SONiCとP4ハードウェアにて実現する SRv6サービスチェイニング

APRESIA Systems株式会社
熊谷　渉

JANOG45 2020.01.23

自己紹介

- 熊谷 渉 (くまがい わたる)
- APRESIA Systems株式会社
  - 2006年より現在まで、ボックス型L3スイッチのソフトウェア開発を担当
  - 初期の頃はコーディングを担当していたが、設計→要件定義と上流工程へシフト
  - 2018年～ 当社がEdgecoreの代理店となったことで、Barefootデバイスを触る機会を頂き、本業の傍らでP4の勉強を始める(コーディングの楽しさを再認識)

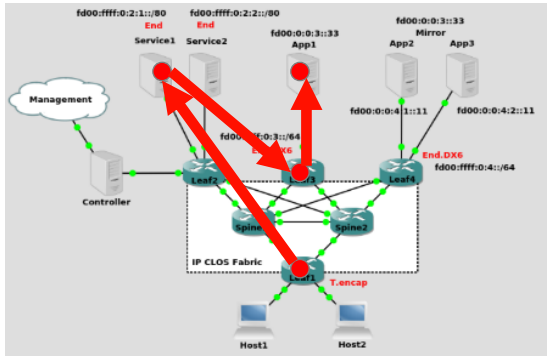
- 過去のJANOG発表 (JANOG42 P4に関する話題)
  - https://www.janog.gr.jp/meeting/janog42/program/SP-P4

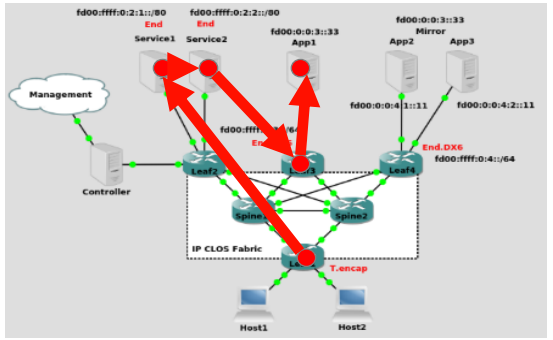
はじめに

APRESIA SYSTEMS

はじめに(1)

◆JANOG43にて、SRv6サービスチェイニングのコンセプトを発表





https://www.janog.gr.jp/meeting/janog43/program/sc

◆上記発表時は、仮想環境上でソフトウェア(Linux)により動作させた。

→性能等の懸念あり、運用を考えるとハードウェア動作させたい・・・

はじめに(2)

当社で試行錯誤の結果・・・

SONiCとP4対応ハードウェアを組み合わせることで
SRv6がハードウェア動作しました！！

今回の発表目的

- SRv6をハードウェア動作させるための知見を共有
- SRv6を使用してどのようなネットワークを構築したいか？を議論したい

概要

APRESIA SYSTEMS

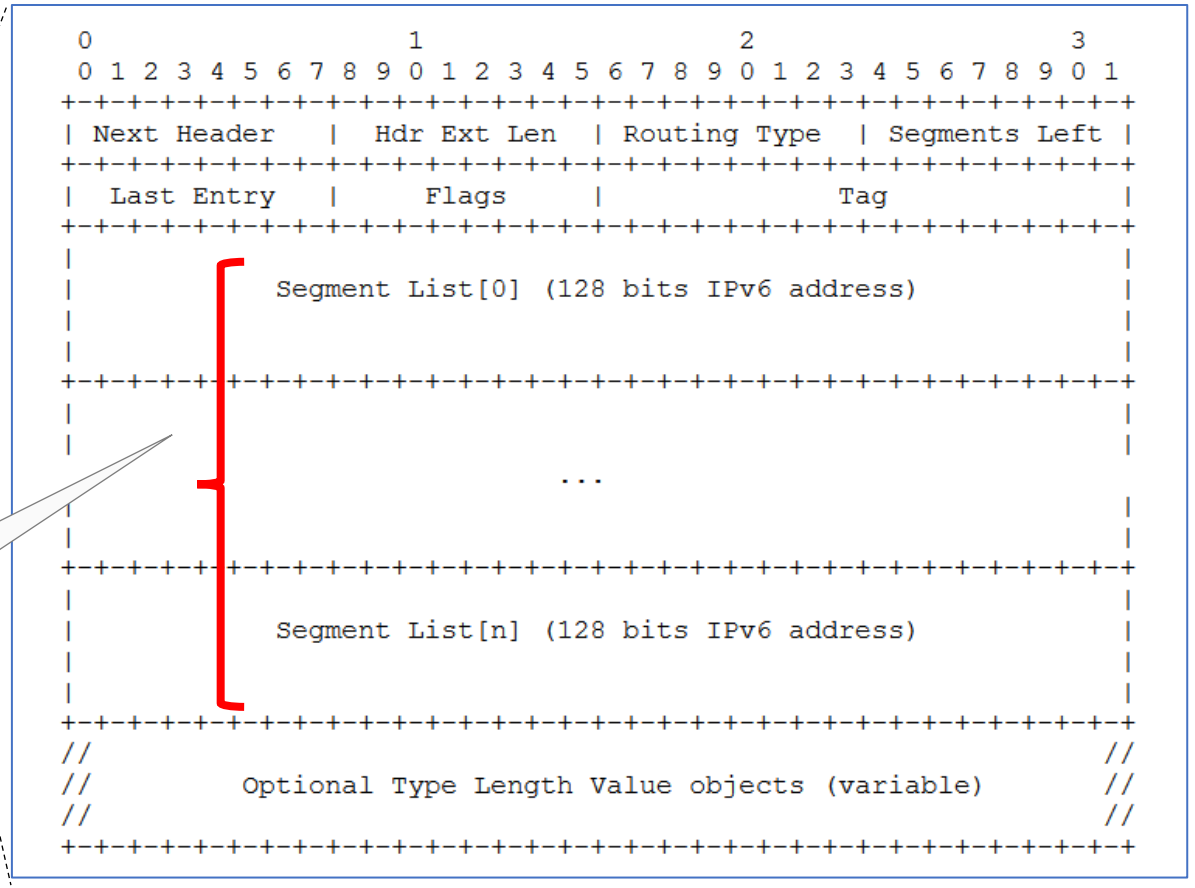
SRv6とは？

- IPv6をベースとしたセグメントルーティング
  - IPv6の拡張ヘッダを使用するため、既存のIPv6対応装置にて中継可能

SRv6 header format

Ether Header
IPv6 Header
IPv6 Optional Header
Payload

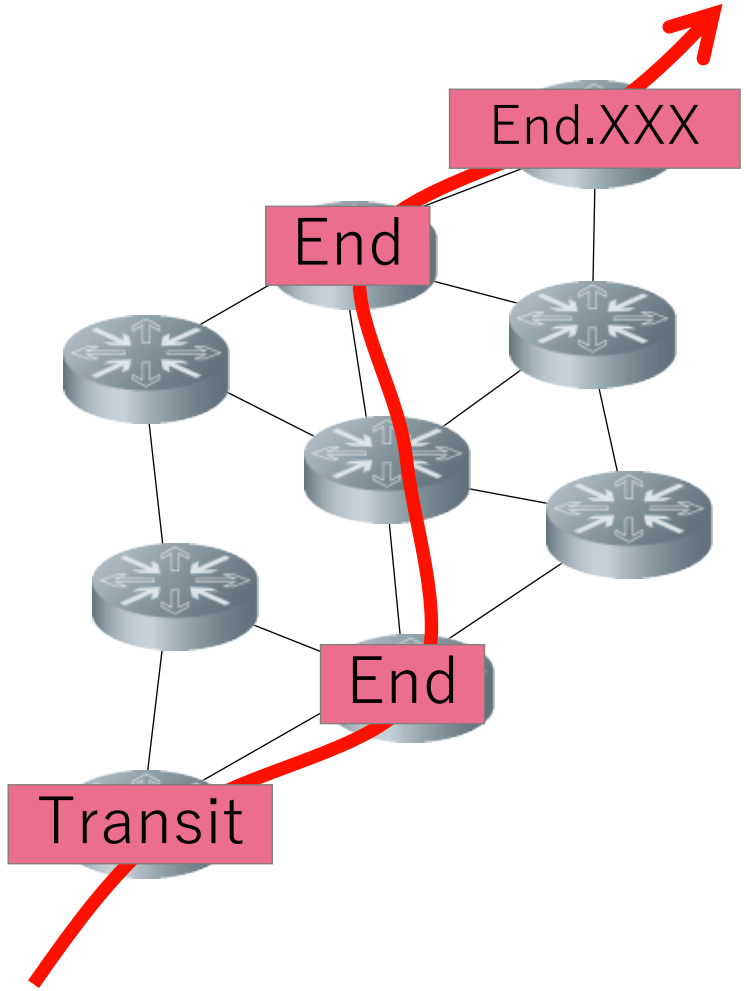
SIDリスト
IPv6形式で格納されている



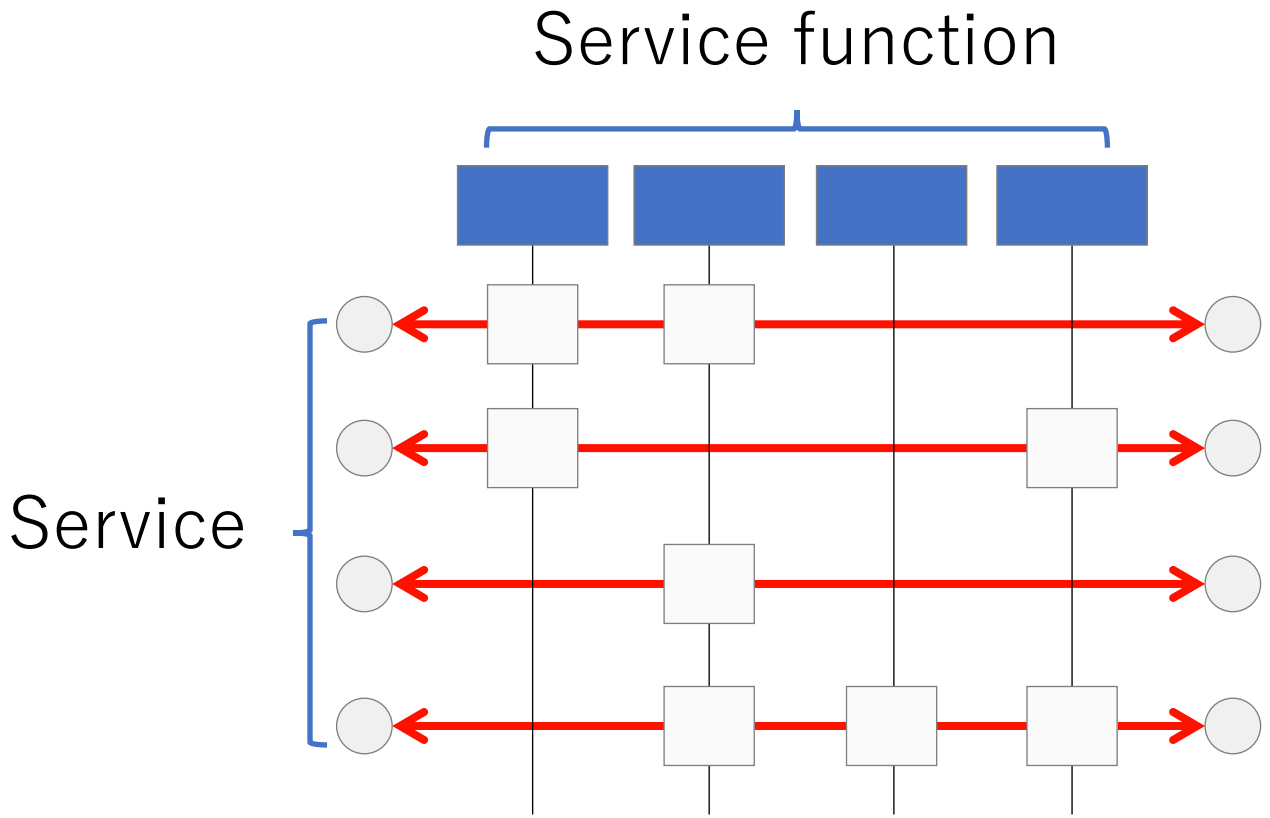
https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-26

