Segment Routing用 Stateful PCEを フルスクラッチで開発した話

2022年06月10日

NTTコミュニケーションズ株式会社 イノベーションセンター
三島 航

自己紹介

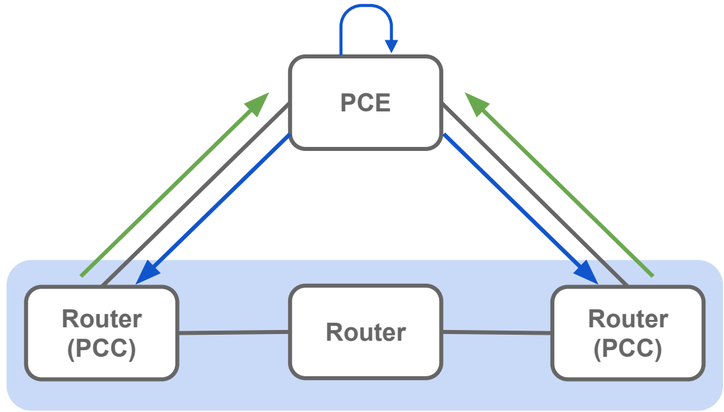
- 三島 航（watal）
  - NTTコミュニケーションズ イノベーションセンター
  - 業務内容はネットワーク全般の技術検証とTestbedの開発・運用
  - Multi-AS Segment Routingアーキテクチャを開発中
    - Keyword: TE / VPN / Service Function Chaining / Multi-AS / EPE / Delay Measurement / Pub/Sub
    - 要素技術: **SR (SRv6/SR-MPLS)** / Telemetry / MP-BGP / **PCEP**
      - **自作Stateful PCE実装を開発＆公開しました。本日はこちらをご紹介します！：https://github.com/nttcom/pola**

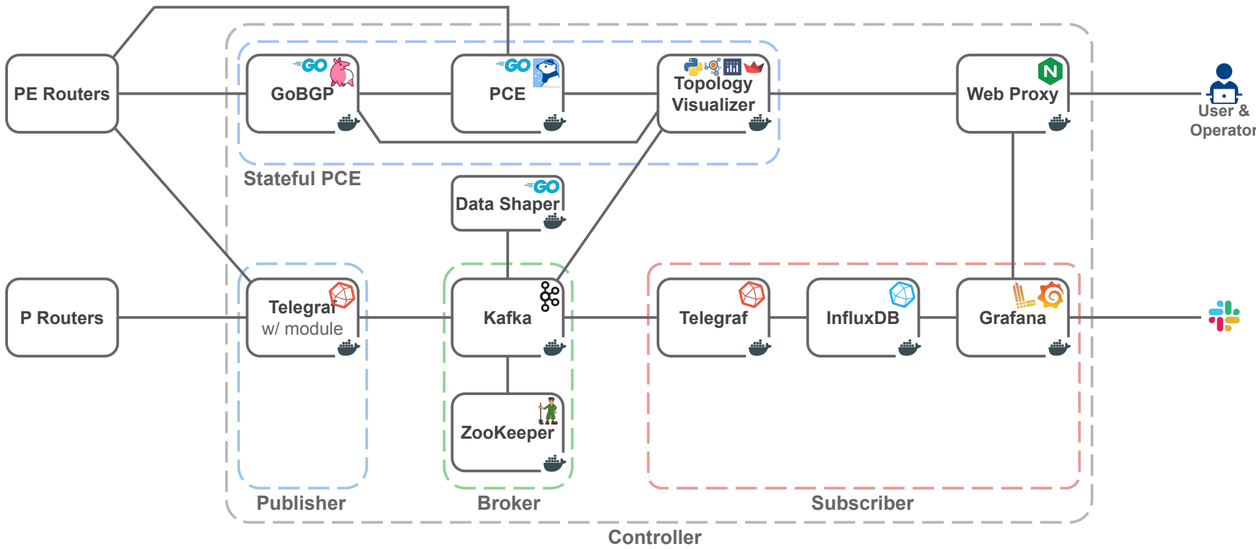
watal_i27e

https://github.com/watal

目次

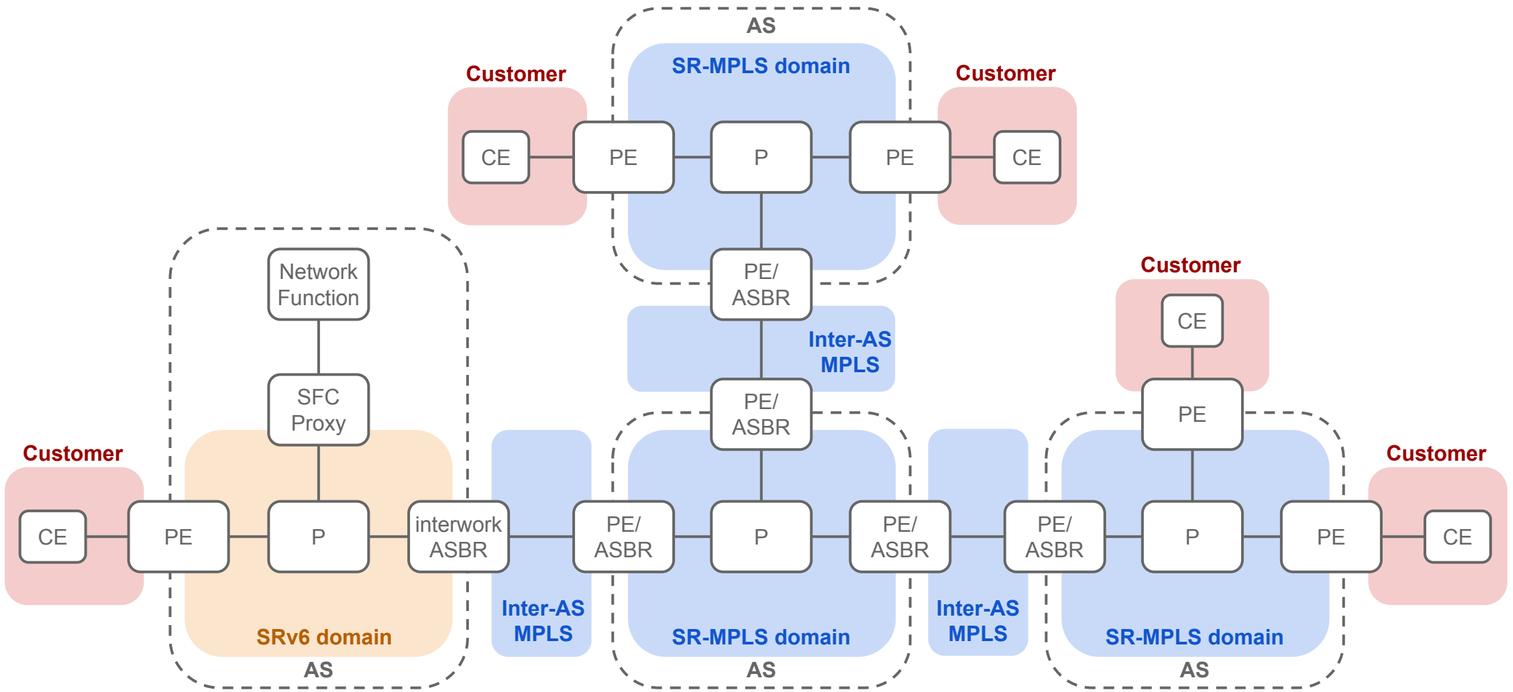
- PCEの導入背景
- PCEとPCEP
  - PCE：概要と種別＆ユースケース
  - PCEP：機能とプロトコル詳細
- 自作Stateful PCE実装 - Pola PCE
  - 実装の紹介
  - 動作デモ
    - Pola PCEによるSR Policy発行
- まとめと今後の展望





PCEの導入背景 – SR網の運用・管理の効率化

- Segment Routingの運用において、経路情報（SR Policy）を効率的に管理したい！
  - 一般的に大規模なSR網では、HeadendとなるPEの数も増大
  - 顧客の要求に応じ、提供するサービスを多様化させたい → SR Policyの数が増加
- それぞれのPEにコンフィグを直接投入する StaticなSR Policy管理は非効率
  - **コントローラによるTraffic Engineering（TE）の集中管理を目指す**

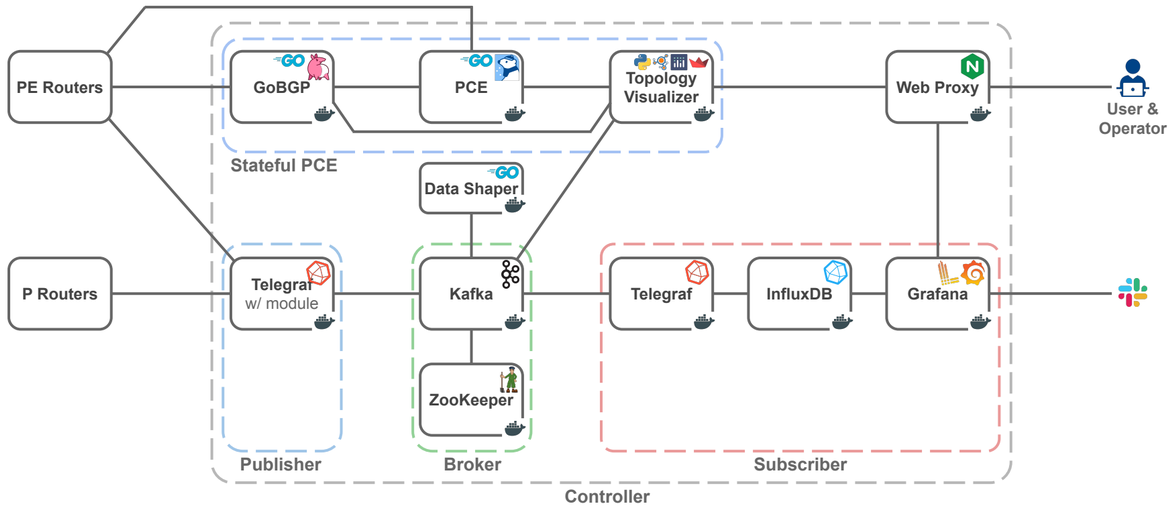


SR Policyの集中的な管理？

- Policyの集中的な管理のため、複数のプロトコルが提案済
  - **PCEP**
    - 本来はMPLS-TEにおいてLSPを管理するためのプロトコル
    - SR-MPLS拡張はRFC8664等で提案済
    - SRv6拡張はDraftで議論中: https://datatracker.ietf.org/doc/html/draft-ietf-pce-segment-routing-ipv6
  - **BGP SR Policy**
    - MP-BGPのNLRI（SAFI 73）としてSR Policyを広告する技術
    - SR-MPLS/SRv6共にDraftで議論中: https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy
  - **NETCONF**
    - SSHを経由し、StaticなSR policyを流し込む手法
    - 制御のため、ベンダや機種ごとに固有のYANGモデルが必要
  - **マルチベンダでの利用と機器への実装状況を考え、今回の取り組みではPCEPを採用**

SRを制御するコントローラを作ろう

- SRの運用改善を実現するためには複数の要素が存在
  - **SR Policyの発行・管理**
    - 発行済経路の可視化・更新・削除・ベンダフリーな経路配布
    - SRv6、Service Function Chaining、Flex-AlgoやEPE対応
    - 遅延最小化・帯域保証など利用者に応じた通信品質の提供
  - **可視化機能**
    - Flex-Algoなどのスライス・サービスメニュー表示
    - FRR/TI-LFAのBackup-path表示、切り替え予測
  - **将来的なMulti-AS対応**
    - Multi-AS SRアーキテクチャの理念である疎結合と相互連携の実現



