LINEにおけるデータセンターでのSRv6ユースケース

Toshiki Tsuchiya
LINE Corporation

ENOG 55 Meeting 2019/02/22

自己紹介

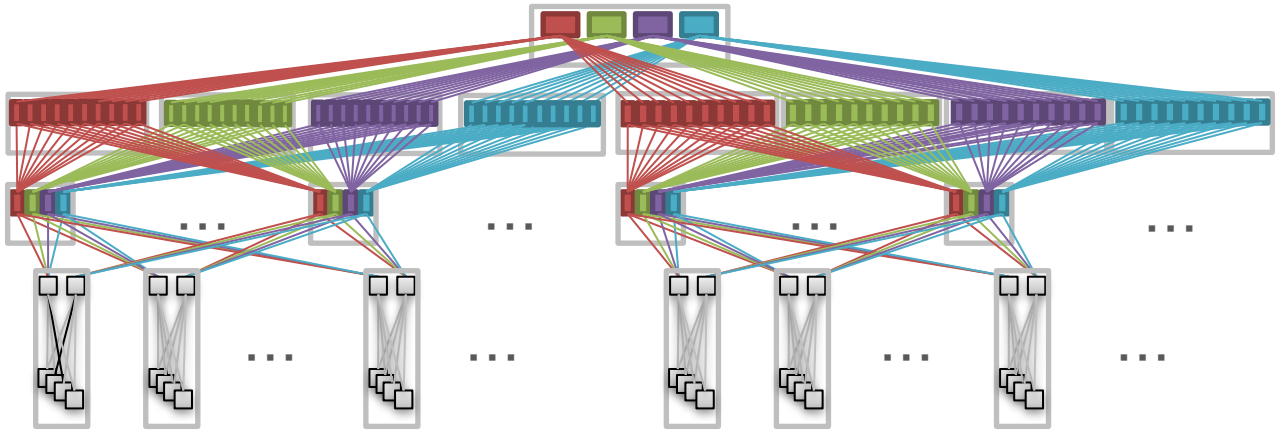
土屋 俊貴

ITサービスセンター ネットワーク室
サービスネットワークチーム (2017-04 ~)

データセンタネットワークの設計・運用
運用ツールの開発

最近のLINEのネットワークと課題

Full-L3 CLOS Network [1]
シングルテナントネットワーク
　　複数サービスが同居 (Messaging, LINE NEWS, Clova …)



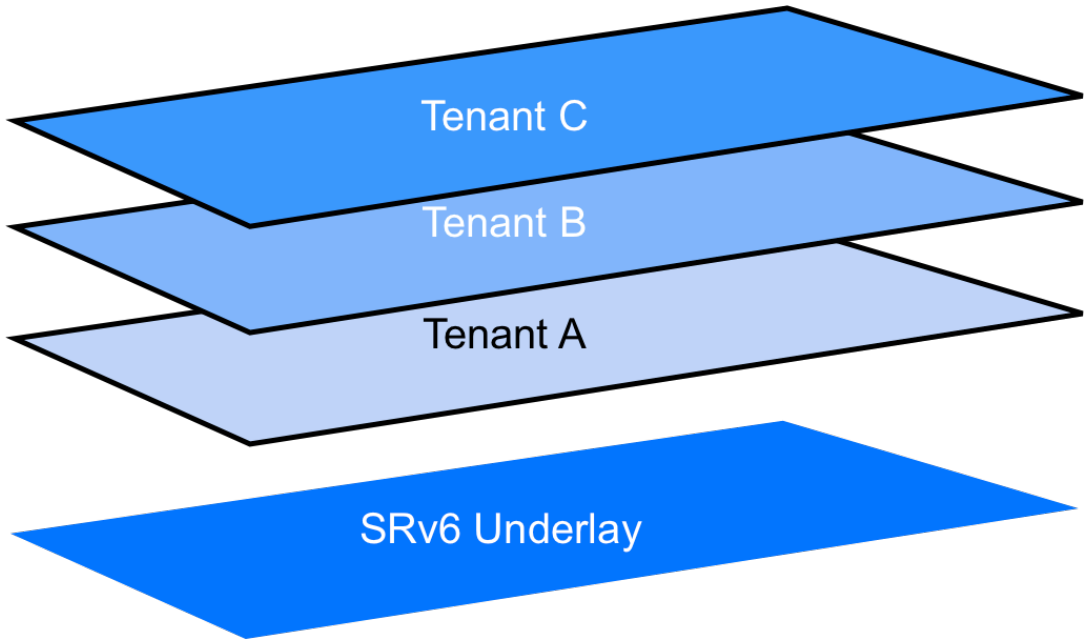
サービス別の専用ネットワーク
今までと異なる要件のサービスが増加
(Fintech系サービスなど)

