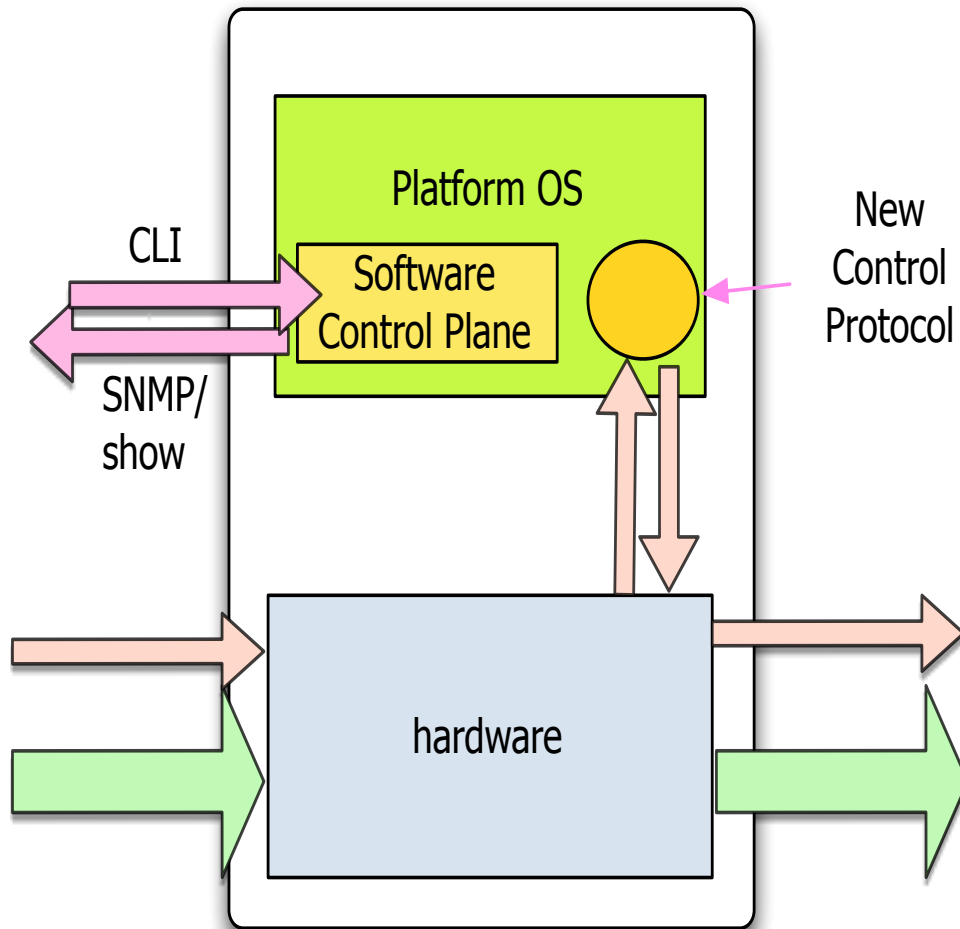
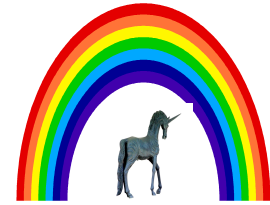
Unicorns and rainbows -- Wouldn't it be great if the HOpen platform existed today...?



