SR環境下におけるコントローラとトラフィックエンジニアリング

JANOG49@Kagoshima (Jan, 2022)

Kohei Suzuki <kosuzuki@juniper.net>

JUNIPER NETWORKS | Driven by Experience™

承前

- 本発表では便宜上、ネットワークコントローラ (WAN Controller / SR Controller) と PCE (Path Computation Element) をほぼ同義として扱っておりますが、実際のところ、ネットワークコントローラはネットワークトポロジーの可視化、フロー収集、サービスオーケストレーションなどの様々な機能を備えたソフトウェアであり、PCEはそれを構成するモジュールの1つとなります。
- メーカーの提供する製品やOSのバージョン、実装によって技術仕様が異なる場合もあります。製品ごとのサポート状況につきましては各メーカーの担当者様へお伺いください。
- 本発表による見解は個人の意向を示したものであり、Juniper Networks, Inc. およびジュニパーネットワークス株式会社の意向を示すものではございません。

発表のきっかけ

ある日の業務中のつぶやき

SR環境下で本当にコントローラーが必要かどうかみたいなディスカッションをJANOGなり何かしらのイベントでもしてみたいんだけど、自分がメーカーの立場でしか喋れないから「コントローラーがあるとこういうことができるよ、ルータ単体だとこれができるよ」としか言えないのが難点。

すごく興味あります。PCEPつらたんを聞いてたので、SRでも同じことにならんかねとかやってる人に素直に聞いてみたいです。メーカの立場で言えない話はネットワークオペレータに話してもらうように整えればいいかと思いました。

JANOG実行委員 伊藤さん

発表のきっかけ

ある日の業務中のつぶやき

SR環境下で本当にコントローラーが必要かどうかみたいなディスカッションをJANOGなり何かしらのイベントでもしてみたいんだけど、自分がメーカーの立場でしか喋れないから「コントローラーがあるとこういうことができるよ、ルータ単体だとこれができるよ」としか言えないのが難点。

応募した結果、採択

すごく興味あります。PCEPつらたんを聞いてたので、SRでも同じことにならんかねとかやってる人に素直に聞いてみたいです。メーカの立場で言えない話はネットワークオペレータに話してもらうように整えればいいかと思いました。

JANOG実行委員 伊藤さん

本発表の内容

SR環境下でトラフィックエンジニアリング(TE)を行う際にあたって、

- ルータ単体でできること (コントローラを使わないSR-TE)
- コントローラを使うことでできること (コントローラを用いたSR-TE)
- TEのユースケース

について紹介し、どういった場面でコントローラ/TEが使えるかをご紹介いたします。

その後、JANOG参加者の皆様から、Segment Routingの導入にあたってコントローラにまつわる検討事項、運用してみてどうだったか、こういう点がしんどかった…など、Segment routingとコントローラに関するご意見をお伺いさせていただければと思います

自己紹介

鈴木 恒平 (すずき こうへい)

- 2019年 ジュニパーネットワークス株式会社 (新卒入社)
- 同年より通信キャリア様のアカウントSEとしてルータ・スイッチ製品、自動化ソフトウェアを始めとしたPre-Sales

JANOGとの関わり

- 2016年 JANOG38@沖縄 (若者支援プログラム)
- 2020年 JANOG45@札幌 (現地参加)
- 2021年 JANOG49@鹿児島 (初登壇)

Agenda

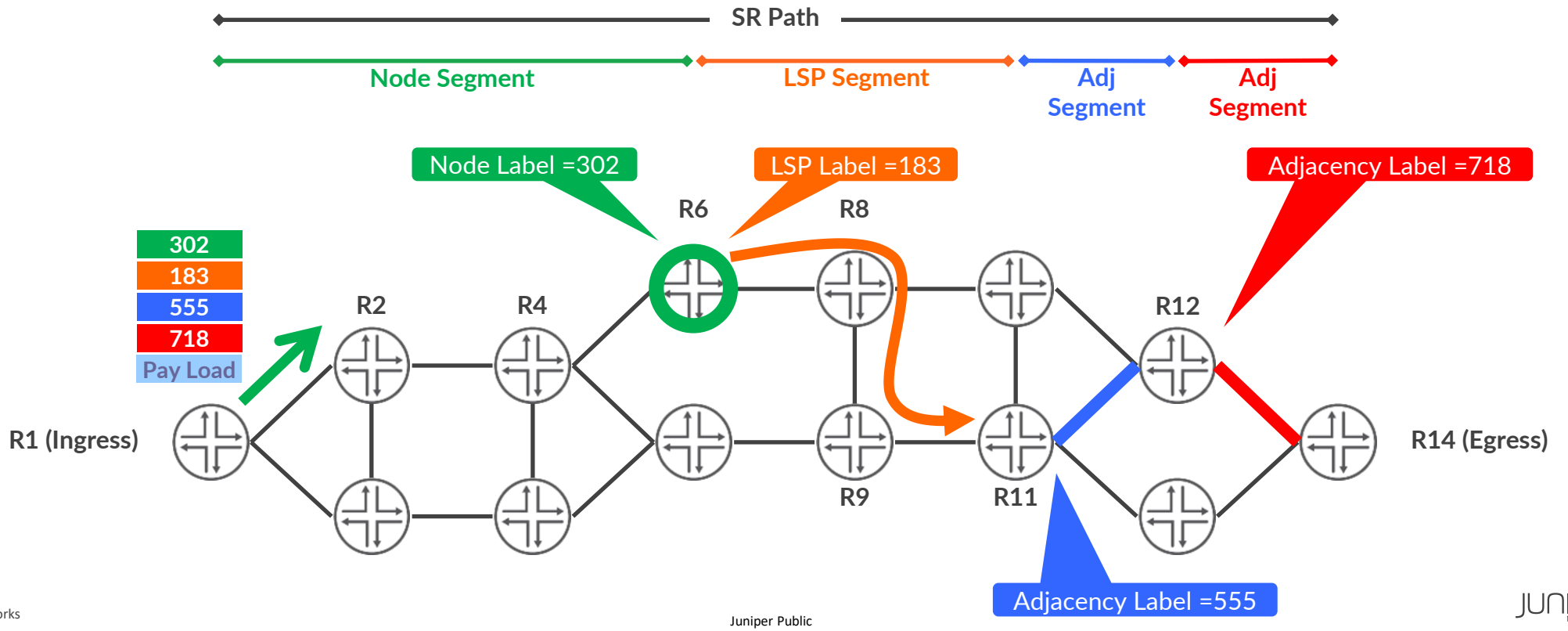
- Segment Routing と コントローラ
- コントローラを使わないSR-TE
- コントローラを用いたSR-TE
- TEのユースケース
- まとめ