サービスごとに専用のネットワークを構築

設計、構築に時間がかかる
運用コストの増加

[1] 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

マルチテナンシー



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

有効な実現方法として<u>SRv6</u>に注目

SRv6への期待

マルチテナンシーの実現はVXLANが使われることが多い

LINEのネットワークにおける懸念点

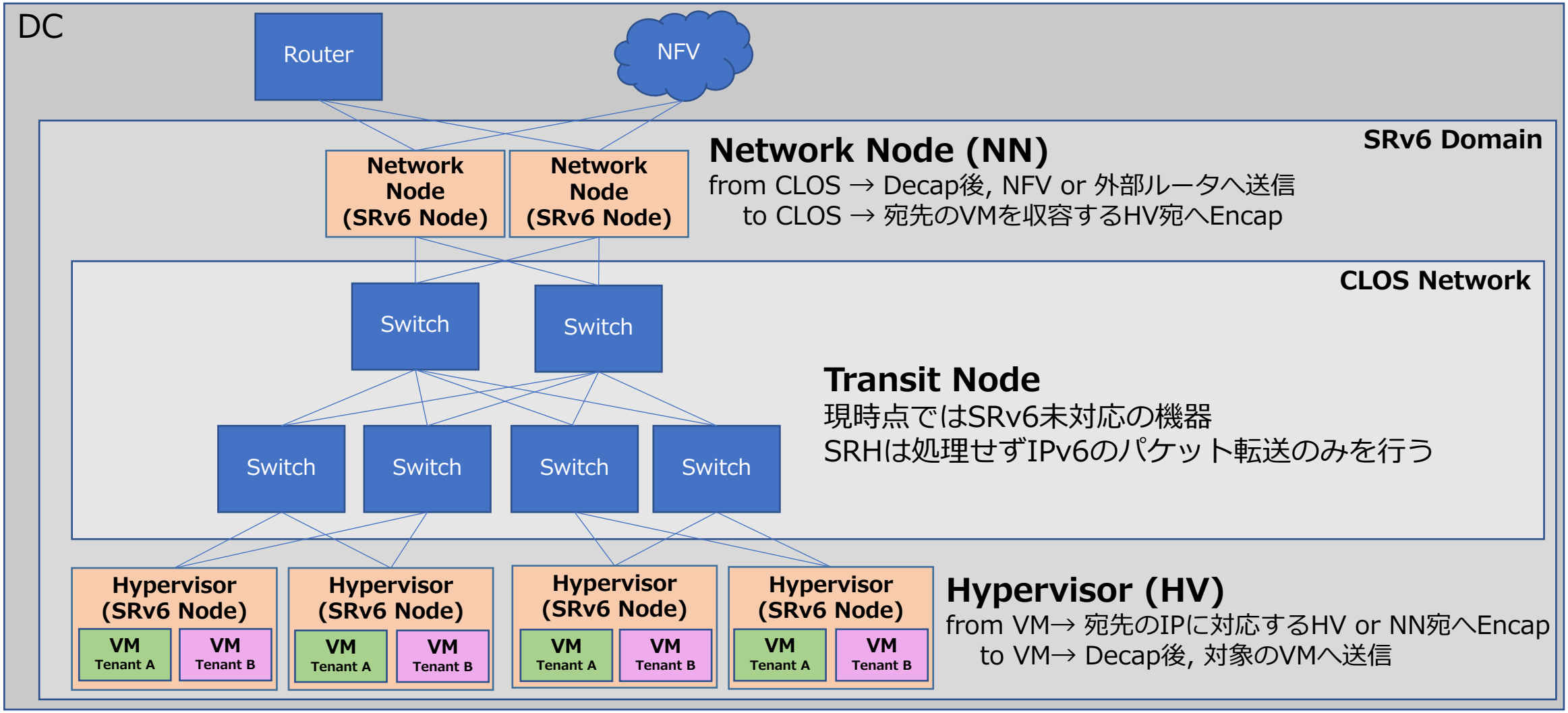
- Full-L3で構成した利点が損なわれるのでは…
- 既存プロトコルの多重化

