Forefront of SRv6

Open Source Implementations

JANOG43 Meeting @甲府 ｜ 2019年1月23日

株式会社トヨタIT開発センター
海老澤 健太郎 Kentaro Ebisawa

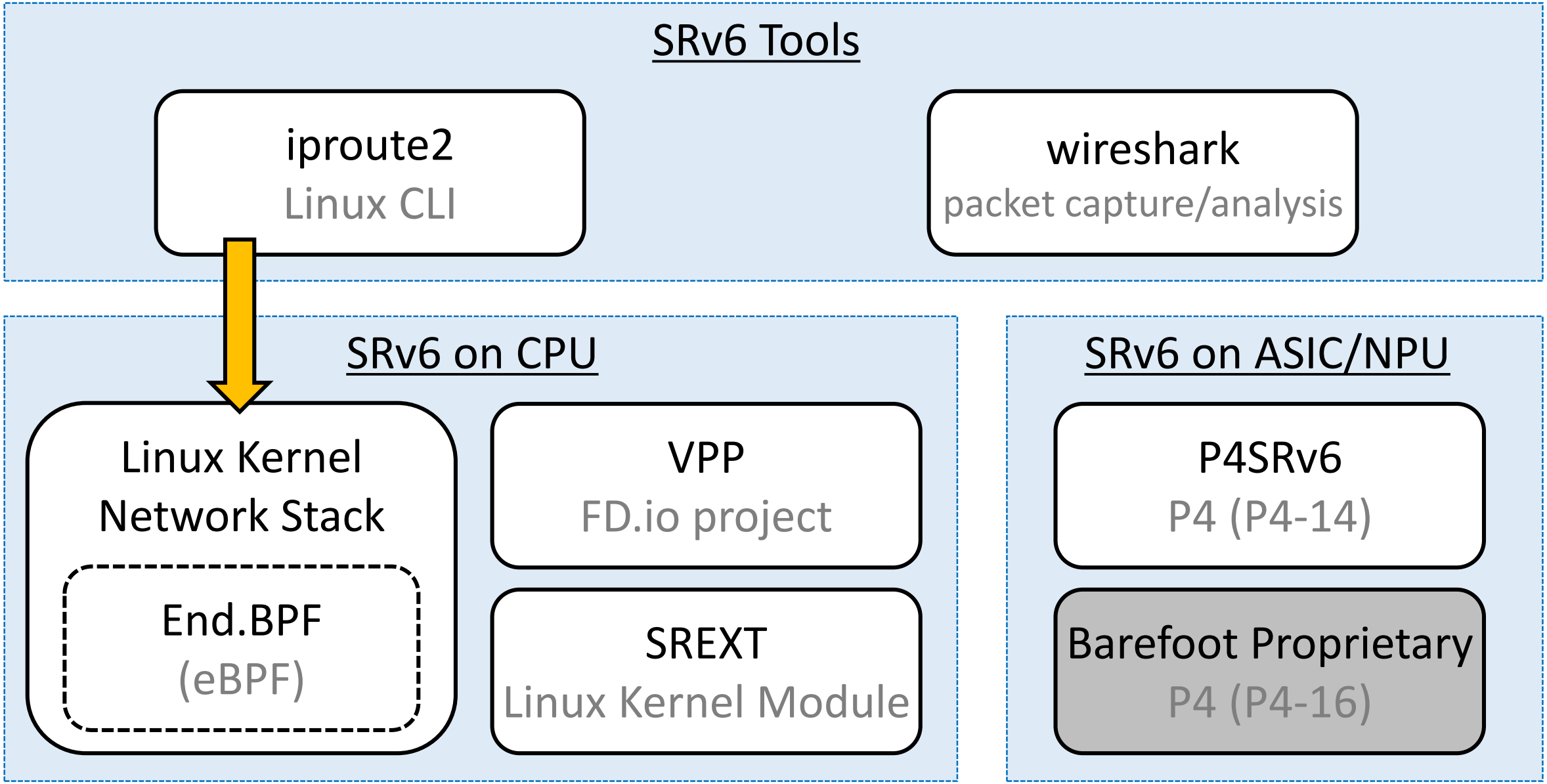
このセッションでお伝えしたいこと

- SRv6のオープンソース実装の紹介
- SRv6 Functions 実装状況
- 設定コマンドと設計思想の違い
  - VPP vs Linux (iproute2)
- どこに・どのようにSRv6が実装されてるか？（Source Code）

Table of Contents

SRv6 Open Source implementations

SRv6 Open Source implementations



TOYOTA INFOTECH

SRv6 in Linux Kernel Network Stack

- Since Linux 4.10
  - By David Lebrun@UCLouvain（ベルギー）
  - https://segment-routing.org/

- “Light Weight Tunnel” (LWTunnel) として実装
  - lwtunnel_encap_types { MPLS, IP, ILA, IP6, SEG6, BPF, SEG6_LOCAL }
  - IPv6 “route” としてルーティングテーブルに設定（デバイスではなく）

```
$ ip -6 route
c0be:fe::/64  encap seg6 mode inline segs 4 [ c0be::1 c0be::2 c0be::3 :: ]
                via 2001:db8::1 dev lxcbr0 metric 1024 linkdown pref medium
fc00::1  encap seg6local action End via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
fc00::2  encap seg6local action End.X nh6 fc00::1:1 via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
fc00::3  encap seg6local action End.T table 100 via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
```

TOYOTA INFOTECH

SRv6 in Linux Kernel Network Stack

- Since Linux 4.10
  - By David Lebrun@UCLouvain (Belgium)
  - https://segment-routing.org/

- Implemented as “Light Weight Tunnel” (LWTunnel)
  - lwtunnel_encap_types { MPLS, IP, ILA, IP6, SEG6, BPF, SEG6_LOCAL }
  - Configured in routing table via IPv6 “route” (not as tunnel devices)

```
$ ip -6 route
c0be:fe::/64  encap seg6 mode inline segs 4 [ c0be::1 c0be::2 c0be::3 :: ]
                via 2001:db8::1 dev lxcbr0 metric 1024 linkdown pref medium
fc00::1  encap seg6local action End via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
fc00::2  encap seg6local action End.X nh6 fc00::1:1 via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
fc00::3  encap seg6local action End.T table 100 via 2001:db8::1
                dev lxcbr0 metric 1024 linkdown pref medium
```

TOYOTA INFOTECH

End.BPF (Linux Kernel eBPF)

- Since Linux 4.18
  - https://segment-routing.org/index.php/Implementation/BPF

- SEG6_LOCAL (LWTunnel) のアクションの1つとして実装
- eBPFを用いてプログラムした独自処理を実行させる

※ 詳細は「SRv6 Academy Update」で

```
ip -6 route add dead::beef encap seg6local action End.BPF
        endpoint object my_code.o section my_function dev eth0
```

eBPFプログラム

TOYOTA INFOTECH

End.BPF (Linux Kernel eBPF)

- Since Linux 4.18
  - https://segment-routing.org/index.php/Implementation/BPF

- One of the actions of SEG6_LOCAL (LWTunnel)
- Invoke eBPF program with custom functionality

※ Details in next session “SRv6 Academy Update”

```
ip -6 route add dead::beef encap seg6local action End.BPF
        endpoint object my_code.o section my_function dev eth0
```

eBPF program

TOYOTA INFOTECH

SREXT (Linux Kernel Module)

- https://netgroup.github.io/SRv6-net-prog/
- By “Networking Group” @ University of Rome Tor Vergata, Italy
- SRv6非対応VNFのサービスチェイニングが可能（End.AD, End.AM）

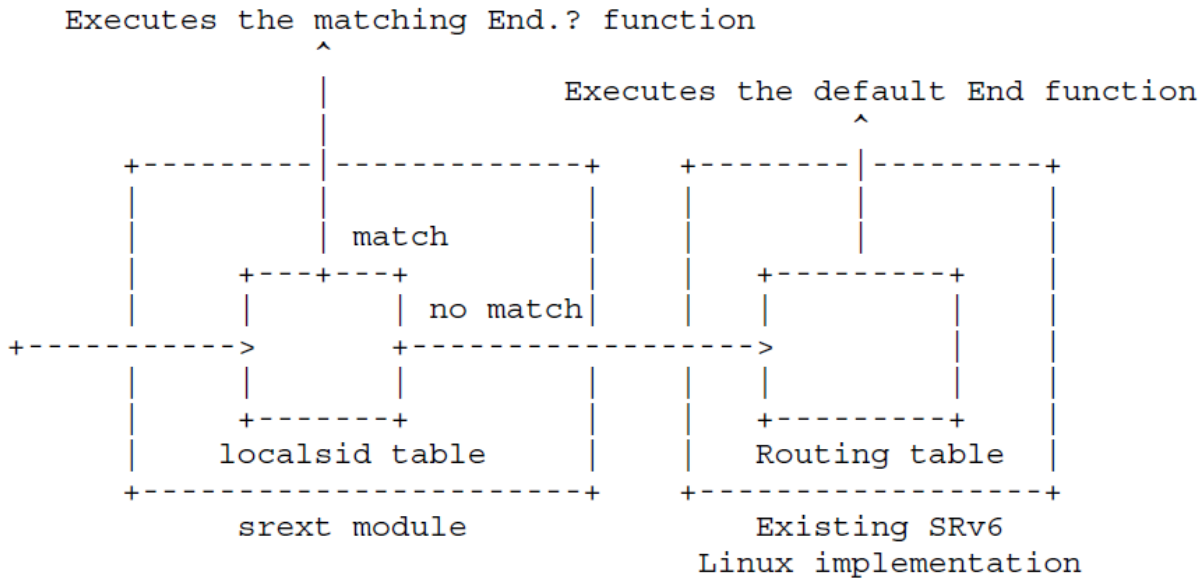


Figure 1 Segment Routing processing of IPv6 packets

Design documents:
“Linux implementation of SRv6 Network Programming model”