SRv6ユースケース

◆トラフィックエンジニアリング



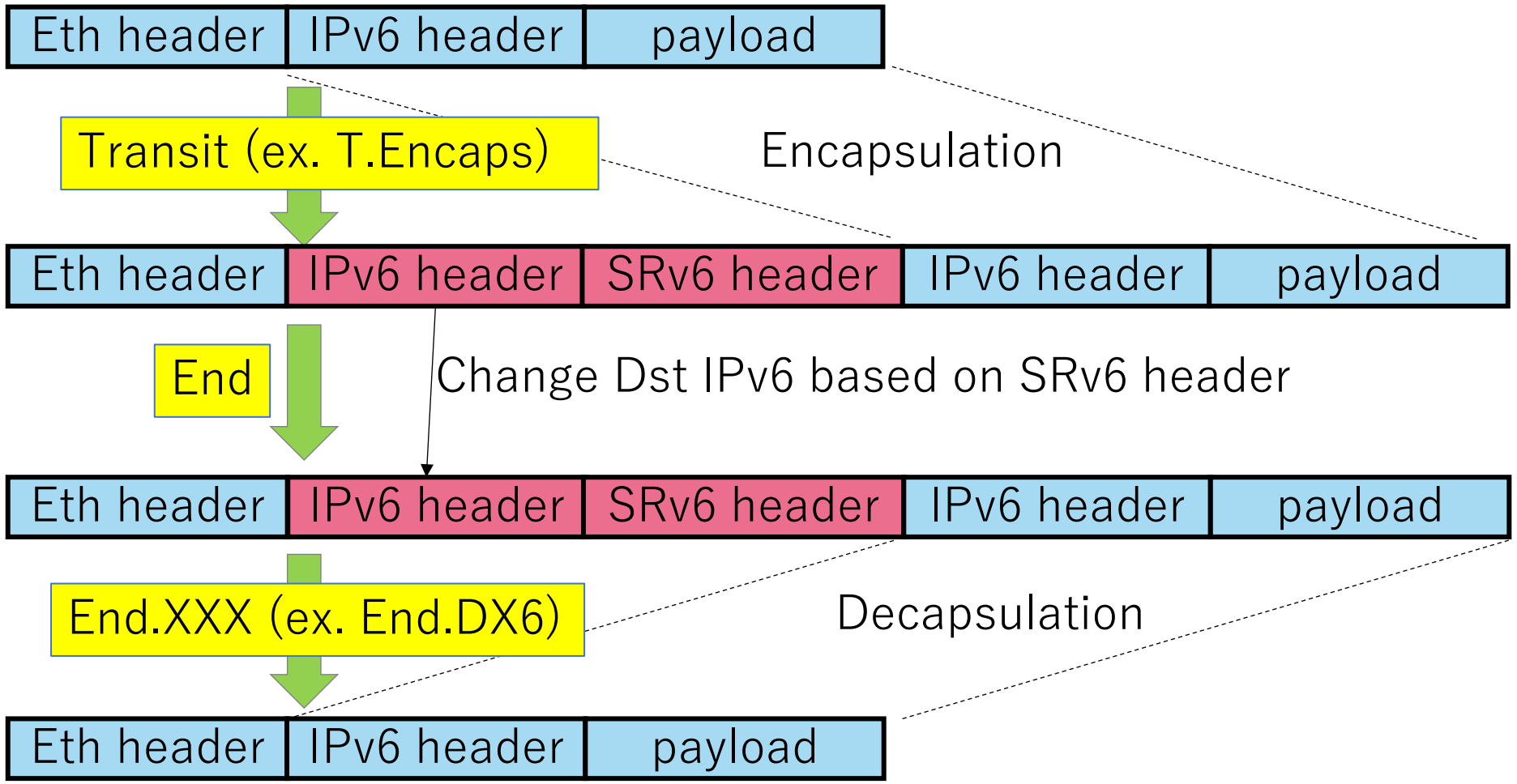
◆サービスチェイニング



今回はこちら

SRv6 Functions

SRv6 function



詳細はJANOG43の資料をご参照ください。
※Day 1でSRv6に関する発表多数あり

SRv6パケットの中継

◆SRv6パケット(宛先：次のSID)を各Functionへ到達させるためには？

SID: fd00:ffff:0:1:0:0:0:11

Locator　　Function

◇SID Locatorのみ
◇Functionを含めた全体

} どちらかを各ルーターへ広告する必要がある

◆今回は

◇SID Locatorを各ルーターへ広告
◇具体的には、Locatorのプレフィックスを持つIPv6アドレスを各装置のループバックアドレスに設定し、BGPにより広告(redistribute connected)

SRv6 実装の現状と新たな選択肢

- スイッチ: Cisco社, Huawei社製デバイス(ハードウェア)
- サーバー: Linux kernel, VPP(ソフトウェア)

新たな選択肢:

P4を使用してSRv6を実装し、P4対応ハードウェアで動作させる

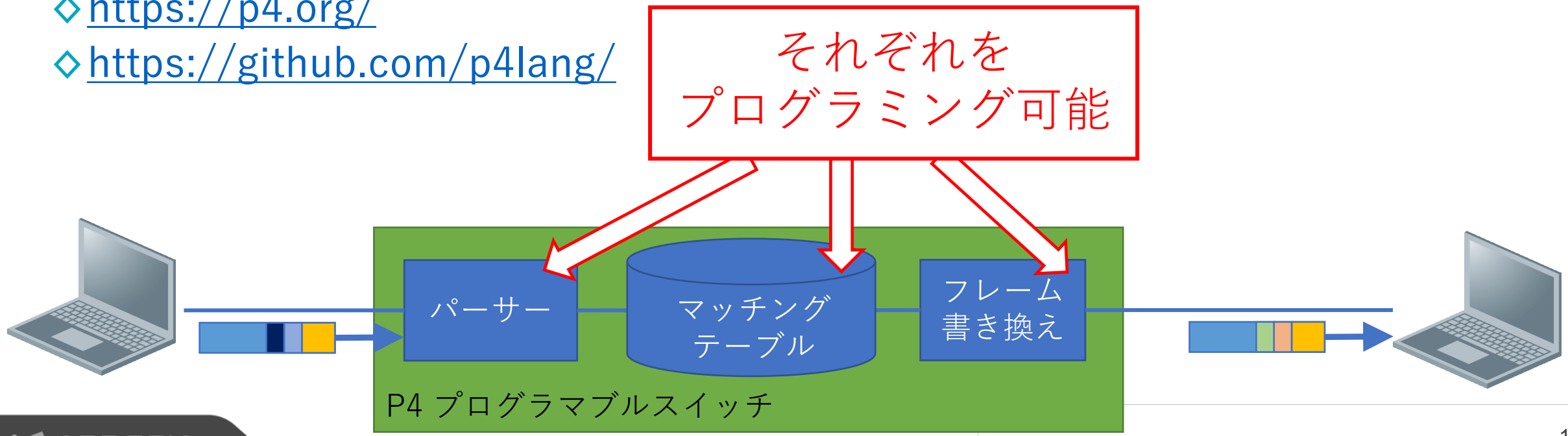
- 必要なFunctionを、必要なスペックで実装可能
- スイッチだけでなくサーバー側もハードウェア実装可能(Smart NIC)

→End to EndでハードウェアSRv6を実現！！

性能向上、負荷のオフロード、低遅延に期待

P4とは？

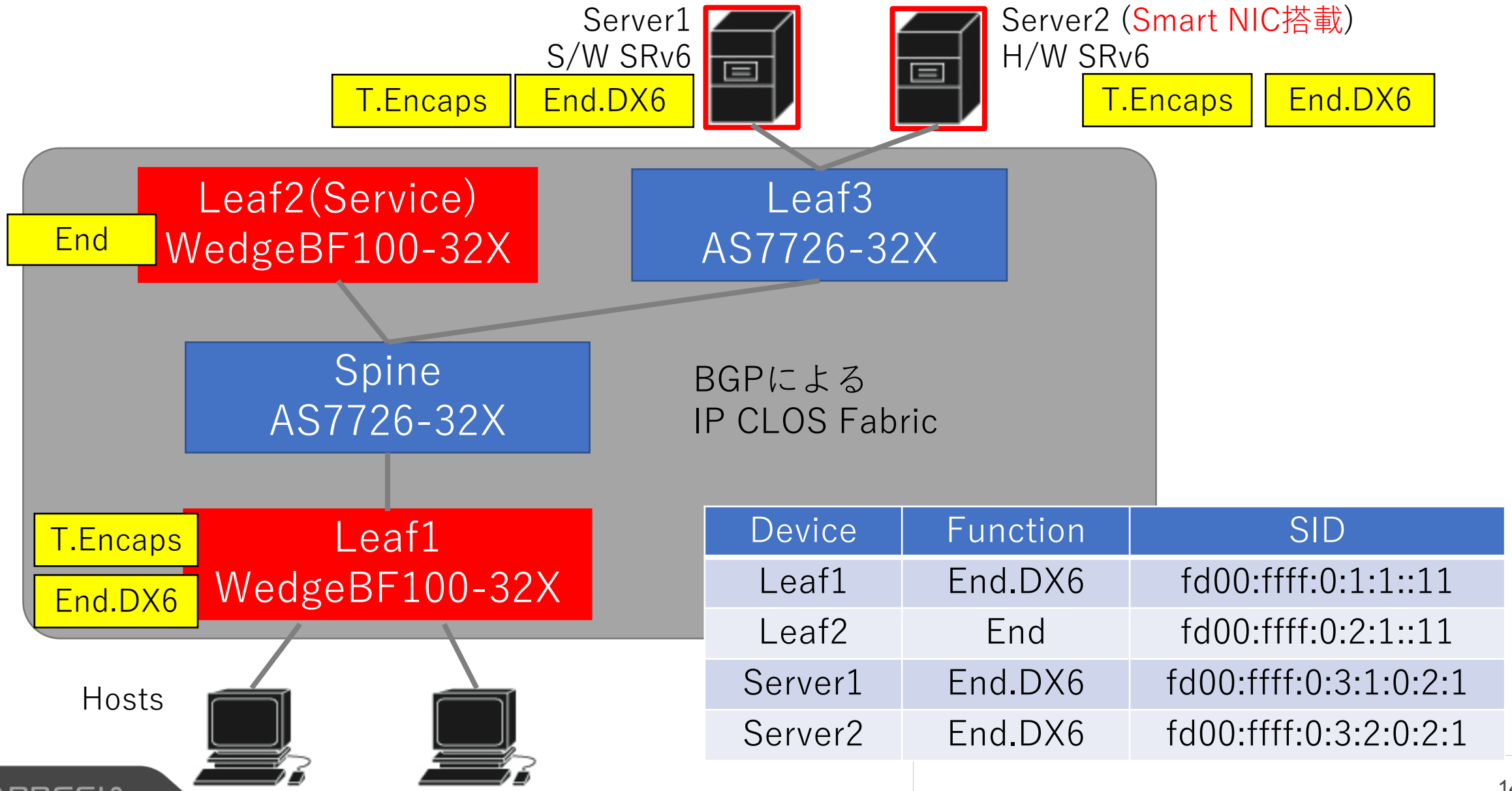
- データプレーンを記述可能なプログラミング言語
  - 言語仕様が2種類存在(P4_14, P4_16)
- (基本的には)ターゲットデバイス非依存
- P4対応ハードウェアへ実装することで、ユーザー定義データプレーンのハードウェア処理が可能となる
  - https://p4.org/
  - https://github.com/p4lang/



使用する装置

- スイッチ(Leaf):
  P4対応プログラマブルASIC(Barefoot Tofino chip)搭載スイッチ
  Edgecore Wedge100BF-32X
- スイッチ(Leaf/Spine):
  ホワイトボックススイッチ(P4未対応/Broadcom ASIC)
  Edgecore AS7726-32X
- サーバー:
  HPE Proliant DL380
    CPU: 2 *インテル® Xeon® Gold 6144
    Smart NIC: P4対応FPGA搭載 インテル® FPGA PAC N3000

構成図



Device	Function	SID
Leaf1	End.DX6	fd00:ffff:0:1:1::11
Leaf2	End	fd00:ffff:0:2:1::11
Server1	End.DX6	fd00:ffff:0:3:1:0:2:1
Server2	End.DX6	fd00:ffff:0:3:2:0:2:1