- マルチベンダにSR Policyを管理可能で、拡張性も高いPCE実装が欲しい → 自作しよう！
  - コントローラの全体像はNTT Tech Conference 2022で発表済:
    - https://speakerdeck.com/watal/da-gui-mo-srwang-falseyun-yong-woxiao-lu-hua-surunetutowakukontororafalsekai-fa-ntt-tech-conference-2022
  - **今回は自作PCE実装部分について解説！**

自作Stateful PCE実装 – Pola PCE

- Go言語によるOSSのStateful PCE実装（https://github.com/nttcom/pola）
  - 発行済みSegment Listの管理・更新 + PCEPライブラリ

- マイクロサービスアーキテクチャに基づいた設計
  - gRPCによるデータ連携
    - GoBGP等のBGP-LS実装や自作Topology Visualizerとの連携を前提に設計

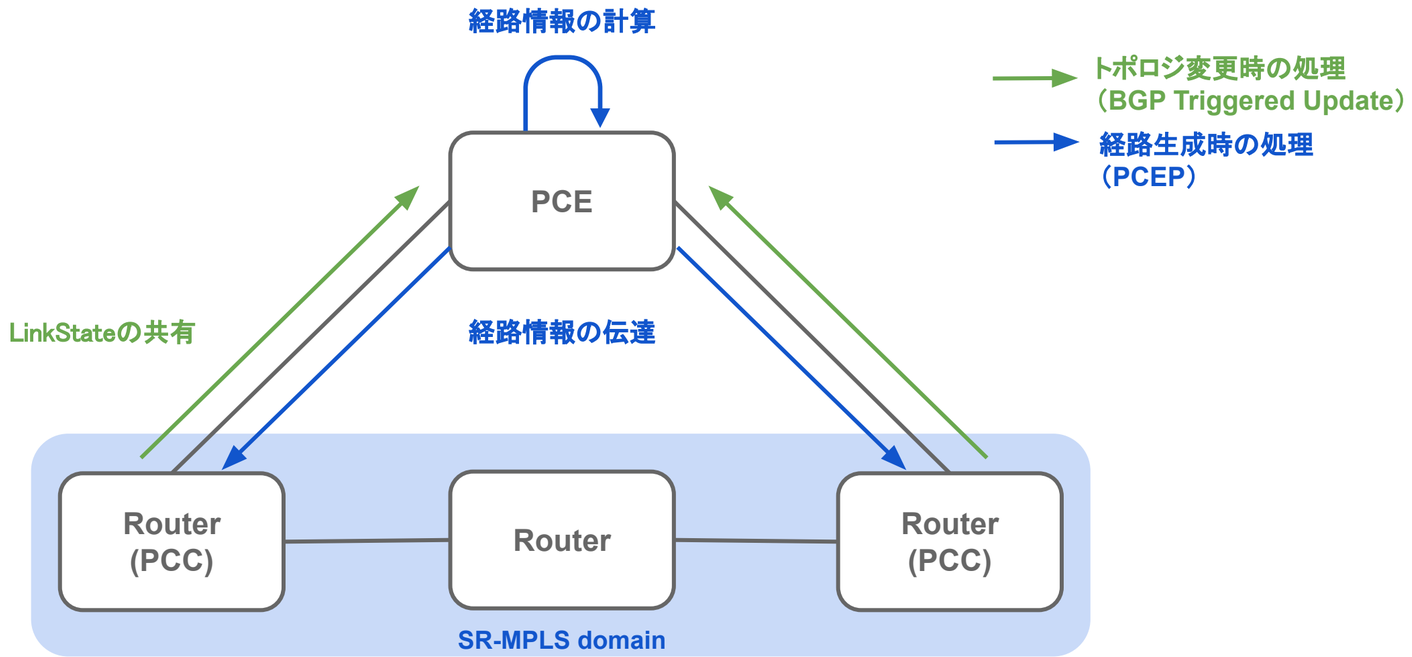
- マルチベンダ対応なPCEP/Stateful PCE実装を0から実装！
  - Multi-AS SR網は3種のベンダ + Linuxで構築中であり、相互接続性を重視
  - 2名体制で開発を実施、実装の半分は新入社員の竹中が担当

- **実装の詳細をご紹介する前に、まずはPCEやPCEPとは何かについて解説していきます！**

PCEとPCEP

Path Computation Element（PCE）

- 中央集権的にTEを実現する計算主体
  - 帯域制御など複雑なポリシーの適用や、発行済みのLSP/SR Policyの管理を実現
    - TE経路はMPLS/RSVP-TEの文脈ではLSPと呼ばれるが、本発表ではSRを扱うためSR Policyと呼称
  - 発行済みのSR Policyを管理しないStateless PCEと、管理するStateful PCEが存在
  - BGP-LSによりLinkStateを認識、PCEPによりSR Policyを配布
    - 経路を受け取るノードはPCC（Path Computation Client）と呼称

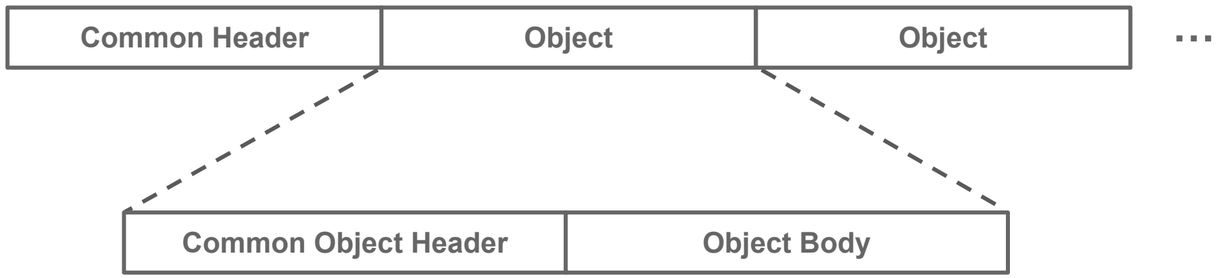


Stateless PCEとStateful PCE

- 発行済みの経路を管理するかどうかで2種類に分類
  - **Stateless PCE**：発行済みのSR Policy等の情報を保持しないPCE
    - PCCからのリクエストに応じ、最適経路を計算＆応答
    - PCEをシンプルな経路計算主体として用いるユースケース
  - **Stateful PCE**：発行済みSR Policyの保持・管理を行うPCE。PassiveとActive の2種が存在
    - **Passive Stateful PCE**：既存のSR Policy情報をもとに経路計算を実施。PCEからは既存SR Policyを更新しない
      - 既存SR Policyや使用済み帯域を考慮することで、PCEを全体最適を行う経路計算主体として用いるユースケース
    - **Active Stateful PCE**：PCCから委任（Delegation）された発行済み経路を操作し、ネットワークの最適化を行う
      - 全体最適に加え、PCEを中央集中的なTEの管理主体として用いるユースケース
  - **今回の取り組みはSR Policyの管理が目的であるため、Active Stateful PCEを採用**

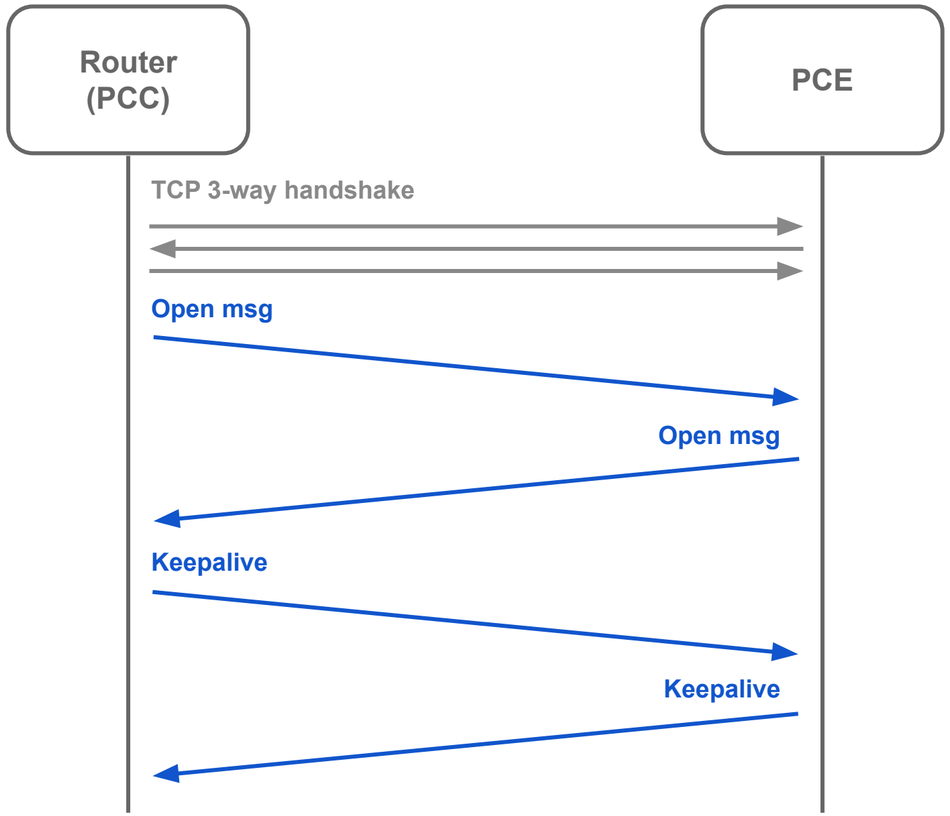
PCEP

- Path Computation Element Communication Protocol (PCEP)
  - RFC 5440, 8231 などで提案された、TCPによるPCE/PCCの通信プロトコル（TCP port 4189）
  - PCEPメッセージはPCEP Common Headerと1つ以上のObjectで構成
    - 各ObjectはCommon Object HeaderとObject Bodyから構成
    - Common Object HeaderのObject-Class＆Object-Typeで識別



- **Object-Class**
- **Object-Type**
- **Flags**
- **Object Length**

PCEP Messageの例



PCC発のInitialization Phase（参考: RFC5440 4.2.1 Figure 1）

（参考）PCEP object

- 役割ごとにさまざまなobjectが存在
  - **ERO**：encodeされた経路を伝達するObject。SRではSegment Listを格納
  - **END-POINTS**：Destination/Source IPアドレスの組を指定
    - 各SegmentはSID + NAI（Node or Adjacency Infromation）の組で指定
  - **LSP**：Stateful PCEの場合に使用。LSP（SR Policy）の状態を伝達
    - PCCで一意なLSPのIDであるPLSP-IDやPolicy名、LSPのUp/Down状態、Sync/DeligateなどPCEに対する状態を持つ
  - **SRP**：Stateful PCE Request Parameters。Stateful PCEの場合に使用
    - PCEの送信した要求とPCCからの応答を紐づけるためのSRP-IDを持つ
  - **LSPA**：LSP Attribute。経路に関するAffinity制約などのパラメータを指定（Exclude-any, Include-any/all）
    - Flex-Algo情報もLSPAに付与する新たなTLVとして議論中
  - **METRIC**：IGP/TEメトリックやホップ数など、経路の様々なパラメータを指定
  - **VENDOR-INFORMATION**：ベンダ固有のobject、例えばIOS XRではColorの独自実装を格納
  - その他OPEN、CLOSE、RP、RRO、NO-PATH、BANDWIDTHなど様々（2022/06/10時点で45種提案済み）
    - パラメータの確認にはIANAのページが便利：https://www.iana.org/assignments/pcep/pcep.xhtml

PCEP messages – Open/Keepalive/Close/PCErr

- **Open**
  - PCEP sessionを確立するために用いられるメッセージ（RFC5440）
- **Keepalive**
  - PCEP sessionを維持するために用いられるメッセージ（RFC5440）
  - Openメッセージが交換された後の応答として送信され、その後は一定のTimerごとに送信
- **Close**
  - 確立したPCEP sessionを閉じるために用いられるメッセージ（RFC5440）
- **PCErr**（Error）
  - エラーの種別や内容を伝えるためのメッセージ（RFC5440、RFC8231等）
  - Push方式で送る場合と、Pull方式で送る場合が存在