Agenda

- **Segment Routing と コントローラ**
- コントローラを使わないSR-TE
- コントローラを用いたSR-TE
- TEのユースケース
- まとめ

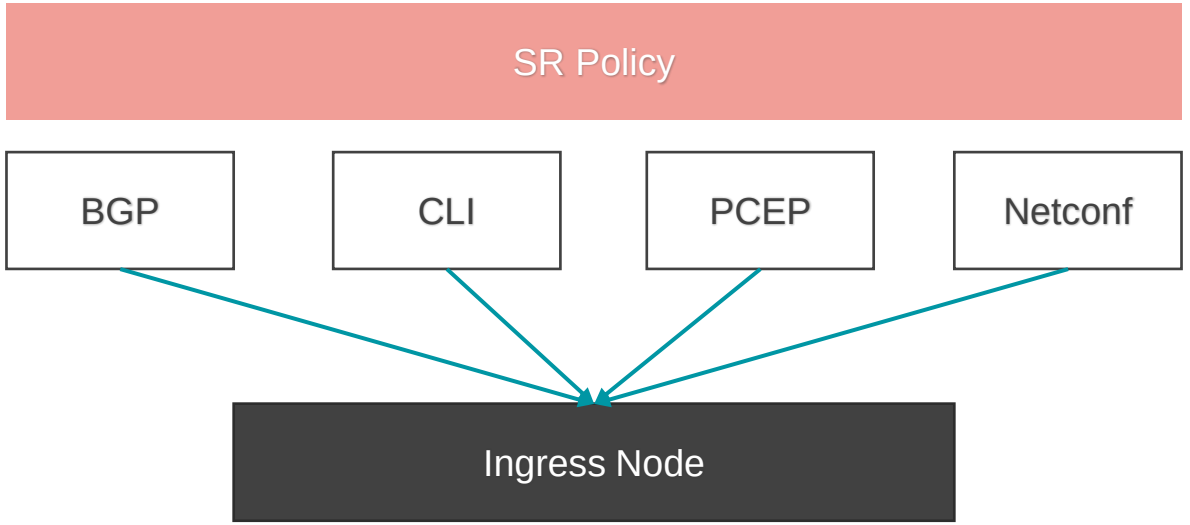
Segment Routing とトラフィックエンジニアリング (SR-TE)

- Ingress Node (Head End) でトラフィックが通過するパスを決定し、通過する要素を Segment (SID) としてパケットのヘッダに埋め込むことで、パケットが順序通りに転送される
  - SR-MPLSの場合はMPLSラベル、SRv6の場合はSRHにSIDが格納される
  - SR-MPLSの場合、送信元ノード以外はスタックされたラベルに基づいた処理を行う
- Segment Routing におけるトラフィックエンジニアリングは通過する要素のリスト(=SR Policy)をどのように決めるかが課題



SR Policy の設定方法

- Head-end (Ingress) ノードは様々な方法で SR Policy を設定・学習
- SR Policy は BSID entryとしてFIBに登録される
- SR Policy が有効で、かつそのプリファレンスが最高値である場合に、そのパスが選択される
- パスの選択ロジックにおいて、パスのプロトコルソースは重要ではない
- 新しいCpathが学習されたとき、または前のCpathが期限切れになったときに、選択プロセスが再実行される