スイッチ上の
ハードウェアSRv6実現

APRESIA SYSTEMS

ハードウェア概要

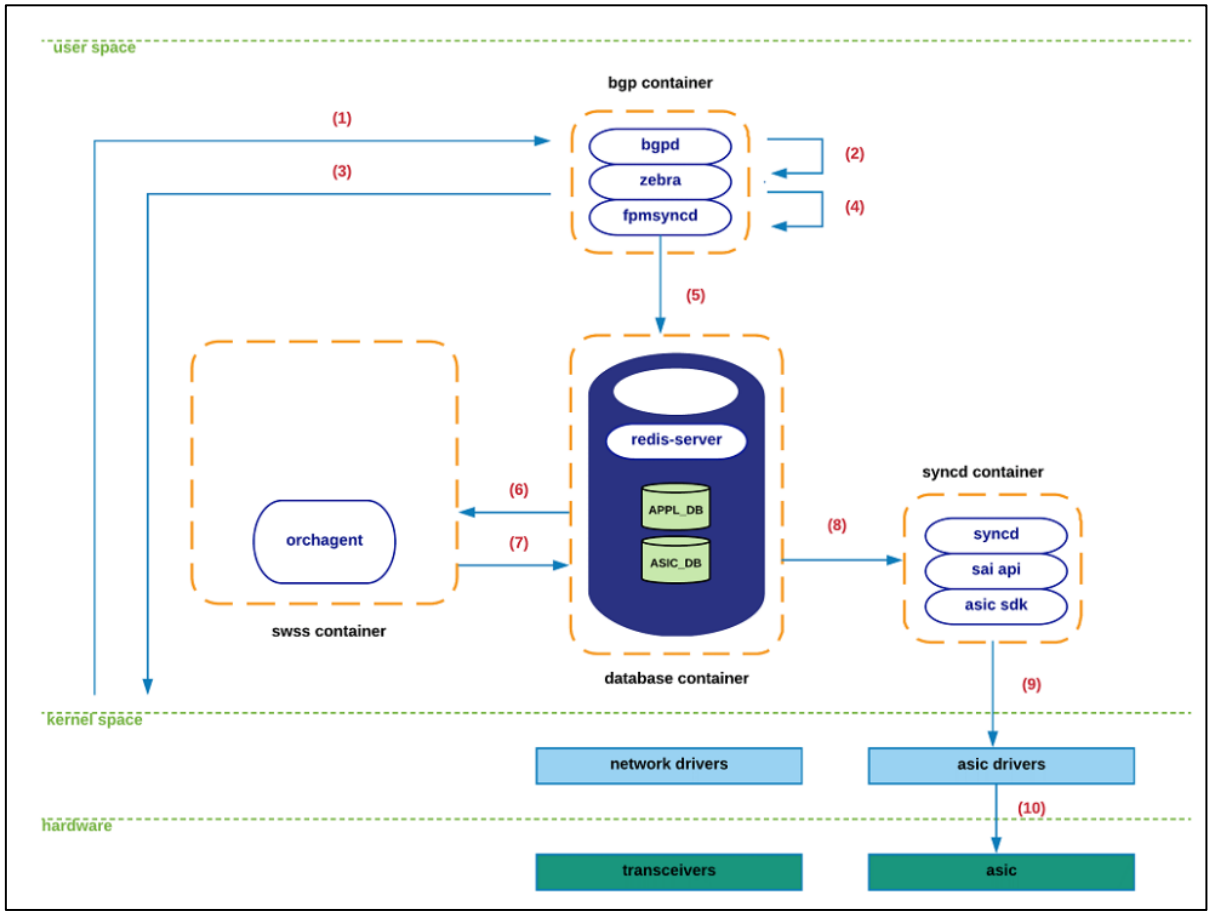
- Edgecore Wedge100BF-32X
  - P4対応プログラマブルチップ Barefoot Tofino 搭載
  - I/F: 100G QSFP28 x 32ポート
  - **SONiC対応**。これによりBGPを使用し、IP CLOS fabricを構成可能

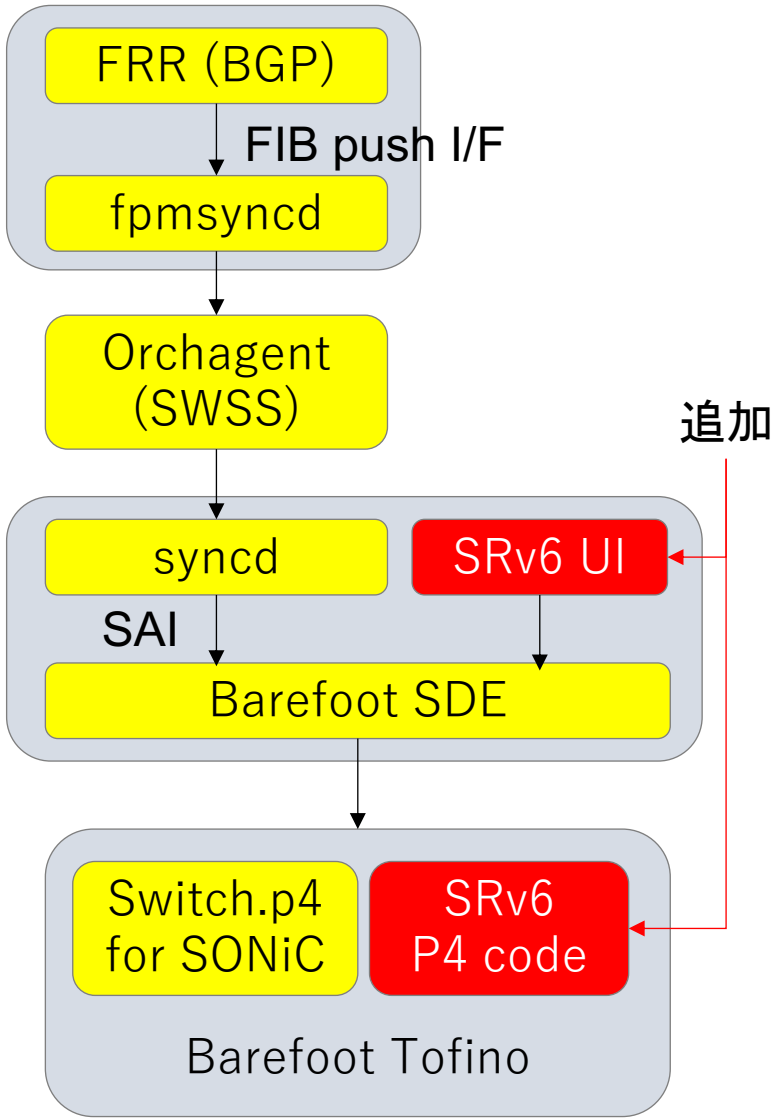
- SONiC参考資料:
  - JANOG44 OSSなWhitebox用NOSのSONiCが商用で使われている理由を考える
  - https://www.janog.gr.jp/meeting/janog44/application/files/1415/6396/6082/janog44_sonic_kuwata-00.pdf

SRv6のP4実装 in SONiC

SONiC Architecture

https://github.com/Azure/SONiC/wiki/Architecture



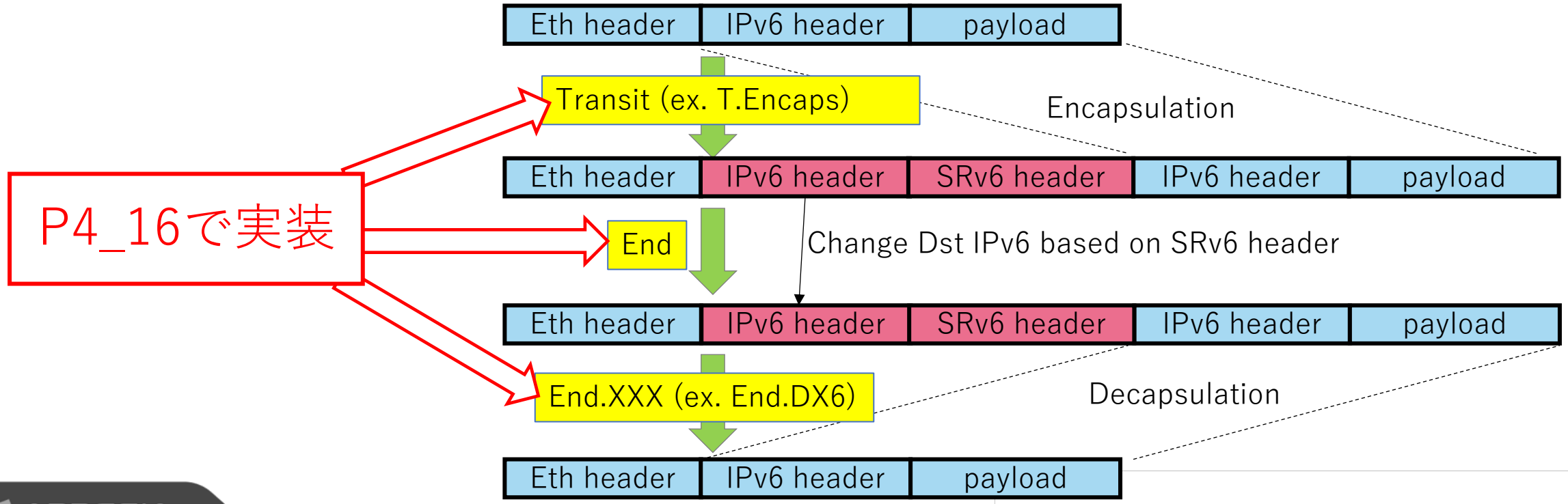


手順

1. SRv6ファンクションをP4で実装
2. 実装したP4コードをTofinoへロード(説明省略)
3. テーブルへマッチアクションのルール設定

1. SRv6ファンクションをP4で実装

- 基本的にはIETF仕様の通り実装すればよい。
- 機能拡張が可能
  - 例：T.EncapsでHostのIPアドレスをチェック(BGPパケットをEncap条件から外す)
- P4_16で記述(TofinoチップはP4_16対応済)



1. SRv6ファンクションをP4で実装

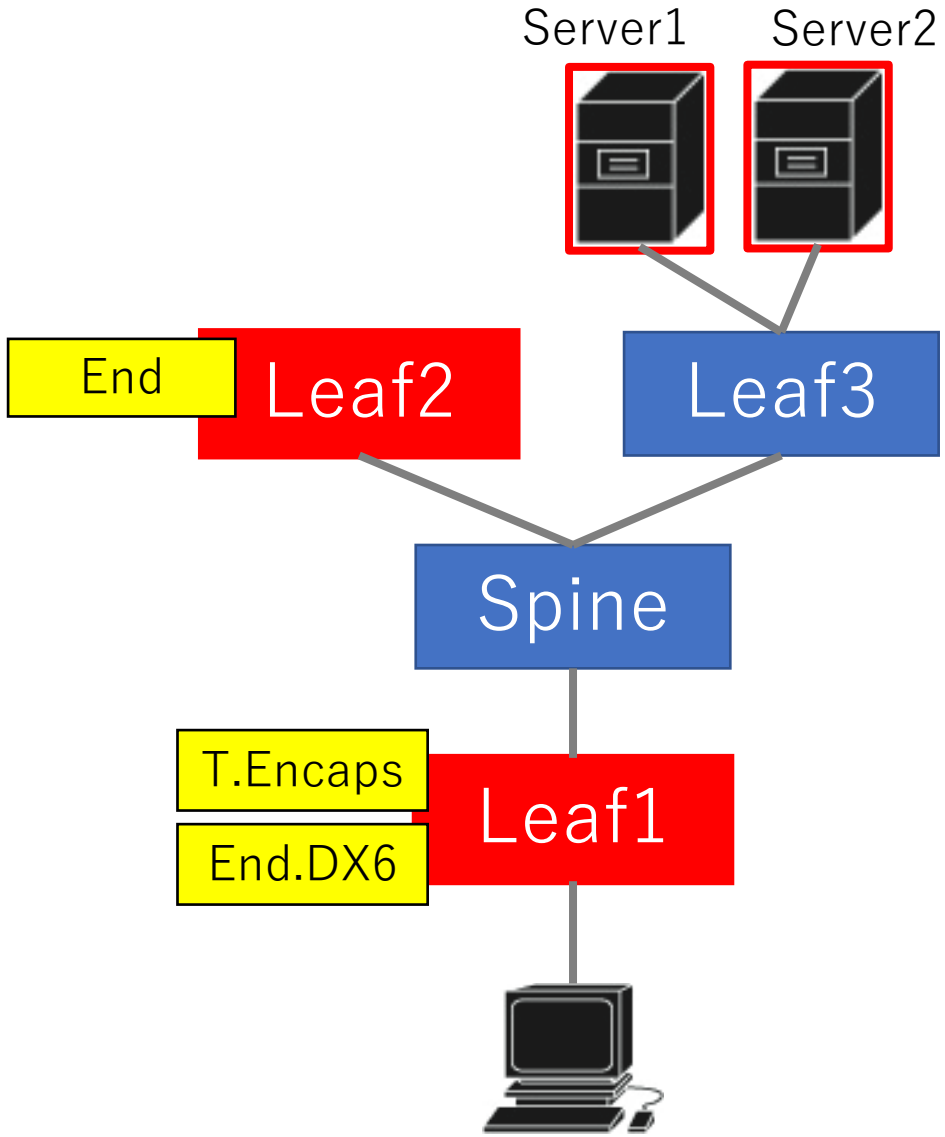
◆マッチアクションテーブルの定義(Leaf1/2共通)