Hybrid platform



Actually it does

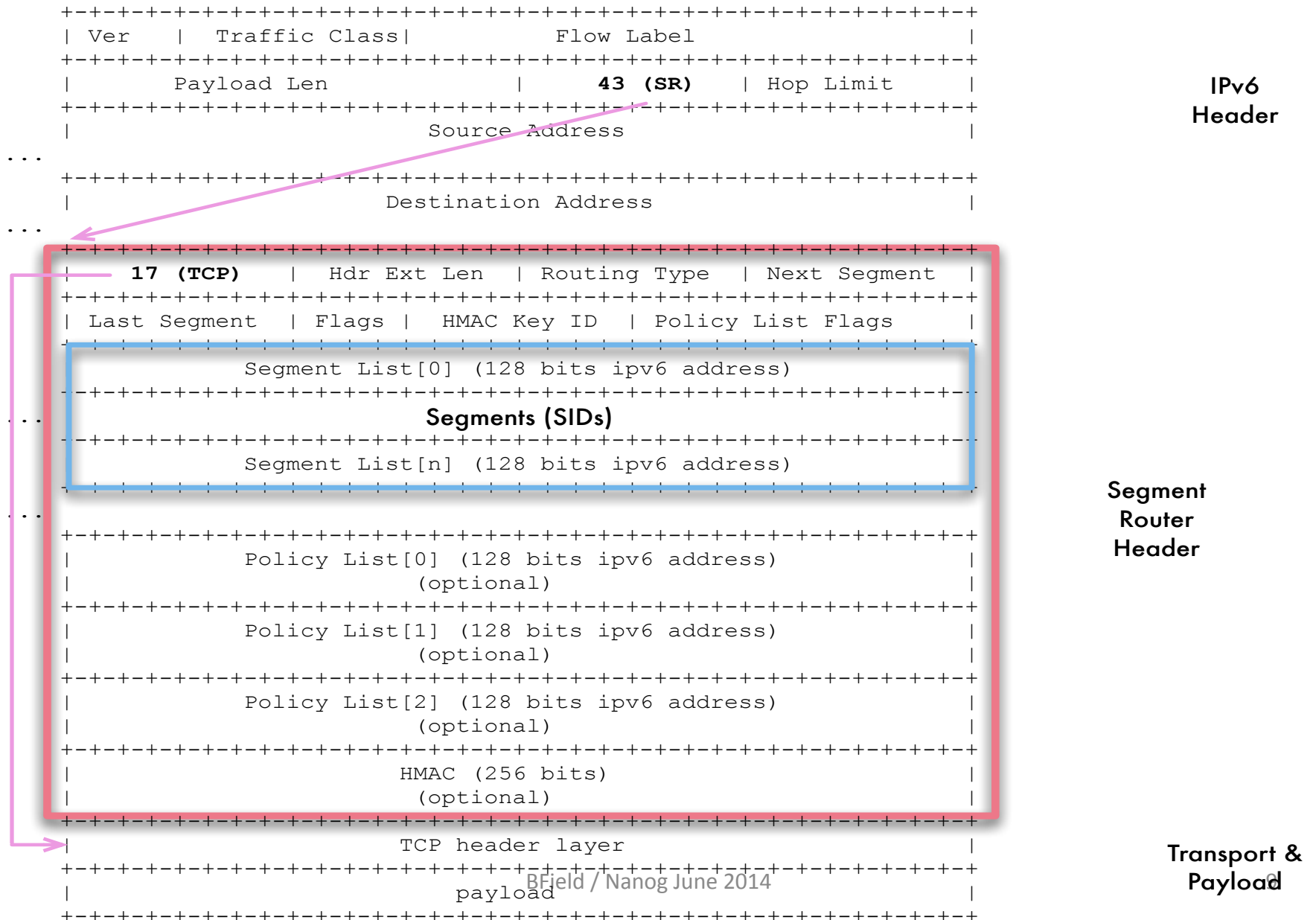


- Linux based
- Drop in your own code / binaries next to vendor code
- What might be an interesting use case to develop on this platform?

