Next Data Center Networking with SRv6 - Data plane -

Toshiki Tsuchiya, LINE Corporation

2019/07/26 JANOG44 Meeting

Agenda

- LINEのサービスとネットワーク
- データプレーン
- コントロールプレーン

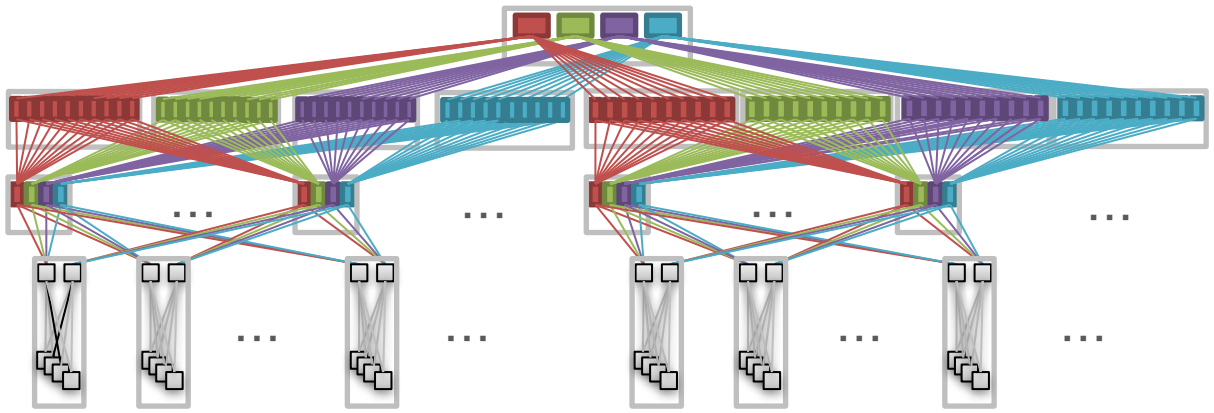
LINEのサービスとネットワーク

LINEのサービスとネットワーク

現状のネットワークと課題

Full L3 CLOS Network*

- シングルテナントネットワーク
- LINEのメッセンジャーやファミリーサービスなどが稼動



特定サービス専用のネットワーク

- サービス固有の要件を持つサービスが稼動
- サービスごとにネットワークを構築

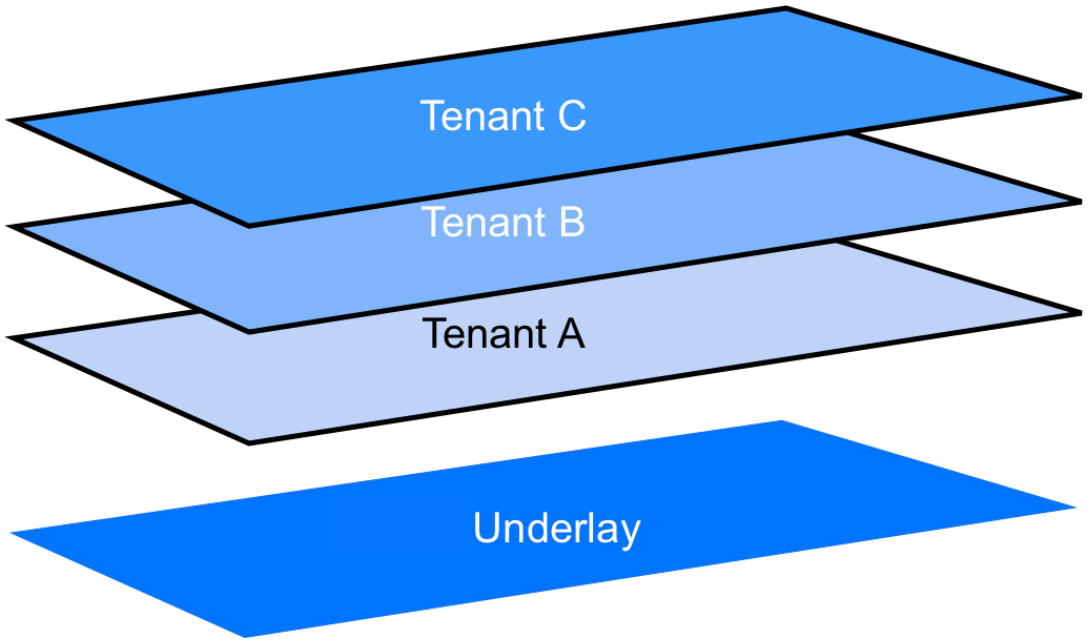
その他 Fintech Business

アンダーレイネットワークの断片化
設計・構築にかかる時間の増加
運用コストの増加

* JANOG43 Meeting: LINEのネットワークをゼロから再設計した話
https://www.janog.gr.jp/meeting/janog43/program/line

解決策: マルチテナンシー

- アンダーレイネットワークを共通化し、運用負荷を軽減
- オーバーレイネットワークでサービス(テナント)ごとのポリシーを実現



柔軟にスケールするオーバーレイ
テナントで分離・独立したセキュリティ