Src IPv6 address	Dst IPv6 address	Segment Left	Action

マッチ条件　　アクション

3. テーブルへマッチアクションのルール設定

- T.Encaps (Leaf1)
  - Match条件: HostのIPv6アドレス
  - Action: T.Encaps (SID Leaf2, Server1) (SID Leaf2, Server2)
    ※経由するデバイスのSIDを指定
- End (Leaf2)
  - Match条件: 宛先IPv6アドレスが fd00:ffff:0:2:1::11 (SID=Leaf2)
  - Action: End
- End.DX6 (Leaf1)
  - Match条件:宛先IPv6アドレスが fd00:ffff:0:1:1::11 (SID=Leaf1)
  - Action: End.DX6



3. テーブルへマッチアクションのルール設定

◆Leaf1のマッチアクションテーブル

Src IPv6 address	Dst IPv6 address	Segment Left	Action
Host1's address	-	-	T.Encaps
Host2's address	-	-	T.Encaps
-	fd00:ffff:0:2:1::11 (Leaf1's SID)	0	End.DX6

◆Leaf2のマッチアクションテーブル

Src IPv6 address	Dst IPv6 address	Segment Left	Action
-	fd00:ffff:0:2:1::11 (Leaf2's SID)	-	End

Smart NIC上の
ハードウェアSRv6実現

APRESIA SYSTEMS

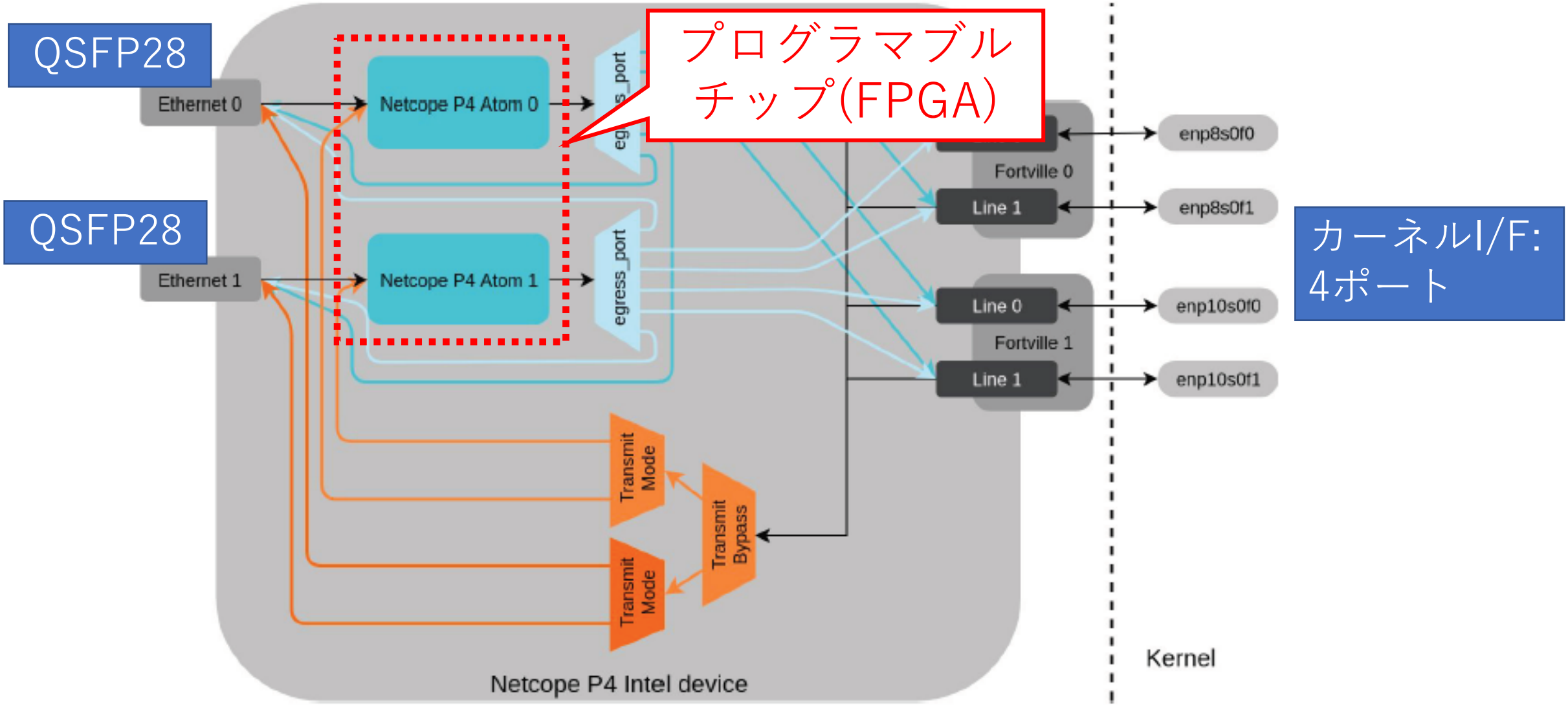
Smart NIC ハードウェア概要

- インテル® FPGA PAC N3000 (マクニカアルティマカンパニー様より借用)
  - I/F: QSFP28 x 2ポート
    - 4 x 10G (2ポート)、2 x 25G (2ポート)、または4 x 25G (1ポート)
  - プログラマブルチップ(FPGA)搭載
  - Netcope社のコンパイラを使用することで、P4で実装したコードをFPGA上にロードすることが可能→P4プログラミングが可能！