PCEP messages – PCReq/PCRep

- **PCReq**（Path Computation Request）
  - 経路計算を要求するPCCからPCEに対して送られるリクエストメッセージ（RFC5440、RFC8231等）
  - RP object（経路計算要求における各条件の優先度を示す）、END-POINTS objectsが必須

- **PCRep**（Path Computation Reply）
  - PCEからPCCに、PCReqに基づく経路計算結果を回答するメッセージ（RFC5440、RFC8231等）
  - RP objectが必須
  - 経路計算要求を満たすことができない場合はNO-PATH Objectを付与

- PCReq/PCRepはActive Stateful PCEでは不要なメッセージであり、現在Pola PCEには未実装
  - Passive Stateful PCE / Stateless PCE等への対応を考慮し、将来的には実装予定

PCEP messages – PCRpt/PCUpd/PCInitiate

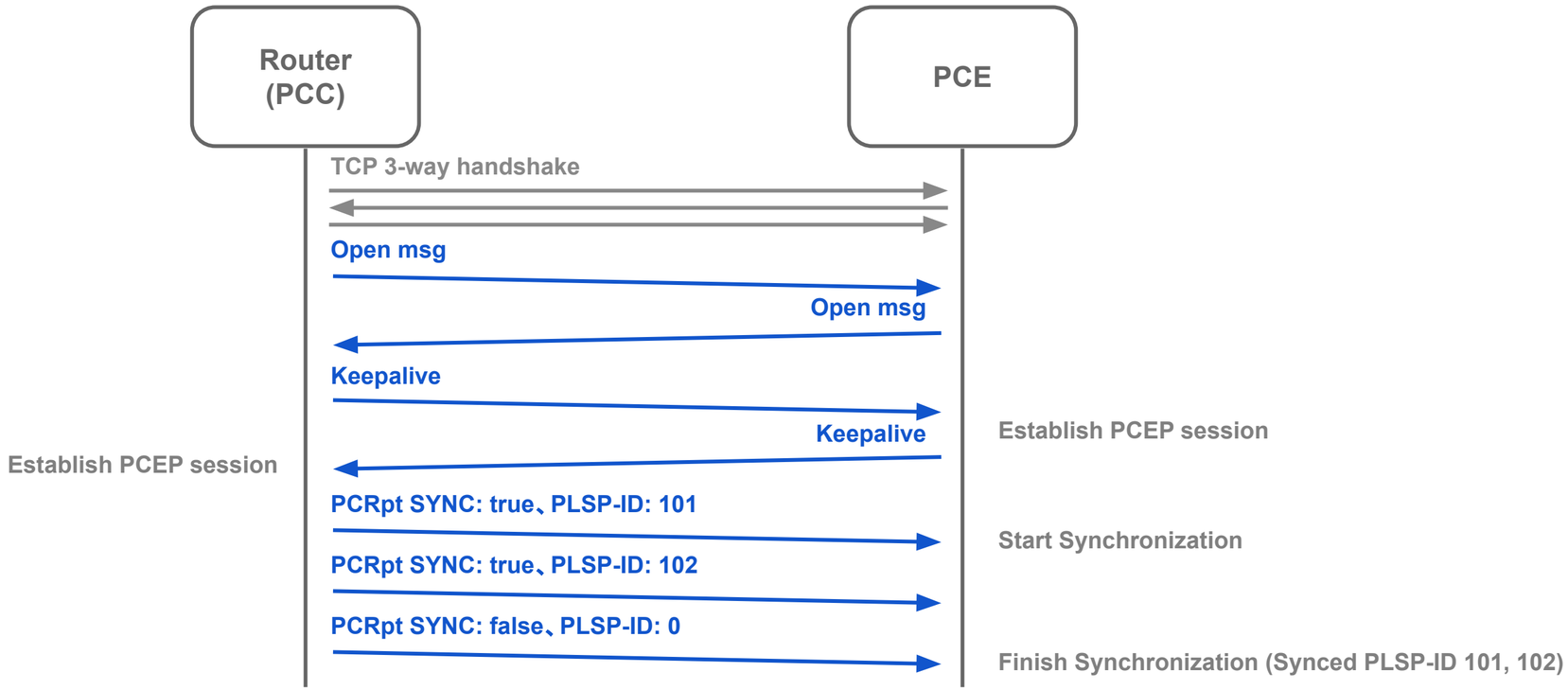
- **PCRpt**（Path Computation LSP State Report message）
  - PCCがPCEに現在のLSP状態を報告するためのメッセージ（RFC8231）
    - PCUpdへの応答としてPCRptを用いる場合はSRP objectを含める
      - この際、PCUpdに含まれるSRP-ID番号を利用して応答することでメッセージを識別

- **PCUpd** (Path Computation LSP Update Request message)
  - PCEがPCCに既存の経路更新を要求するためのメッセージ（RFC8231）
    - SRP object、LSP object、ERO objectが必須

- **PCInitiate** (LSP Initiate Request message)
  - PCEがPCCにLSPの作成または削除を依頼するためのメッセージ (RFC8281)
    - SRP object、LSP objectが必須

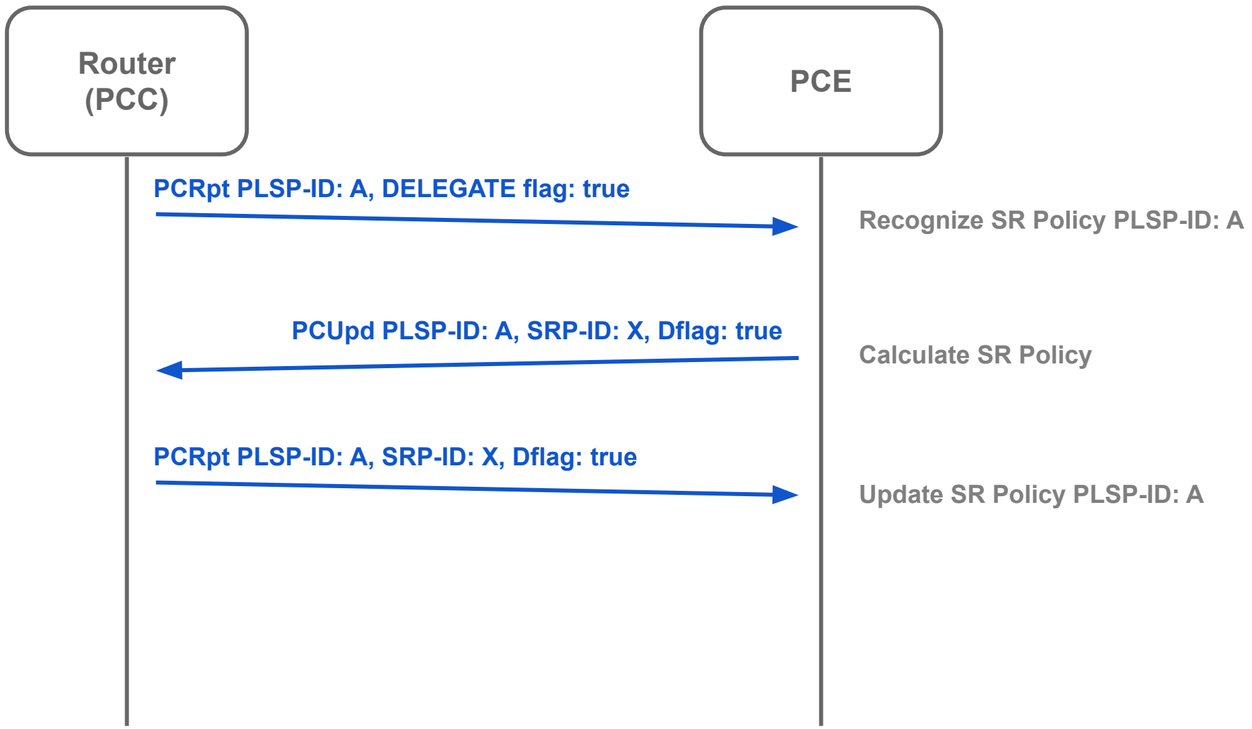
Stateful PCEの通信（Peer構築～SR Policy同期）

- PCC発の3-way handshakeでPCEPセッションを構築する例
  1. PCC - Stateful PCE間で3-way handshakeを交換
  2. Open messageを交換後、Keepaliveを交換することでPCEPセッションを確立
  3. PCCが自らの持つSR policyの情報を、SYNC flagがtrueのPCRptとしてStateful PCEに報告し同期を開始
  4. PLSP-IDが0かつSYNC flagがfalseであるメッセージを送信することで同期を完了



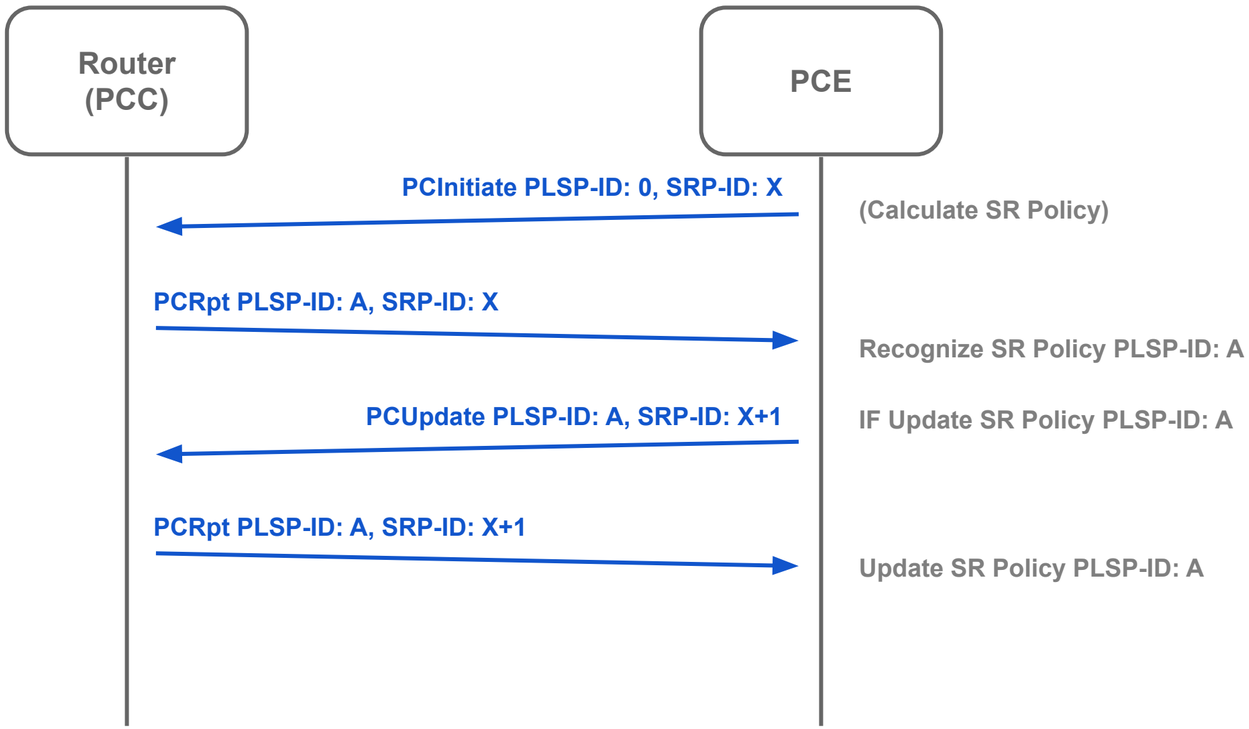
Stateful PCEの通信（SR Policy発行：Delegation）

- PCCがPCEPに委任するSR Policyを作成し、PCRptによりStateful PCEに経路を委任する例
  1. PCCがPLSP-ID: AかつDELEGATE flag（Dflag）がtrueのPCRptをPCEに送信
  2. PCEがDflagがtrueなPCRptを受け取った場合、経路計算を実施し、結果をPCUpdとして送信
  3. PCCはPCUpdに基づきSR Policyを更新後、最終的なSR Policy情報をPCRptでPCEに送信