SRにおけるネットワークコントローラ/PCE

トラフィックが通過する経路 (TE Path) を比較的簡単に作成・制御できるSR環境下では、より多くのLSP/ネットワークスライスが作成されることが想定される

- 様々な制約事項を考慮したうえで最適な TE Path を計算する必要性
- ネットワークトポロジの変化に応じて動的に TE Path を最適化したい

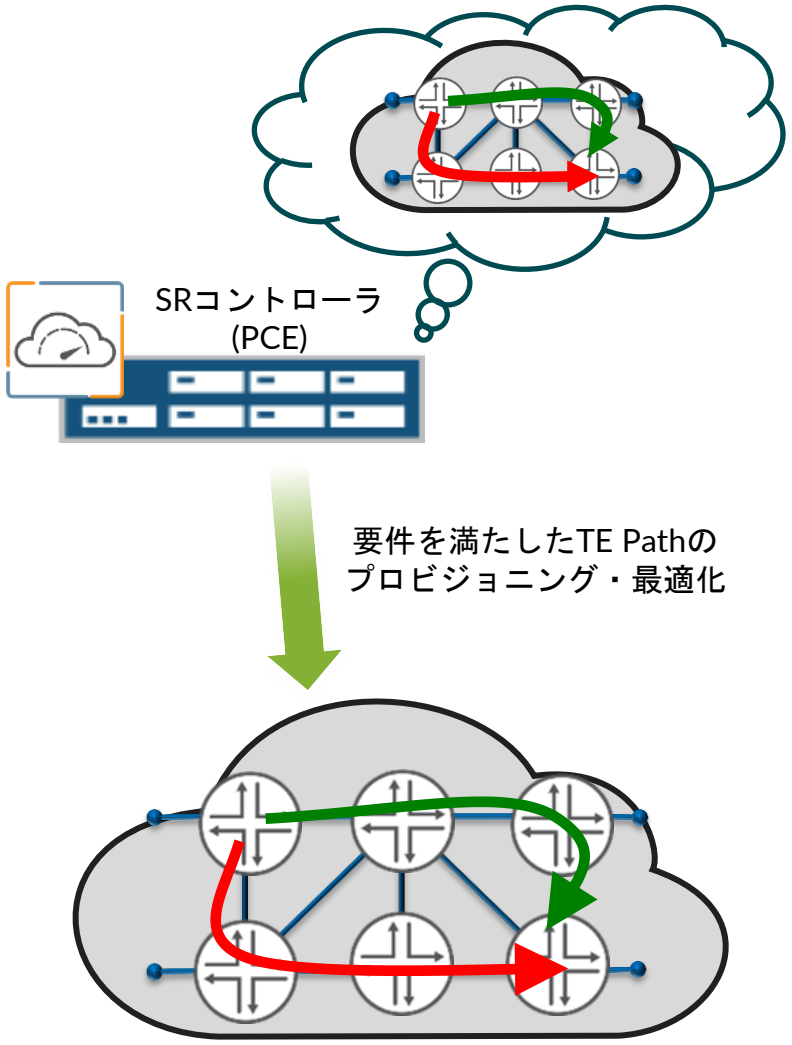
PCE (Path Computation Element) による中央集権的なパス計算/プロビジョニング

Stateless PCE (RFC8281)

- ネットワーク内のLSP情報を管理しない
- PCC(ルータ)からのパス計算のリクエストに対して応答するだけ

Stateful PCE (RFC8664)

- ネットワーク内のLSP情報を管理する
- PCEから新規にLSPを確立したり、既存LSPのパス変更が可能



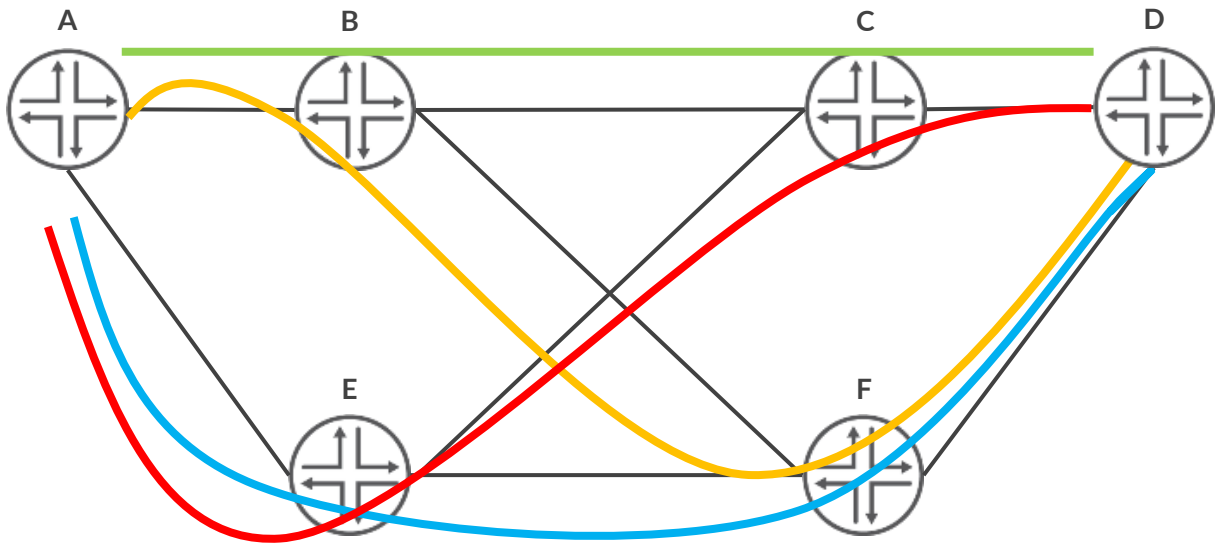
で、コントローラありとなし、どっちが良いの??

Controller (PCE)

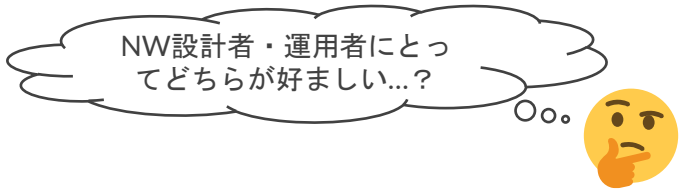
最適な経路のパス計算を行うPCE[*1]としての役割を果たす上で、ネットワーク全体のステータスを知っている必要がある

TE Path (LSP) の設定
- 手動による設定
- NETCONF
- PCEP

ネットワーク状態の収集
- ノード情報 (SNMP/Telemetry)
- トポロジ情報 (IGP/BGP-LS)
- フローの収集 (IPFIX etc.)



