SRv6 Network Programming
FD.io VPP and Linux

Pablo Camarillo - Software Engineer @ Cisco Systems
Ahmed Abdelsalam – PhD student @ Gran Sasso Science Institute
Who are we?

Ahmed Abdelsalam
PhD student @ Gran Sasso Science Institute
ahmed.abdelsalam@gssi.it

Pablo Camarillo
Software Engineer @ Cisco Systems
pcamaril@cisco.com
Agenda

1. SRv6 101
2. Deployment use-cases
3. SRv6 on FD.io VPP
4. SRv6 on Linux
5. SERA
Segment Routing

- Source Routing
  - the topological and service (NFV) path is encoded in packet header

- Scalability
  - the network fabric does not hold any per-flow state for TE or NFV

- Simplicity
  - automation: TILFA sub-50msec FRR
  - protocol elimination: LDP, RSVP-TE, NSH...

- End-to-End
  - DC, Metro, WAN
Two dataplane instantiations

IPv6
• leverages RFC8200 provision for source routing extension header
• 1 segment = 1 address
• a segment list = an address list in the SRH

Segment Routing

MPLS
• leverage the mature MPLS HW with only SW upgrade
• 1 segment = 1 label
• a segment list = a label stack

IPv6
• leverages RFC8200 provision for source routing extension header
• 1 segment = 1 address
• a segment list = an address list in the SRH
IPv6 adoption is a reality

Global IPv6 traffic grew 241% in 2016

Globally IPv6 traffic will grow 16-fold from 2016 to 2021

IPv6 will be 37% of total Internet traffic in 2021

Sources: 6lab.cisco.com - Web content
Cisco VNI Global IP Traffic Forecast, 2016-2021
IPv6 provides reachability
SRv6 – Segment Routing & IPv6

- Simplicity
  - Protocol elimination
- SLA
  - FRR and TE
- Overlay
- NFV
- SDN
  - SR is de-facto SDN architecture
- 5G

SR for anything else
IPv6 for reachability
SRv6 for underlay

IPv6 for reachability

Simplification through protocol reduction

SLA through automated FRR and TE

De-facto SDN architecture
SRv6 for underlay and overlay

- NSH for NFV → Additional Protocol and State
- UDP+VxLAN Overlay → Additional Protocol just for tenant ID
- SRv6 for Underlay → Simplification, FRR, TE, SDN
- IPv6 for reachability

Multiplicity of protocols and states hinder network economics

(C) FOSDEM 2018
SR for anything: Network as a Computer
Network instruction

- 128-bit SRv6 SID
  - Locator: routed to the node performing the function
  - Function: any possible function
    - either local to NPU or app in VM/Container
  - Flexible bit-length selection
Network instruction

• 128-bit SRv6 SID
  • Locator: routed to the node performing the function
  • Function: any possible function
    either local to NPU or app in VM/Container
  • **Arguments**: optional argument bits to be used only by that SID
  • Flexible bit-length selection
Network Program

Next Segment

Locator 1 | Function 1
Locator 2 | Function 2
Locator 3 | Function 3
Network Program in the Packet Header

IPv6 header

Source Address

Locator 1
Function 1

Segment Routing Header

Active Segment

Locator 1
Function 1

Locator 2
Function 2

Locator 3
Function 3

IPv6 payload

TCP, UDP, QUIC
Argument shared between functions

“Global” Argument

Metadata TLV
Group-Based Policy
SRv6 Header

TAG

Segments Left

Locator 1  Function 1
Locator 2  Function 2
Locator 3  Function 3

Metadata TLV

Segment List[0] (128 bits IPv6 address)

... 

Segment List[n] (128 bits IPv6 address)

// Optional Type Length Value objects (variable)
SRv6 LocalSIDs
Endpoint function

SR: \{A4::1, A6::1, A8::\}

• For simplicity function 1 denotes the most basic function
• Shortest-path to the Node
For simplicity $A_k::C_j$ denotes:

- Shortest-path to the Node K and then x-connect (function C) to the neighbor J
Deployment use-cases
TILFA

• **50msec Protection** upon local link, node or SRLG failure

• **Simple** to operate and understand
  • automatically computed by the router’s IGP process
  • 100% coverage across any topology
  • predictable (backup = postconvergence)

• **Optimum backup path**
  • leverages the post-convergence path, planned to carry the traffic
  • avoid any intermediate flap via alternate path

• **Incremental deployment**

• **Distributed and Automated Intelligence**
Centralized TE

Input Acquisition
- BGP-LS
- Telemetry

Policy Instantiation
- PCEP
- BGP-TE
- Netconf / Yang

Algorithm
- SR native

Low-Latency to 7 for application ...