Segment Routing

- Introduced to IETF in 2013
- Isn't supported on deployed platforms
- SR 101:
 - Traffic engineering
 - Service chaining
- Comcast interested in IPv6 SR

Segment Router Packet

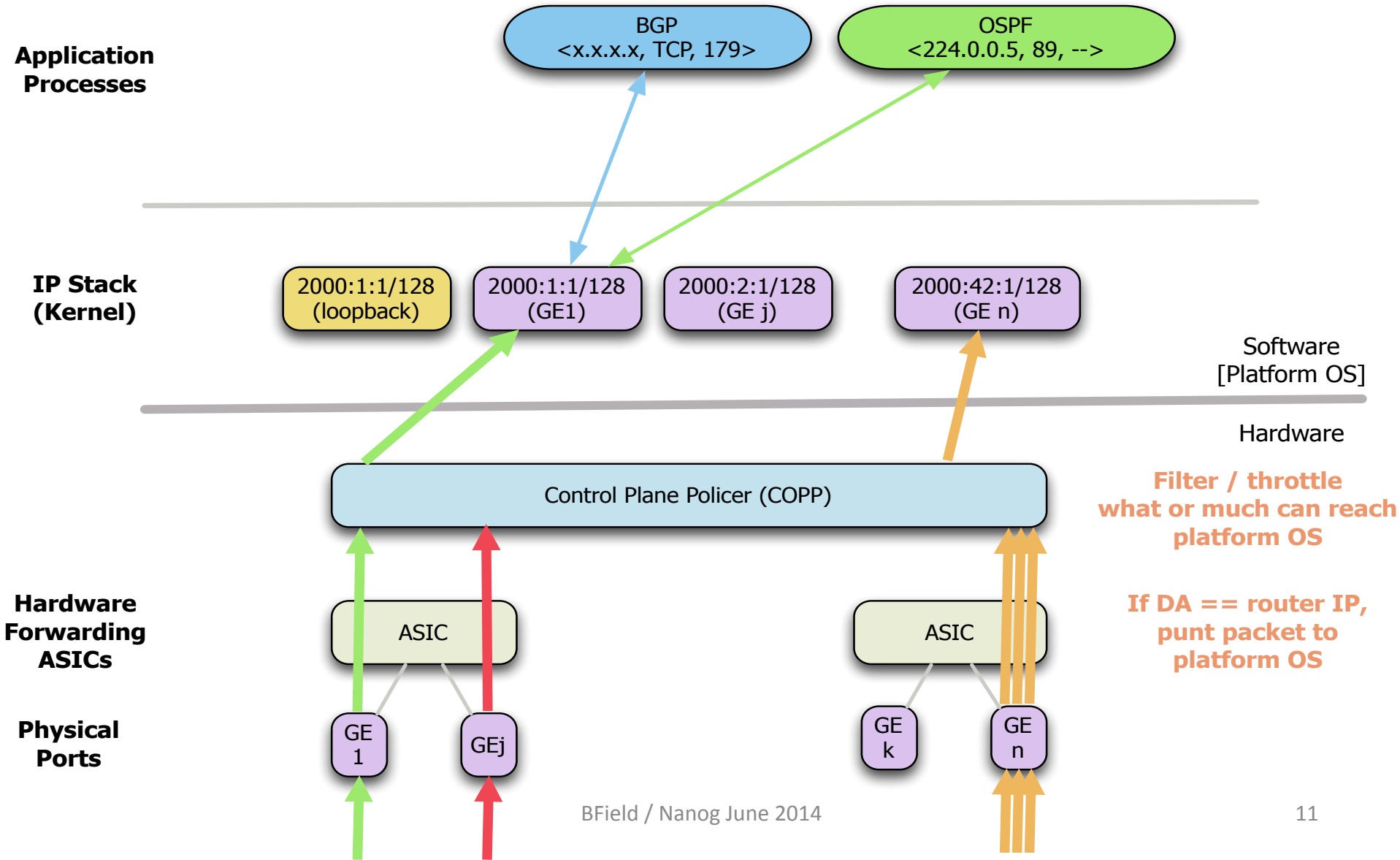


Segment Routing Operation

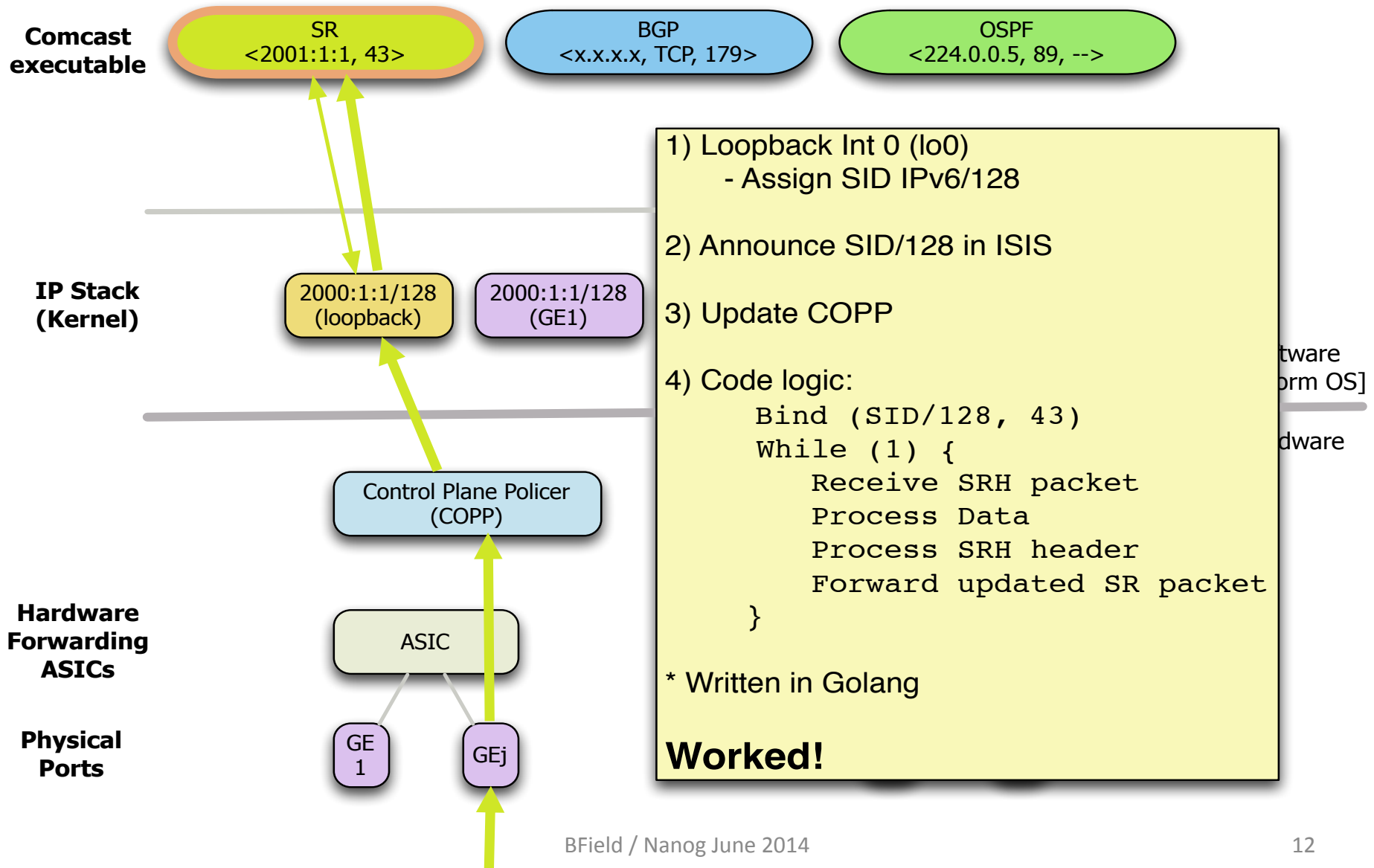
- IPv6 packet with SRH is delivered to Dest Addr .
 - Router or application
- Process data, then consult SRH and determine next “Segment” ID (SID)
- Update
 - SRH
 - IPv6 packet with next SID as next DA. Forward.
- Details shared in NANOG 58 – June 2013.
 - <http://www.nanog.org/sites/default/files/wed.general.segment.files.13.pdf>