将来的なサービスチェイニング

シンプルなL3のアンダーレイ

マルチテナンシー

VXLAN vs SRv6

VXLAN

Pros

- 運用実績や情報が多くある
- 多くのネットワーク機器で実装されている

Cons

- full-L3にした恩恵が損なわれるのでは...
- Service chainingを実現するために、
  追加のプロトコルが必要となる
  → プロトコルの多重化/複雑性が増す

IPv6 Segment Routing (SRv6)

Pros

- アンダーレイはIPv6 forwardingのみ
- Segment IDを駆使することで、
  SegregationやService chainingが実現できる
  → サービスの要件に合ったポリシーが実現できる

Cons

- データセンターネットワークでの導入事例・情報が無い
- ネットワーク機器での実装がほぼ無い

+ SRv6の将来性に期待

採用を決定

SRv6の概要

Segment ID (SID)

- 128bitのIPv6アドレス表記
- **Locator**: SRv6を処理するノード(ペアレントノード)へルーティングするための情報
- **Function**: ペアレントノードが実行する処理を示す情報

Segment Routing Header (SRH)

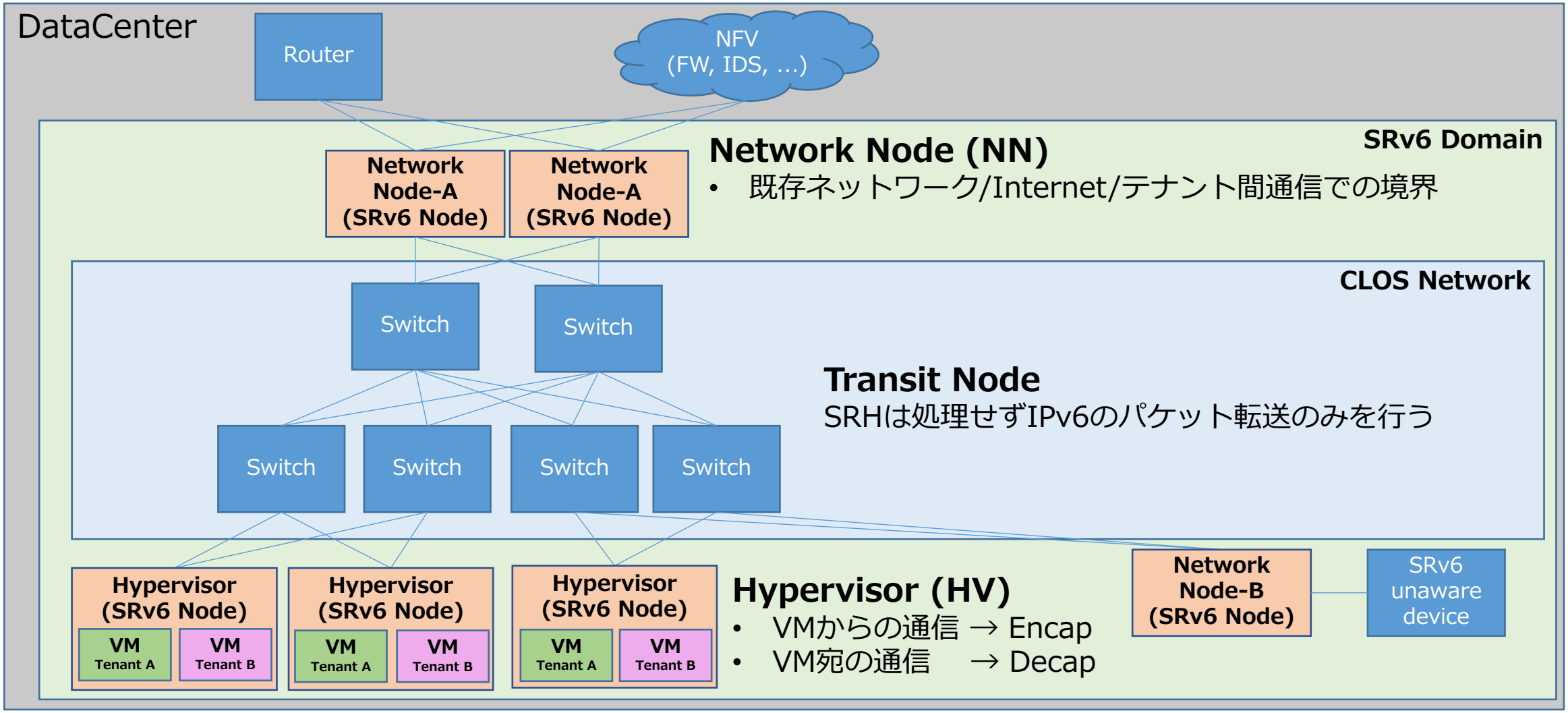
- IPv6の拡張ヘッダー
- 複数のSIDをまとめたSegment Listと、Segment Listの現在のポインタであるSegment Leftなどから構成