SRv6であれば

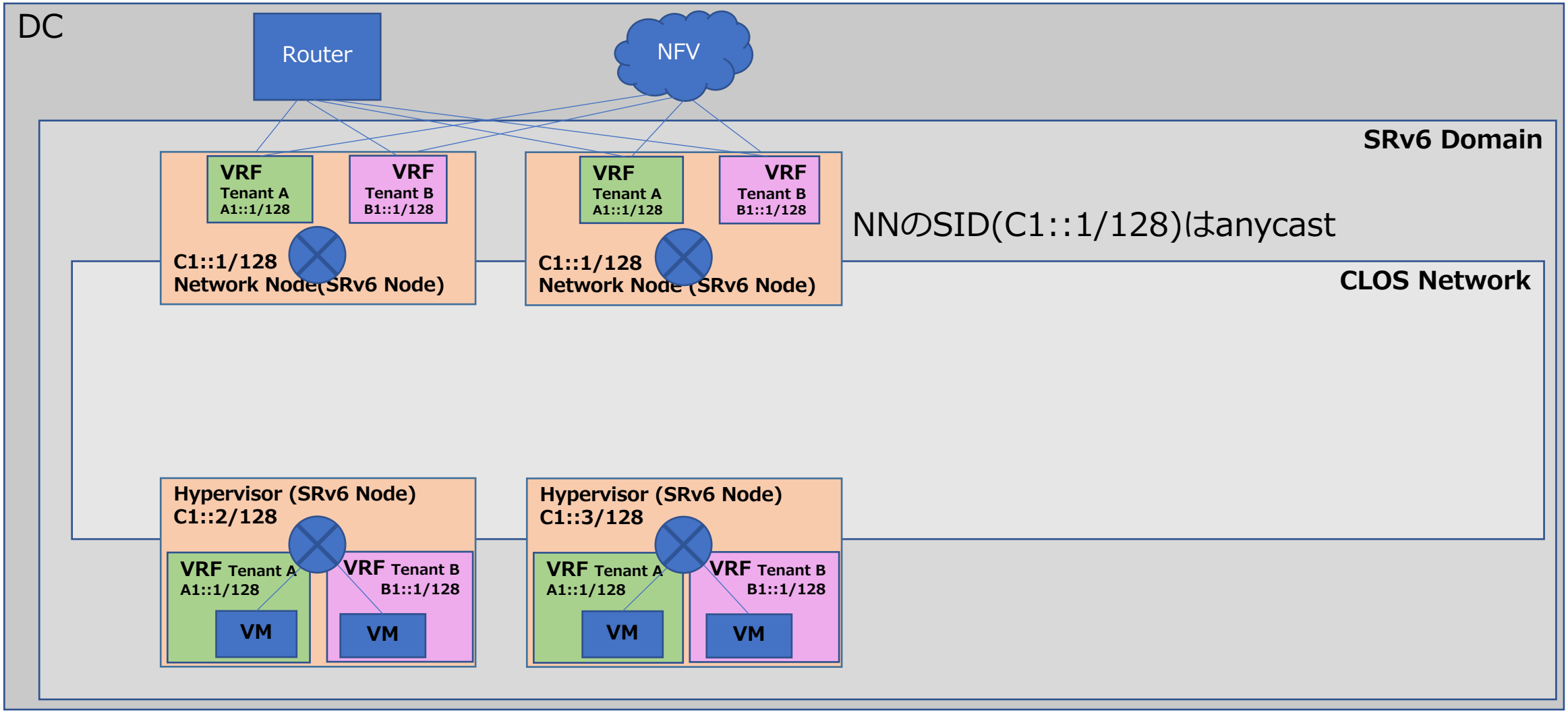
アンダーレイはIPv6のパケット転送ができれば良い

セグメントをうまく設計することで実現したい要件を表現できる

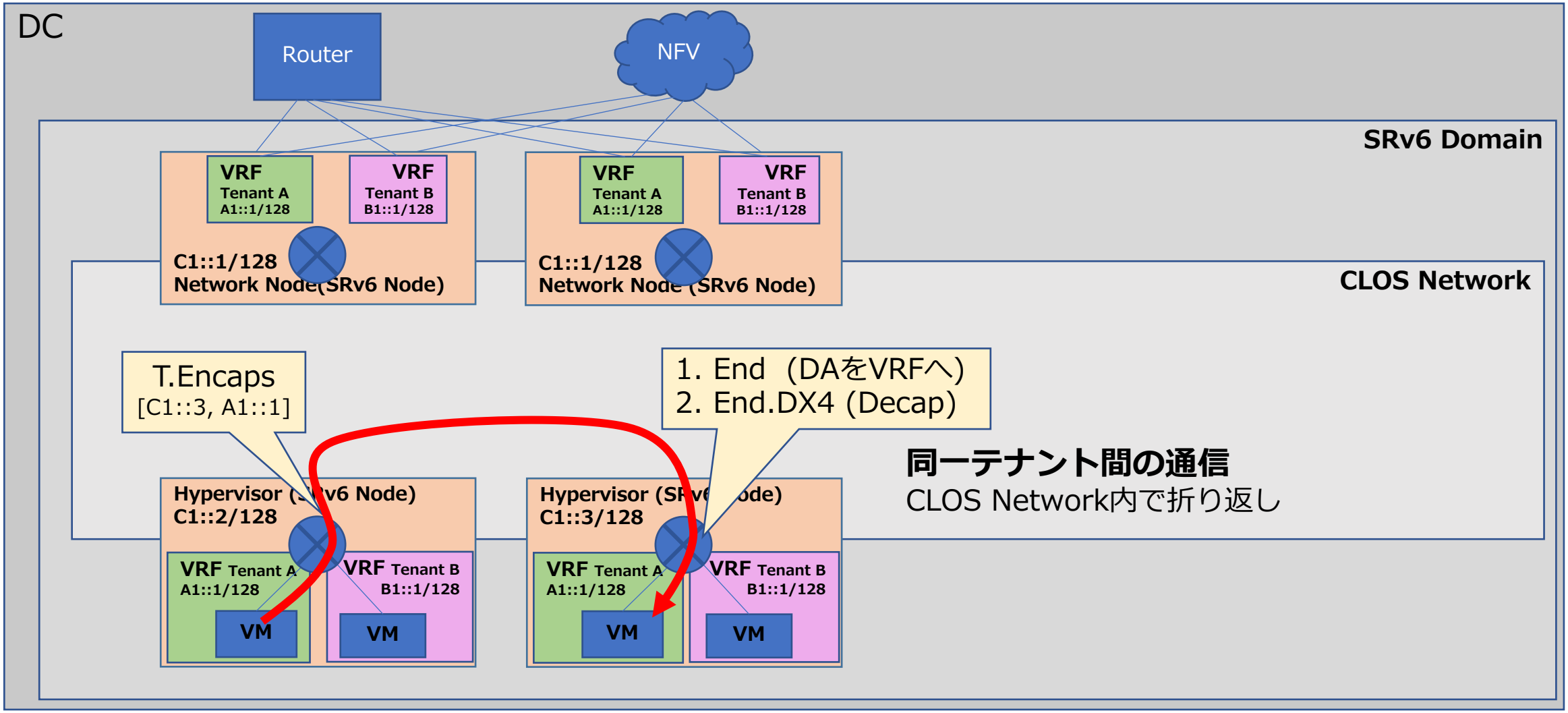
コンセプトデザイン (Data Plane)