Stateful PCEの通信（SR Policy発行：Initiate）

- PCEが経路を生成し、PCCに送る例
  1. Stateful PCEはSR Policyを発行し、PCInitiateを利用してPCCに送信
  2. PCCは受信したSegment Listを登録すると共に、PLSP-IDをローカルに生成し、PCEに送信
  3. もしPCEが当該のSR Policyを再度更新する場合、SRP-IDをインクリメントし、PCUpdを送信

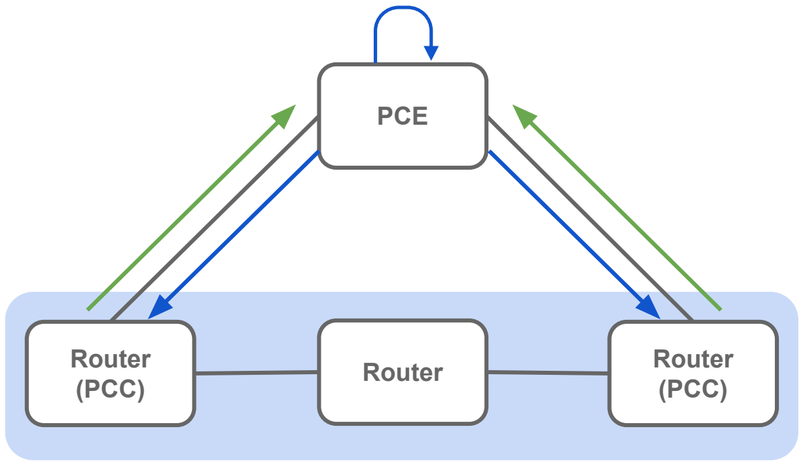


ここまでのまとめ – PCEとPCEP

- PCEの導入により、SR Policyの集中管理が実現可能
  - 発行済 SR policyや網の運用ポリシーを一元管理可能
    - → SR網のスケーラビリティ向上

- 発行済み経路のStateを持つかどうかにより、Stateless PCEとStateful PCEが存在
  - Stateful PCEを採用することで、帯域保証やLSPの管理、全体最適化などが実現可能
  - PCE側発の経路更新を行うActive Stateful PCEと、行わないPassive Stateful PCEが存在

- PCE-PCC間の通信はPCEPを採用
  - TCPセッションを利用したPCEのコミュニケーションプロトコル
  - HeaderとObjectによるシンプルなメッセージ構成
  - 役割に応じた複数のPCEPメッセージが存在



自作Stateful PCE実装 – Pola PCE

Pola PCE（再掲）

- Go言語によるOSSのStateful PCE実装（https://github.com/nttcom/pola）
  - 各種PCEP Message実装（OPEN / Close / PCRpt / PCUpd / PCInitiate / PCError）
  - 発行済みSegment Listの管理・更新
  - Goの強みを生かした非同期処理

- マイクロサービスアーキテクチャに基づいた設計
  - gRPCによるデータ連携
    - GoBGP等のBGP-LS実装や自作Topology Visualizerとの連携を前提に設計

- マルチベンダ対応なPCEP/Stateful PCE実装を0から実装！
  - Multi-AS SR網は3種のベンダ + Linuxで構築中であり、相互接続性を重視
  - 2名体制で開発を実施、実装の半分は新入社員の竹中が担当

Pola PCEのライブラリ構成

- polad/polaの2つのコマンドとPCEPライブラリ
  - **cmd/polad**: Stateful PCE daemon
    - 実行すると、confファイルの設定通りにStateful PCEを起動
  - **cmd/pola**: CLIで実行するPCEの状態確認コマンド
    - PCEP peerの一覧、発行済みSR Policy一覧の確認、SR Policyの発行（PCInitiate）
  - **pkg/packet:** Goでimportして使用可能なPCEPライブラリ
- その他サンプル
  - **examples/sr-mpls_l3vpn**: Tinetを用いたサンプルネットワーク
    - 実行するとSR-MPLS + VPNv4のコンフィグが投入されたFRRoutingによるサンプルネットワークが起動
  - **utils/grpc**: GoやPythonによるシンプルなgRPCクライアントの作成例

（参考）Pola PCEの利用形態

- Pola PCEは3種類の使い方を想定
  - polaコマンド等からの、gRPC経由での設定確認・投入
    - **実装済み**。あらかじめpoladを立ち上げておき、外部からgRPC経由でSR Policy追加・確認・更新を実施
  - poladコマンドによる初期値付きでのコンフィグ投入
    - **未実装**。polad.yamlに初期値としてSR Policyを設定しておき、立ち上げ直後にPC Initiateを実行
  - Native PCE Libraryとしての呼び出し（c.f. GoBGP）
    - **未実装**。PCEの各イベントをハンドラから参照できるようにし、Pola PCEをGo言語から直接活用可能

Pola PCEの使い方 1/2（polad - Stateful PCE起動）

```
## polad/polaの一括インストール
$ go install github.com/nttcom/pola/cmd/…@latest

## polad.confの作成
$ vi polad.conf
---
global:
  pcep:
     address: "192.0.2.1"
     port: 4189
  grpc:
     address: "192.0.2.1"
     port: 50051
  log:
     path: "/var/log/pola/"
     name: "polad.log"

## poladの起動
$ sudo polad -f polad.yaml
2022-06-05T22:57:59.823Z        info    gRPC Listen  {"listenInfo": "192.0.2.1:50051", "server": "grpc"}
2022-06-05T22:57:59.823Z        info    PCEP Listen  {"listenInfo": "192.0.2.1:4189"}
```

Pola PCEの使い方 2/2（pola – SR Policy発行）

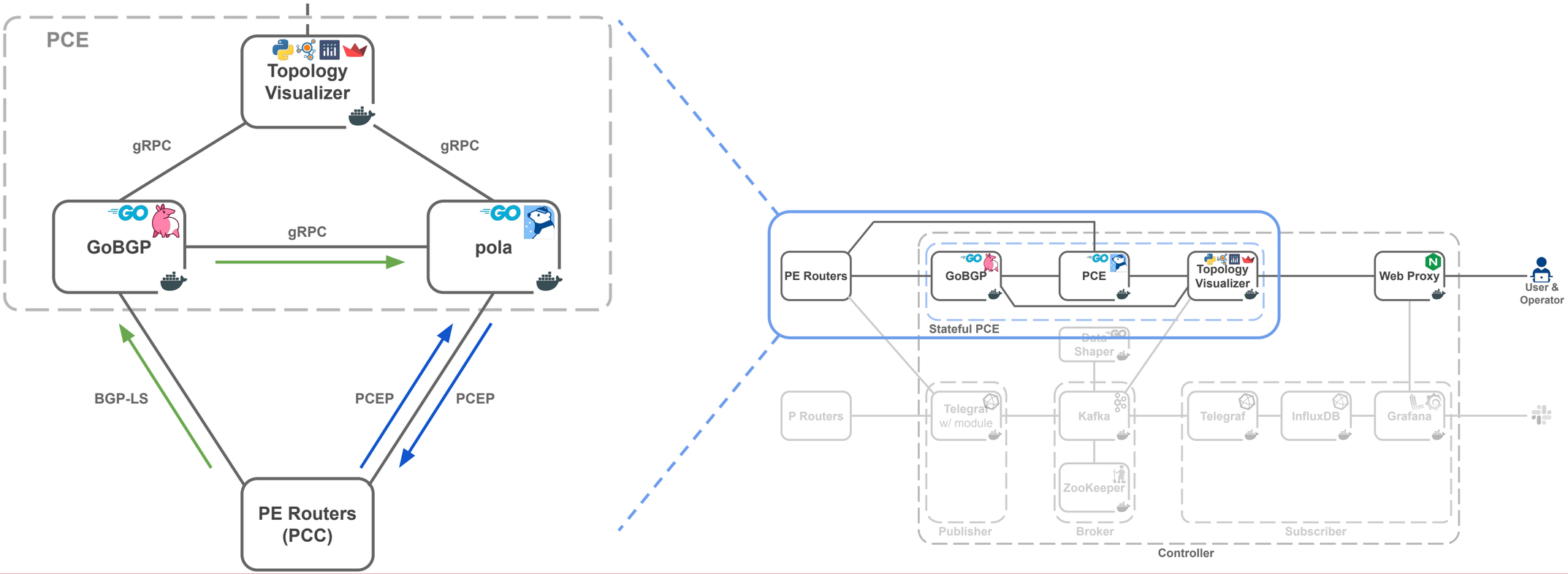
```
## polaによるPCEPセッション・SR Policyの確認
$ pola session
$ pola lsp list

## srpolicy.yamlの作成
$ vi srpolicy.yaml
---
srPolicy:
    name: name
    peerAddr: 10.0.255.1
    srcAddr: 10.255.0.1
    dstAddr: 10.255.0.3
    color: 1
    segmentlist:
        - sid: 16002
          nai: 10.255.0.2
        - sid: 16004
          nai: 10.255.0.4
        - sid: 16003
          nai: 10.255.0.3

## polaによるSR Policyの投入
$ pola lsp add -f policy1.yaml
```

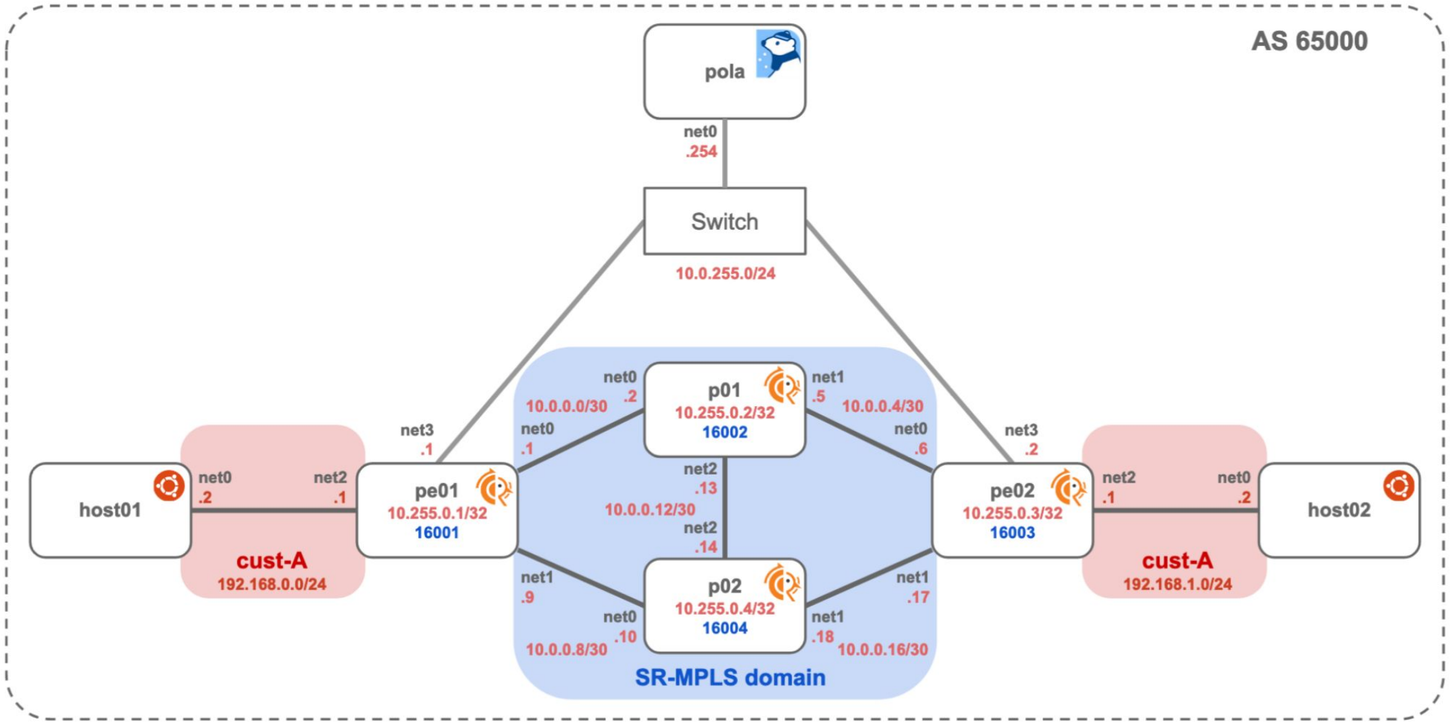
(参考)マイクロサービスの活用例

- gRPCを介して、Topology Visualizer・GoBGP・その他クライアントApp等と連携
  - GoBGP: BGP-LSをPE RouterやRRと交換
  - Pola PCE: BGP-LSを取得しLinkStateを把握、PCEPに基づいてSR Policyを発行
  - Topology Visualizer: BGP-LSによるトポロジ可視化、PCEと連携したSR Policyの発行、TE Pathの可視化