Router Control Plane Operation

[Punt traffic 101]



What we did



The screenshot displays the Wireshark network protocol analyzer interface. The title bar indicates the file being opened is 'sr packets between arista VMs.pcapng' and the version is 'Wireshark 1.10.5 (SVN Rev 54262 from /trunk-1.10)'. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. The toolbar contains various icons for file operations, packet navigation, and analysis.

The packet list pane shows five captured packets, all of type SR (Segment Routing), with lengths of 190 bytes. The packet details pane for the first packet (Frame 1) shows the following structure:

- Ethernet II, Src: 0a:00:27:00:00:02 (0a:00:27:00:00:02), Dst: CadmusCo_a3:73:a4 (08:00:27:a3:73:a4)
- Internet Protocol Version 6, Src: 2000:10::5 (2000:10::5), Dst: 2000:10::14 (2000:10::14)
- Segment Routing Extension Header
 - Next header: 17
 - Hdr Ext Len: 16
 - Routing Type is SRH: 4
 - Next Segment: 0
 - Last Segment: 16
 - Flags: 0 [Clean-Up]
 - HMAC Key ID: 0
 - No HMAC
 - Policy List Flags: 0
 - Policy Field 1: 0
 - Policy Field 2: 0
 - Policy Field 3: 0
 - Policy Field 4: 0
 - Number of Segments: 8
 - Segment: 0
 - Segment: 1
 - Segment: 2
 - Segment: 3
 - Segment: 4

The packet bytes pane shows the raw data in hexadecimal and ASCII. The status bar at the bottom indicates 'Text item (text), 128 bytes', 'Packets: 5 · D...', and 'Profile: Default'.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2000:10::5	2000:10::14	SR	190	
2	0.005471000	2000:10::14	2000:10::16	SR	190	
3	0.022799000	2000:10::16	2000:10::15	SR	190	
4	0.035512000	2000:10::15	2000:10::16	SR	190	
5	0.049522000	2000:10::16	2000:10::14	SR	190	

```

> Frame 1: 190 bytes on wire (1520 bits), 190 bytes captured (1520 bits) on interface 0
> Ethernet II, Src: 0a:00:27:00:00:02 (0a:00:27:00:00:02), Dst: CadmusCo_a3:73:a4 (08:00:27:a3:73:a4)
> Internet Protocol Version 6, Src: 2000:10::5 (2000:10::5), Dst: 2000:10::14 (2000:10::14)
  Segment Routing Extension Header

```

```

Next header: 17
Hdr Ext Len: 16
Routing Type is SRH: 4
Next Segment: 0
Last Segment: 16
Flags: 0 [Clean-Up]
HMAC Key ID: 0
No HMAC
Policy List Flags: 0
Policy Field 1: 0
Policy Field 2: 0
Policy Field 3: 0
Policy Field 4: 0

```

Number of Segments:	8
Segment:	0
Segment:	1
Segment:	2
Segment:	3
Segment:	4

[illegible]

Text item (text), 128 bytes Packets: 5 · D... Profile: Default

What did we just do? Took the first step



Extending ... Wouldn't it be great if...

- We had the ability for the SR process to tag it's prefix announcements in ISIS that the /128 is "SR" capable?
 - Extend this to other services/features we might want to support per internal /128
- Crud– this vendor's ISIS implementation doesn't support ISIS admin tags...
- Wait, lets hack this new feature into open source ISIS and run that instead of the vendor's ISIS!