DC (BGP-SR) to PEER

Low Lat, Low BW

Default ISIS cost metric: 10
Overlay

- Automated
  - No tunnel to configure
- Simple
  - Protocol elimination
- Efficient
  - SRv6 for everything

IPv6 Hdr
SA = T::1, DA = V::2
Payload

IPv6 Hdr
SA = A1::0, DA = A2::C4
IPv6 Hdr
SA = T::1, DA = V::2
Payload

Green Overlay
V/64 via A2::C4
• SRv6 does not only eliminate unneeded overlay protocols
• SRv6 solves problems that these protocols cannot solve

Green Overlay V/64 via A2::C4 with Latency
## Endpoint behaviors specs summary

<table>
<thead>
<tr>
<th>Codename</th>
<th>Behavior</th>
<th>Flavor</th>
</tr>
</thead>
<tbody>
<tr>
<td>End</td>
<td>Endpoint</td>
<td>[PSP/USP flavors]</td>
</tr>
<tr>
<td>End.X</td>
<td>Endpoint with Layer-3 cross-connect</td>
<td>[PSP/USP flavors]</td>
</tr>
<tr>
<td>End.B6</td>
<td>Endpoint bound to an SRv6 policy</td>
<td></td>
</tr>
<tr>
<td>End.B6.Encaps</td>
<td>Endpoint bound to an SRv6 Encapsulation policy</td>
<td></td>
</tr>
<tr>
<td>End.DX6</td>
<td>Endpoint with decapsulation and IPv6 cross-connect</td>
<td>(per-CE VPN label)</td>
</tr>
<tr>
<td>End.DX4</td>
<td>Endpoint with decapsulation and IPv4 cross-connect</td>
<td>(per-CE VPN label)</td>
</tr>
<tr>
<td>End.DT6</td>
<td>Endpoint with decapsulation and specific IPv6 table lookup</td>
<td>(per-VRF VPN label)</td>
</tr>
<tr>
<td>End.DT4</td>
<td>Endpoint with decapsulation and specific IPv4 table lookup</td>
<td>(per-VRF VPN label)</td>
</tr>
<tr>
<td>End.DX2</td>
<td>Endpoint with decapsulation and Layer-2 cross-connect</td>
<td></td>
</tr>
</tbody>
</table>
Service chaining
Service Chaining with SRv6

Packets from are steered through a sequence of services on their way to the server.

- Services are expressed with segments
  - Flexible
  - Scalable
  - Stateless
Integrated NFV

• Stateless
  • NSH creates per-chain state in the fabric
  • SR does not
• App is SR aware or not
• App can work on IPv4, IPv6 or L2
Integrated NFV

- Integrated with underlay SLA
Integrated NFV

- Stateless
  - NSH creates per-chain state in the fabric
  - SR does not
- App is SR aware or not
- App can work on IPv4, IPv6 or L2
Integrated NFV

• Integrated with Overlay
Service Chaining with **SRv6**

**SR-Aware VNFs:**
- Leverage SRv6 Kernel support to create smarter applications
- SERA: SR-Aware Firewall (extension to iptables)

**SR-UnAware VNFs:**
- Application is not aware of SR at all
- Leverage VPP as a vm/container vSwitch to do SRv6 processing

**Types of VNFs**
SRv6 support in VPP
Vector Packet Processing

- Extensible framework that provides out-of-the-box production quality switch/router functionality (dataplane only)
- We’ve implemented the entire SRv6 Network Programming on it
SR-UnAware VNFs

- End.AM – Endpoint to SR-unaware app via masquerading
- End.AD – Endpoint to SR-unaware app via dynamic proxy
- End.ASM – Endpoint to SR-unaware app via shared memory
End.AM – Endpoint to SR-unaware app via masquerading

RFC2460:
“A Routing header is not examined or processed until it reaches the node identified in the Destination Address field of the IPv6 header.”