Well-known function (後述の説明で使うfunctionのみ)

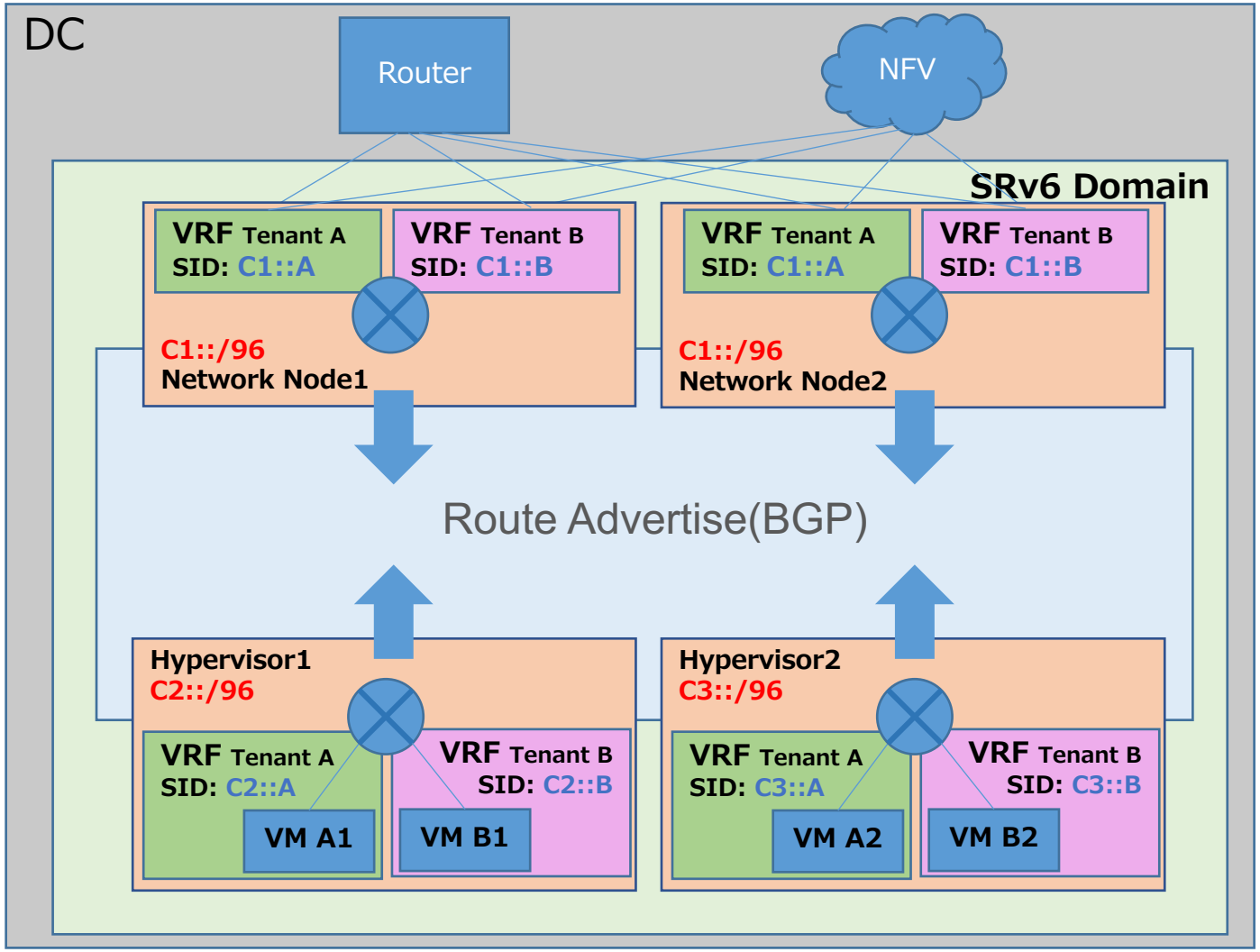
- **T.Encaps(Encap)**: パケットにIPv6ヘッダー、SRHを付与してカプセル化する
- **End.DX4(Decap)**: SRv6のパケットからIPv6ヘッダーとSRHを外して、指定されたNexthopに転送
- **End.DT4(Decap)**: SRv6のパケットからIPv6ヘッダーとSRHを外して、指定されたRoutingTableをルックアップして転送
  (Linux Kernelで実装されていないため今回の構成では使用していません。本来であればDX4よりDT4が使いたい)