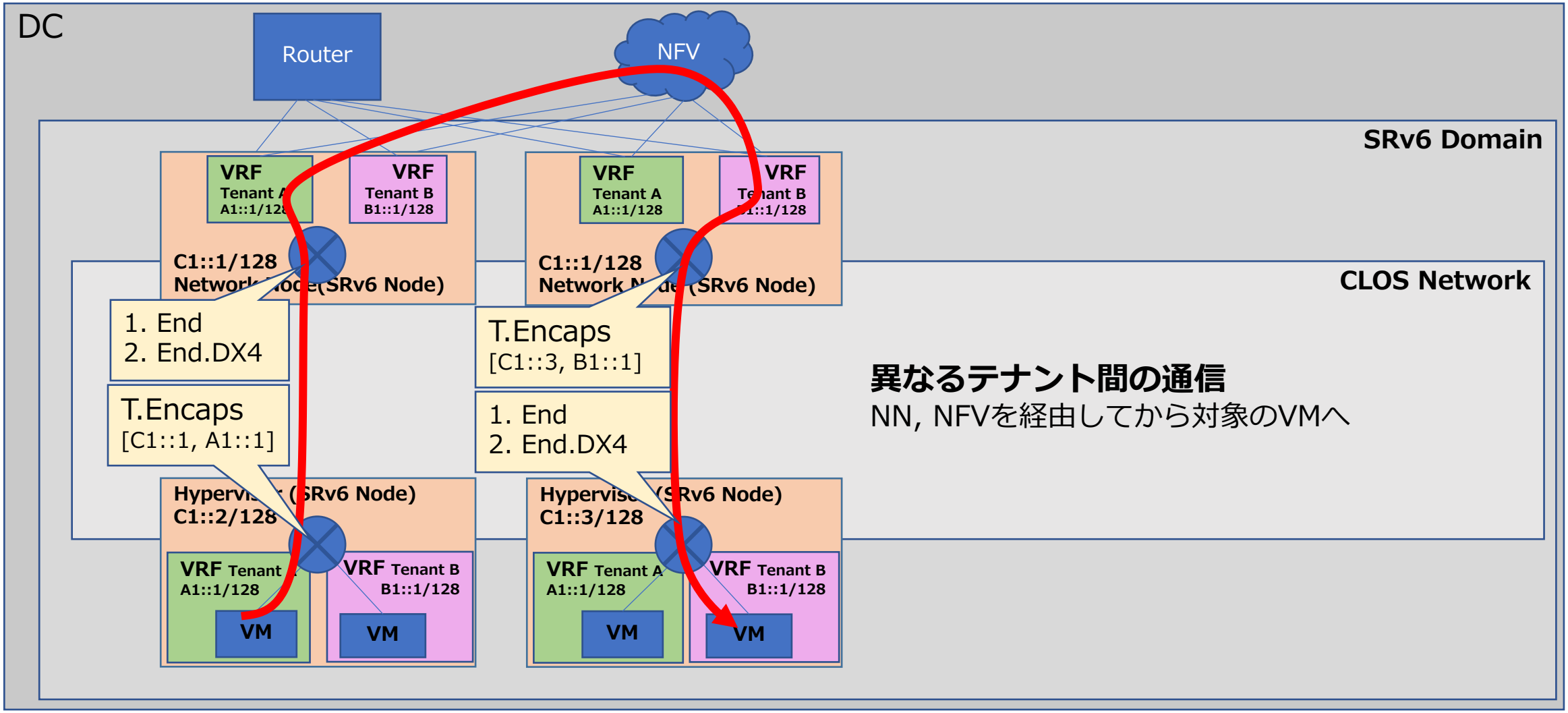
コンセプトデザイン (Data Plane)