PCEを使わなくても(動的に)パス設定を行う方法も存在する
- CLIによる逐次設定
- ルータ上でのパス計算 (CSPF)



[*1] PCE (Path Computation Element) については下記資料もご参照ください
- PCE～MPLSネットワークのSDN化を本気で実現する”唯一の”方法～ (宮坂拓也, 2017, JANOG39.5)

Agenda

- Segment Routing と コントローラ
- **コントローラを使わないSR-TE**
- コントローラを用いたSR-TE
- TEのユースケース
- まとめ

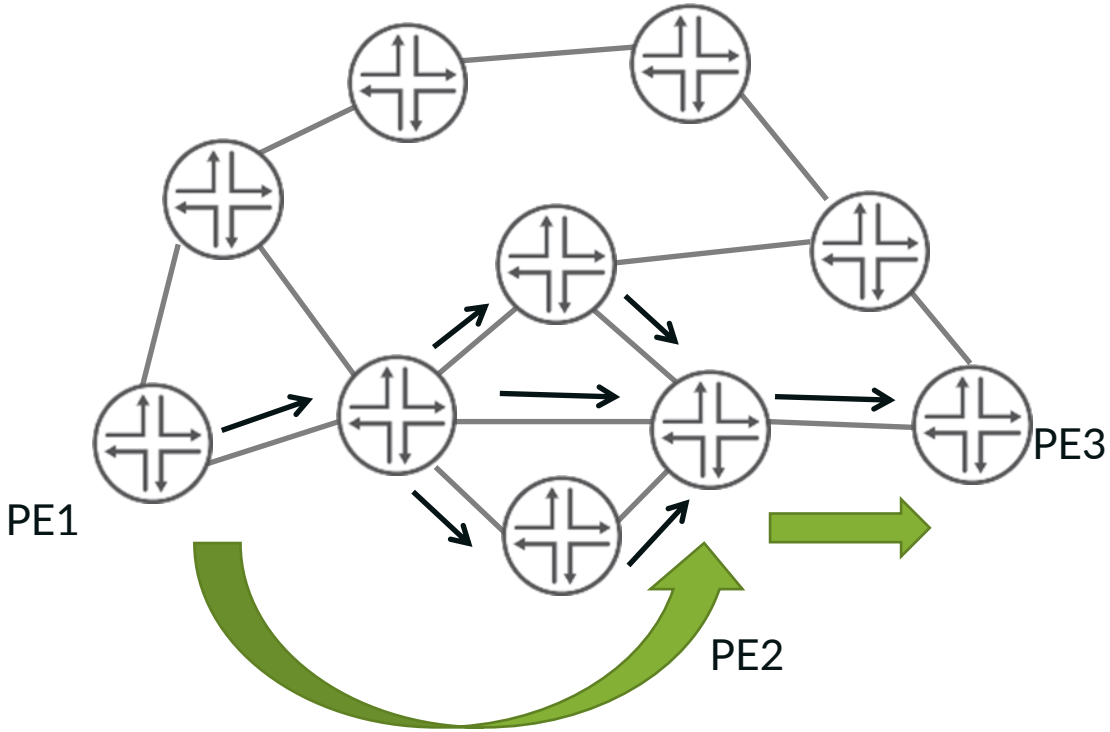
コントローラを使わないSR-TE

中央集権的なコントローラを使わずに行うトラフィックエンジニアリングを行う方法

- 手動による設定
- CSPF計算に基づくTE

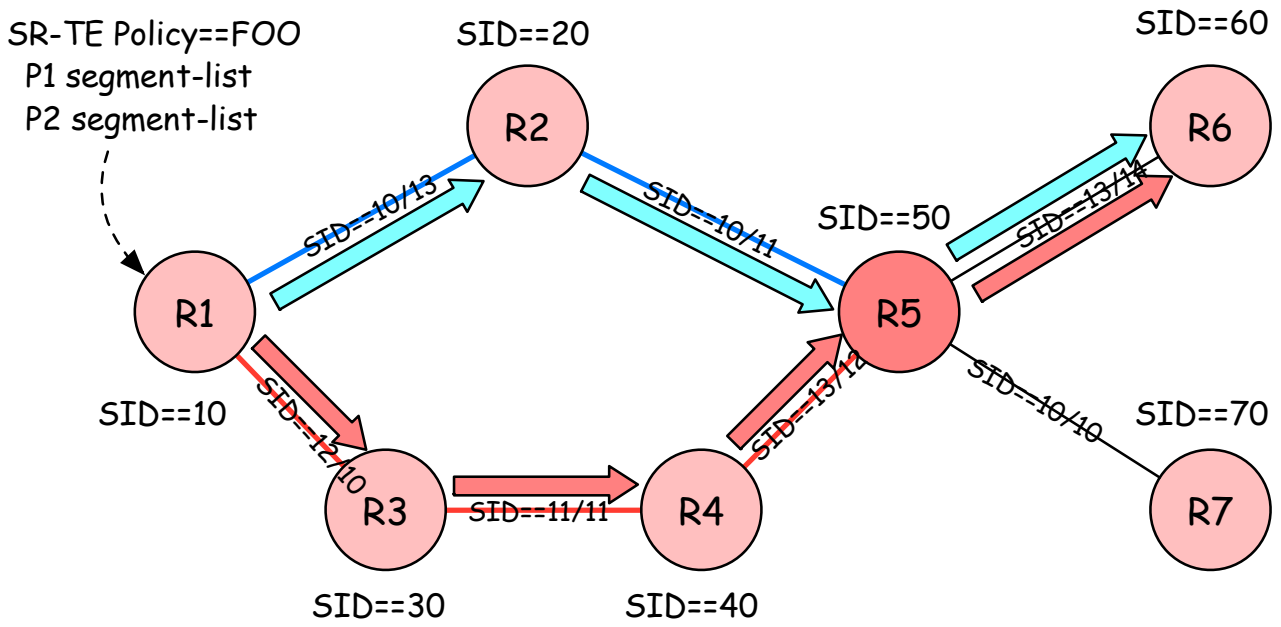
特徴

- パス計算はHead-endルータで行われる
- CSPFでは admin-color などの要素を元に計算がされる



SR-TE LSP の設定

- Junosの場合は `protocol source-packet-routing source-routing-path` 以下に SR-TE LSP を設定
- segment-list 以下に通過するHOP名とSIDを設定
  - Non-colored SR-TE LSP の場合、First HopはIPアドレスを指定
- Weighted ECMP, Secondary Path の設定も可能




```
edit protocols source-packet-routing
source-routing-path FOO {
    to 10.0.0.6;
    color 10;
    binding-sid 1000;
    primary {
        p1 weight 2;
        p2 weight 1;
```

```
segment-list p1 {
        hop1 label 120;
        hop2 label 150;
        hop3 label 160;
    }
```

SRGB==100-200

Node-SID based path

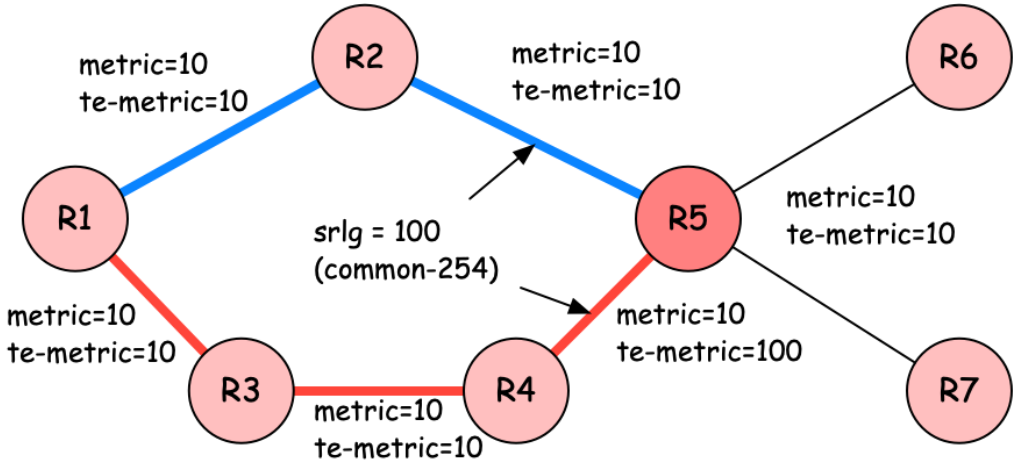
```
    segment-list p2 {