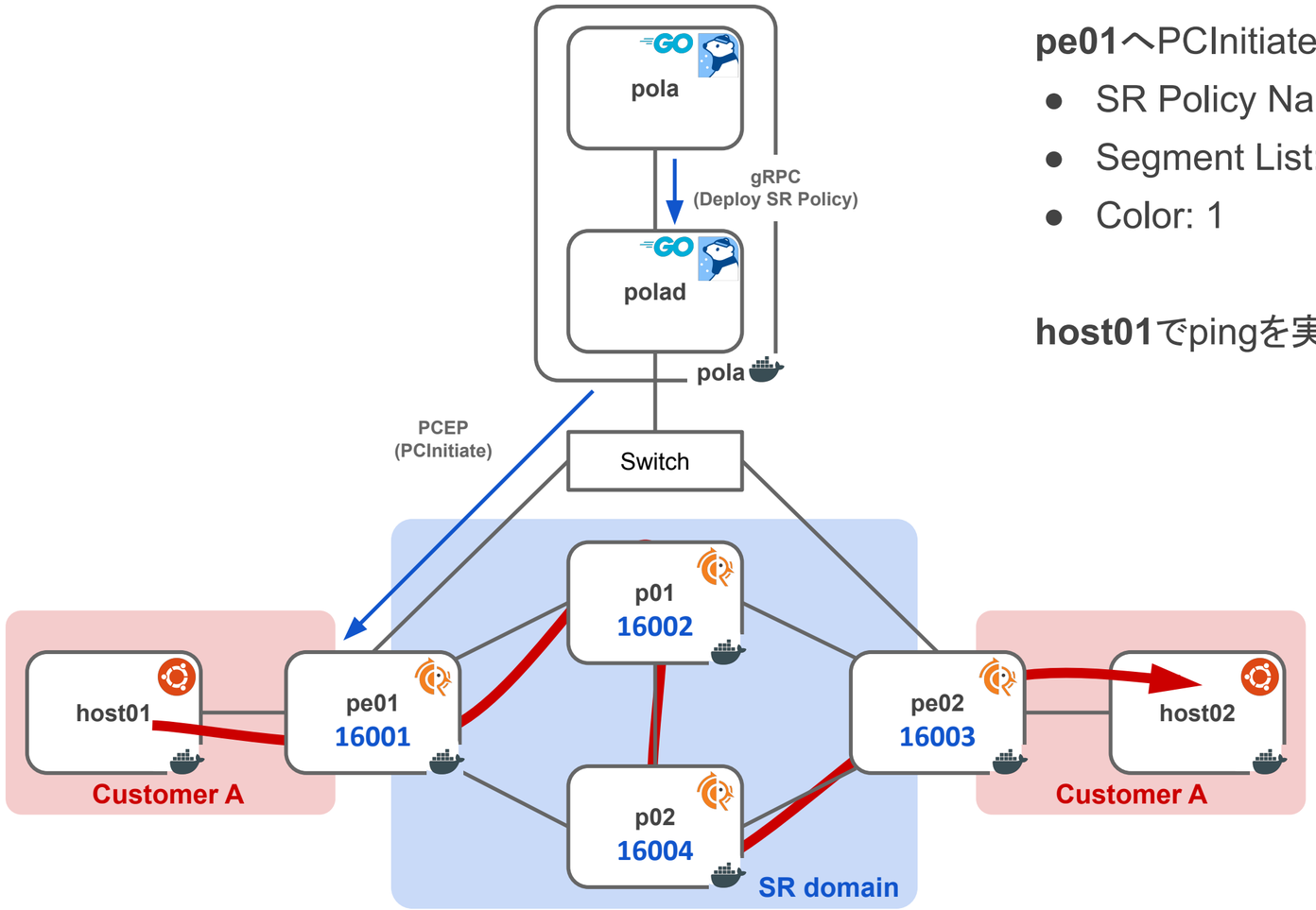
Pola PCEのSample Network

- 試験用Neworkを即座に入手し、誰でも検証を行うことのできるサンプルファイル
  - tinynetwork/tinetを利用。SR-MPLS + L3VPNのネットワーク環境が数分で構築＆起動
    - https://github.com/nttcom/pola/tree/main/examples/sr-mpls_l3vpn
  - コンテナであれば組み込めるため、例えばspec.yamlを編集することでcRPDなども検証可能



デモ：Pola PCEによるSegment List投入

- デモ環境を使用し、FRRoutingへSR Policyを投入



pe01へPCInitiateを用いて下記のSR Policyをデプロイ

- SR Policy Name: enog_policy1
- Segment List: **16002/16004/16003**
- Color: 1

host01でpingを実行、**p01**でパケットキャプチャし、SR Headerを確認

Future Work

- Pola PCEの機能拡充
  - 経路計算機能の追加、Flex-AlgoやDelay Metric、帯域保証など
  - Multi-AS対応

- 各種SRv6対応
  - Pola PCEに機能を追加し、Multi-AS SR内のSRv6網上で検証を実施
  - Underlay IPv6 Onlyなども検討中

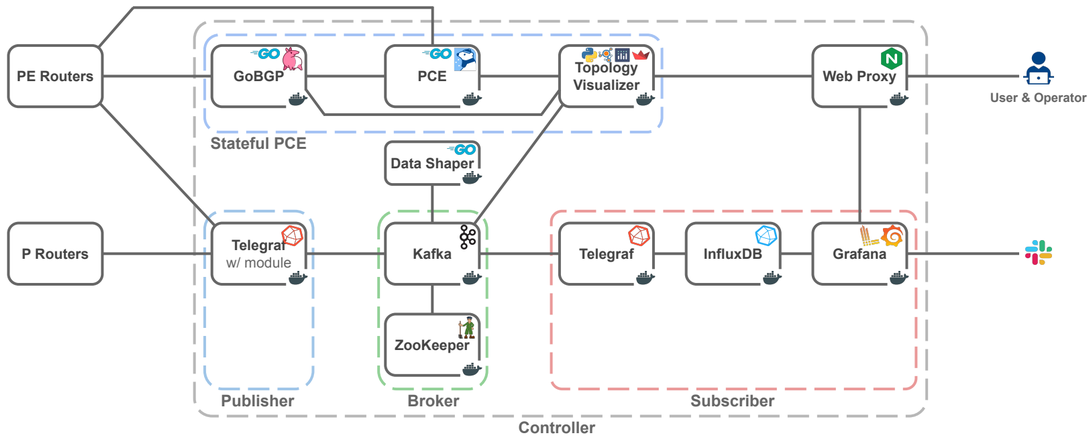
- VPN/TE/SFCサービスの更なる付加価値向上
  - SFCの活用、Web GUIからのChain選択
  - 顧客ごとのSlicing機能の追加
  - 可視化ツールとの連携、Segment List/SFC/Slice/FastReRouteなどの可視化、TWAMP測定結果等の表示

まとめ

- PCE/PCEPの概要について解説
  - Stateful PCE：SR Policyの一元管理とPCInitiateによるPush型のSR Policy発行が可能
  - PCEP：PCC-PCE間の通信プロトコル。SR Policyの同期や更新を実現

- SR網の効率的な管理を目指し、自作Stateful PCEを開発＆OSSとして公開
  - 発行済みSR Policyの管理＆更新を中央集中のコントローラから実現可能に
  - 是非お手元の環境でご活用いただき、機能要望/Issue、PRなどいただけると嬉しいです

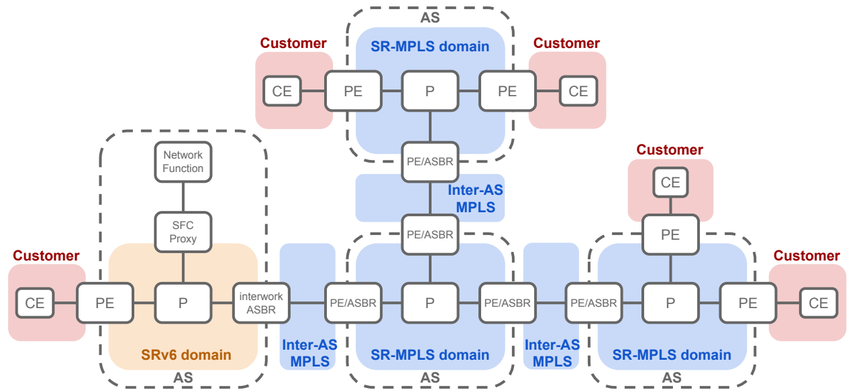
- **より効率的なSR網の運用・管理を目指し、今後も開発・検証を続けていきます！**



最後に...