TOYOTA INFOTECH

SREXT (Linux Kernel Module)

- https://netgroup.github.io/SRv6-net-prog/
- By “Networking Group” @ University of Rome Tor Vergata, Italy
- Support Service Chaining of non-SRv6 capable VNFs (End.AD, End.AM)

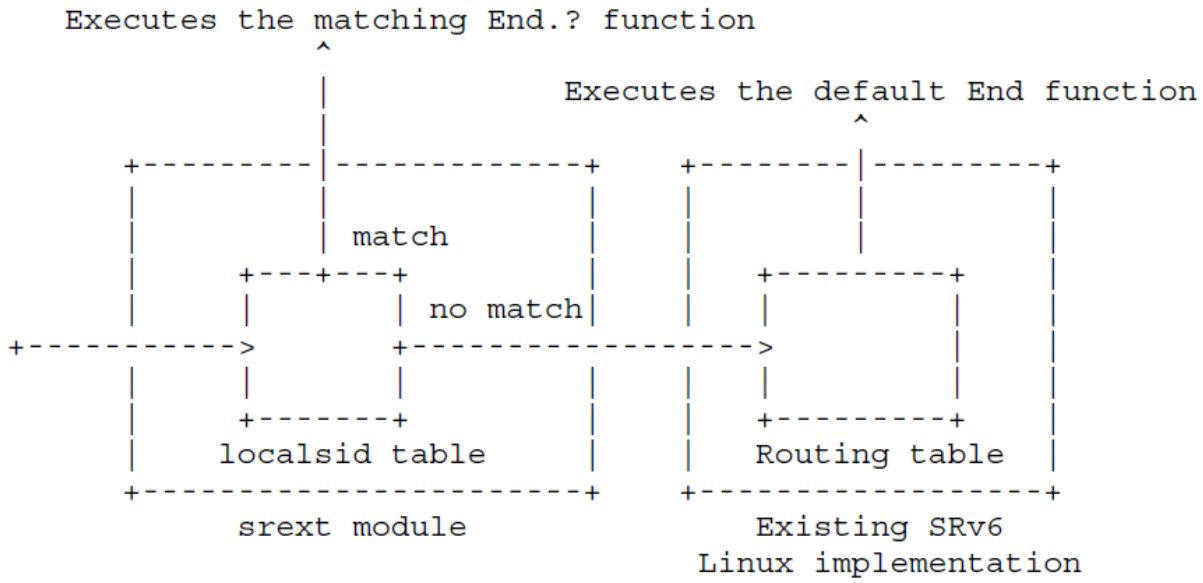


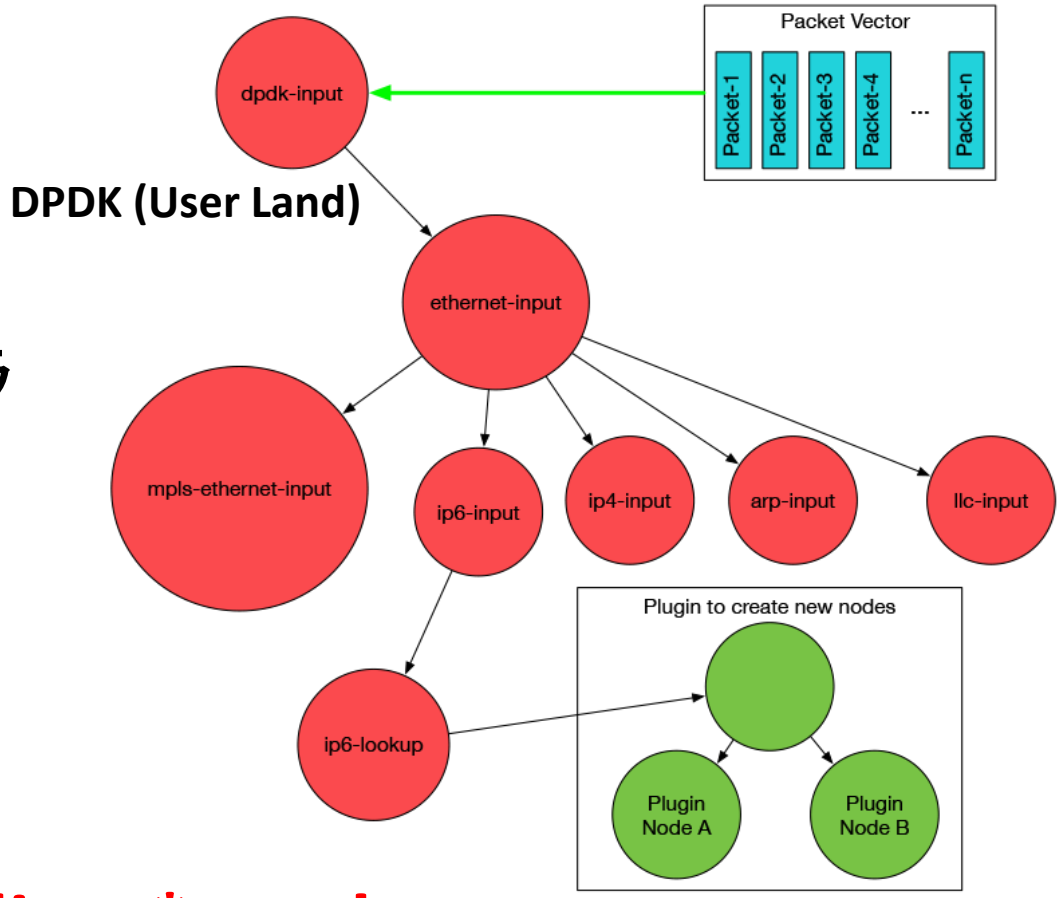
Figure 1 Segment Routing processing of IPv6 packets

Design documents:
“Linux implementation of SRv6 Network Programming model”

SRv6 on VPP (Vector Packet Processing)

- パケットをベクトル単位で入力
- node（ethernet, mpls, ip6, arp …）のグラフ順に処理を実行

- 新しい処理はnodeの改造やPluginを作成し追加可能



SRv6 node: sr_localsid_node, sr_policy_*_node

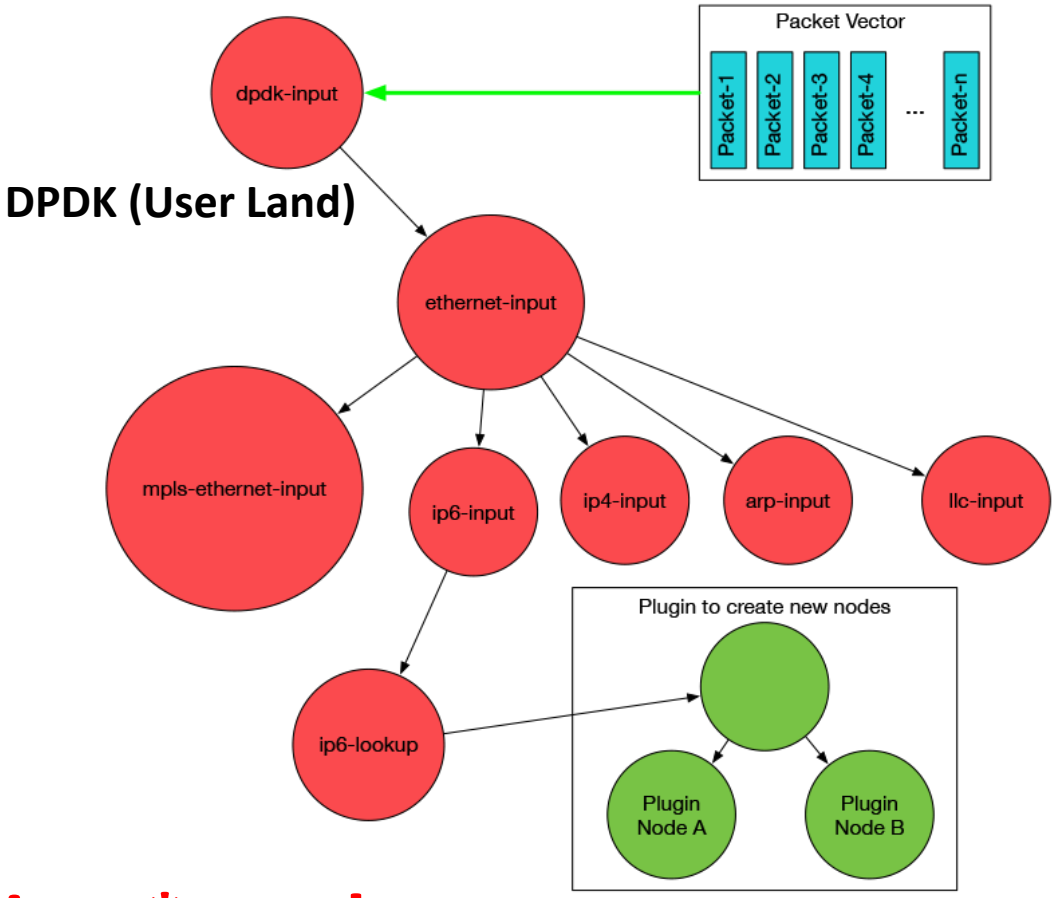
https://fd.io/technology/

TOYOTA INFOTECH

SRv6 on VPP (Vector Packet Processing)

- Input as Packet Vector
- Process based on graph of nodes (ethernet, mpls, ip6, arp …)

- New functionality could be added by customizing node or creating plugin



SRv6 node: sr_localsid_node, sr_policy_*_node

https://fd.io/technology/

TOYOTA INFOTECH

P4SRv6 (SRv6 on P4-14)

README.md

p4srv6 ... proto-typing SRv6 functions with P4 lang.