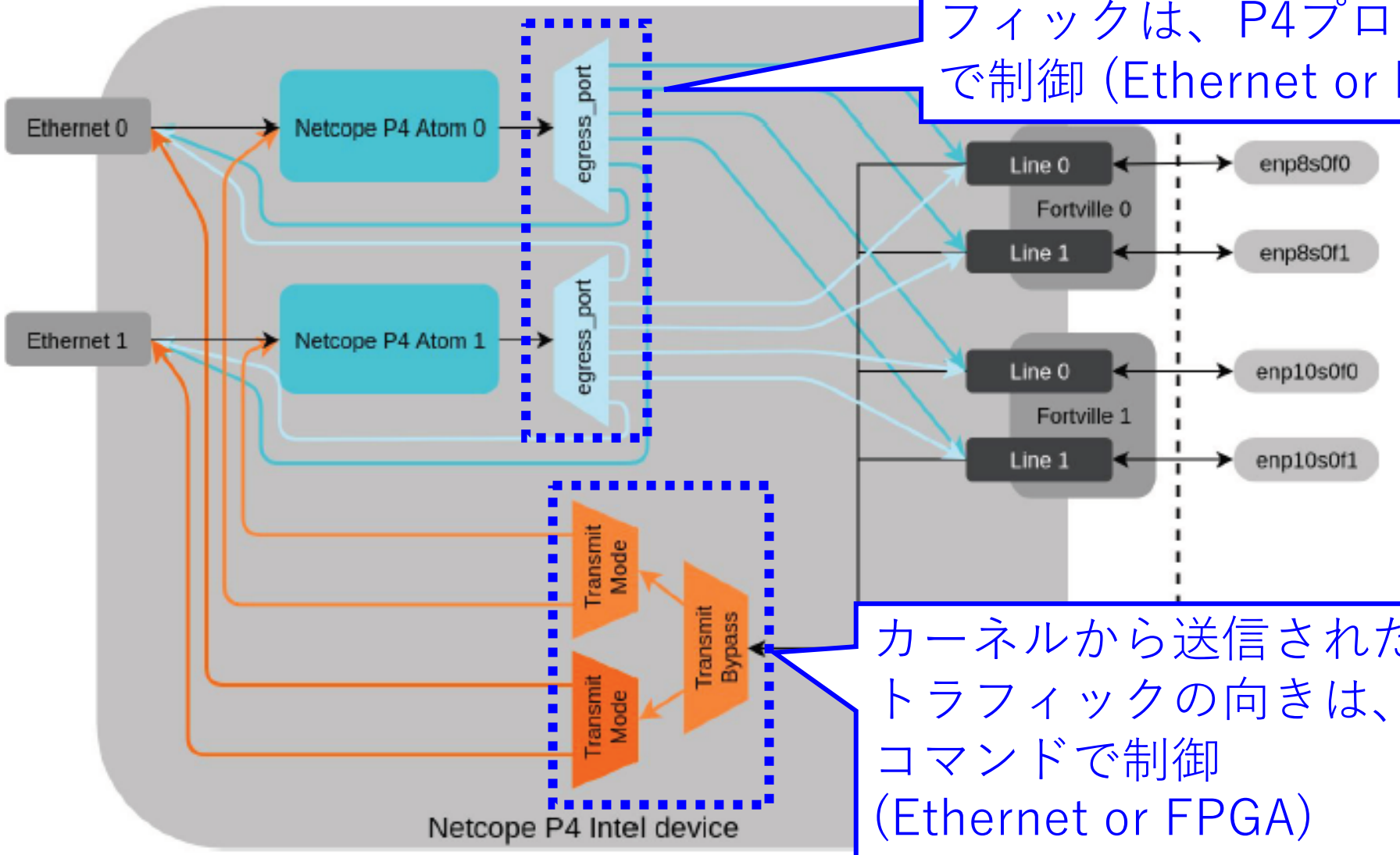
ALTIMA
A Macnica Division Company

NETCOPE
TECHNOLOGIES

インテル® FPGA PAC N3000 ブロック図



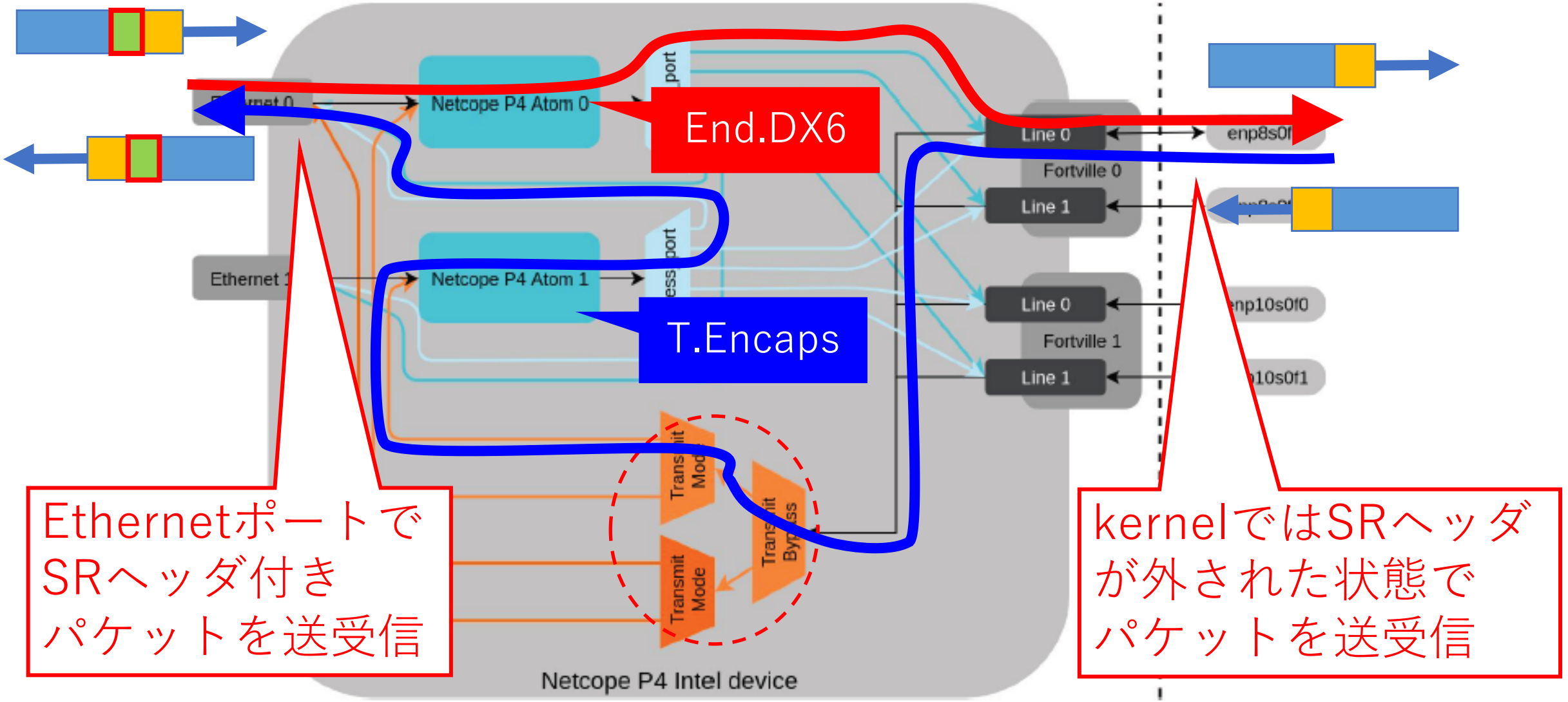
インテル® FPGA PAC N3000 ブロック図



FPGAから送信されるトラフィックは、P4プログラムで制御 (Ethernet or kernel)

カーネルから送信されたトラフィックの向きは、コマンドで制御 (Ethernet or FPGA)

今回実装したパケットフロー

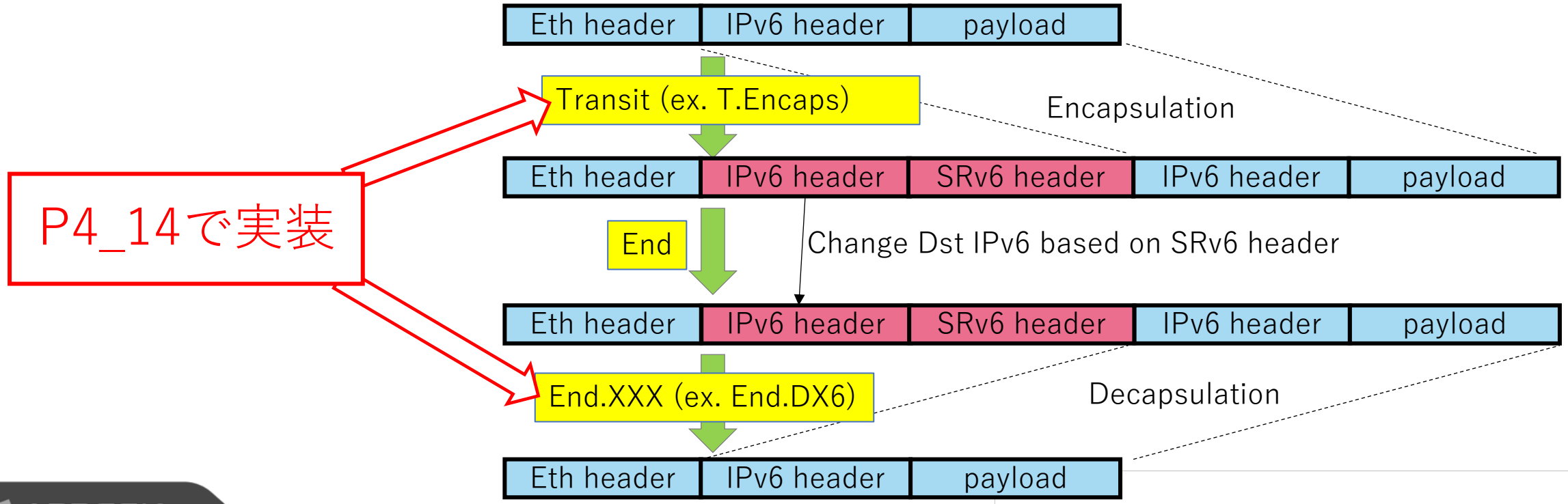


手順

1. SRv6ファンクションをP4で実装
2. 実装したP4コードをFPGAデータに変換
3. FPGAデータを、NIC上のFPGAへロード(説明省略)
4. テーブルへマッチアクションのルール設定

1. SRv6ファンクションをP4で実装

- 基本的にはスイッチと同様のコードでよいはず
- 但し、今回の試作時はコンパイラがP4_14のみ対応のため、NIC用のP4_14コードを別に作成
  - P4_16対応は2月予定