- Multi-AS SRの取り組みについて
  - **JANOG 50でMulti-AS SRアーキテクチャの内容と検証・運用結果を発表予定！**
    - 日時: 2022年7月14日(木) 15:15 - 16:00
    - 場所: 武道館A＋B
  - **NTT Com Engineers' Blogにて Multi-AS SR + L3VPNの連載記事を執筆中！**
    - https://engineers.ntt.com/
    - まずは6/13に第0回（導入）・第1回（Single-AS L3VPN）の記事を公開予定

- **一緒に開発業務に取り組む仲間を募集中！**
  - 大規模なネットワーク開発、Multi-AS SRアーキテクチャに興味がある方
  - コントロールプレーン・データプレーン技術の開発に興味がある方
  - ぜひ一緒に最先端ネットワークの技術開発に取り組みましょう！
    - ジョブ型採用: https://hrmos.co/pages/nttcom0033/jobs/1692786
    - インターンシップ予定等も順次公開予定

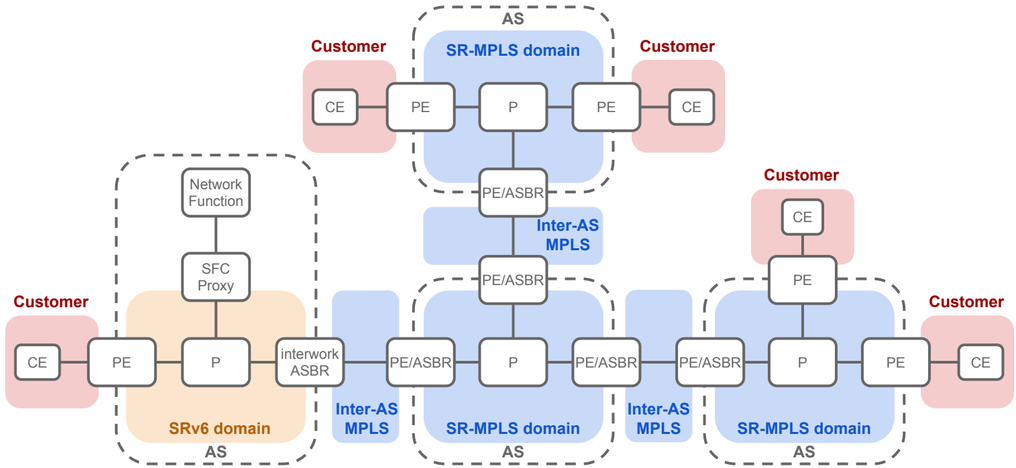


以降は質疑用スライド

Multi-AS SRの運用における課題

- NTT Com独自のMulti-AS Segment Routingアーキテクチャを考案、運用中
  - マルチASかつマルチベンダにおけるSR-MPLS TEの実現: https://mpls.jp/2021/presentations/20211101_MPLSJAPAN_NTTCOM_tajima_pub.pdf

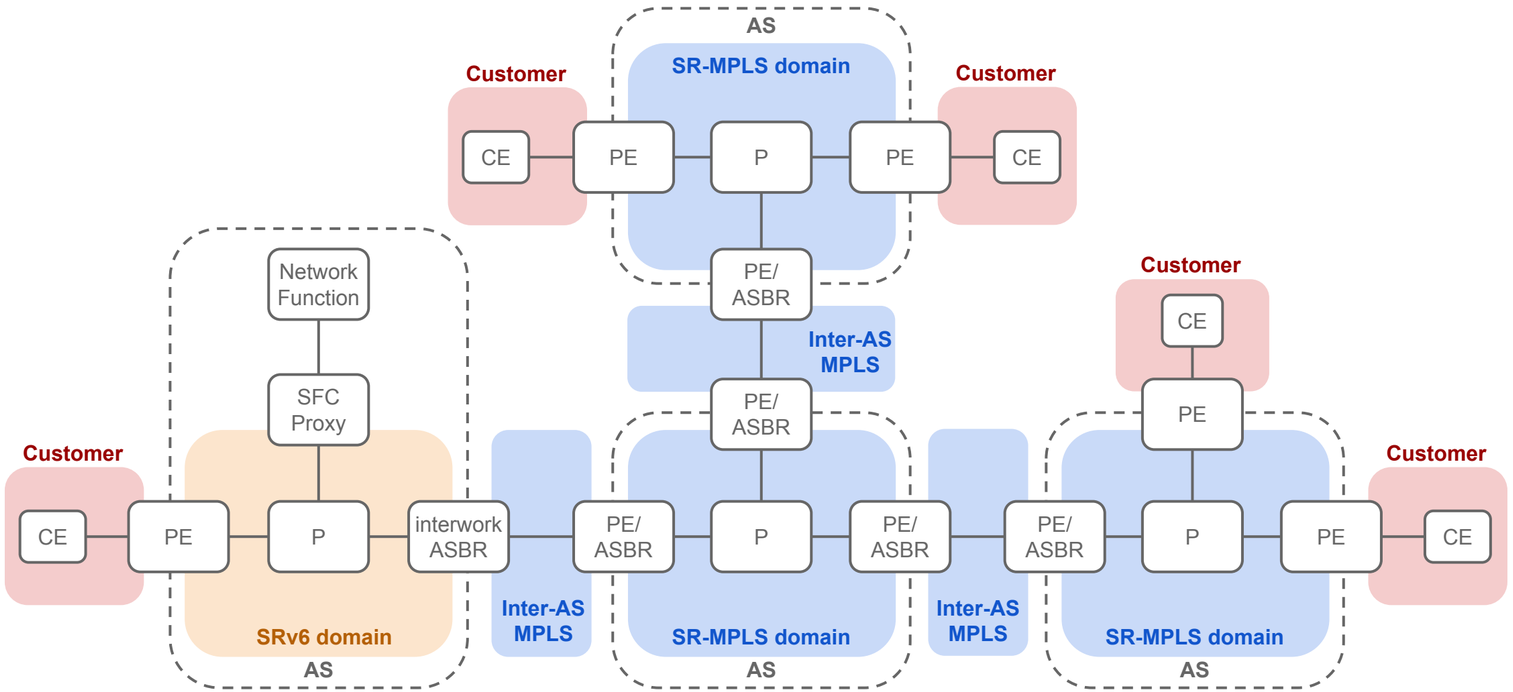
- 大規模SR網には多くの特長があるが、運用課題も存在
  - **ネットワーク状態の把握や効率的な経路管理が困難**
    - トポロジ、Interface状態、コンフィグやログを一元管理したい
    - 経路・Backup-PathやSFCを直感的に把握したい
  - **利用者の増加に伴う管理の煩雑化や作業時間の肥大化**
    - 上記の解決のため、以前にユーザによるセルフマネージ化を提案
      - ユーザによるセルフマネージ可能なネットワークサービス運用システムの事例: https://onic.jp/_cms/wp-content/uploads/2019/11/onic2019_tajima.pdf
    - 利用者自身がサービスを直感的に理解し、権限の中で自ら選択



→ 運用効率化や検証業務改善のため、可視化とネットワーク制御を実現するコントローラが欲しい！

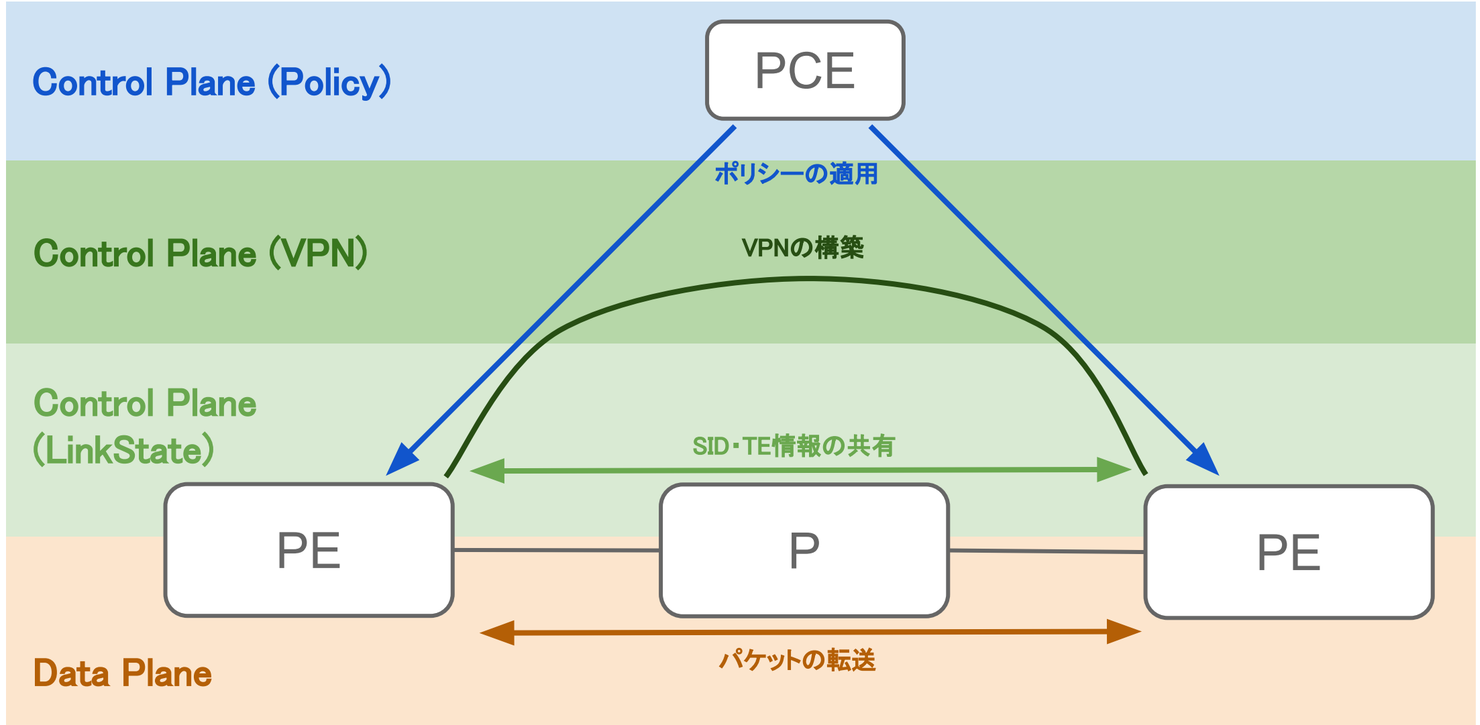
Multi-AS Segment Routing

- 運用者・NW種別・拠点などの管理単位でASを分離 & 疎結合にVPN/TE/SFCを連携
  - **高いスケーラビリティ・シンプルさ・マイグレーション能力というSRの利点を強化**
  - SRv6とSR-MPLSの相互接続や、**SR-MPLSへのSFC提供も実現**
  - Flex-Algo / EPE / Delay Measurement / TI-LFAなどの先端技術を検証し導入、マルチベンダに実現
  - **ASを超えた大規模なネットワーク提供や付加価値向上を実現！→ どうやって管理しよう？**



データプレーン/コントロールプレーン

- ネットワークの持つ機能を役割ごとに分離し扱う概念
  - データプレーン：転送機能
  - コントロールプレーン：経路共有機能。SRではさらに3種類の役割に分離（以下はNTT Com独自の分類のため注意）
    - **Control Plane (Link State)**　：オーバレイの構築（Node SIDとTE機能の共有）
    - **Control Plane (VPN)**　：オーバレイ上のVPNの構築（VPN経路とVPNラベルの共有）
    - **Control Plane (Policy)**　：ポリシーの適用（Segment Listの作成と配布）



役割：経路の設定（Segement Listの作成と配布）
プロトコル：BGP-LS（トポロジ吸い上げ）
PCEP（SegmentListの適用）

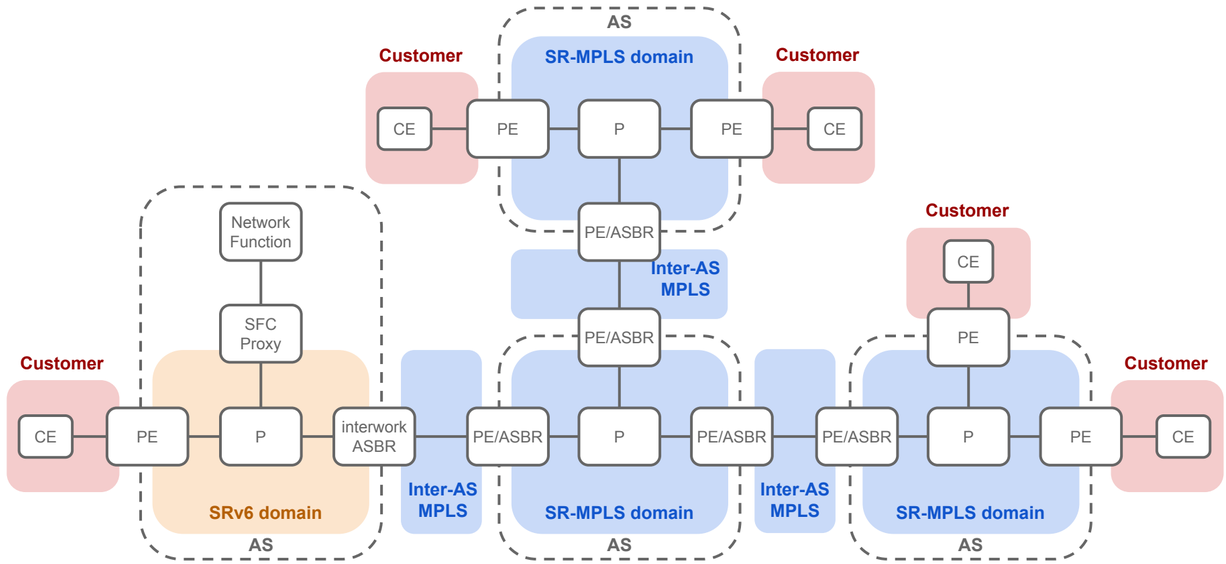
役割：オーバーレイ上のVPNの構築
（BGP Prefix-SIDの広告）
プロトコル：BGP（VPNv4/VPNv6/EVPN等）