SRv6 Data Center Network
データプレーン

データプレーン
全体構成



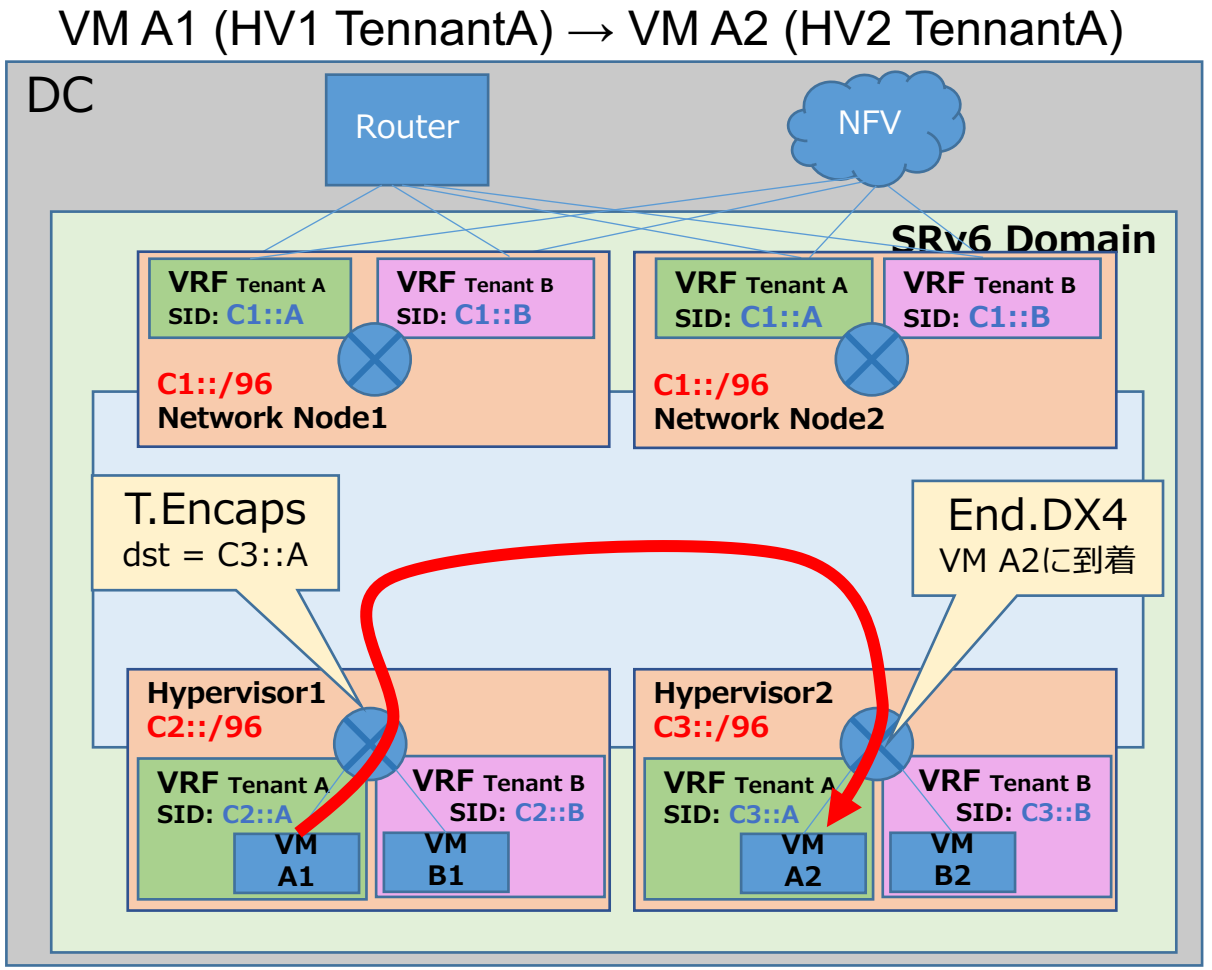
データプレーン
SID、Routing



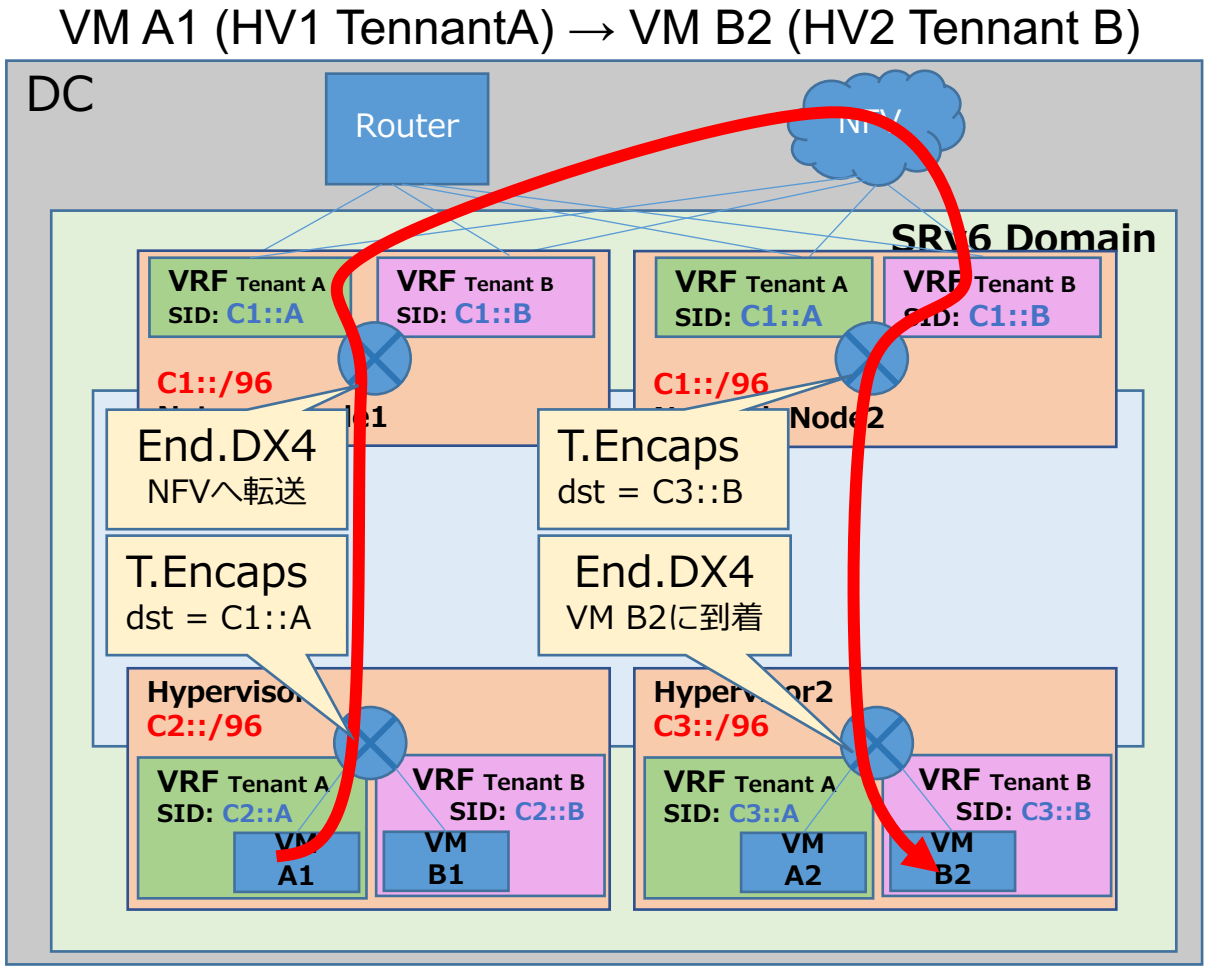
- NN, HVにはテナント単位で VRF (l3master device)を作成
- 各ノード(NN, HV)にはIPv6 アドレスを /96 単位で割り当て **(Locator)**
- テナントIDを**Function**としてSIDを設定