- Ingress:
  - Active SID is E1::A where function 0xA is associated with End.AM
  - Replace DA with the last segment B::
  - Forward to VNF (OIF, NH)

- Egress:
  - Inspect SRH and update DA with active segment C3::

```
IPv6Hdr   SA = A::, DA = E1::A
SRHdr     ( B::, C3::, E1::A ) SL=2
Payload
```

```
IPv6Hdr   SA = A::, DA = C3::
SRHdr     ( B::, C3::, E1::A ) SL=1
Payload
```
End.AD – Endpoint to SR-unaware app via dynamic proxy

- **Ingress:**
  - Active SID is E1::B where function 0xB is associated with End.AD
  - Pop and **store** outer IP and SR headers
  - Forward to VNF (OIF, NH)

- **Egress:**
  - Push the IP and SR headers
  - Forward based on next segment

- **Valid for IPv4 and IPv6 traffic**
- **Per-chain dynamic configuration**

```plaintext
VPP: show sr localsid
LocalSID  Behavior
E1::B  End.AD {OIF: TenGE0/1/0, NH: 2001::a, IIF: TenGE0/2/0}
Total SR LocalSIDs: 1

Ingress:
- Active SID is E1::B where function 0xB is associated with End.AD
- Pop and store outer IP and SR headers
- Forward to VNF (OIF, NH)

Egress:
- Push the IP and SR headers
- Forward based on next segment
```

- **IPv6 Header**
  - SA = A::, DA = B::
  - Payload

- **SR Header**
  - SL = 2

- **IPv6 Header**
  - SA = C1::, DA = E1::C
  - Payload
End.ASM – Endpoint to SR-unaware app via shared mem.

1. Put the received packet in a shared memory region
2. Perform SR processing on the host
   Pass a `pointer` of the inner packet to S2
3. Perform SR processing on the host
   Pass a `pointer` of the inner packet to S3
4. Move the packet from the shared memory into the output iface buffer ring

- Valid for IPv4 and IPv6 traffic
- Max. theoretical achievable performance
SRv6 LocalSID development kit

• Users can write their own SRv6 LocalSIDs functions as VPP plugins
• There is a SR LocalSID plugin template
  • Starting point for your own developments
  • We do the ‘SRH’ processing for you
SRv6 support in Linux
Kernel 4.10

- The first support of SRv6 in Linux kernel, released in February 2017.
- **End, T.Insert6, and T.Encaps6** behaviors were supported
- The SRv6 support is enabled on interface basis. All IPv6 addresses assigned to an SRv6-enabled interface are treated as local SID
- `iproute2` was extended to support creating an SR policy.

```
sysctl -w net.ipv6.conf.all.seg6_enabled=1
sysctl -w net.ipv6.conf.<device>.seg6_enabled=1
ip -6 route add <prefix> encap seg6 mode <encapmode> segs <segments>  dev <device>
```
Kernel 4.14

• Another milestone in SRv6 support in Linux, released in November 2017

• More SRv6 behaviors were supported: \texttt{T.Encaps.L2, T.Insert4, T.Encaps4, End.X, End.T, End.DX2, End.DX4, End.DX6, End.DT6, End.B6, End.B6.Encaps.}

• \texttt{IProute2} extended to associate one of the new behaviors to a local SID

\begin{verbatim}
ip -6 route add <segment> encap seg6local action <action> <params> \ dev <device> table localsid
\end{verbatim}
SREXT

- An external kernel module provides advanced Segment Routing functions.
- Complements the existing SRv6 kernel implementation.
- Supports several proxy behaviors that enable SR-unaware service functions to be included in an SRv6 service chain.
SR-aware Service Functions

• No more state information is required per a VNF.

• Leverage SID arguments for local parameters

• Leverage TLVs to pass metadata between services

• Have a vision of the whole packet path for better protection
SERA
SERENA

- SEgment Routing Aware firewall
- The first-ever SRv6-aware Network function
- An advanced SR aware firewall, with extended matching and actions capabilities.
- It allows matching information from original packet, SRH, and the outer packet.
- It’s capable of performing SR-specific actions.
SERA has a full view of SR Encapsulated packets

It has no way to analyze inner packet - matching capabilities limited to outer packet.

We can match all headers of a received packet.