        hop1 label 12;
        hop2 label 11;
        hop3 label 12;
        hop4 label 14;
```

Adj-SID based path

CSPFによる SR-TE Path の計算

Head-EndでSR-TEパスの計算を実施し、動的にパスを決定

- TE Path は CSPFによって計算され、ネットワークの変更に応じて再計算
- IGPのフラッディング・コンバージェンスに依存
- Segment list ごとに制約条件を設定可能



1 LSP, 2 segment-lists, constraints per SL, & WECMP

```
protocols {
    source-packet-routing {
        compute-options{
            optimize-timer 3600;
        }
        compute-profile blue-lsp {
            include-any blue;
            maximum-segment-list-depth 5;
        }
        compute-profile red-lsp {
            include-any red;
            maximum-segment-list-depth 5;
        }
    }
    source-routing-path sr-te-lsp-to-r6 {
        to 1.1.1.6;
        primary {
            1st-seg-to-r6 {
                weight 25;
                compute red-lsp;
            }
            2nd-seg-to-r6 {
                weight 75;
                compute blue-lsp;
```

CSPFで計算できるメトリック

- IGP metric
- TE metric
- Delay metric

Agenda

- Segment Routing と コントローラ
- コントローラを使わないSR-TE
- **コントローラを用いたSR-TE**
- TEのユースケース
- まとめ

