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

Legacy Router stack

- BGP, ISIS, PIM, etc.
- Proprietary CLI
- Proprietary OS/chassis
- Proprietary ASICs

OF enabled router stack

- BGP, ISIS
- Proprietary CLI
- Proprietary OS/chassis
- Proprietary ASICs

OF Stack

- Controller
- OpenFlow

OpenFlow API

- BGP, ISIS, PIM, etc
- Open source network code
- Linux
- mumble hw
- COTS hw

Open source router stack

Web stack (role model)

HTTP

Apache

Linux

Linux

COTS hw

How do we transition across this gap?
We glue them together... “HOpen”
In theory, over time....

- **Vendor** Open
- **Proprietary ASICs**
- **Linux**

**HOpen platform (today)**

**HOpen platform (tomorrow)**

**HOpen platform (tomorrow++)**
Unicorns and rainbows -- Wouldn’t it be great if the HOpen platform existed today...?

Hybrid platform

Legacy open protocols

Insert ideas here

Vendor Proprietary code

Insert code here

Linux

Proprietary ASICs
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

![IPv6 Header](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>Protocol version (4 bits)</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>Traffic class (8 bits)</td>
</tr>
<tr>
<td>Flow Label</td>
<td>Flow label (20 bits)</td>
</tr>
<tr>
<td>Payload Len</td>
<td>Payload length (16 bits)</td>
</tr>
<tr>
<td>Hop Limit</td>
<td>Hop limit (8 bits)</td>
</tr>
<tr>
<td>Source Address</td>
<td>Source address (128 bits)</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination address (128 bits)</td>
</tr>
</tbody>
</table>

#### IPv6 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 (TCP)</td>
<td>Protocol number (6 bits)</td>
</tr>
<tr>
<td>Hdr Ext Len</td>
<td>Header extension length (16 bits)</td>
</tr>
<tr>
<td>Routing Type</td>
<td>Routing type (8 bits)</td>
</tr>
<tr>
<td>Next Segment</td>
<td>Next segment identifier (8 bits)</td>
</tr>
<tr>
<td>Last Segment</td>
<td>Last segment identifier (8 bits)</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags (16 bits)</td>
</tr>
<tr>
<td>HMAC Key ID</td>
<td>HMAC key identifier (32 bits)</td>
</tr>
<tr>
<td>Policy List Flags</td>
<td>Policy list flags (16 bits)</td>
</tr>
<tr>
<td>Segment List[0]</td>
<td>Segment list (128 bits IPv6 address)</td>
</tr>
<tr>
<td>Segment List[n]</td>
<td>Segment list (128 bits IPv6 address)</td>
</tr>
<tr>
<td>Policy List[0]</td>
<td>Policy list (128 bits IPv6 address) (optional)</td>
</tr>
<tr>
<td>Policy List[1]</td>
<td>Policy list (128 bits IPv6 address) (optional)</td>
</tr>
<tr>
<td>HMAC</td>
<td>HMAC (256 bits) (optional)</td>
</tr>
</tbody>
</table>

#### Transport & Payload

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP header layer</td>
<td>TCP header layer</td>
</tr>
<tr>
<td>payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>

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.
Router Control Plane Operation

[Punt traffic 101]

Application Processes

IP Stack (Kernel)

Hardware Forwarding ASICs

Physical Ports

Control Plane Policer (COPP)

BGP <x.x.x.x, TCP, 179>

OSPF <224.0.0.5, 89, -->

2000:1:1/128 (loopback)

2000:1:1/128 (GE1)

2000:2:1/128 (GE j)

2000:42:1/128 (GE n)

ASIC

ASIC

GE 1

GE j

GE k

GE n

If DA == router IP, punt packet to platform OS

Filter / throttle what or much can reach platform OS

BField / Nanog June 2014
What we did

1) Loopback Int 0 (lo0)
   - Assign SID IPv6/128

2) Announce SID/128 in ISIS

3) Update COPP

4) Code logic:
   Bind (SID/128, 43)
   While (1) {
     Receive SRH packet
     Process Data
     Process SRH header
     Forward updated SR packet
   }

* Written in Golang

Worked!
SR packets on the wire
What did we just do? Took the first step

Legacy Router stack

- BGP, ISIS, PIM, etc.
- Proprietary CLI
- Proprietry OS/chassis
- Proprietary ASICs

OF enabled router stack

- BGP, ISIS
- Proprietary CLI
- Proprietary OS/chassis
- Proprietary ASICs

Controller

- OpenFlow

Vendor Open

- OF API

- Linux

- Proprietary ASICs

HOpen platform

- BGP, ISIS, PIM, etc
- Open source network code
- Linux
- mumble hw

open source router stack
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

1) Arista EOS running Quagga ISISd code
2) SYSID
3) New command dev’d into Quagga ISISd
4) Second Arista EoS running stock ISIS has "UP AD" with Arista based Quagga ISISd
5) Quagga ISISd generating new TLV
6) We picked 200 as Type for new TLV
7) Structure for new TLV
8) 6 <tag, prefix> values encoded using defined TLV structure
Second step towards HOpen...

Legacy Router stack

Proprietary OS/chassis

Proprietary ASICs

OF enabled router stack

Proprietary OS/chassis

Proprietary ASICs

OpenFlow

OF API

Controller

Vendor Open

Vendor Open

Linux

Proprietary ASICs

Proprietary ASICs

BGP, ISIS, PIM, etc.

Proprietary CLI

Propriety OS/chassis

Proprietary ASICs

BGP,ISIS,PIM,etc.

Open source network code

Linux

mumble hw

open source router stack

BGP, ISIS, PIM, etc.

Open source network code

Linux

mumble hw

open source router stack
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