コンセプトデザイン (Data Plane)



コンセプトデザイン (Data Plane)



コンセプトデザイン (Control Plane)

Network Node, Hypervisorは
宛先のVMを収容しているHV宛にEncapする必要がある

LINEではOpenStack baseのPrivate Cloudを運用
→ OpenStackのControllerは**HVが収容しているVM**を認識

Network Node, HypervisorにAgentを配置
→ Controllerの指示でEncap, Decapのルールを設定

PoC (Data Plane)

コンセプトデザインをVM複数台で構成
→ Linux Kernelの機能を使用して想定通りに動くことを確認

実施環境

- Linux Kernel: 4.18, iproute2-ss180813
  - 使用している機能
    - SRv6 (T.Encaps, End, End.DX4)
    - VRF (L3 master device)

性能検証

Linux Kernelの実装でどの程度性能が出るかを検証

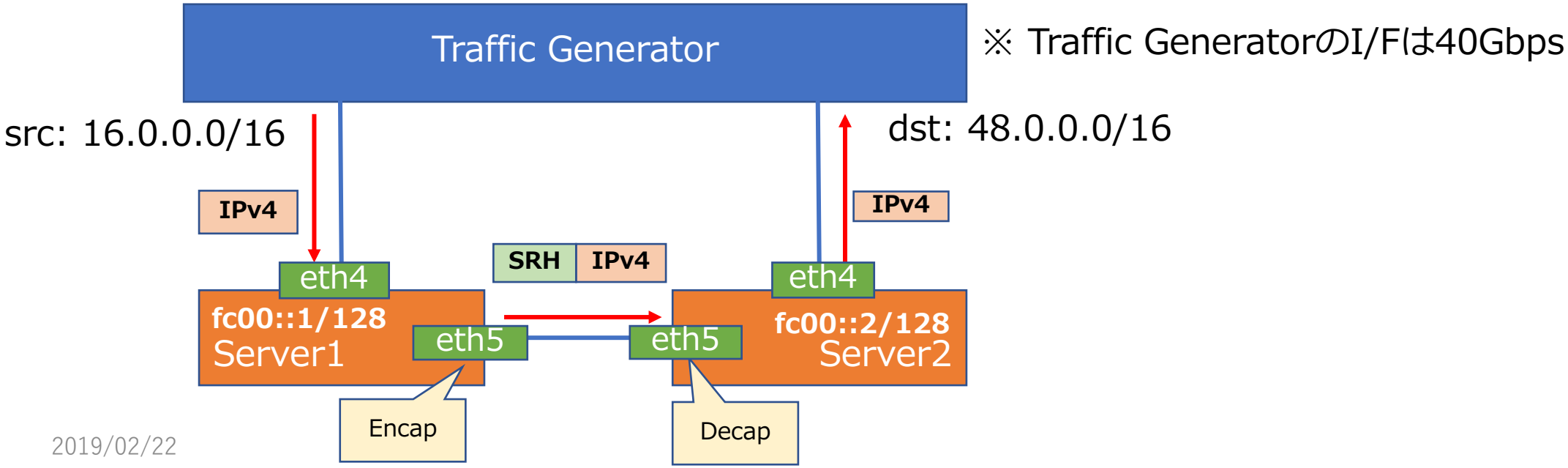
検証環境

検証対象機

Linux Kernel: 4.19.9, iproute2-ss180813

CPU: Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz

NIC: Mellanox ConnectX-4 (100Gbps x2port)