Running Modified Open Source ISIS on HOpen Platform

5) Quagga ISISd generating new TLV

1) Arista EOS running Quagga ISISd code

2) SYSID

3) New command dev'd into Quagga ISISd

8) 6 <tag, prefix> values encoded using defined TLV structure

6) We picked 200 as Type for new TLV

4) Second Arista EoS running stock ISIS has "UP" ADJ with Arista based Quagga ISISd

7) Structure for new TLV

The image is a collage of screenshots illustrating the process of running modified Open Source ISIS on HOpen Platform. It includes several key components:

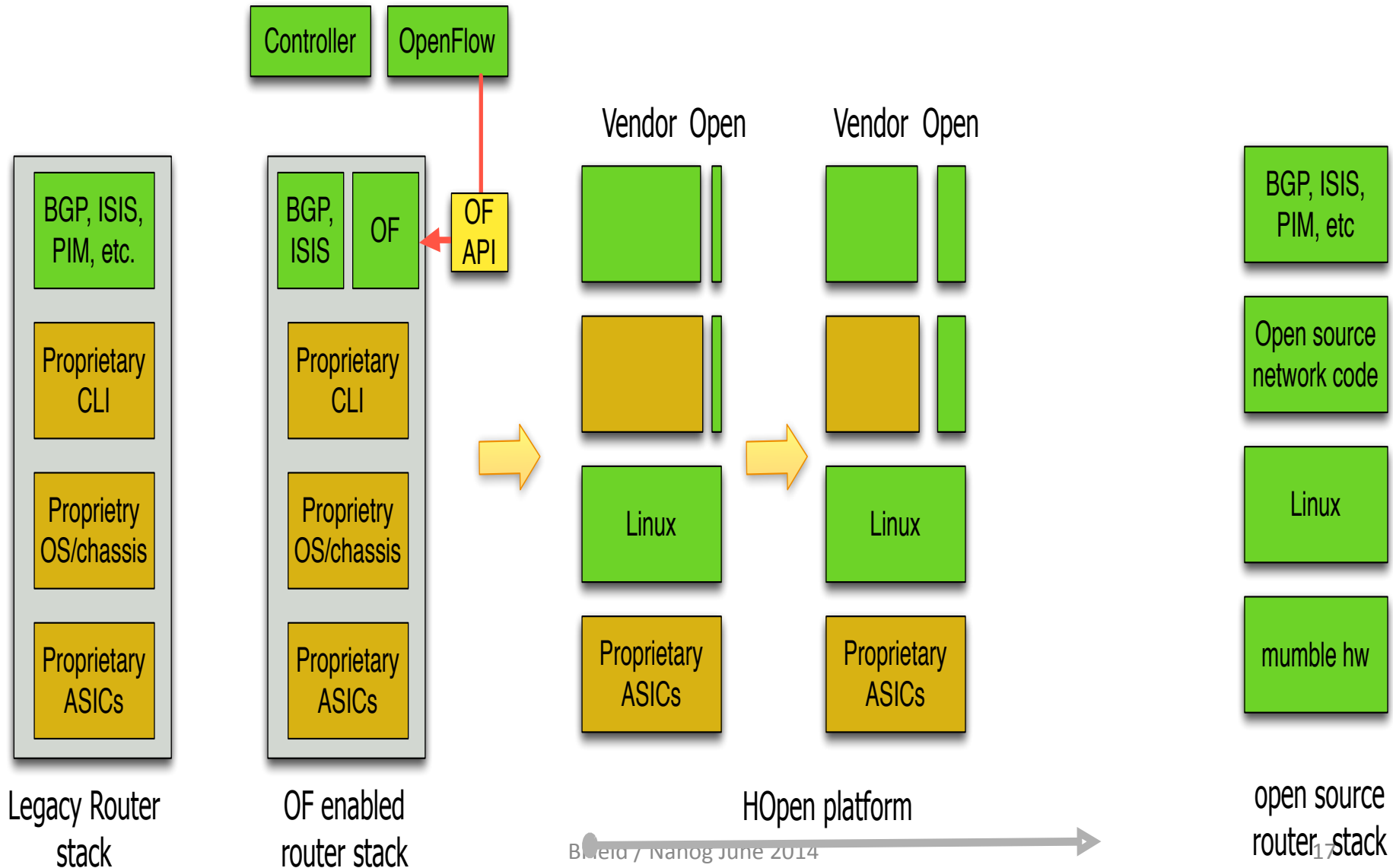
- Wireshark Packet Capture:** A screenshot of the Wireshark 1.10.5 interface showing a packet capture of ISIS traffic. The packet list shows several ISIS packets from source 'CadmusCo_2a:dd:67'. The packet details pane shows the 'IS Reachability (12)' section, which includes 'Area address(es) (4)', 'IP Interface address(es) (4)', 'IP Internal reachability (12)', 'IPv6 reachability (14)', 'Unknown code 200 (102)', and 'IS Reachability (12)'. The packet bytes pane shows the raw data of the packet.
- Arista EOS Configuration:** A screenshot of the Arista EOS CLI showing the configuration of ISIS. The configuration includes:


```
router isis 10
  net 49.0000.1000.0000.0014.00
  is-type level-2-only
  tag-v6prefix 42 42:a:a:a:a:a:a:a:a:a:a:a:a:a:a:a:a
  tag-v6prefix 52 52:b:b:b:b:b:b:b:b:b:b:b:b:b:b:b:b
  tag-v6prefix 62 62:c:c:c:c:c:c:c:c:c:c:c:c:c:c:c:c
  tag-v6prefix 71 72:d:d:d:d:d:d:d:d:d:d:d:d:d:d:d
  tag-v6prefix 72 72:d:d:d:d:d:d:d:d:d:d:d:d:d:d:d
  tag-v6prefix 73 72:d:d:d:d:d:d:d:d:d:d:d:d:d:d:d
  lsp-lifetime 65
```