The objective of this project is to implement SRv6 functions still under discussion using P4 Lang to make runni available for testing and demo. Since there is no Open Source P4 switch implementation supporting SRv6, this basic switch features required to test SRv6.

List of SRv6 functions of interest and status:

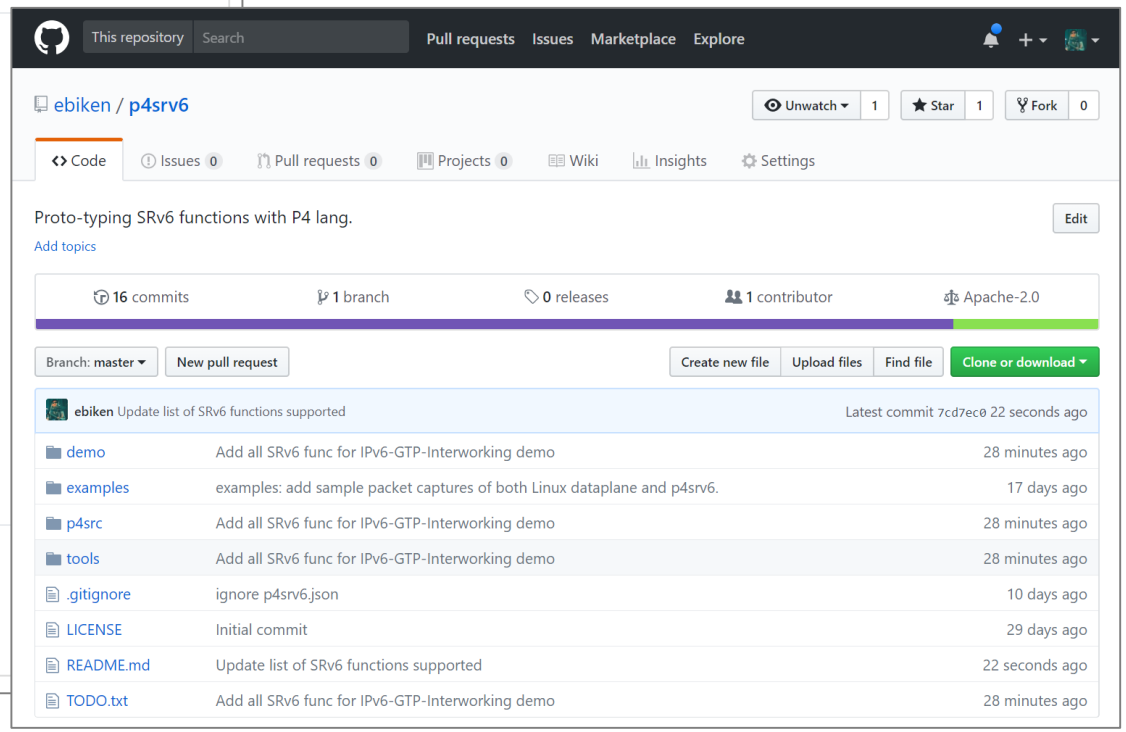
- draft-filsfils-spring-srv6-network-programming-04
  - T.Insert
  - T.Encaps, T.Encaps.Red
  - End, End.DT6
- draft-ietf-dmm-srv6-mobile-uplane-01
  - End.MAP (future)
  - End.M.GTP6.D
  - End.M.GTP6.E
  - End.M.GTP4.E (future)
  - T.M.Tmap (future)
  - End.Limit (not planned)

List of helper actions:

- GTP
  - Encap/Decap GTP-U

https://github.com/ebiken/p4srv6

- SRv6 mobile user plane functions 実装 (draft-ietf-dmm-srv6-mobile-uplane-01)
- 制約が多いので P4-16 で書き直す予定



TOYOTA INFOTECH

P4SRv6 (SRv6 on P4-14)

README.md

p4srv6 ... proto-typing SRv6 functions with P4 lang.

The objective of this project is to implement SRv6 functions still under discussion using P4 Lang to make runni available for testing and demo. Since there is no Open Source P4 switch implementation supporting SRv6, this basic switch features required to test SRv6.

List of SRv6 functions of interest and status:

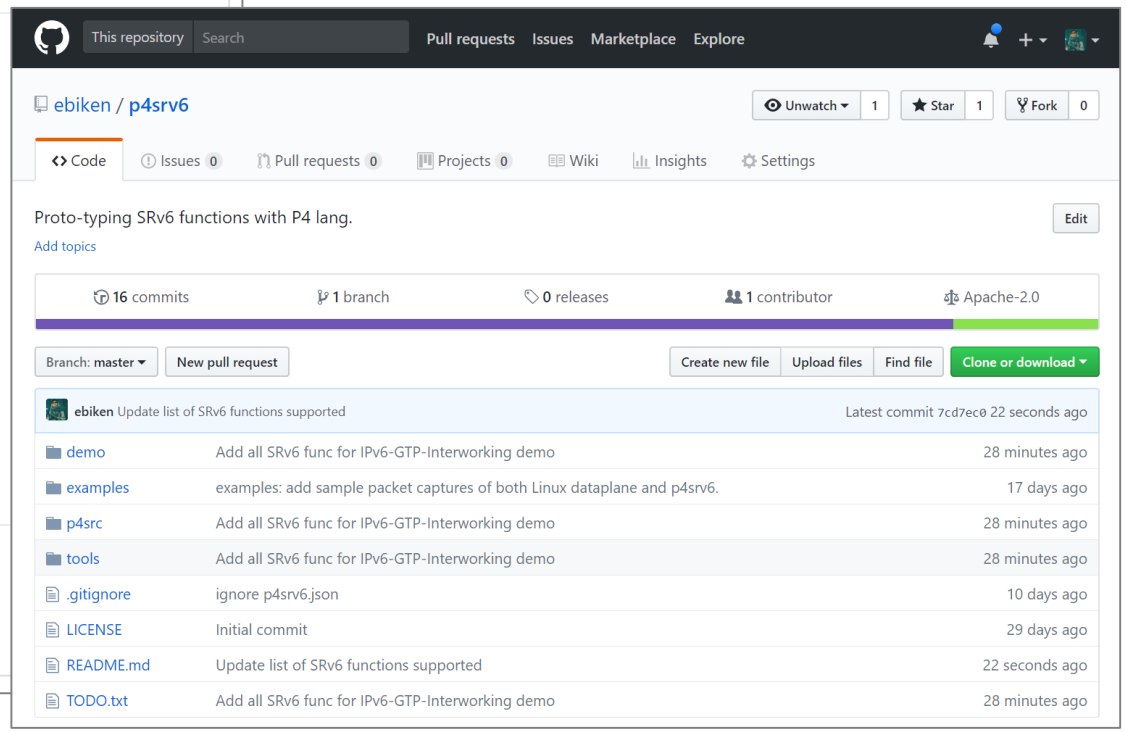
- draft-filsfils-spring-srv6-network-programming-04
  - T.Insert
  - T.Encaps, T.Encaps.Red
  - End, End.DT6
- draft-ietf-dmm-srv6-mobile-uplane-01
  - End.MAP (future)
  - End.M.GTP6.D
  - End.M.GTP6.E
  - End.M.GTP4.E (future)
  - T.M.Tmap (future)
  - End.Limit (not planned)

List of helper actions:

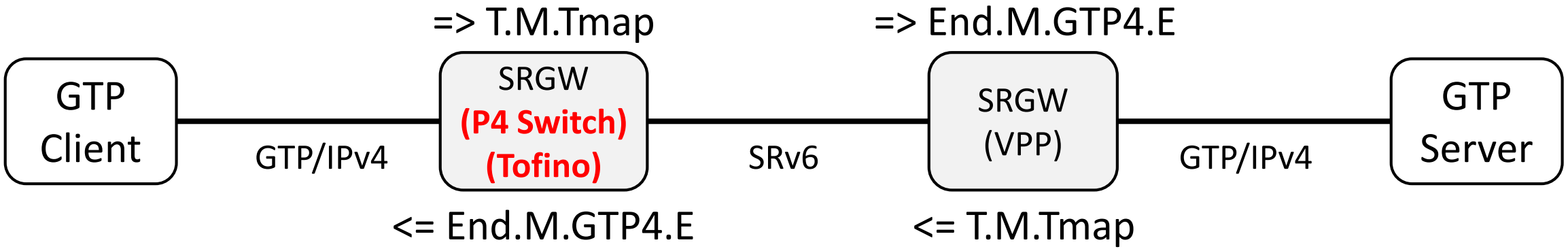
- GTP
  - Encap/Decap GTP-U

https://github.com/ebiken/p4srv6

- SRv6 mobile user plane functions (draft-ietf-dmm-srv6-mobile-uplane-01)
- Planning to re-write using P4-16