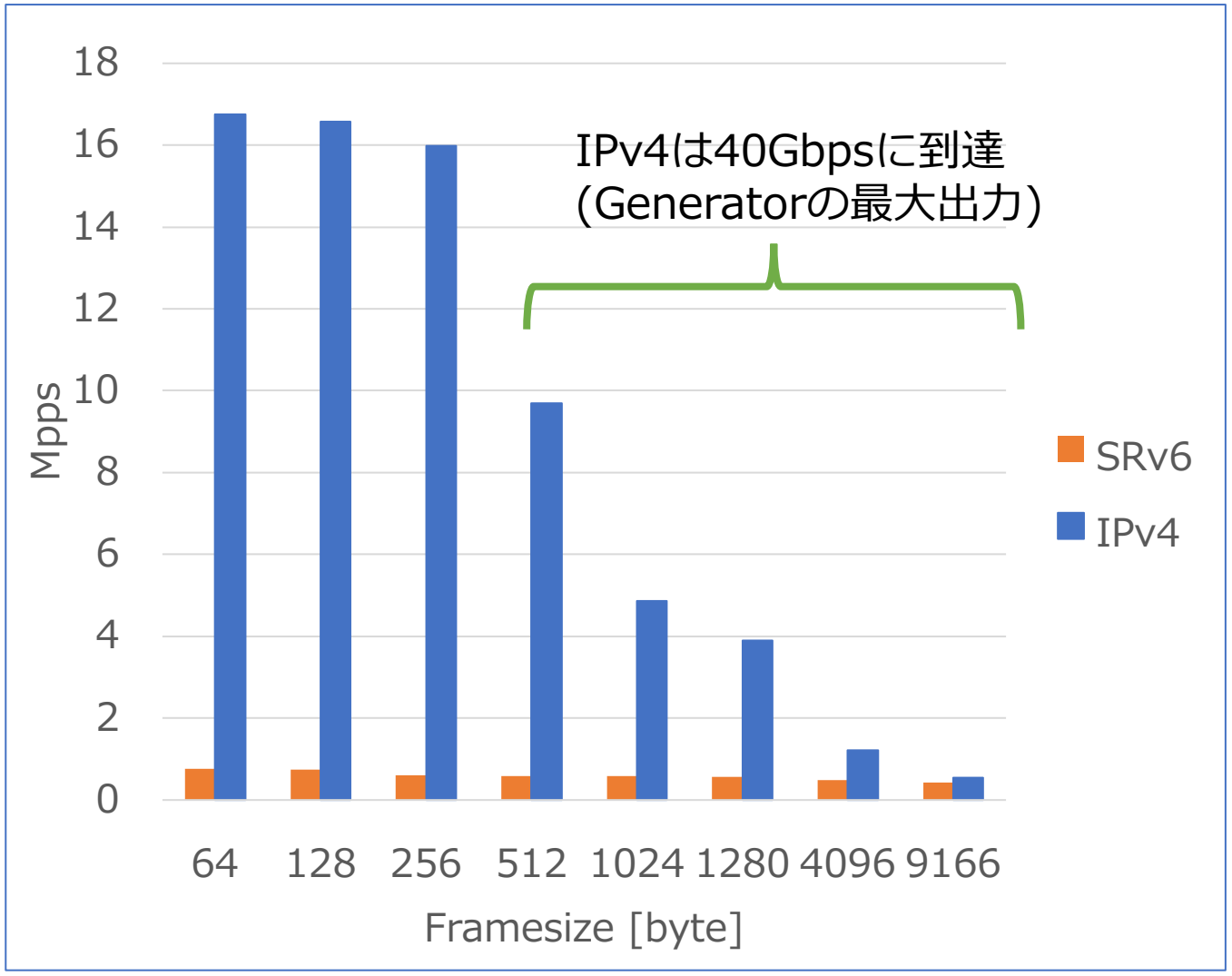
性能検証1: T.Encaps, End.DX4

単純なIPv4 forwardingとSRv6の比較

Framesize: 64byte

IPv4: 16.75Mpps
SRv6: 0.75Mpps

→ IPv4の4.5%の性能しか出ていない



性能検証1: T.Encaps, End.DX4

性能低下の原因: 特定のコアへのソフト割り込みの集中 (赤枠)

RSS(Receive-Side Scaling)が効いていない

→ EncapされたパケットはDecap側では
(src: fc00::1, dst: fc00::2) のフロー1種類に見えるため分散しない

IPv6 flowlabelの計算を有効にする

```
sysctl -w net.ipv6.seg6_flowlabel=1
```

→ NICの実装依存

今回検証で使用しているNICでは効果はなかったが、別のNICでは分散された

RPS(Receive Packet Steering)を有効にする
→ 多少は分散される? **(青枠)**

特定のHV間で大量に通信が流れる場合はあまり性能が出ない

トラフィック印加時のDecap側サーバのCPU使用率 (一部抜粋)

CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
all	0.00	0.00	0.01	0.00	0.00	1.50	0.00	0.00	0.00	98.48
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
15	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
23	0.00	0.00	0.00	0.00	0.00	8.79	0.00	0.00	0.00	91.21
24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

※ RSS: 受信パケットをマルチコアで処理できるようにする**NIC**の機能 (Hardware)
RPS: 受信パケットをマルチコアで処理できるようにする**Linux Kernel**の機能 (Software)