- Arista EOS Neighbor Status:** A screenshot of the Arista EOS CLI showing the output of the 'show isis neighbors' command. It displays a table of ISIS neighbors with columns for Instance, VRF, System Id, Type, Interface, SNPA, State, Hold time, and Circuit Id. The output shows two neighbors in the 'UP' state.
- Quagga ISISd Development:** A screenshot of a terminal window showing the development of a new TLV structure. The terminal output shows the command 'grep 200 ./isis_tlv.h | grep BF' and the resulting output:


```
[fedoravm (39) ~/google-quagga/quagga/isisd]: grep 200 ./isis_tlv.h | grep BF
#define TAG_V6PREFIX 200 /* BF -- TBD new prefix-tag TLV code */
```
- Quagga ISISd Structure:** A screenshot of a terminal window showing the structure of the new TLV. The terminal output shows the command 'grep 200 ./isis_tlv.h | grep BF' and the resulting output:


```
/* BF: New ISIS TLV structure */
struct prefix_tag
{
  u_char tag;
  u_char prefix[16];
};
```


Second step towards HOpen...



What does this mean?

- HOpen is an alternative SDN paradigm:
 - Leverage the control plane work the vendor has done
 - Supplement with Operator code or / Open Source for new features
- HOpen -- Best of both worlds:
 - Vendor support for “legacy” features
 - Operator can develop new features as needed, prove value, vendor rolls into their code base.
 - Maybe in an Open Source kinda way

Rough consensus and working code...

- This paradigm enables [unaffiliated] individuals to create, develop and deploy new control plane protocols on production routing platforms
- Does this change how things work in IETF?
- Might this be good?
- Does NANOG start to have:
 - Hackathon's?
 - Inter-op work, etc.?

Quote time

“...With great power comes great responsibility...”

-- Voltaire (and later FDR, Ben Parker— Spiderman's uncle)

Thanks!

brian_field@cable.comcast.com