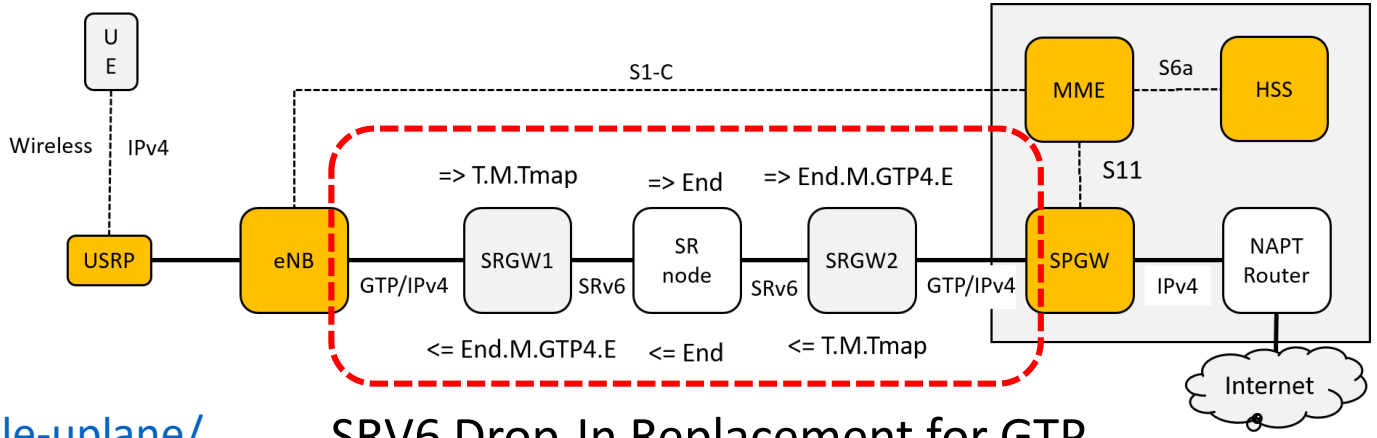
（参考）SRv6 モバイル機能のプロトタイピング



VPP拡張はPablo@Ciscoが開発

Switch ASICでGTP/SRv6変換を実行

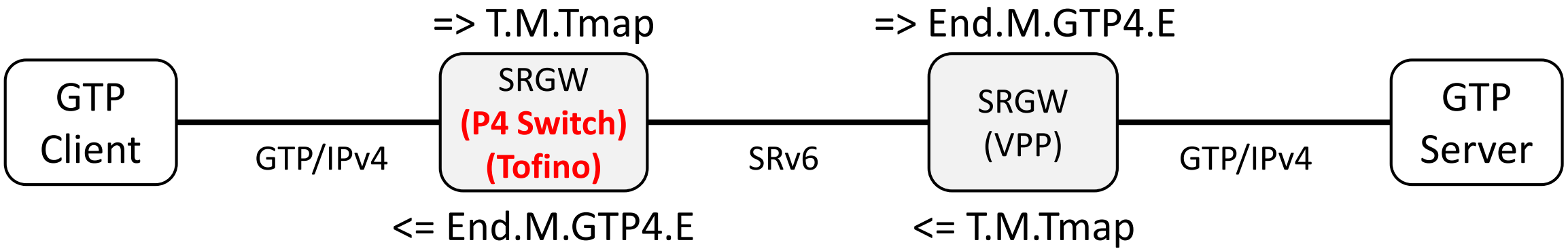
Barefoot P4-16 SRv6 実装をベースに
Function を追加（by APRESIA Systems）



SRV6 Drop-In Replacement for GTP

Segment Routing IPv6 for Mobile User Plane
https://datatracker.ietf.org/doc/draft-ietf-dmm-srv6-mobile-uplane/

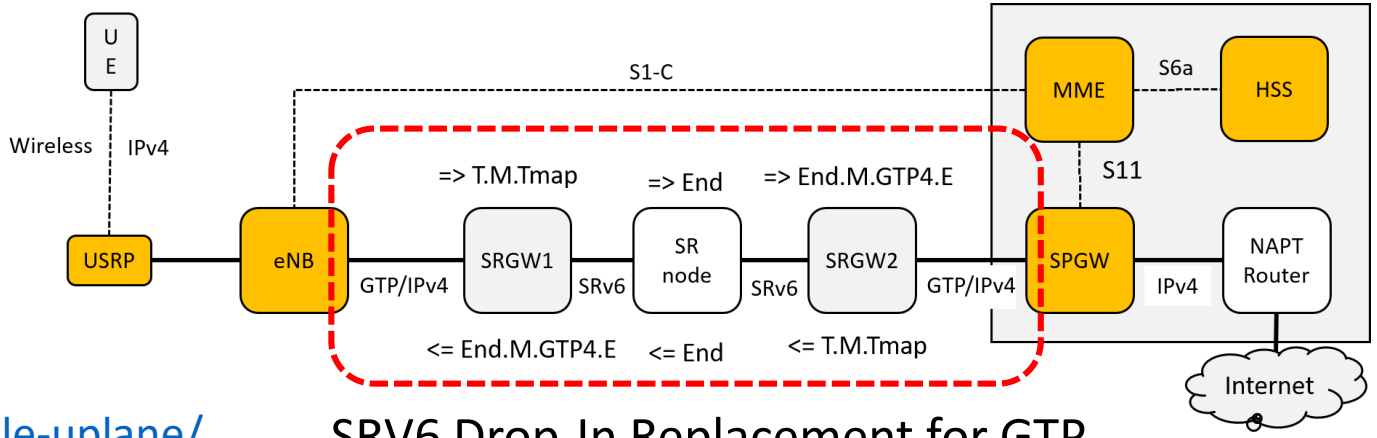
（FYI）SRv6 mobile functions (proto type)



VPP extension developed by Pablo@Cisco

GTP/SRv6 translation on Switch ASIC

Based on Barefoot P4-16 SRv6 code.
Functions added by APRESIA Systems.



SRV6 Drop-In Replacement for GTP

Segment Routing IPv6 for Mobile User Plane
https://datatracker.ietf.org/doc/draft-ietf-dmm-srv6-mobile-uplane/

TOYOTA INFOTECH

iproute2

- Linux network stack の CLI で、netlink を用いて設定・参照

- Linux Kernel と SRv6 をサポートしているバージョンが異なる
  - v4.12.0 : SEG6_IPTUN_MODE_ENCAP & INLINE (encap / inline)
  - v4.14.0 : SEG6_IPTUN_MODE_L2ENCAP (l2encap)
  - v4.14.0 : End.*
  - v4.18.0 : +End.BPF

- 欲しいバージョンを自分で make する（必要に応じて）
  - Tar ball: https://www.kernel.org/pub/linux/utils/net/iproute2/
  - Source: git clone git://git.kernel.org/pub/scm/network/iproute2/iproute2.git

TOYOTA INFOTECH

iproute2

- CLI for Linux network stack using netlink to monitor / configure.

- Supported SRv6 function differs between Linux Kernel and iproute2 versions
  - v4.12.0 : SEG6_IPTUN_MODE_ENCAP & INLINE (encap / inline)
  - v4.14.0 : SEG6_IPTUN_MODE_L2ENCAP (l2encap)
  - v4.14.0 : End.*
  - v4.18.0 : +End.BPF

- Build required version by your self (if required)
  - Tar ball: https://www.kernel.org/pub/linux/utils/net/iproute2/
  - Source: git clone git://git.kernel.org/pub/scm/network/iproute2/iproute2.git

TOYOTA INFOTECH

Wireshark

```
Internet Protocol Version 6, Src: db8::1, Dst: db8::2
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0100 0000 1011 1000 1100 = Flow Label: 0x40b8c
    Payload Length: 160
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 64
    Source: db8::1
    Destination: db8::2
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPv6 (41)
      Length: 6
      [Length: 56 bytes]
      Type: Segment Routing (4)
      Segments Left: 2
      First segment: 2
    > Flags: 0x00
      Reserved: 0000
      Address[0]: c0be::3
      Address[1]: c0be::2 [next segment]
      Address[2]: db8::2
    > [Segments in Traversal Order]
> Internet Protocol Version 6, Src: db8::1, Dst: db8:b::3
> Internet Control Message Protocol v6
```

WIRESHARK

https://www.wireshark.org/

- SRHの中身をグラフィカルに表示
- Linux & Windows & macOS 対応

TOYOTA INFOTECH

Wireshark

```
Internet Protocol Version 6, Src: db8::1, Dst: db8::2
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0100 0000 1011 1000 1100 = Flow Label: 0x40b8c
    Payload Length: 160
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 64
    Source: db8::1
    Destination: db8::2
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPv6 (41)
      Length: 6
      [Length: 56 bytes]
      Type: Segment Routing (4)
      Segments Left: 2
      First segment: 2
    > Flags: 0x00
      Reserved: 0000
      Address[0]: c0be::3
      Address[1]: c0be::2 [next segment]
      Address[2]: db8::2
    > [Segments in Traversal Order]
> Internet Protocol Version 6, Src: db8::1, Dst: db8:b::3
> Internet Control Message Protocol v6
```

WIRESHARK

https://www.wireshark.org/

- Graphical view of SRH
- Runs on Linux & Windows & macOS

SRv6 Functions ... Implementation Status

TOYOTA InfoTech

SRv6 on VPP (by FD.io project)

Supported functions as of 2019/01/16

http://www.segment-routing.net/open-software/vpp/

Name	Description	Release
T.Insert	Transit behavior with insertion of an SRv6 Policy	**17.04** (April 2017)
T.Encaps	Transit behavior with encapsulation in an SRv6 policy	**17.04** (April 2017)
T.Encaps.L2	T.Encaps behavior of the received L2 frame	**17.04** (April 2017)