SR Information is hidden from the firewall.
SERA – Netfilter extensions

• Implemented as an extension to the existing Linux ip6tables firewall

• Three new extensions have been to the netfilter implementation as follows:
  • `ip6t_srh`: matches information of SRH
  • `ip6t_inner6`: matches information of inner packet
  • `ip6t_SEG6`: performs SR- Specific actions
SERA – Iptables extensions

• Three new shared libraries have been added to the iptables user-space utility implementation to support the new features:
  • libip6t_srh: to add rules that match based on SRH
  • libip6t_inner6: to add rules that match based on inner packet
  • libip6t_SEG6: to add rules that perform SR-specific actions
$ ip6tables -m srh -h
........
srh match options:
[!] --srh-next-hdr
[!] --srh-hdr-len-eq
[!] --srh-hdr-len-gt
[!] --srh-hdr-len-lt
[!] --srh-segs-left-eq
[!] --srh-segs-left-gt
[!] --srh-segs-left-lt
[!] --srh-last-entry-eq
[!] --srh-last-entry-gt
[!] --srh-last-entry-lt
[!] --srh-tag

$ ip6tables -m srh -h
........
srh match options:
[!] --inner6-src ip6_addr[/mask]
[!] --inner6-dst ip6_addr[/mask]

$ ip6tables -j SEG6 -h
........
SEG6 target options:
--seg6-action ACTION
ACTION: go-next | skip-next | go-last
SERA - Examples

ip6tables -I INPUT -m inner6 --inner6-src fc00:a::/64 \ 
--inner6-dst fc00:b::/64 -m srh --srh-next-hdr 6 \ 
--srh-segs-left-gt 5 --srh-tag 0 -j DROP

ip6tables -I INPUT -m inner6 --inner6-src fc00:a::/64 \ 
--inner6-dst fc00:b::/64 -m srh --srh-next-hdr 6 \ 
--srh-segs-left-gt 5 --srh-tag 0 -j SEG6 --seg6-action go-last
Conclusion
SRv6 timeline

First SRv6 demo: Spray use-case
- VPP
- ASR9k
- ASR1k

Fretta
- First SRv6 HW demo in merchant silicon
- VPN DP use-case

Cisco Live US
- SRv6 VPN
  - ASR1k
  - ASR9k
  - Fretta
  - VPP+NFV
  - BGP Control Plane

SD-WAN summit
- SRv6 for the SD-WAN
  - ASR1k

Apr 2016
- First SRv6 demo: Spray use-case

Mar 2017
- Fretta
  - First SRv6 HW demo in merchant silicon
  - VPN DP use-case

Apr 2017
- Cisco Live US
  - SRv6 VPN
  - ASR1k
  - ASR9k
  - Fretta
  - VPP+NFV
  - BGP Control Plane

May 2017
- SD-WAN summit
  - SRv6 for the SD-WAN
  - ASR1k

Jun 2017
- Barefoot
  - SRv6 VPN HW demo

Aug 2017
- SR VPN InterOp
  - Fretta
  - ASR9k
  - ASR1k
  - VPP
  - Linux (srext)
  - Barefoot

Sep 2017
- More to come...
  - 5G + Network slicing

2018

CC FOSDEM 2018
IPv6 provides reachability
SRv6 unleashes IPv6 potential
Homework time!

- Go on [www.segment-routing.net](http://www.segment-routing.net) and check the latest demos
- Read the IETF draft: [draft-filsfils-spring-srv6-network-programming](http://draft-filsfils-spring-srv6-network-programming)
- Play with VPP and srext

- Create your own SRv6 aware apps. There is business. It’s easy. Contact us!
Thank you!

www.segment-routing.net
SRv6 timeline

First SRv6 demo: Spray use-case
- VPP
- ASR9k
- ASR1k

Fretta
- First SRv6 HW demo in merchant silicon
- VPN DP use-case

Cisco Live US SRv6 VPN
- ASR1k
- ASR9k
- Fretta
- VPP+NFV
- BGP Control Plane

SD-WAN summit SRv6 for the SD-WAN
- ASR1k

SRv6 VPN+NFV: MPLS World Con.
- VPP
- Linux (srext)

Barefoot
- SRv6 VPN HW demo

SR VPN InterOp
- Fretta
- ASR9k
- ASR1k
- VPP
- Linux (srext)
- Barefoot

More to come...
- 5G + Network slicing

(CC) FOSDEM 2018