- 各ノードは Locatorとして割り当てられた IPv6の/96をBGPで経路広報

データプレーン
パケットフロー: テナント内通信



データプレーン
パケットフロー: テナント間通信



データプレーン

実際の設定 (Network Node A)

Encap設定

```
[NetworkNode-A]# ip route show table 12
10.122.12.113  encap seg6 mode encap segs 1 [ ■■■■■■■■■■■■■■■■4d8d:a9fe:108 ] dev vrf5c0594737b87 scope link
10.122.12.114  encap seg6 mode encap segs 1 [ ■■■■■■■■■■■■■■■■4d8e:a9fe:108 ] dev vrf5c0594737b87 scope link
10.122.12.115  encap seg6 mode encap segs 1 [ ■■■■■■■■■■■■■■■■4d8f:a9fe:108 ] dev vrf5c0594737b87 scope link
```

Segment List

宛先VMのIPv4アドレス

Locator(HVのアドレス) Function (テナントID)

Decap設定

```
[NetworkNode-A]# ip -6 route show table local
local ■■■■■■■■■■■■■■■■:4d87:a9fe:102 encap seg6local action End.DX4 nh4 169.254.1.2 dev vrf01b1db9dd10f metric 1024 pref medium
local ■■■■■■■■■■■■■■■■:4d87:a9fe:104 encap seg6local action End.DX4 nh4 169.254.1.4 dev vrf01b1db7f5d2b metric 1024 pref medium
local ■■■■■■■■■■■■■■■■:4d87:a9fe:108 encap seg6local action End.DX4 nh4 169.254.1.8 dev vrf5c0594737b87 metric 1024 pref medium
...
```

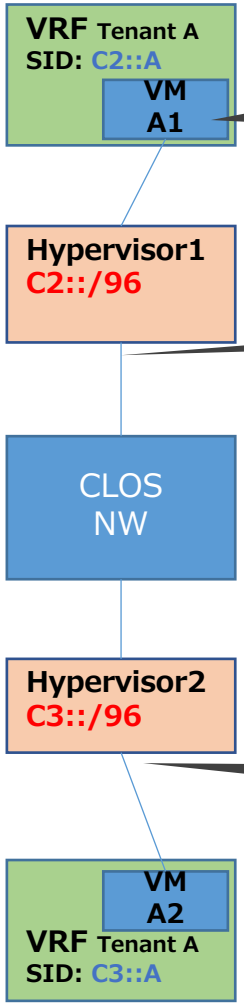
Locator(HVのアドレス) Function (テナントID)