Name	Description	Release
End	Endpoint function	**17.04** (April 2017)
End.X	Endpoint function with Layer-3 cross-connect	**17.04** (April 2017)
End.T	Endpoint function with specific IPv6 table lookup	**17.10** (October 2017)
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect	**17.04** (April 2017)
End.DX6	Endpoint with decapsulation and IPv6 cross-connect	**17.04** (April 2017)
End.DX4	Endpoint with decapsulation and IPv4 cross-connect	**17.04** (April 2017)
End.DT6	Endpoint with decapsulation and IPv6 table lookup	**17.04** (April 2017)
End.DT4	Endpoint with decapsulation and IPv4 table lookup	**17.04** (April 2017)
End.B6	Endpoint bound to an SRv6 policy	**17.04** (April 2017)
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy	**17.04** (April 2017)
End.BM	Endpoint bound to an SR-MPLS Policy	In development
End.S	Endpoint in search of a target in table T	In development
End.AS	Endpoint to SR-unaware APP via static proxy	**18.04** (April 2018)
End.AD	Endpoint to SR-unaware APP via dynamic proxy	**18.04** (April 2018)
End.AM	Endpoint to SR-unaware APP via masquerading	**18.04** (April 2018)

TOYOTA InfoTech

SRv6 Functions on Linux (as of 2019/01/21)

Function	Linux	iproute2	Description
T.Insert	4.10, SREXT	4.12	Transit behavior with insertion of an SRv6 Policy
T.Encaps	4.10, SREXT	4.12	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.L2	4.14	4.14	T.Encaps behavior of the received L2 frame

Function	Linux	iproute2	Description
End.BPF	4.18	4.18	Endpoint for eBPF program invocation

Function	Linux	iproute2	Description
End.AM	SREXT	4.14	Endpoint to SR-unaware APP via masquerading
End.AS	-	4.14	Endpoint to SR-unaware APP via static SR proxy
End.AD4 (AD6)	SREXT	-	Endpoint to IPv4 (v6) SR-unaware APP via dynamic proxy
End.EAD4 (EAD6)	SREXT	-	Extended End.AD4 (AD6) behavior that allow SR-unaware VNFS to be the last SF in SFC

Function	Linux	iproute2	Description
End	4.14, SREXT	4.14	Endpoint function
End.X	4.14, SREXT	4.14	Endpoint function with Layer-3 cross-connect
End.T	4.14	4.14	Endpoint function with specific IPv6 table lookup
End.DX2	4.14, SREXT	4.14	Endpoint with decapsulation and Layer-2 cross-connect
End.DX6	4.14, SREXT	4.14	Endpoint with decapsulation and IPv6 cross-connect
End.DX4	4.14, SREXT	4.14	Endpoint with decapsulation and IPv4 cross-connect
End.DT6	4.14	4.14	Endpoint with decapsulation and IPv6 table lookup
End.DT4	-	4.14	Endpoint with decapsulation and IPv4 table lookup
End.B6	4.14	4.14	Endpoint bound to an SRv6 policy
End.B6.Encaps	4.14	4.14	Endpoint bound to an SRv6 encapsulation Policy
End.BM	-	4.14	Endpoint bound to an SR-MPLS Policy
End.S	-	4.14	Endpoint in search of a target in table T

Use Linux Kernel v4.14+

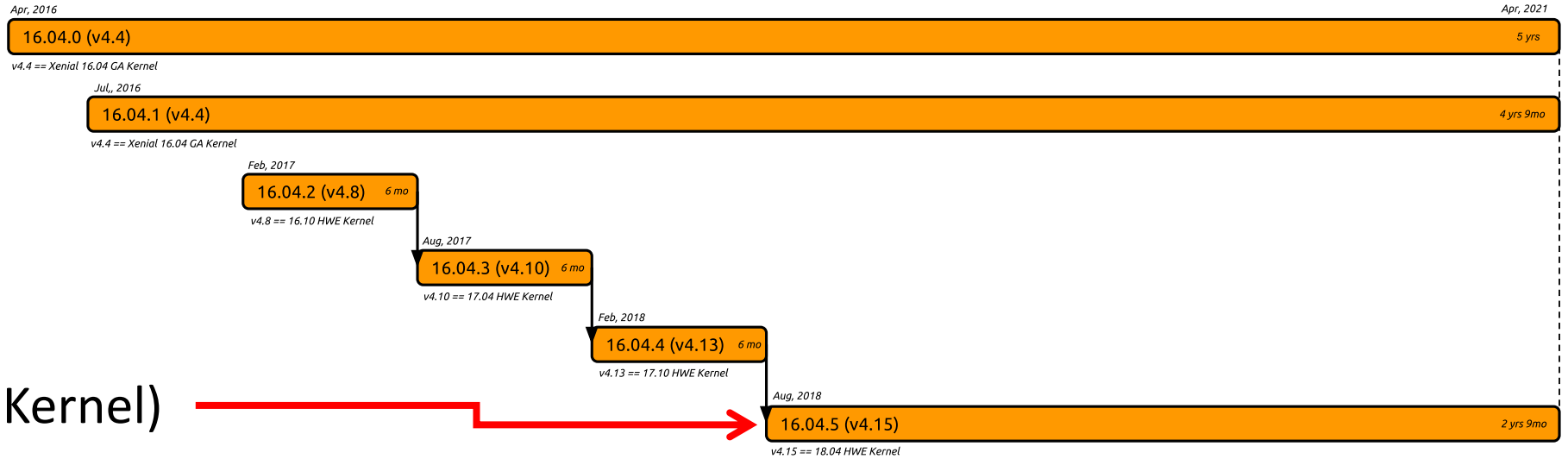
End.BPF : v4.18

Internet-Drafts:

- https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-network-programming/
- https://datatracker.ietf.org/doc/draft-xuclad-spring-sr-service-programming/

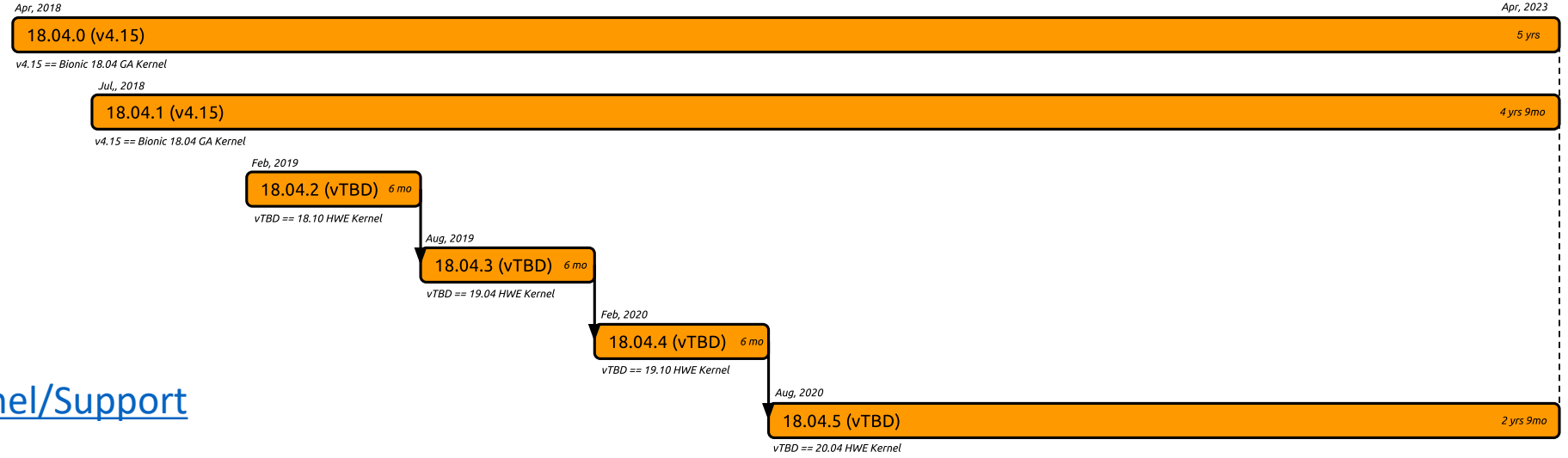
SRv6 Functions ... Implementation Status

16.04.x Ubuntu Kernel Support Schedule



Linux v4.14+
Ubuntu 16.04.05 (HWE Kernel)
Ubuntu 18.04

18.04.x Ubuntu Kernel Support Schedule



https://wiki.ubuntu.com/Kernel/Support

SRv6 configuration command

Comparison between “VPP” and “Linux iproute2”

TOYOTA INFOTECH

Transit node on Linux

```
ip -6 route add <prefix> encap seg6 mode <encapmode>
      segs <segments> [hmac <keyid>] (dev <device> | via <nexthop>)
```

examples

```
ip -6 route add fc00:b::10/128 encap seg6 mode inline
          segs fc00:3::11,fc00:3::12,fc00:3::13 via fc00:a::a

ip -6 route add fc00:b::10/128 encap seg6 mode encap
          segs fc00:3::11,fc00:3::12,fc00:3::13 via fc00:a::a
```

Reference: http://www.segment-routing.org/index.php/Implementation/Configuration

TOYOTA InfoTech

End segments (functions) on Linux

```
ip -6 route add <segment> encap seg6local action <action> <params>
                    (dev <device> | via <nexthop>) [table localsid]
```

examples

```
ip -6 route add fc00::1/128 encap seg6local
      action End                                    via 2001:db8::1
      action End.X          nh6 fc00::1:1           via 2001:db8::1
      action End.T          table 100               via 2001:db8::1
      action End.DX2        oif lxcbr0              via 2001:db8::1
      action End.DX6        nh6 fc00::1:1           via 2001:db8::1
      action End.DX4        nh4 10.0.3.254          via 2001:db8::1
      action End.DT6        table 100               via 2001:db8::1
      action End.B6         srh segs beaf::1,beaf::2 via 2001:db8::1
      action End.B6.Encaps  srh segs beaf::1,beaf::2 via 2001:db8::1
```