役割：オーバーレイの構築
（Node SID/Adjacency SIDの伝搬）
プロトコル：IS-IS, OSPF, BGP-LS（Multi-AS等の場合）

役割：パケットの転送
プロトコル：MPLS, IPv6

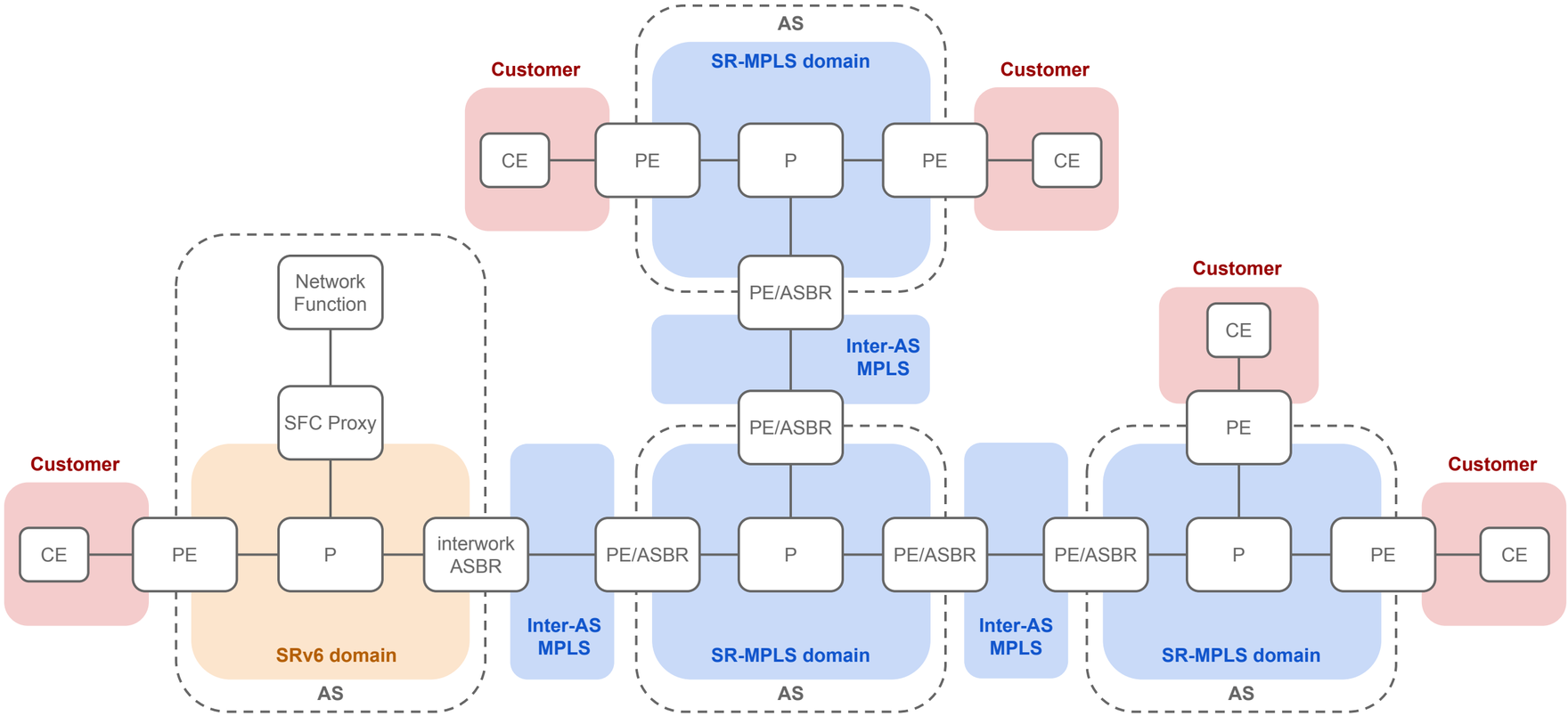
Multi-AS Segment Routingアーキテクチャ

- 社内網でSegment Routingを活用中。更なる発展を目指してMulti-AS SRアーキテクチャを提案
  - スケーラビリティ・マイグレーション能力・シンプルさを向上
    - 運用者・用途・地理的要因などに基づき、ネットワークを自由に分割可能
  - SR-MPLSへのSFC提供など、更なる付加価値提供を実現
    - ASを超えたサービスの相互提供が可能！
  - マルチベンダでの相互接続検証や応用的な技術についても検証中
    - C-Plane/D-Planeのマルチベンダ接続
    - Flex-Algo / Delay Measurement / EPE / TI-LFAなど



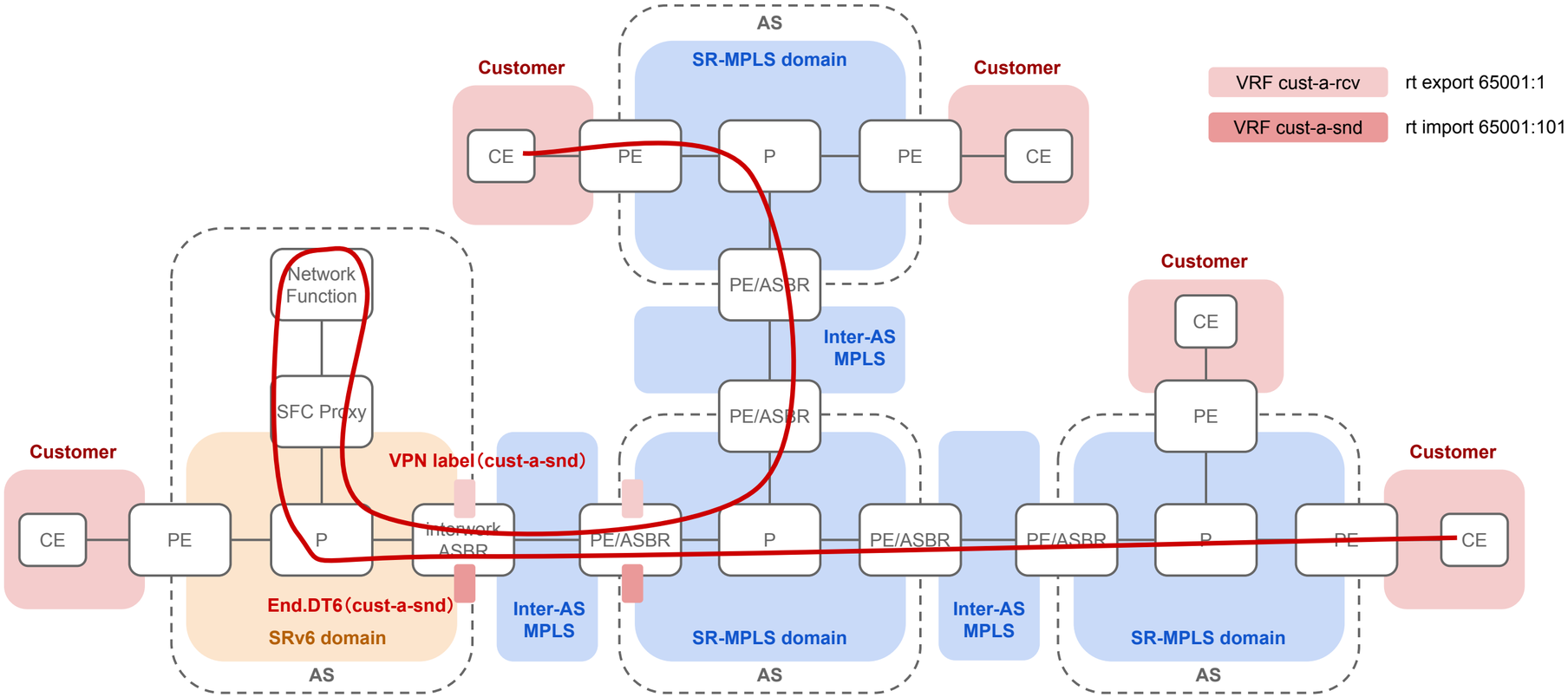
特長① 高いスケーラビリティ

- ASごとの分離・疎結合な連携による高いスケーラビリティ
  - Inter-AS OptionB（RFC4364）によりASを超えたEnd-to-EndなVPNを提供 ＆ TEはASごとに提供
  - D-Plane/C-Plane共に追加のプロトコルは不要
  - **スケーラビリティを向上させると共に、ASを分離したシンプルな運用が可能**



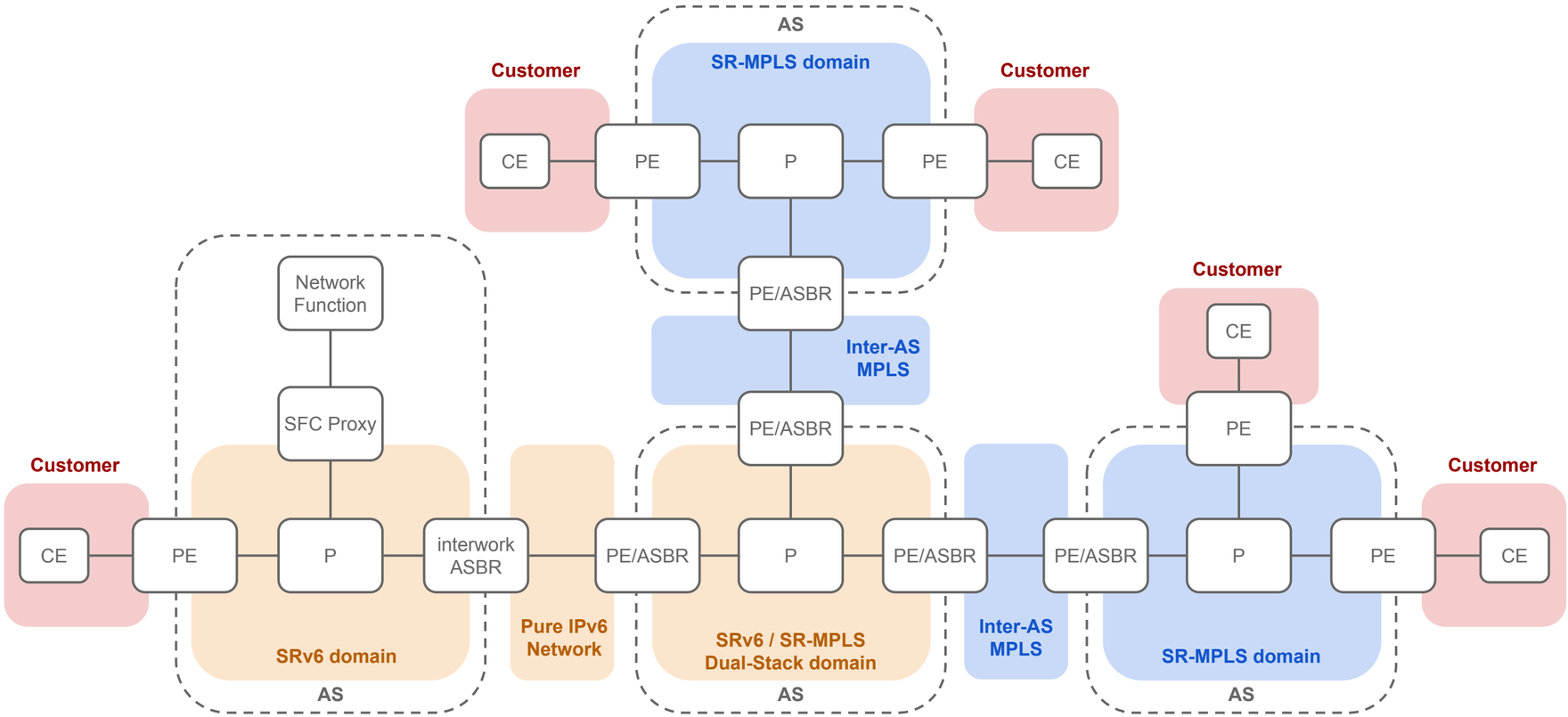
特長② SRv6/SR-MPLS相互接続による付加価値提供

- SR-MPLSとSRv6で一連のVPN/TEを実現
  - 既存リソースの有効活用やサービスの相互提供
- SR-MPLS網にもSFCによる付加価値を提供
  - Inter-AS部分のルーティング面分割によりSR-MPLS - SRv6折り返しを実現。SR-MPLS同士のVPNにもSFCを提供