コントローラを使ったSR-TE

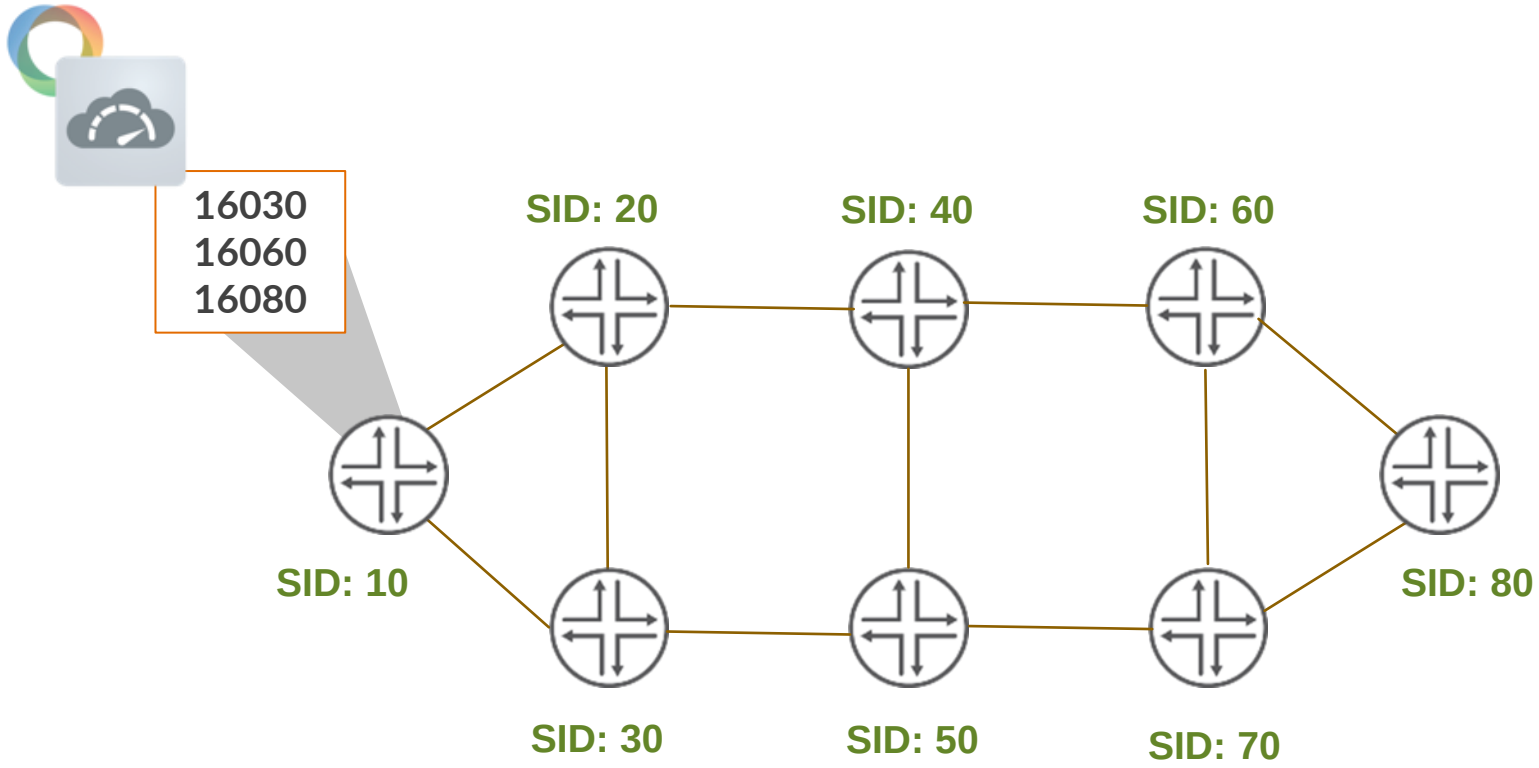
特徴

トポロジ情報の取得

- IGP
- BGP-LS

TE Pathのプロビジョニング

- BGP-LU
- BGP SR-TE
- PCEP
- Netconf/Yang



PCE (Path Computation Element)

PCE: Path Computation Element (RFC4655)

- ネットワーク上の経路を計算を行うエンティティ
- ネットワークノード (Ingress Router)、サーバ、アプリケーションなどが考えられる。

PCE: Path Computation Element

- 経路を計算する

PCC: Path Computation Client

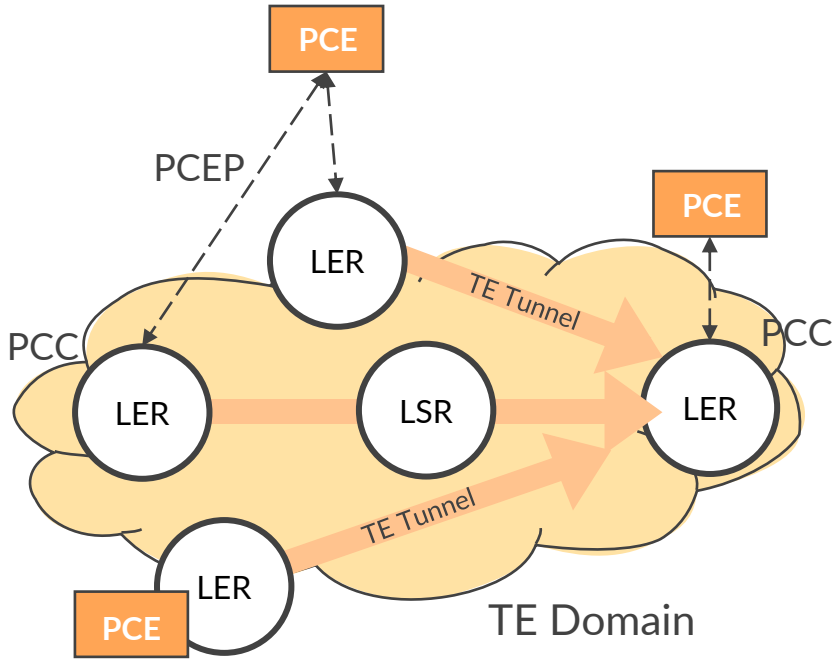
- パス計算クライアント

PCEP: PCE protocol (RFC5440)