性能検証2: 複数フローに変更

通常のトラフィックの場合: (Encap, Decap)の組み合わせは複数

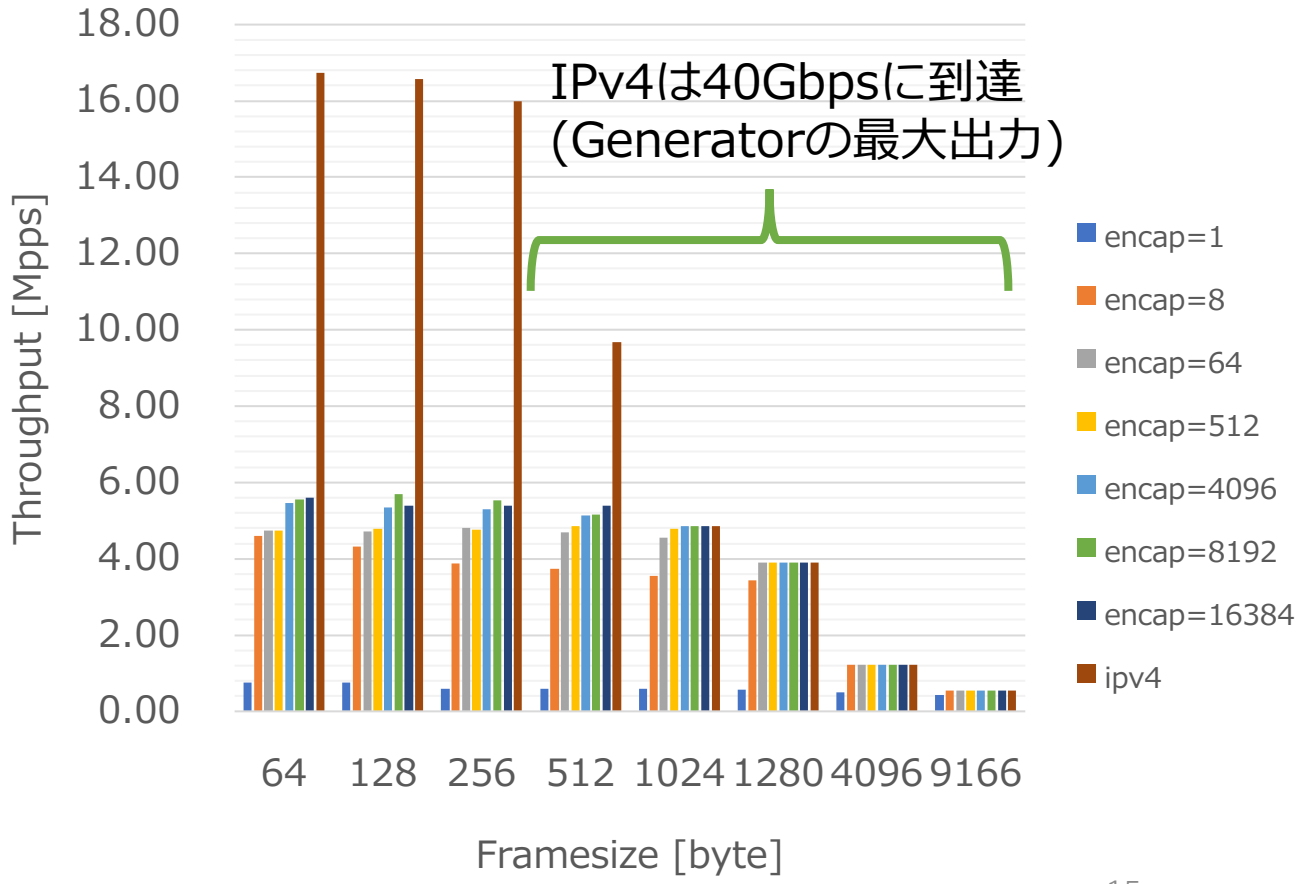
→ Decap側SIDを複数にして検証

Framesize: 64byte

IPv4: 16.75Mpps
SRv6(encap=16384): 5.61Mpps

IPv4比 33.5% (encap=1のときは4.5%)