1. SRv6ファンクションをP4で実装

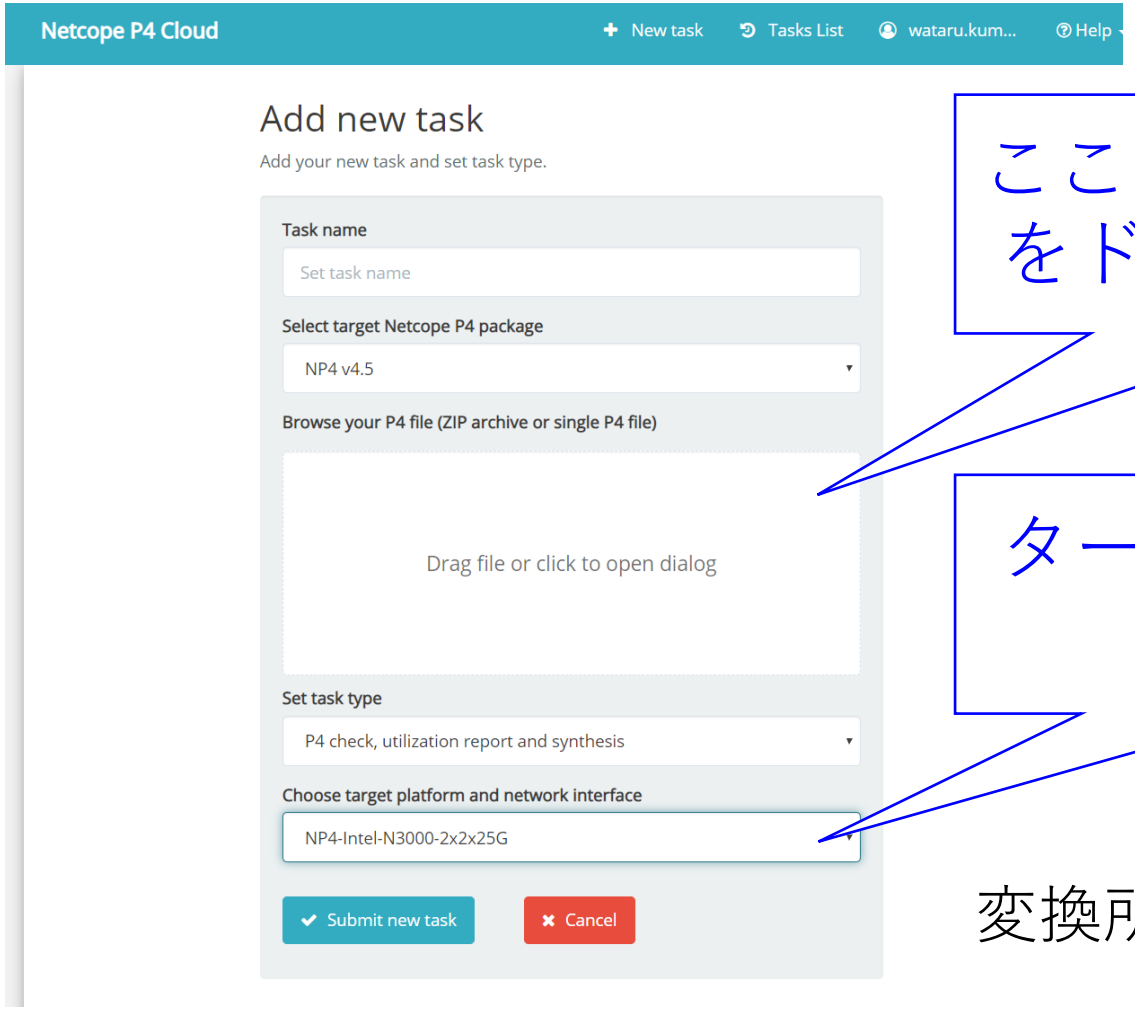
◆マッチアクションテーブル

Dst IPv6 address	Action

マッチ条件　　アクション

2. 実装したP4コードをFPGAデータに変換

◆Netcope社のWebサイト(Netcope P4 Cloud)で実施

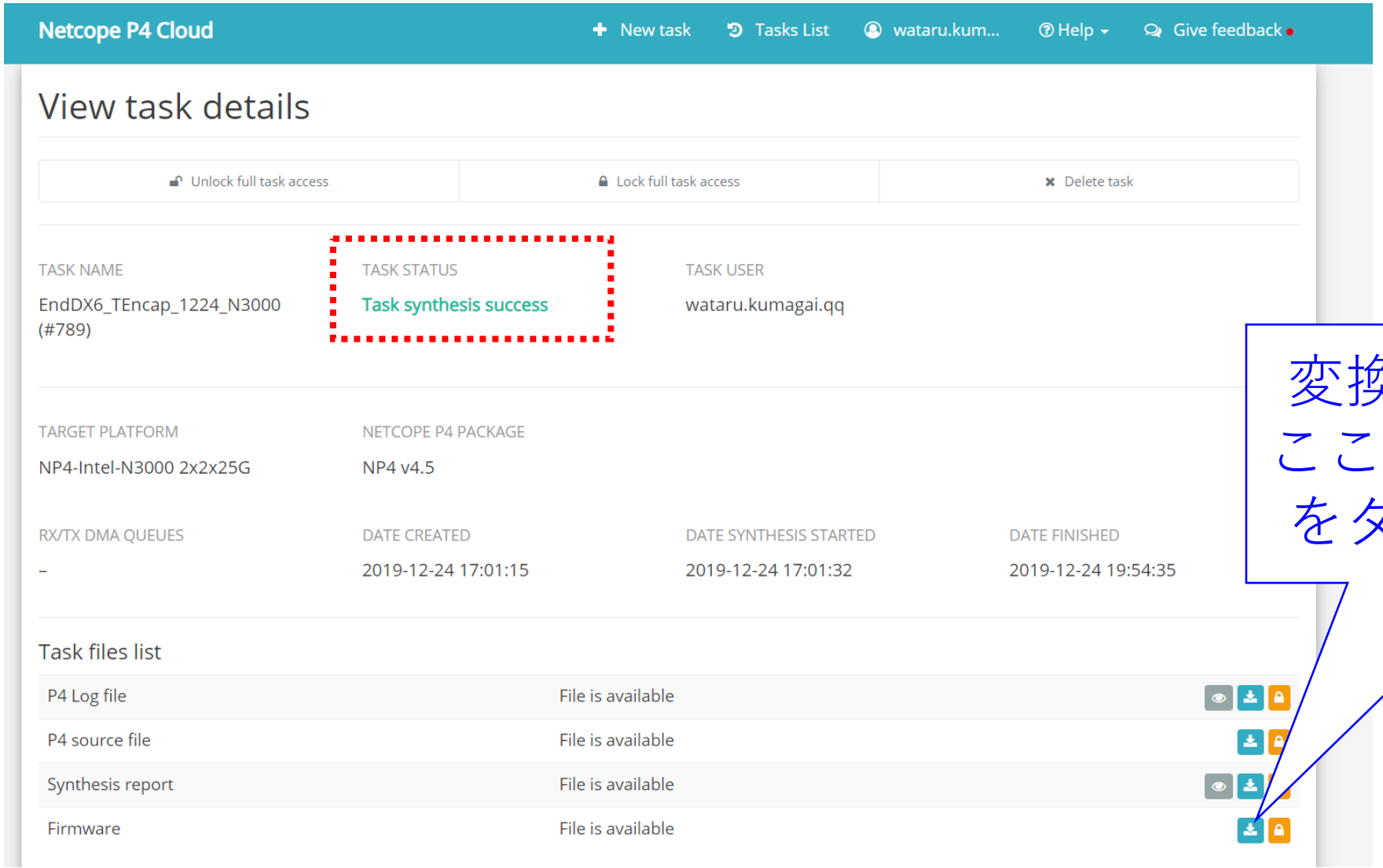


ここにP4ソースコードをドラッグ&ドロップ

ターゲットデバイスを選択

変換所要時間：3時間前後

2. 実装したP4コードをFPGAデータに変換

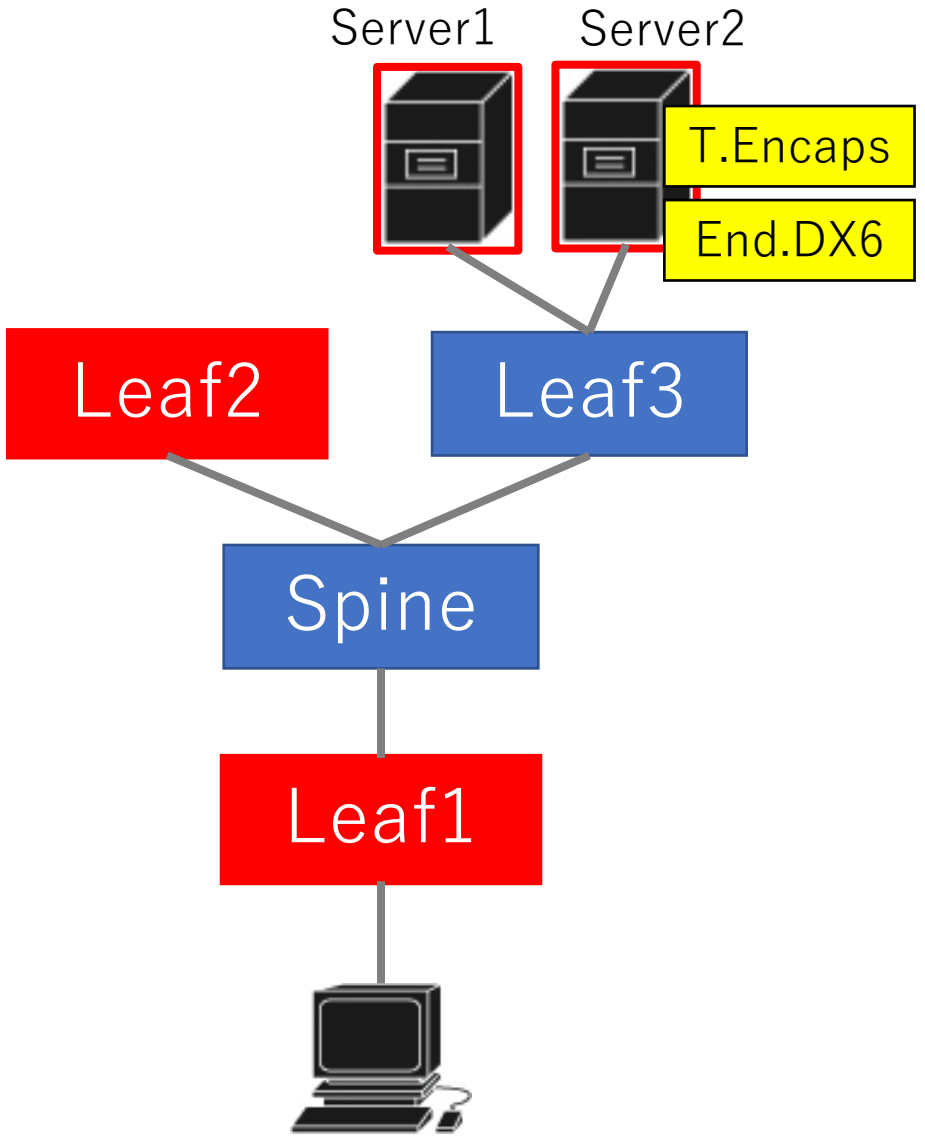


変換に成功すると、ここからFPGAデータをダウンロード可能

4. テーブルへマッチアクションのルール設定

- T.Encaps
  - Match条件: HostのIPv6アドレス
  - Action: T.Encaps (SID Leaf1, Leaf2)
    ※経由するデバイスのSIDを指定

- End.DX6
  - Match条件:宛先IPv6アドレスが fd00:ffff:0:3:2:0:2:1 (自分のSID)
  - Action: End.DX6



4. テーブルへマッチアクションのルール設定

◆マッチアクションテーブル

Dst IPv6 address	Action
Host1's IP address	T.Encaps
Host2's IP address	T.Encaps
fd00:ffff:0:3:2:0:2:1 (Server2's SID)	End.DX6

動作確認、性能評価

APRESIA SYSTEMS

P4ソースコード紹介

P4コード抜粋_Wedg

P4コード抜粋_SmartN

構成

Device	IPv6 address
Server2	fd00::3:2:0:0:2

Server2 (Smart NIC搭載)
H/W SRv6

T.Encaps　End.DX6

End

Leaf2(Service)
WedgeBF100-32X

Leaf3
AS7726-32X

Spine
AS7726-32X

キャプチャ
ポイント

T.Encaps

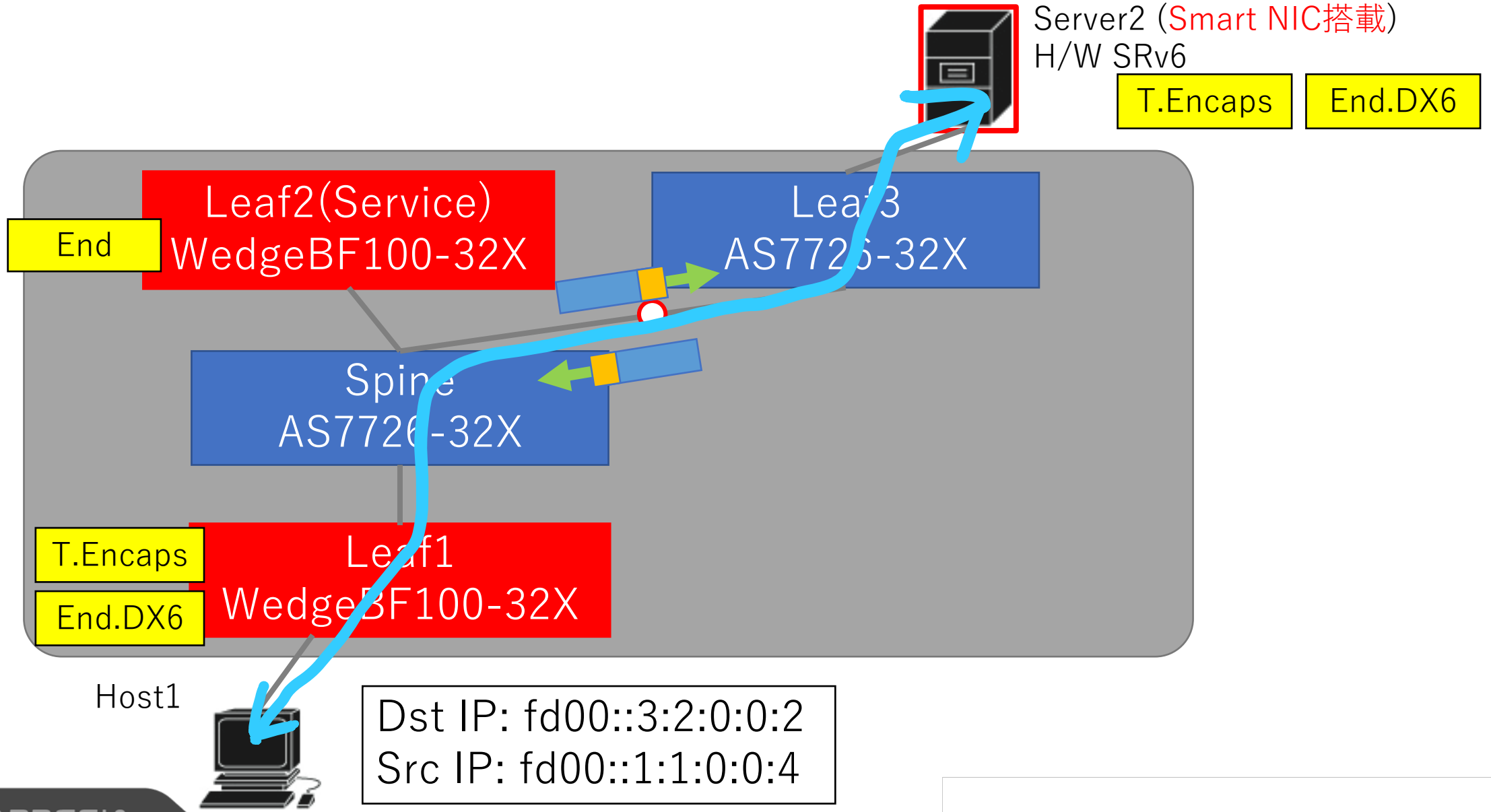
End.DX6

Leaf1
WedgeBF100-32X

Host1

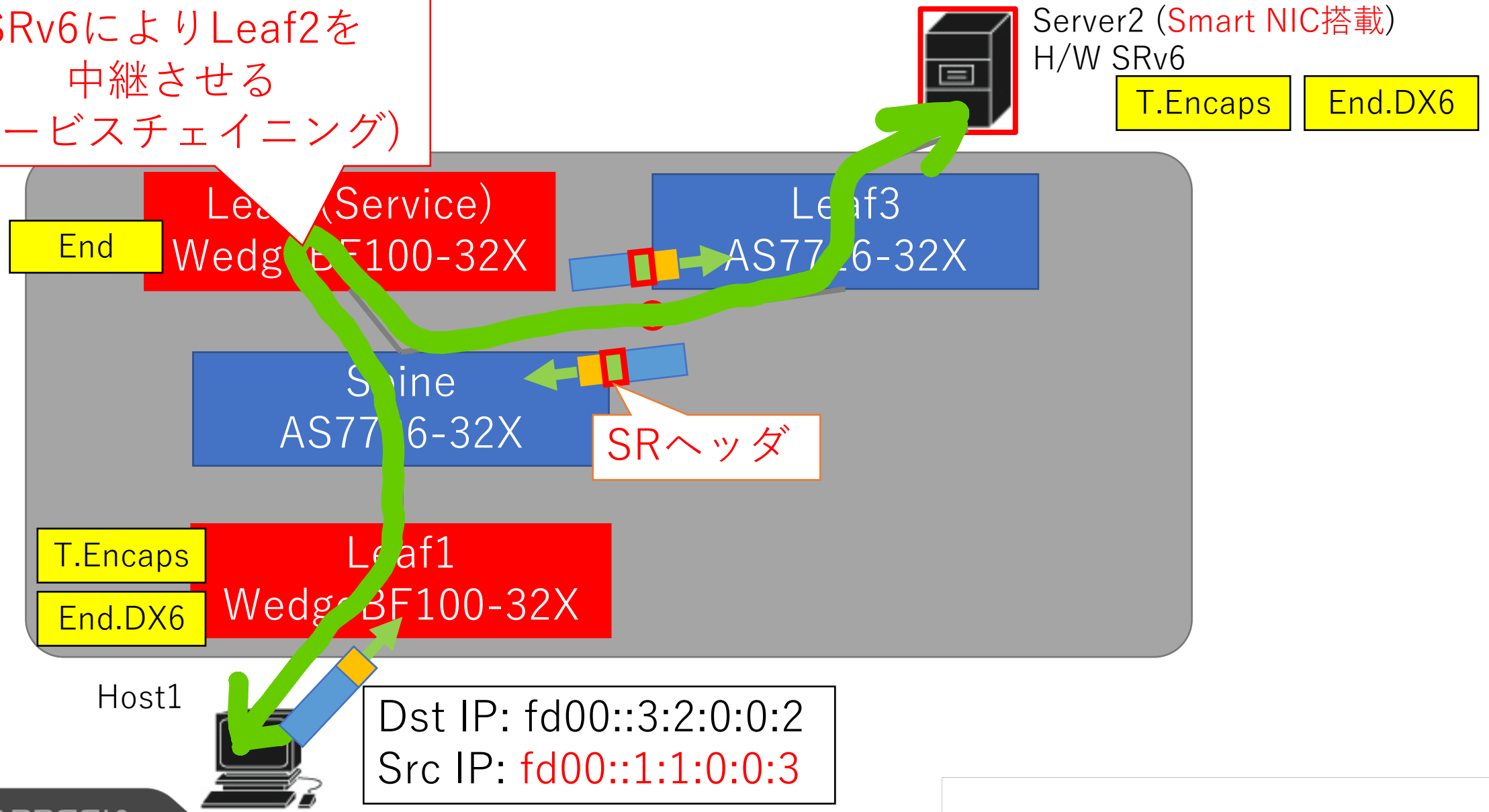
Device	IPv6 address	備考
Host1	fd00::1:1:0:0:4	SRv6無効
Host1	fd00::1:1:0:0:3	**SRv6有効**