Reference: http://www.segment-routing.org/index.php/Implementation/AdvancedConf

TOYOTA INFOTECH

Transit node on VPP

```
sr policy add bsid 2001::1 next A1:: next B1:: next C1:: (encap)
sr steer l3 <address> via bsid <bsid>
sr steer l2 <interface> via bsid <bsid>
```

examples

```
sr policy add bsid cafe::1 next A1:: next B1:: next C1:: encap

sr steer l3 2001::/64 via bsid cafe::1
sr steer l3 10.0.0.0/16 via bsid cafe::1
sr steer l2 TenGE0/1/0 via bsid cafe::1
```

bsid = binding SID

Reference: https://docs.fd.io/vpp/19.01/srv6_doc.html

TOYOTA InfoTech

End segments (functions) on VPP

```
sr localsid address <sid> behavior <function> <parameters...>
```

examples

```
sr localsid address XX::YY behavior end
sr localsid address XX::YY behavior end.x GE0/1/0 2001::a
sr localsid address XX::YY behavior end.dx6 GE0/1/0 2001::a
sr localsid address XX::YY behavior end.dx4 GE0/1/0 10.0.0.1
sr localsid address XX::YY behavior end.dx2 GigabitE0/11/0
sr localsid address XX::YY behavior end.dt6 5
sr localsid address XX::YY behavior end.dt6 5
```

Reference: https://docs.fd.io/vpp/19.01/srv6_doc.html

TOYOTA INFOTECH

Comparison between “VPP” and “Linux iproute2”

VPP

```
sr policy add bsid 2001::1 next A1:: next B1:: next C1:: (encap)
sr steer l3 <address> via bsid <bsid>
sr steer l2 <interface> via bsid <bsid>
```

Linux iproute2

```
ip -6 route add <prefix> encap seg6 mode <encapmode>
      segs <segments> [hmac <keyid>] (dev <device> | via <nexthop>)
```

VPP：binding SID を定義し、そこにパケットを向ける（steer）
Linux：bsid を用いず、直接ルーティングエントリとして定義

TOYOTA INFOTECH

Comparison between “VPP” and “Linux iproute2”

VPP

```
sr policy add bsid 2001::1 next A1:: next B1:: next C1:: (encap)
sr steer l3 <address> via bsid <bsid>
sr steer l2 <interface> via bsid <bsid>
```

Linux iproute2

```
ip -6 route add <prefix> encap seg6 mode <encapmode>
      segs <segments> [hmac <keyid>] (dev <device> | via <nexthop>)
```

VPP : Define binding SID and steer packet to it

Linux : Directly set as routing entry without using bsid

Where and How SRv6 functions are implemented?

TOYOTA InfoTech

Linux Kernel, iproute2 and End.BPF

Where and How SRv6 functions are implemented?

TOYOTA INFOTECH

“netlink” is used to configure / show SRv6 rules

```
RTNetlink
// rtattr_type_t
    RTA_DST
    RTA_OIF
    RTA_ENCAP_TYPE (0x15)
    RTA_ENCAP (0x16)
```

```
RTA_ENCAP_TYPE (0x15)
lwtunnel_encap_types {
    LWTUNNEL_ENCAP_MPLS
    LWTUNNEL_ENCAP_IP
    LWTUNNEL_ENCAP_ILA
    LWTUNNEL_ENCAP_IP6
    LWTUNNEL_ENCAP_SEG6 (5)
    LWTUNNE_ENCAP_BPF
    LWTUNNEL_ENCAP_SEG6_LOCAL (7)
}
```

TYPE = SEG6

TYPE = SEG6_LOCAL

```
RTA_ENCAP (0x16)
    SEG6_IPTUNNEL_SRH
    encap mode {
        SEG6_IPTUN_MODE_INLINE
        SEG6_IPTUN_MODE_ENCAP
        SEG6_IPTUN_MODE_L2ENCAP
    }
    SRH { ... }
```

```
RTA_ENCAP (0x16)
// seg6local types
    SEG6_LOCAL_ACTION
    SEG6_LOCAL_SRH,
    SEG6_LOCAL_TABLE,
    SEG6_LOCAL_NH4,
    SEG6_LOCAL_NH6,
    SEG6_LOCAL_IIF,
    SEG6_LOCAL_OIF,
```

```
SEG6_LOCAL_ACTION (0x01)
seg6local action types {
    SEG6_LOCAL_ACTION_END           = 1
    SEG6_LOCAL_ACTION_END_X         = 2
    SEG6_LOCAL_ACTION_END_T         = 3
    SEG6_LOCAL_ACTION_END_DX2       = 4
    SEG6_LOCAL_ACTION_END_DX6       = 5
    SEG6_LOCAL_ACTION_END_DX4       = 6
    SEG6_LOCAL_ACTION_END_DT6       = 7
    SEG6_LOCAL_ACTION_END_DT4       = 8
    SEG6_LOCAL_ACTION_END_B6        = 9
    SEG6_LOCAL_ACTION_END_B6_ENCAP  = 10
    SEG6_LOCAL_ACTION_END_BM        = 11
    SEG6_LOCAL_ACTION_END_S         = 12
    SEG6_LOCAL_ACTION_END_AS        = 13
    SEG6_LOCAL_ACTION_END_AM        = 14
}
```

- Example when setting route
- SEG6 = Transit node
- SEG6LOCAL = End node (Local Segment)
- Select one value for items in { }
- Select multiple without { }

TOYOTA INFOTECH

netlink message example (SEG6_LOCAL)

```
Linux rtnetlink (route netlink) protocol
> Netlink message header (type: Add network route)
  Address family: AF_INET6 (10)
  Length of destination: 128
  Length of source: 0
  TOS filter: 0x00
  Routing table ID: 254
  Routing protocol: boot (0x03)
  Route origin: global route (0x00)
  Route type: Gateway or direct route (0x01)
  Route flags: 0x00000000
> Attribute: Route destination address
v Attribute: RTA_ENCAP
    Len: 32
  v Type: 0x0016, RTA_ENCAP (22)
      0... .... .... .... = Nested: 0
      .0.. .... .... .... = Network byte order: 0
      Attribute type: RTA_ENCAP (22)
    Data: 080001000200000014000500fc000000000000000000000000...
v Attribute: RTA_ENCAP_TYPE
    Len: 6
  > Type: 0x0015, RTA_ENCAP_TYPE (21)
    Data: 0700                                  -> ENCAP_SEG6_LOCAL (7)
> Attribute: Output interface index: 3
```

```
0000  08 00 01 00 02 00 00 00 14 00 05 00 fc 00 00 00
0010  00 00 00 00 00 00 00 00 00 00 01 00 01
```

08 00 01 00 | len: 8bytes, type: SEG6_LOCAL_ACTION (0x01)
02 00 00 00 | data: SEG6_LOCAL_ACTION_END_X (0x02)
14 00 05 00 | len: 20bytes, type: SEG6_LOCAL_NH6 (0x05)
fc 00 00 00 00 00 00 00 00 00 00 00 00 01 00 01 | data: IPv6 addr

netlink メッセージの確認方法

```
modprobe nlmon
ip link add nlmon0 type nlmon
ip link set nlmon0 up
tshark -i nlmon0 -V
```

TOYOTA InfoTech

Linux Source Code (where to look)

Linux Kernel Source Code

- include/uapi/linux/
  - rtnetlink.h
  - lwtunnel.h
  - seg6_genl.h
  - seg6.h
  - seg6_hmac.h
  - seg6_iptunnel.h
  - seg6_local.h
- net/core/
  - lwtunnel.c
- net/ipv6/
  - seg6.c
  - seg6_hmac.c
  - seg6_iptunnel.c
  - seg6_local.c

iproute2

- ip/
  - ipseg6.c
  - iproute_lwtunnel.h
  - iproute_lwtunnel.c

Linux: https://github.com/torvalds/linux
iproute2 git://git.kernel.org/pub/scm/network/iproute2/iproute2.git

TOYOTA INFOTECH

End.BPF action (Linux eBPF)

SEG6_LOCAL_BPF : ~/linux/net/ipv6/seg6_local.c

```
static int input_action_end_bpf(struct sk_buff *skb,
                                struct seg6_local_lwt *slwt)
{
        struct seg6_bpf_srh_state *srh_state =
                this_cpu_ptr(&seg6_bpf_srh_states);
        struct seg6_bpf_srh_state local_srh_state;
        struct ipv6_sr_hdr *srh;
        int srhoff = 0;
        int ret;

        srh = get_and_validate_srh(skb);
        if (!srh)
                goto drop;
        advance_nextseg(srh, &ipv6_hdr(skb)->daddr);

        /* preempt_disable is needed to protect the per-CPU buffer srh_state,
         * which is also accessed by the bpf_lwt_seg6_* helpers
         */
        preempt_disable();
        srh_state->hdrlen = srh->hdrlen << 3;
        srh_state->valid = 1;

        rcu_read_lock();
        bpf_compute_data_pointers(skb);
        ret = bpf_prog_run_save_cb(slwt->bpf.prog, skb);
        rcu_read_unlock();

        local_srh_state = *srh_state;
        preempt_enable();

        switch (ret) {
        case BPF_OK:
        case BPF_REDIRECT:
                break;
        case BPF_DROP:
                goto drop;
```

How to write eBPF code for End.BPF

https://segment-routing.org/index.php/Implementation/BPF

Implementation / Programming network actions with BPF

Edit

As of Linux 4.18, custom SRv6 network functions can be implemented in eBPF and installed in the kernel. eBPF (for *extended Berkeley Packet Filter*), is a general-purpose 64 bits RISC-like virtual machine included in Linux. It provides a programmable interface to adapt kernel components at run-time to user-specific behaviours. eBPF programs are written in C and compiled to eBPF bytecode using LLVM, and then can be attached to predetermined hooks in the kernel. The eBPF program is then executed for each packet going through the datapath associated to its hook. The program can read and, for some hooks, modify the packet.