1024byte以上ではIPv4と同等のスループット



性能検証3: End → End.DX4

実際のコンセプトデザインに近い設定での検証

Decap側での処理: End → End.DX4の二段で実施

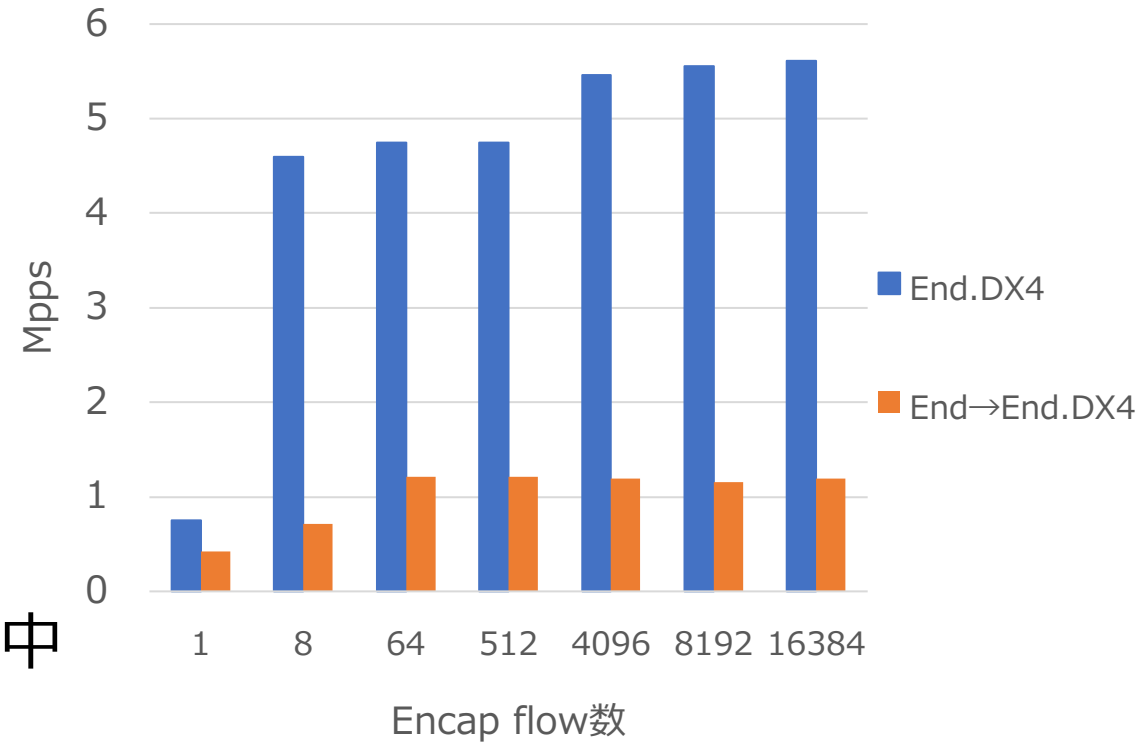
Framesize: 64byte (Encap=16384)

End.DX4のみ:　5.6Mpps

End→End.DX4: 1.2Mpps

原因: Linux KernelのSRv6実装上、
End, End.DX4で2回Route tableのlookupが動作

→ End.DX4 1回で実現できる構成を検討中

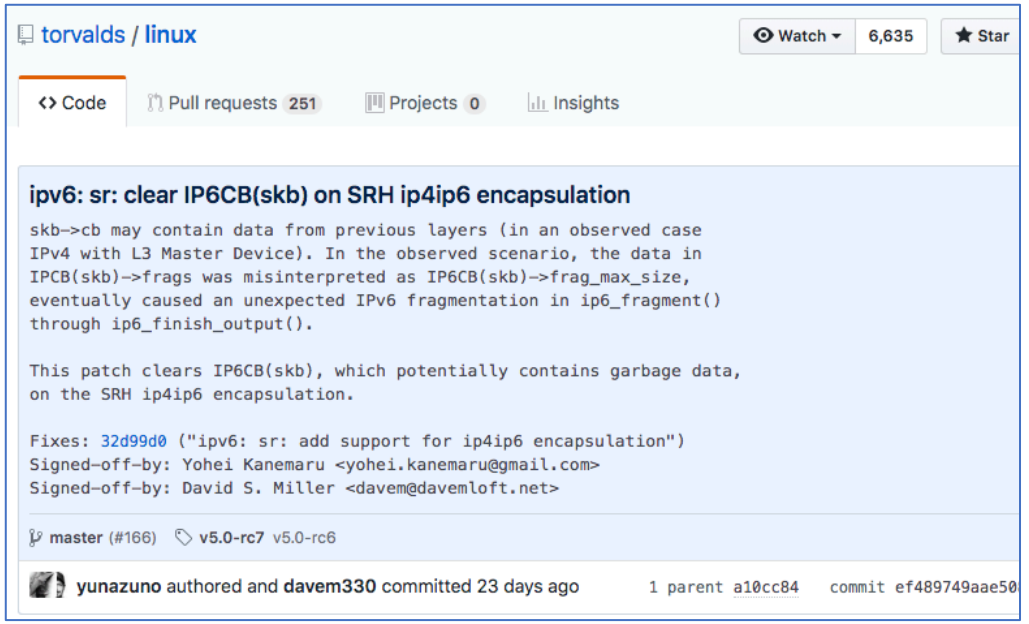


SRv6(Linux Kernel)へのContribution

問題:
VRFを使用している状況で、1280byteより大きいパケットを
T.Encapして送出するとMTUを変更していてもパケットが必ずfragmentする

原因:
Decap前のIPv4のIPCB(skb)->fragsが誤ってIP6CB(skb)->frag_max_sizeに変換されて
必ずfragmentされるような状態になっていた

IP6CB(skb)をclearするように変更して解決!
→ Kernelへパッチを送って無事取り込まれました



まとめ

データセンターでのSRv6ユースケース

- 現在マルチテナンシーを検討中
- 将来的にはサービスチェイニングも

SRv6でのマルチテナンシー

- 現在実装されているLinux Kernelの機能でPoCを実現
- Control Planeは実装が必要
- 性能面ではLinux Kernel実装以外も検討
  - DPDK(VPP), SmartNIC, P4 etc…