SRv6無効時

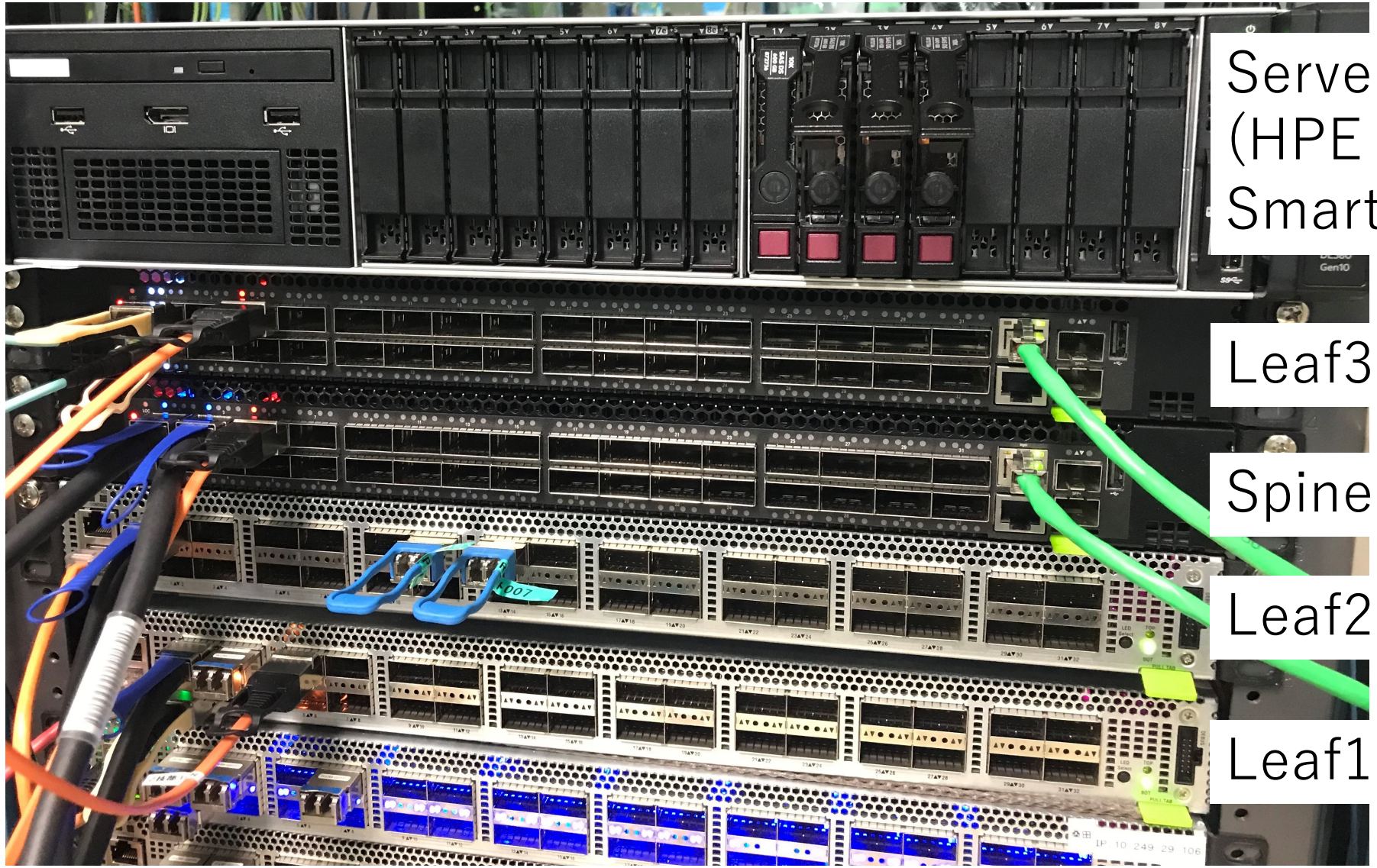

Server2 (Smart NIC搭載)
H/W SRv6
T.Encaps
End.DX6
End
Leaf2(Service)
WedgeBF100-32X
Leaf3
AS7726-32X
Spine
AS7726-32X
T.Encaps
End.DX6
Leaf1
WedgeBF100-32X
Host1
Dst IP: fd00::3:2:0:0:2
Src IP: fd00::1:1:0:0:4

SRv6有効時

SRv6によりLeaf2を
中継させる
(サービスチェイニング)



ラック写真



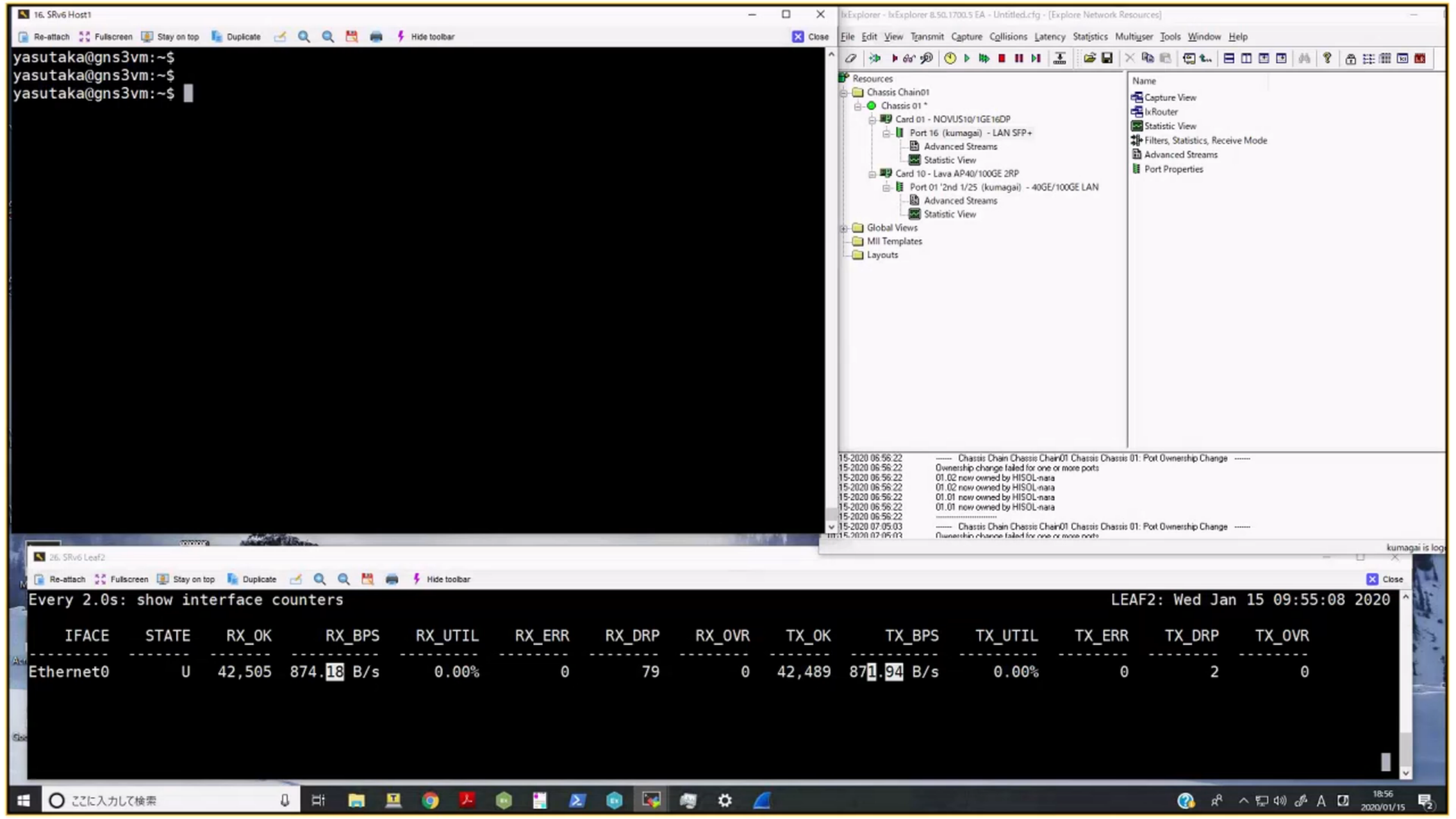
Server2
(HPE Proliant DL380)
SmartNIC実装

Leaf3(AS7726)

Spine(AS7726)

Leaf2(Wedge100BF)

Leaf1(Wedge100BF)


16. SRv6 Host1
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
yasutaka@gns3vm:~$
yasutaka@gns3vm:~$
yasutaka@gns3vm:~$
IxExplorer - IxExplorer 8.50.1700.5 EA - Untitled.cfg - [Explore Network Resources]
File Edit View Transmit Capture Collisions Latency Statistics Multiuser Tools Window Help
Resources
Chassis Chain01
Chassis 01 *
Card 01 - NOVUS10/1GE16DP
Port 16 (kumagai) - LAN SFP+
Advanced Streams
Statistic View
Card 10 - Lava AP40/100GE 2RP
Port 01 '2nd 1/25 (kumagai) - 40GE/100GE LAN
Advanced Streams
Statistic View
Global Views
MII Templates
Layouts
Name
Capture View
IxRouter
Statistic View
Filters, Statistics, Receive Mode
Advanced Streams
Port Properties
15-2020 06:56:22 ------ Chassis Chain Chassis Chain01 Chassis Chassis 01: Port Ownership Change ------
15-2020 06:56:22 Ownership change failed for one or more ports
15-2020 06:56:22 01.02 now owned by HISOL-nata
15-2020 06:56:22 01.02 now owned by HISOL-nata
15-2020 06:56:22 01.01 now owned by HISOL-nata
15-2020 06:56:22 01.01 now owned by HISOL-nata
15-2020 06:56:22 ----------
15-2020 07:05:03 ------ Chassis Chain Chassis Chain01 Chassis Chassis 01: Port Ownership Change ------
kumagai is log
26. SRv6 Leaf2
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
Every 2.0s: show interface counters
LEAF2: Wed Jan 15 09:55:08 2020
IFACE STATE RX_OK RX_BPS RX_UTIL RX_ERR RX_DRP RX_OVR TX_OK TX_BPS TX_UTIL TX_ERR TX_DRP TX_OVR
Ethernet0 U 42,505 874.18 B/s 0.00% 0 79 0 42,489 871.94 B/s 0.00% 0 2 0
ここに入力して検索
18:56
2020/01/15