- PCEとPCCの間のやり取りに使われるプロトコル

LSPのシグナリングと定義

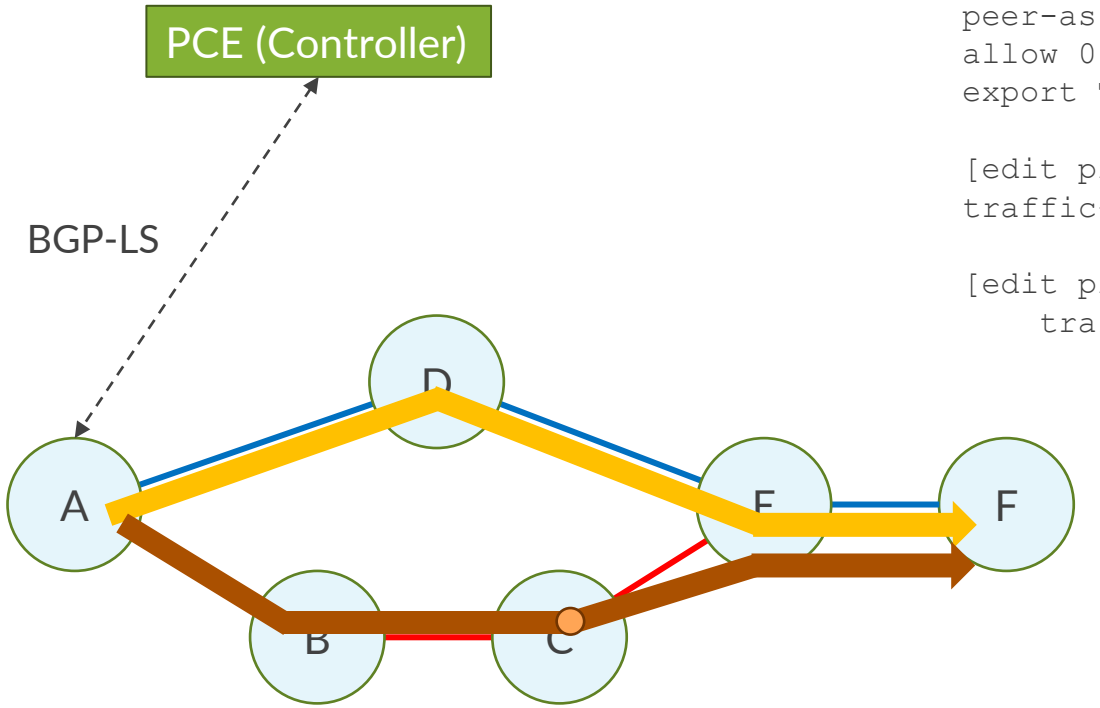
- Static, RSVP, SR-TE, SRv6-TE, …



BGP-LS によるトポロジー情報の export

BGP-LS により IGP トポロジーをコントローラーへ export (RFC 8571)

- SR ドメイン内の任意のルータで BGP-LS を設定 (PE ルータ・RR など)
- コントローラで最適なパスを計算

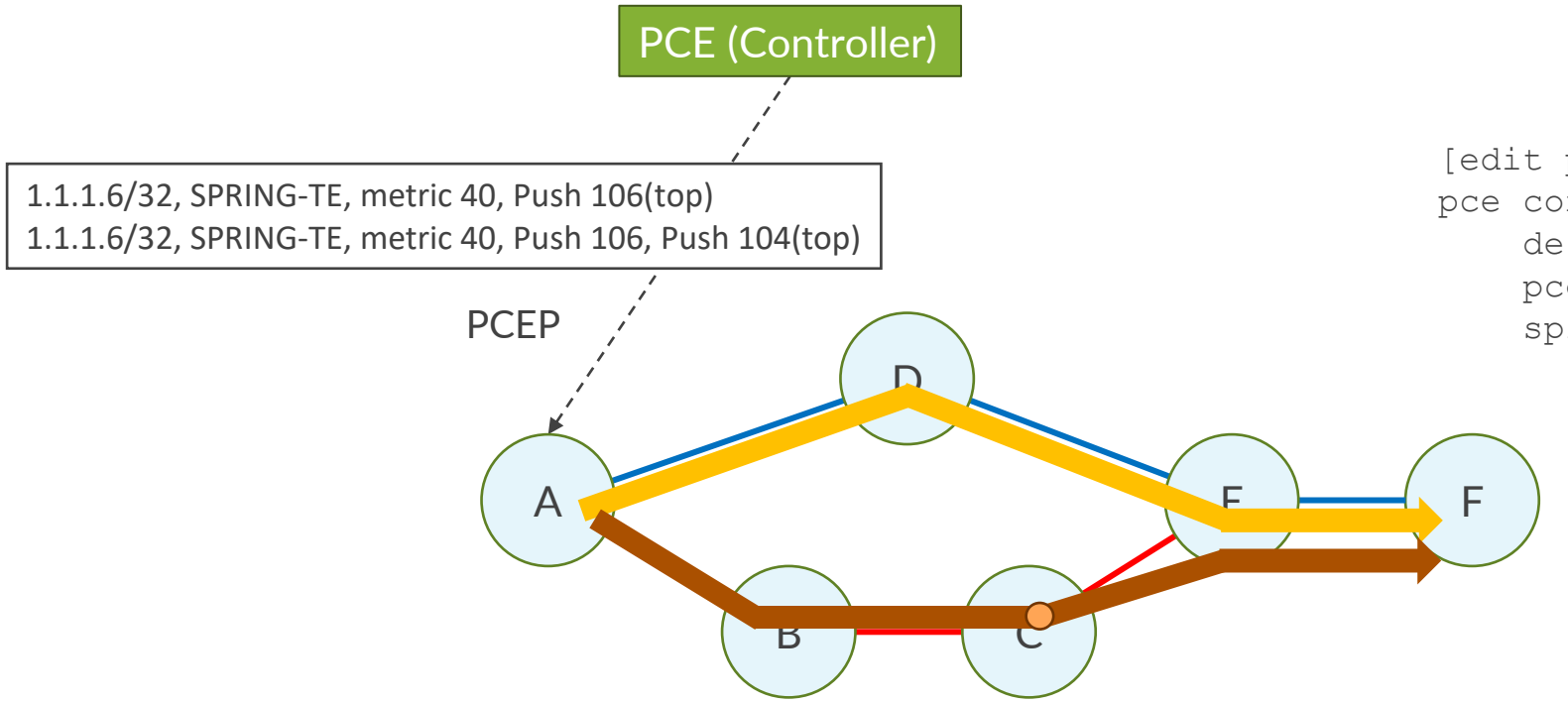



```
[edit policy-options]
family traffic-engineering {
    from family traffic-engineering
    then accept;

[edit protocols bgp group bgp-ls]
family traffic-engineering {
    unicast
peer-as 65000
allow 0.0.0.0/0
export TE

[edit protocols ospf]
traffic-engineering l3-unicast-topology

[edit protocols mpls]
    traffic-engineering {
        database {
            import {
                policy TE;
```