VRFに設定したダミーのIPアドレス
(End.DX4でVRFのテーブルルックアップをするための小細工)

データプレーン

実際の挙動

IP: 10.122.12.36



IP: 10.122.12.35

```
[VM-A1]$ ping 10.122.12.35 -c 10
PING 10.122.12.35 (10.122.12.35) 56(84) bytes of data.
64 bytes from 10.122.12.35: icmp_seq=1 ttl=63 time=0.356 ms
64 bytes from 10.122.12.35: icmp_seq=2 ttl=63 time=0.461 ms
...
64 bytes from 10.122.12.35: icmp_seq=10 ttl=63 time=0.415 ms
--- 10.122.12.35 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
```

HV2: Decap

IPv6, SR headerが外れる

```
▶ Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▶ Ethernet II, Src: 7e:55:34:6d:37:3a (7e:55:34:6d:37:3a), Dst: fa:16:
▼ Internet Protocol Version 4, Src: 10.122.12.36, Dst: 10.122.12.35
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x0000 (0)
  ▶ Flags: 0x4000, Don't fragment
    Time to live: 63
    Protocol: ICMP (1)
    Header checksum: 0x0e6f [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.122.12.36
    Destination: 10.122.12.35
▶ Internet Control Message Protocol
```

HV1: Encap

IPv6, SR headerが挿入

```
▼ Internet Protocol Version 6, Src: [redacted]:4d8e:a9fe:102, Dst
    0110 .... = Version: 6
  ▶ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (D
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 108
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 63
    Source: [redacted]:4d8e:a9fe:102
    Destination: [redacted]:4d8f:a9fe:102
  ▼ Routing Header for IPv6 (Segment Routing)
      Next Header: IPIP (4)
      Length: 2
      [Length: 24 bytes]
      Type: Segment Routing (4)
      Segments Left: 0
      First segment: 0
    ▶ Flags: 0x00
      Reserved: 0000
      Address[0]: [redacted]:4d8f:a9fe:102
    ▼ [Segments in Traversal Order]
        Address[0]: [redacted]:4d8f:a9fe:102
▶ Internet Protocol Version 4, Src: 10.122.12.36, Dst: 10.122.12.35
▶ Internet Control Message Protocol
```

設計・構築で苦労した点

SRv6

- 情報が無い
- Best Practiceが無いのでどうやってSIDを設計すればいいのかわからない
- Linux KernelのSRv6を動かすのに一苦労

Linux networking

- VRF(l3mdev)の挙動
  - SRv6と組み合わせたときに必ずfragmentされる
    - バグ: Linux kernelへパッチ提出 (mainstreamに取り込まれています)
  - netfilterとの組み合わせが悪い
    - HV(Decap) → VMに通信した場合にFORWARDチェインでは無くOUTPUTチェインで引っかかる
    - VMからHV外への通信に関してはtapデバイスをout interfaceに設定したルールに反応しない
- NICのoffloadが効かず、Decap時の性能がかなり落ちる
  - メーカーと協力して調査を実施。修正パッチによって、性能が大幅改善

データプレーン: まとめ

達成できたこと

- SRv6を利用したオーバーレイネットワークを設計
  - SIDやRoutingの設計などを1から設計
- 実環境で構築し、マルチテナンシーが実現できていることを確認 → まもなく実サービス投入予定

今後の課題

- 性能問題
  - 現時点ではNetworkNode, HypervisorともにLinux KernelのSRv6実装を使用
    今後パフォーマンスが問題になることが目に見えている
    → 性能向上に向けて色々と検討中
    (SRv6対応スイッチ/NOS、 P4スイッチ、 Smart NIC、NIC HW offload、DPDK、XDP、...)
- ネットワーク構成
  - NFV(実際はFW)を通す部分はVRFとIP Routingで頑張っている状態のため、かなり複雑
    → SRv6でService chainingできる構成に作り変えていきたい