A BPF hook has been introduced in the seg6local infrastructure as a regular seg6local action, called *End.BPF*. Using *End.BPF*, network operators can implement their own SRv6 functions. Moreover, SRv6-specific BPF helpers are provided and allow *End.BPF* functions to leverage advanced SRv6 features such as executing basic SRv6 actions (End.X, End.T, ...) or adding TLVs.

This page describes the intrinsic properties of *End.BPF*, the explications described here assume that the reader is familiar with BPF. For more information regarding all the capabilities of BPF, see the BPF reference guide.

Overview of *End.BPF*

Every instance of an *End.BPF* action is bound to a given eBPF program. As its name implies, *End.BPF* behaves as an endpoint, i.e. it advances the SRH (*Segment Routing Header*) to the next segment, and subsequently executes the associated eBPF code. Only IPv6 packets containing a SRH with `Segments Left > 0` are accepted. `Segments Left` is decremented and the next segment to be processed is copied to the IPv6 `Destination Address` field by the action before the eBPF function is executed.

End.BPF programs must return one of the three following return values. These values indicate to *End.BPF* the final step of the SRv6 processing that must be applied on the packet:

- `BPF_OK` : a regular FIB lookup must be performed on the next segment, the packet will be forwarded on the egress interface returned by the lookup.
- `BPF_DROP` : the packet must be dropped.
- `BPF_REDIRECT` : the default endpoint lookup must not be performed, and the packet must be forwarded to the destination already set in the packet metadata. To be used altogether with the `bpf_lwt_seg6_action` helper.

Furthermore, this action has been designed with two key principles in mind:

- BPF code cannot compromise the stability of the kernel.
- BPF code should be able to leverage all the functionalities of the SRv6 data plane.

As a consequence of the first principle, direct write access by BPF code is prohibited. Only specific fields of the outermost SRH can be modified using the `bpf_lwt_seg6_store_bytes` helper. To enable *End.BPF* functions to leverage SRv6 features, two others seg6 helpers are available. Mainly, `bpf_lwt_seg6_action` allows to use classic SRv6 actions (End.X, End.T, End.B6 and End.B6.Encaps) inside *End.BPF*. The specifications of these helpers are given below.

TOYOTA InfoTech

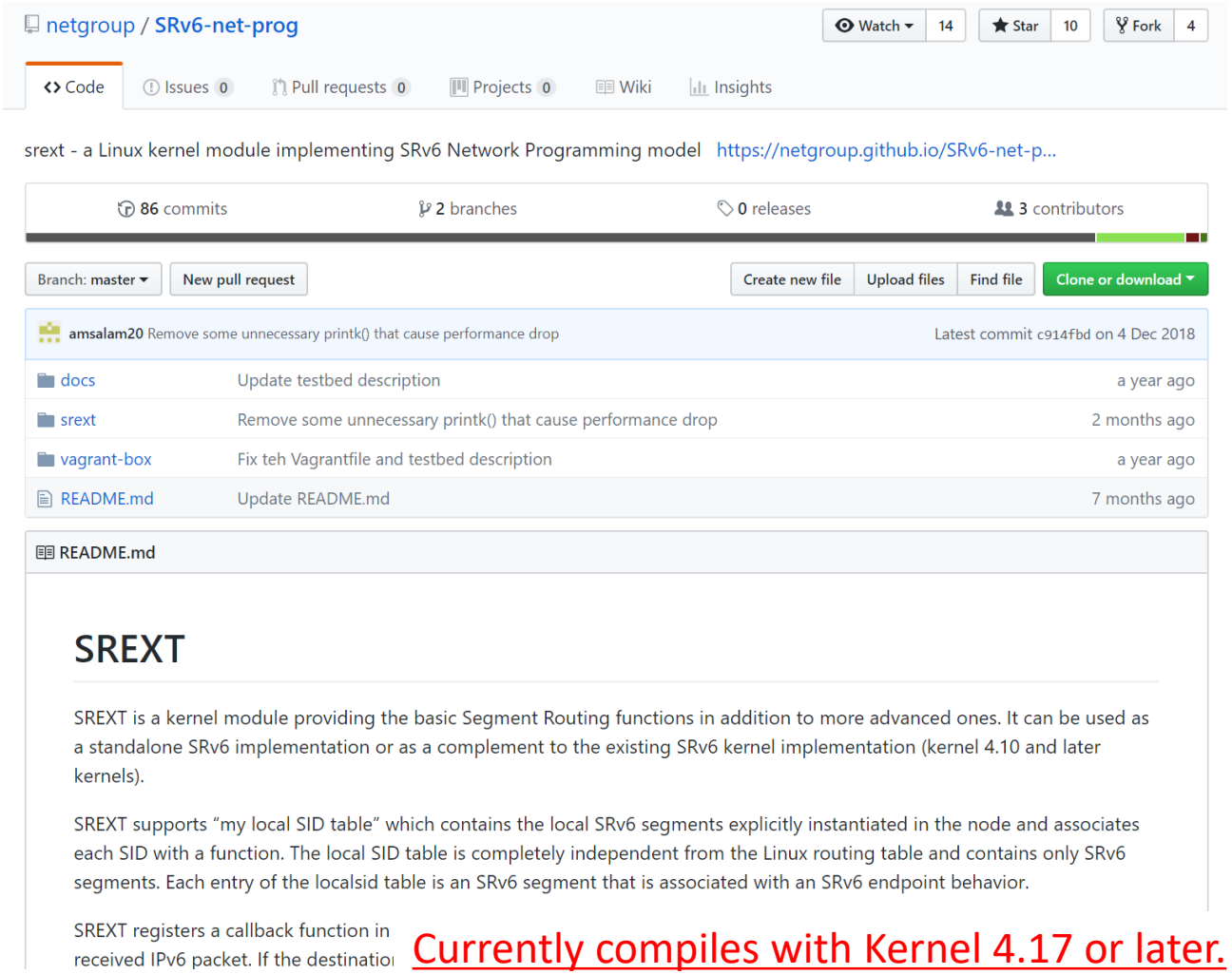
Linux SREX

Where and How SRv6 functions are implemented?

TOYOTA INFOTECH

SREXT Implementation

https://github.com/netgroup/SRv6-net-prog



Currently compiles with Kernel 4.17 or later.

~/SRv6-net-prog/srext/kernel/sr_hook.c

```
/**
 * add_end()
 * Adds a localsid with End behavior to my localsid table
 * [CLI]... "srconf localsid add SID end"
 * @sid: SRv6 SID
 * @behavior: SRv6 behavior
 */

int add_end(const char *sid, const int behavior)
{
    int ret = 0;
    u32 hash_key;
    struct in6_addr bsid;
    struct sid6_info *s6;
    char * err_msg = "add_end - ";
```

```
/**
 * add_end_x()
 * Adds localsid with End.X or End.DX6 behavior to my localsid table
 * [CLI]..."srconf localsid add SID {end.x | end.dx6} NEXTHOP6 TARGETIF"
 * @sid: SRv6 SID
 * @behavior: SRv6 behavior
 * @nh_ip6: IPv6 address of next hop
 * @mac: MAC address of next hop
 * @oif: target interface
 */

int add_end_x(const char *sid, const int behavior, const char *nh_ip6, const
unsigned char *mac,
              const char *oif)
{
```

TOYOTA InfoTech

SRv6 in VPP

Where and How SRv6 functions are implemented?

TOYOTA INFOTECH

Extending SRv6 functions

https://docs.fd.io/vpp/19.04/srv6_plugin_doc.html

Sample SRv6 LocalSID documentation

Introduction

This plugin is an example of how an user can create a new SRv6 LocalSID behavior by using VPP plugins with the appropiate existing SR code.

This **example** plugin registers a new localsid behavior, with cli keyword 'new_srv6_l recival of a packet, this plugin will enforce the next IP6 lookup in the specific fib-tabl the fib_table n+1 (since for the shake of the example we increment the fib-table.)

Notice that the plugin only 'defines' a new SRv6 LocalSID behavior, but the existing S LocalSIDs. Notice that there are callback functions such that when you create or rem parameters through the functions in this plugin.

Variables to watch for

- srv6_localsid_name: This variable is the name (used as a unique key) identifyi
- keyword_str: This is the CLI keyword to be used for the plugin. In this exampl behavior new_srv6_localsid <parameters>)
- def_str: This is a definition of this SR behavior. This is printed when you do 'sh
- params_str: This is a definition of the parameters of this localsid. This is printe

Functions to watch for

- srv6_localsid_creation_fn: This function will be called every time a new SR Loc plugin.
- srv6_localsid_removal_fn: This function will be called every time a new SR Loc This function tends to be used for freeing up all the memory created in the pre
- format_srv6_localsid_sample: This function prints nicely the parameters of eve
- unformat_srv6_localsid_sample: This function parses the CLI command when parses all the parameters and ensures that the parameters are correct.
- format_srv6_localsid_sample_dpo: This function formats the 'show ip6 fib' me behavior.