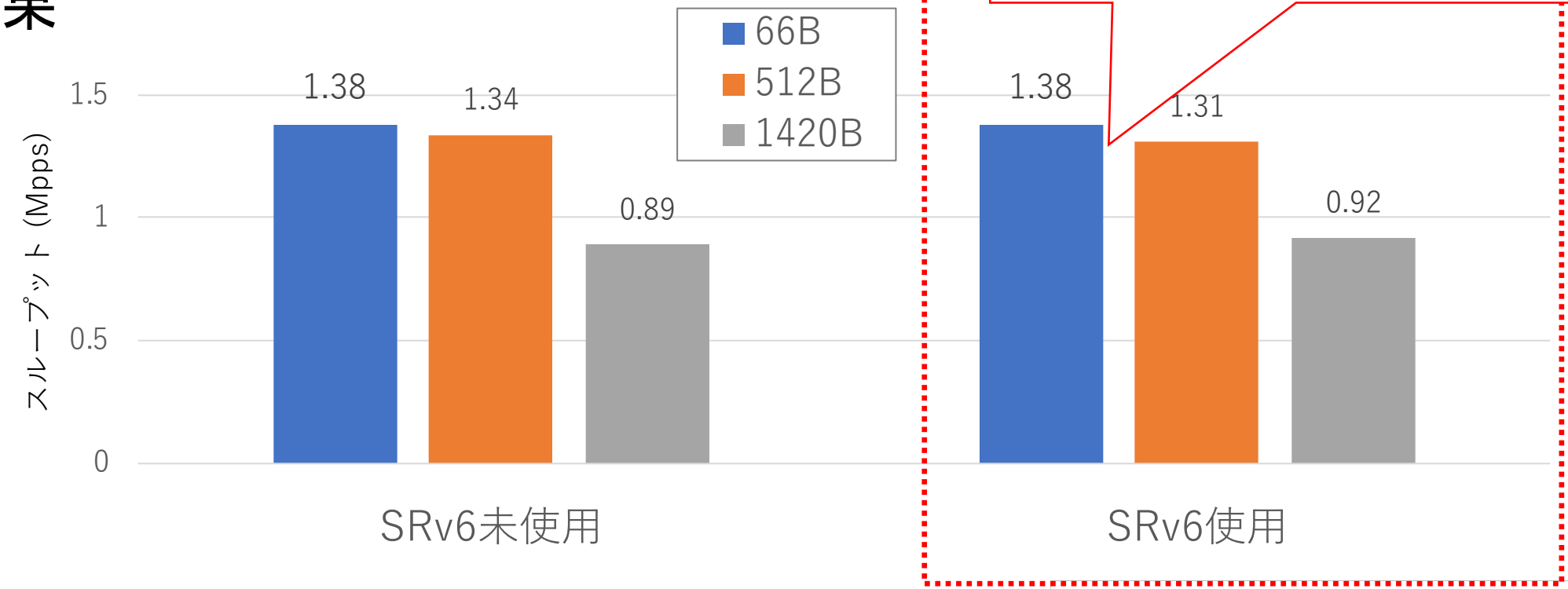
性能測定(スループット)

◆測定条件

- ◇Leaf1に接続した測定器からServer2宛てに2MppsでICMPを送信し、戻りのレートを測定
- ◇SRv6未使用時、使用時それぞれ測定

◆結果

想定よりも性能出ない…
サーバーのチューニングが十分でない可能性
※SRv6処理の影響は見られず、SmartNICの効果は出ている

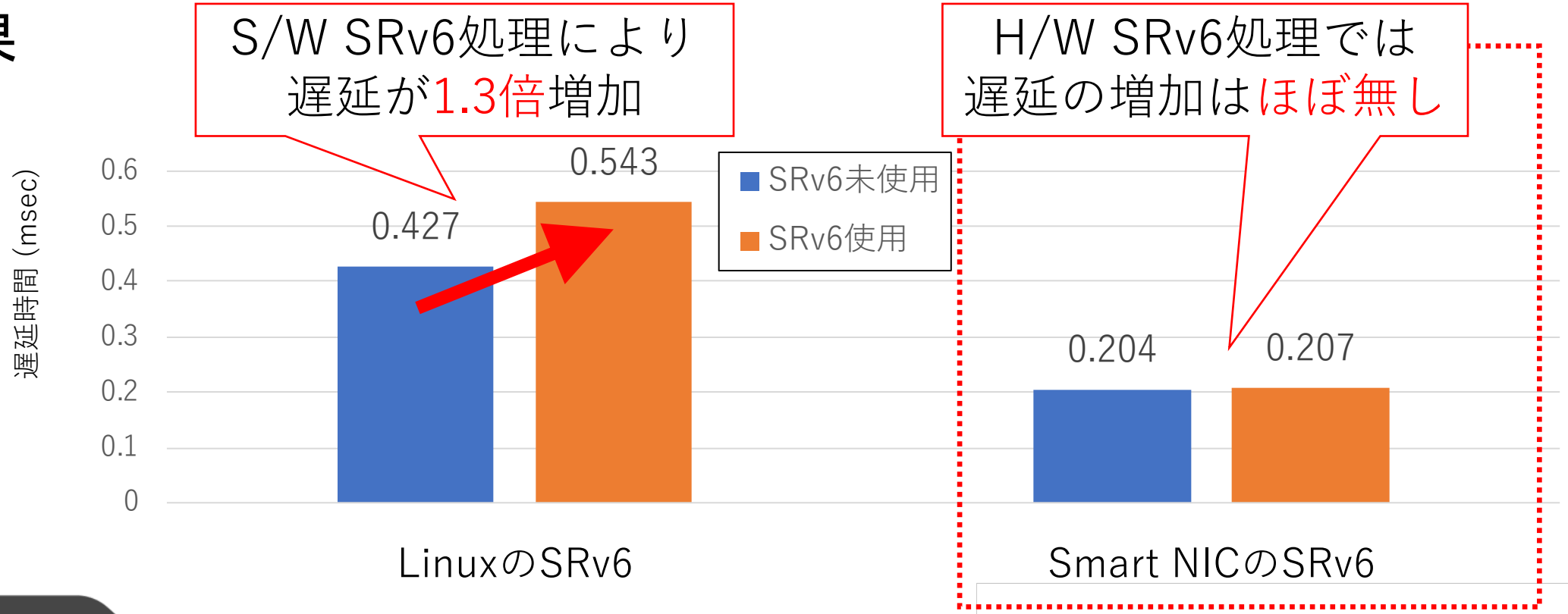


性能測定(通信遅延)

◆測定条件

- ◇Host1-Server2間でICMPv6通信
- ◇Host1からICMPv6 request送信～reply受信までの時間を測定
- ◇比較用に、Host1-Server1(LinuxのSRv6)でも同様に測定

◆結果



まとめと今後の課題

APRESIA SYSTEMS

まとめ

- P4ハードウェアにSRv6のFunctionを実装できた
- **ハードウェアでSRv6サービスチェイニングが動作した！**

- スイッチでのP4 SRv6
  - BarefootデバイスとSONiCにより、IP CLOS fabricとSRv6機能の共存が可能
- サーバー(NIC)でのP4 SRv6
  - 高性能(高スループット、低遅延)、負荷のオフロード等、必要な箇所に対してピンポイントにハードウェアSRv6を導入可能

- P4の実装自体もそこまで難しくない
  - 実装規模はSmartNICで400行程度 (T.Encaps/End.DX6)
  - 要件の整理と設計(テーブルマッチの仕様決め)が重要

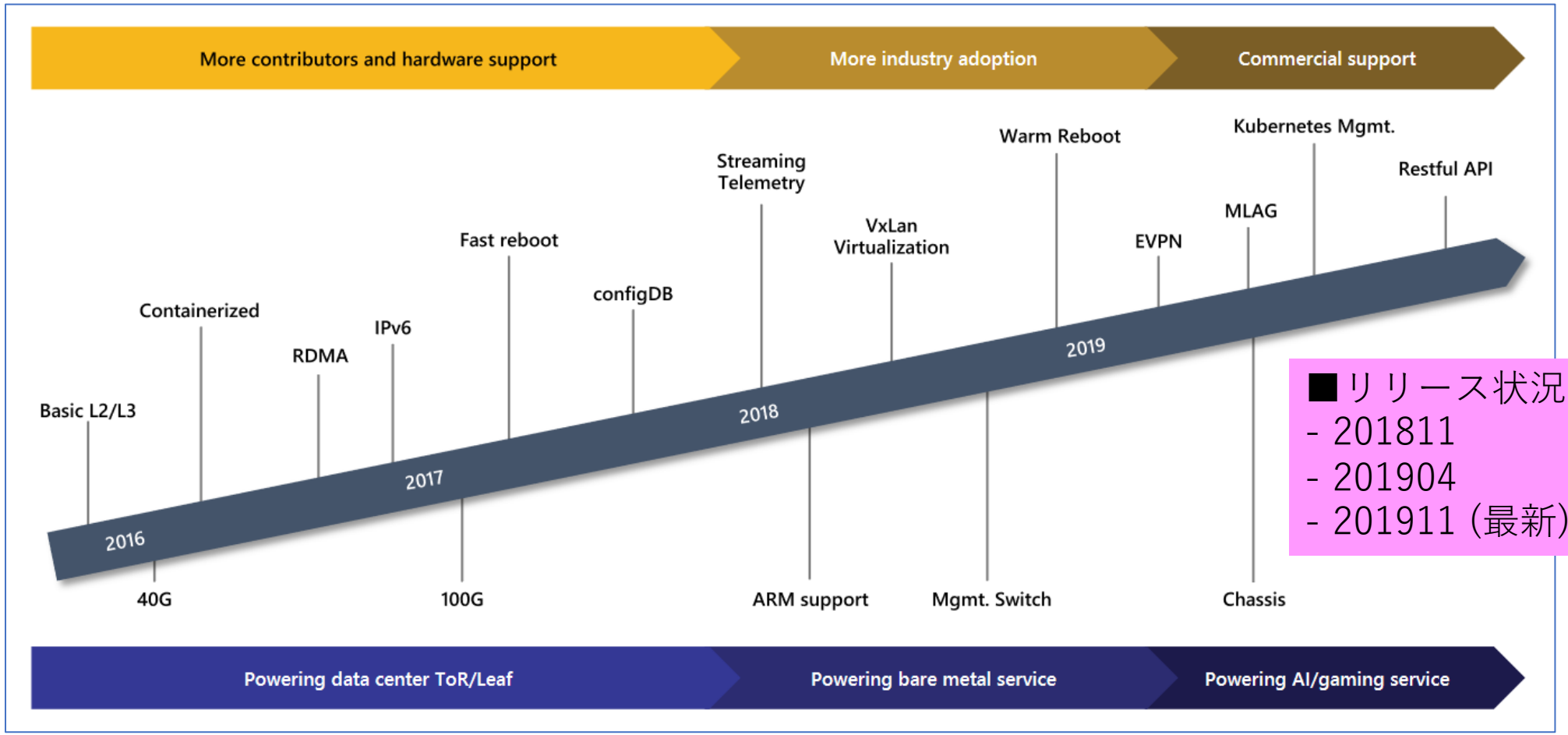
今後の課題

- 他のSRv6 Function実装
  - サービスチェイニングのためのEnd function
  - https://tools.ietf.org/html/draft-ietf-spring-sr-service-programming-01
- サーバーの性能向上
  - 送受信チューニング、DPDKの導入等
  - Smart NIC使用帯域増強 (2x25G, 4x25G)
- Smart NICのP4プログラムをP4_16に変更し、コードをスイッチと共通化

参考資料
SONiC 今後の予定

APRESIA SYSTEMS

SONiCロードマップ



■リリース状況
- 201811
- 201904
- 201911 (最新)

2019 OCP Global Summit “SONiC/SAI and It's Rapid Growing Ecosystem”
https://146a55aca6f00848c565-a7635525d40ac1c70300198708936b4e.ssl.cf1.rackcdn.com/images/f86f93bac8c32db39dd851dcbbe5101c64321d91.pdf

ロードマップ詳細は以下を参照
https://github.com/Azure/SONiC/wiki/Sonic-Roadmap-Planning

201911リリース機能(予定)

- ZTP
- BFD SW 100ms interval from FRR
- NAT
- STP/PVST
- Mgmt VRF
- Multi-DB optimization
- sFlow
- Error handling enhancements
- L2 functional and performance enhancements
- L3 perf enhancement
- VRF
- Platform test
- SSD diagnostic Tolling
- Management Framework
- Platform Driver Development Framework
- Build Improvements
- Dynamic Break Out
- Sub-port support
- Threshold(BST)
- Inband Flow Analyzer
- Debug Framework
- Platform Development Environment
- Build time improvements
- ONIE FW tools
- Egress mirroring and ACL action support check via SAI
- Configurable drop counters
- Core File Manager
- Log analyzer to pytest
- HW resource monitor
- MLAG

https://github.com/Azure/SONiC/wiki/Release-Progress-Tracking-201911