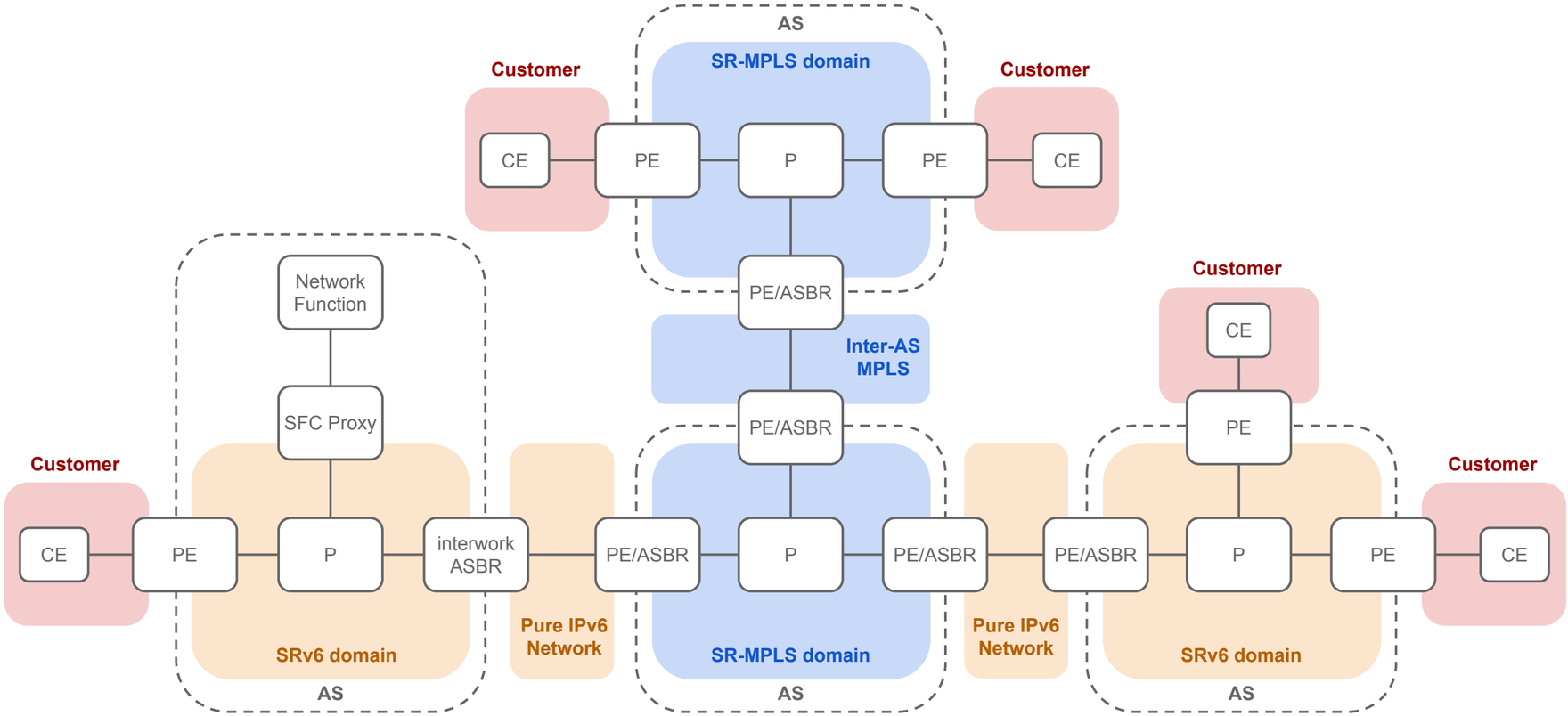
特長③ SR Migration（1/2）

- UnderlayやOverlayサービスへ影響を与えることなく、一部のASのみの移行が可能
  - SRv6 onlyなネットワーク設計をウォーターフォールとするとアジャイルの発想
    - 実装が早く、ある程度枯れたSR-MPLSでサービスを提供＆必要な部分にのみ先端的なSRv6の機能を導入可能
    - 多くの機器ではSR-MPLS→SRv6の移行はコンフィグ変更のみで実現可能であり、移行が容易



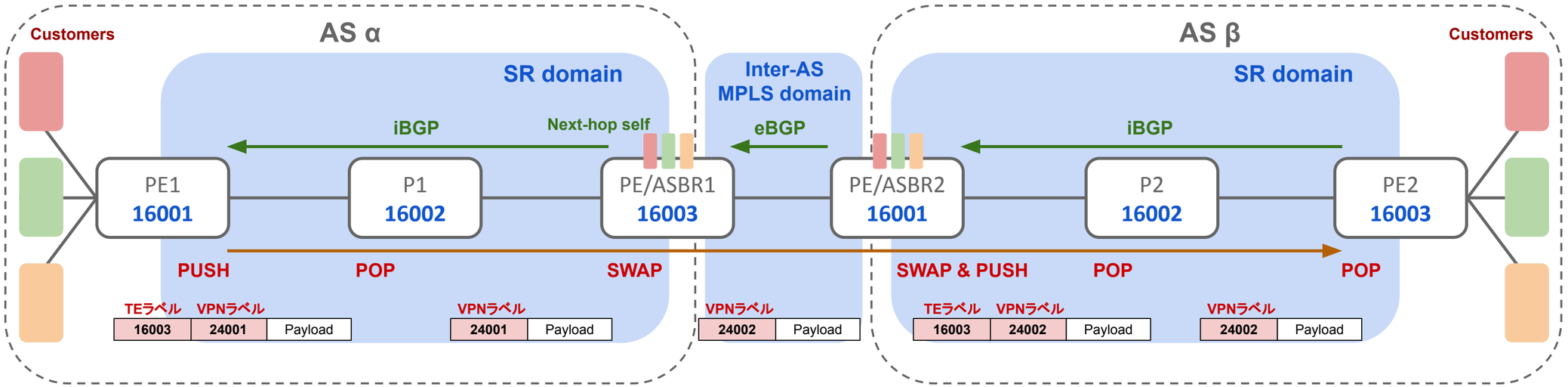
特長③ SR Migration（2/2）

- 好きなASを自由なプロトコルで構成可能
  - UnderlayをIPv6で構成しておくことにより、離れたSRv6網同士の接続なども可能
    - SR-MPLS&SRv6のDual planeネットワークを経ることで移行が容易になる可能性あり
  - 既存網の一部に新たなSR網を接続することも自由自在
    - Pure IPv6なSRv6のメリットと、Option-Bによる疎結合なAS連携のメリット



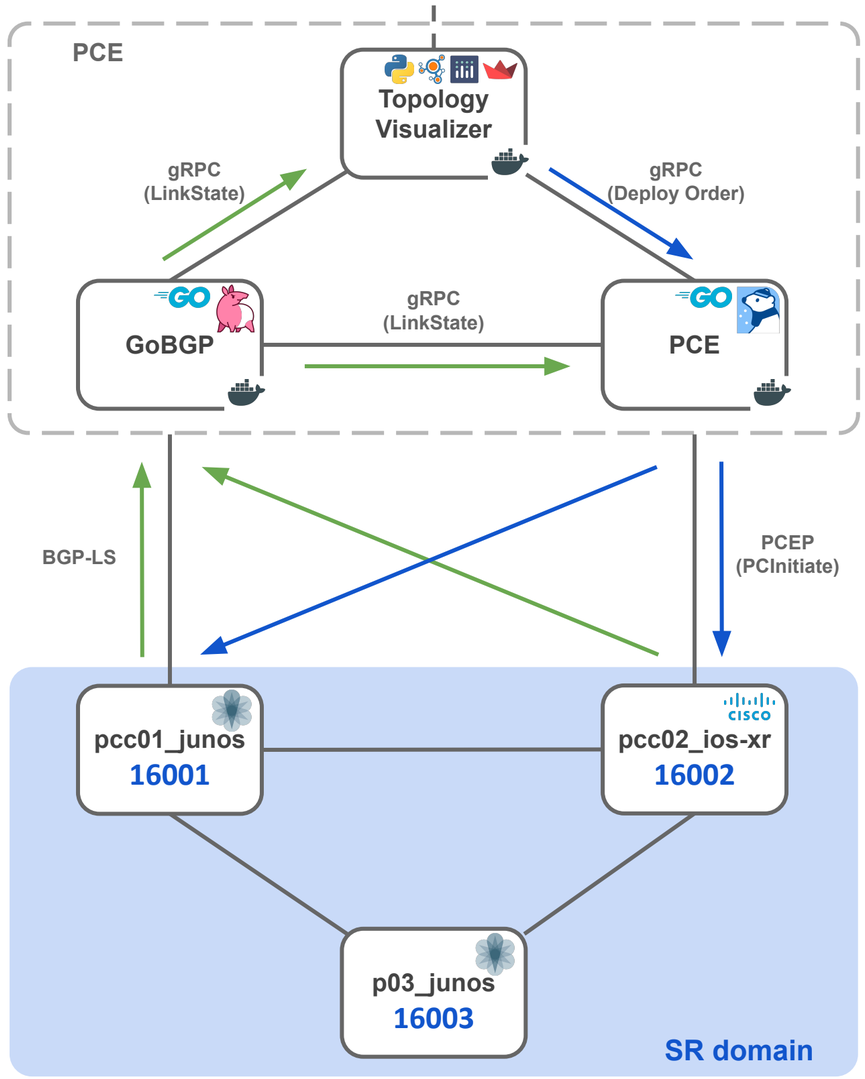
Inter-AS Option B

- RFC4364で提案されたAS接続オプションの一つ
- ASBR間でVPN経路を広告、VPNラベルのSWAPでAS間を接続する方式
  - ASBRはVRFに紐づくインターフェースを持たないため、経路のRIB Installが不要
  - AS間は単一のインターフェースで接続可能
  - ASBR間もVPNで接続。AS内のVPNとSWAPし、一連のVPNを構築
  - → **ASを分離し、疎結合に接続することでスケーラビリティの確保が可能**



デモ: Polaを用いた疎結合なコントローラの例

- pcc01（Junos）、pcc02（IOS-XR）双方にSegment Listをデプロイする



Topology Visualizerへ必要なSegment Listを入力しデプロイ
その後各機器にログインしtracerouteでTEを確認

PCC - PEE 間は同一リンクを使用しており、
BGP-LSとPCEPがそれぞれ別のポートを使用

pcc01_junos に対し Segment List **16002/16003/16001/16002** のループを含む経路を、
pcc02_ios-xr に対し **16003/16002/16003/16001** という往復が含まれた経路をデプロイ