Graph node

The current graph node uses the function 'end_srh_processing' to do the Segment R cleanup of a Segment Routing header (as per the SRv6 behavior specs). This functio /src/vnet/srv6/sr_localsid.c In case that by some other reason you want to do decapsulation, or SRH clean_up you can use the functions 'end_decaps_srh_processing' or 'end_psp_srh_processing' respectively.

```
$ git clone https://gerrit.fd.io/r/vpp

~/vpp/src/examples/srv6-sample-localsid$ ls
node.c                      srv6_localsid_sample.h
srv6_localsid_sample.c      srv6_sample_localsid_doc.md
```

```
static uword
srv6_gtp_fn (vlib_main_t * vm, vlib_node_runtime_t * node, vlib_frame_t * frame)
{
  u32 n_left_from, * from, * to_next;
  u32 next_index;

  ip6_sr_main_t * sm = &sr_main;

  from = vlib_frame_vector_args (frame
  n_left_from = frame->n_vectors;
  next_index = node->cached_next_index
  u32 thread_index = vlib_get_thread_i

  while (n_left_from > 0)
  {
    u32 n_left_to_next;

    vlib_get_next_frame (vm, node, nex
                         to_next, n_left_to_
```

```
VLIB_REGISTER_NODE (srv6_gtp_node) = {
  .function = srv6_gtp_fn,
  .name = "srv6-gtp",
  .vector_size = sizeof (u32),
  .format_trace = format_srv6_gtp_trace,
  .type = VLIB_NODE_TYPE_INTERNAL,
  .n_errors = SRV6_LOCALSID_N_COUNTERS,
  .error_strings = srv6_localsid_counter_strings,
  .n_next_nodes = 2,
  .next_nodes = {
      [SRV6_GTP_NEXT_IP4LOOKUP] = "ip4-lookup",
      [SRV6_GTP_NEXT_ERROR] = "error-drop",
  },
```

TOYOTA InfoTech

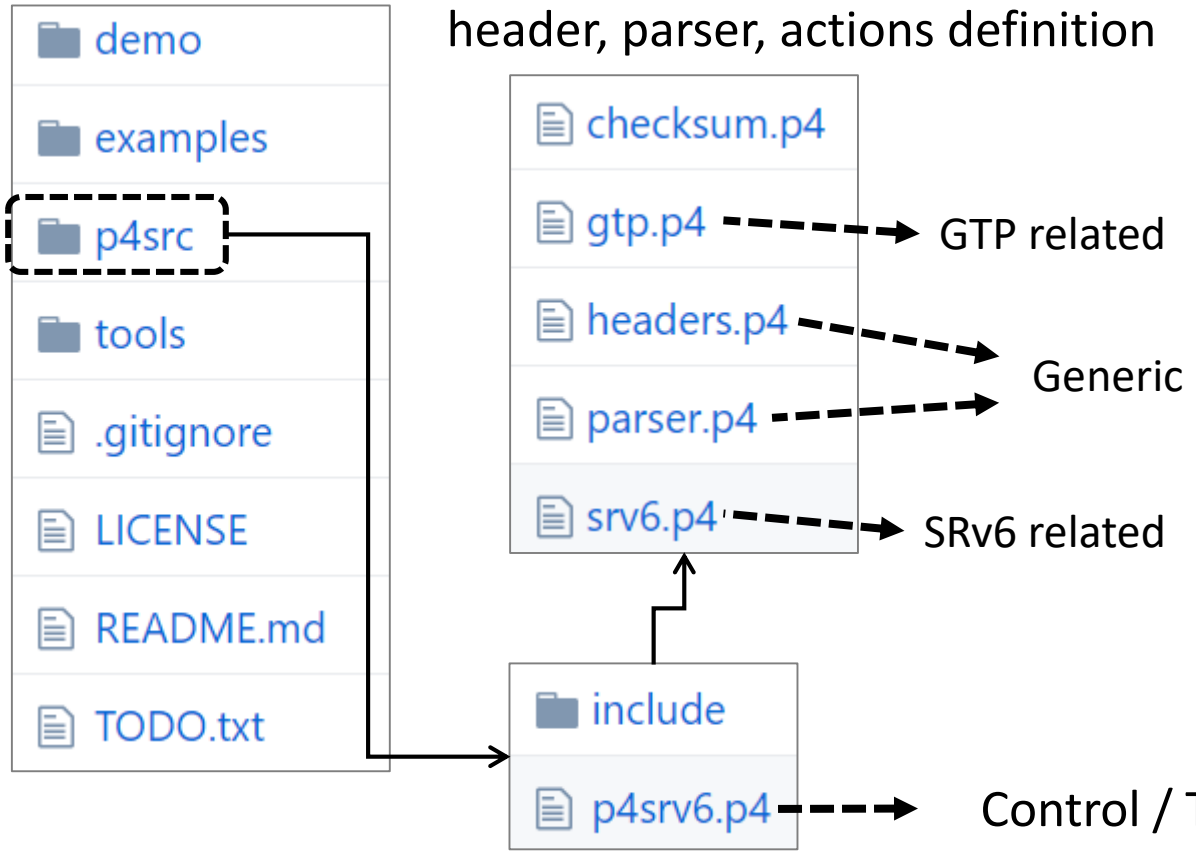
SRv6 on P4

Where and How SRv6 functions are implemented?

TOYOTA InfoTech

P4SRv6 (P4-14)

https://github.com/ebiken/p4srv6




```
table srv6_localsid {
    reads {
        ipv6.dstAddr: exact;
    }
    actions {
        srv6_T_Insert1; srv6_T_Insert2; srv6_T_Insert3;
        srv6_T_Encaps2; srv6_T_Encaps1; srv6_T_Encaps3;
        srv6_T_Encaps_Red2; srv6_T_Encaps_Red3;
        srv6_End0; srv6_End1;
        srv6_End_DT6;
        srv6_End_M_GTP6_D2; srv6_End_M_GTP6_D3;
        srv6_End_M_GTP6_E;
    }
}
```

Wireshark

Where and How SRv6 functions are implemented?

TOYOTA InfoTech

Wireshark

https://github.com/wireshark/wireshark/blob/master/epan/dissectors/packet-ipv6.c#L1066

```
1066  /* Segment Routing Header (Type 4) */
1067  /* draft-ietf-6man-segment-routing-header-05 */
1068  static int
1069  dissect_routing6_srh(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree, void *data)
1070  {
1071      struct ws_rthdr *rt = (struct ws_rthdr *)data;
1072      proto_item *ti;
1073      int offset = 0;
1074      gint offlim, offstart;
1075      gint idx;
1076      gint srh_first_seg, srh_addr_count;
1077      const ws_in6_addr *addr;
1078      proto_tree *rthdr_srh_addr_tree;
1079      static const int *srh_flags[] = {
1080          &hf_ipv6_routing_srh_flag_unused1,
1081          &hf_ipv6_routing_srh_flag_p,
```

Conclusion | まとめ

Conclusion

TOYOTA INFOTECH

複数のオープンソース実装でテスト可能

→ Linux Network Stack | Kernel Module
VPP (FD.io)

ハードウェア（ASIC）上でもクローズドながら
オープンなプログラミング言語（P4）で実装済み

→ ASIC上で動作する
独自Functionをプログラム可能

成熟度はまだこれから
実装されてないSRv6 Functionも多数
コントロールプレーン（BGP/ISIS/OSPF）の不在

→ まだまだ貢献可能な隙間が沢山☺

SRv6の実装・実験に興味ある方はお声がけを！！
（特にオープンソースで）

Conclusion

TOYOTA InfoTech

Multiple Open Source implementations are available to start testing SRv6 functionality.

→ Linux Network Stack | Kernel Module
VPP (FD.io)

Hardware is also available on closed platform (Barefoot Tofino) but in open language (P4)

→ We can start writing new SRv6 functions on hardware.

Still in early stage of implementation
Not all functions are implemented
No control plane (BGP/ISIS/OSPF etc.)

→ Still large space left to contribute ☺

Talk to me if you are interested in testing / implementing SRv6 !!
(Especially as an Open Source Software)

！！続きは “ENOG55@燕三条” で！！

http://enog.jp/archives/2014

参加登録

お知らせ

ENOG55 Meeting 開催のお知らせ

by masakazu • 2018年12月13日 • 0 Comments

開催概要

日時

2019/2/22(金) 14:00～19:00(予定)

会場

味覚天国たまや
（新潟県三条市須頃1丁目67-1）

- http://www.week.co.jp/gourmet/101726/

勉強会

- “SRv6 入門”
  - – 浅間 正和 (有限会社銀座堂)
- “Control Plane的何か（仮）”
  - – 河野 美也 (Cisco Systems)
- “（仮）SRv6の標準化やPoC”
  - – 松嶋 聡 (ソフトバンク)
- “（仮）SRv6データプレーン実装とモバイルへの適用”
  - – 海老澤 健太郎 (TOYOTA ITC)
- “LINEにおけるデータセンターでのSRv6ユースケース”
  - – 土屋 俊貴 (LINE株式会社)
- “IP anycast + SRv6によるNetwork APIの提供”
  - – 宮坂 拓也 (KDDI総合研究所)
- “Trellis (SR based DC Fabric Solution)(仮)”
  - – 調整中 (SK Telecom)
- “SKT’s R&D on Telco Networks(仮)”
  - – 調整中 (SK Telecom)
- “Kamuee SRv6対応 調査と報告と実装ステータス”
  - – 城倉 弘樹 (NTTコミュニケーションズ)
- “Huawei SRv6 update”
  - – 高嶋 隆一 (Huawei Technologies Japan K.K.)

SRv6の実装例
詳しく話します

※今回は SRv6 に関する発表のみのプログラムとなります。