PCEPによるLSPの制御・検出

LSP の Head-end となるルータとPCE間で PCEP を介してコミュニケーション (RFC 8231, 8281, 8623)

- 各LSPの管理をPCEに委譲し、PCEからLSPの制御を実施 (Stateful PCEの場合)
- importしたトポロジー情報を元に最適なパスを決定
- PCEPまたはNetconfによるLSPのプロビジョニング

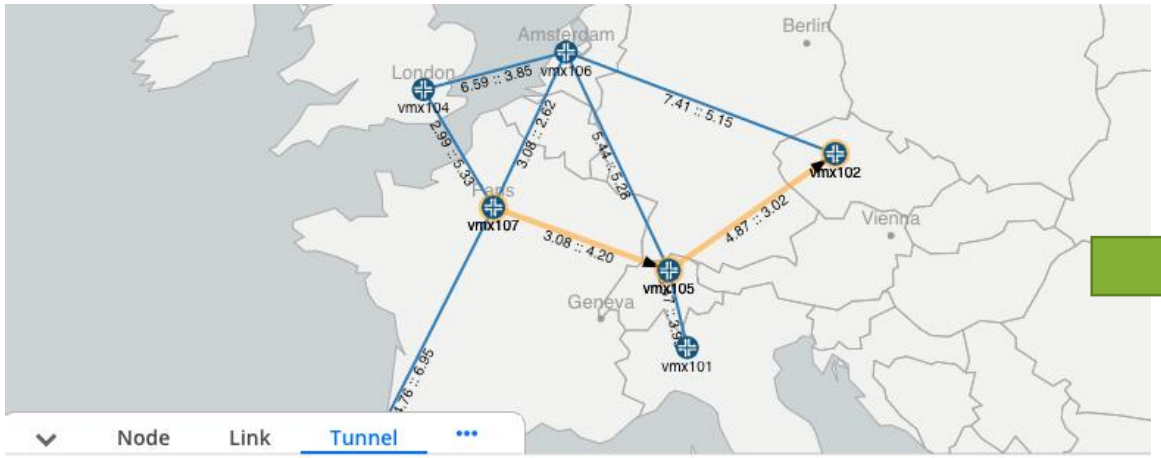



```
[edit protocols pcep]
pce controller {
    destination-ipv4-address 10.123.123.254;
    pce-type active stateful;
    spring-capability;
```

Agenda

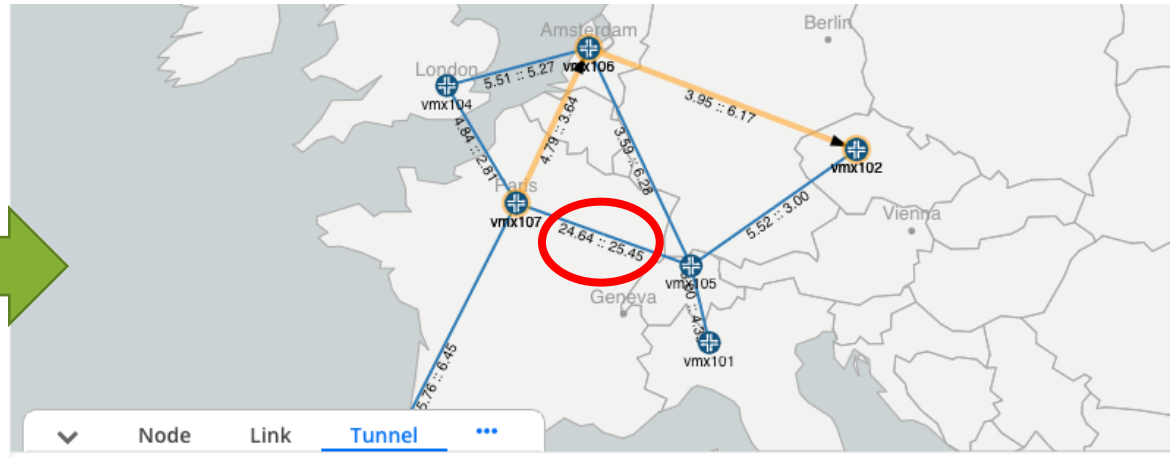
- Segment Routing と コントローラ
- コントローラを用いたSR-TE
- コントローラを使わないSR-TE
- **TEのユースケース**
- まとめ

リンク遅延に基づいたルーティング



Tunnels 1 selected

Name	Node A	Node Z	Type	Op Status	Path Selection
LL_VMX107VMX102	vmx107	vmx102	SR	▲ Active	dynamic



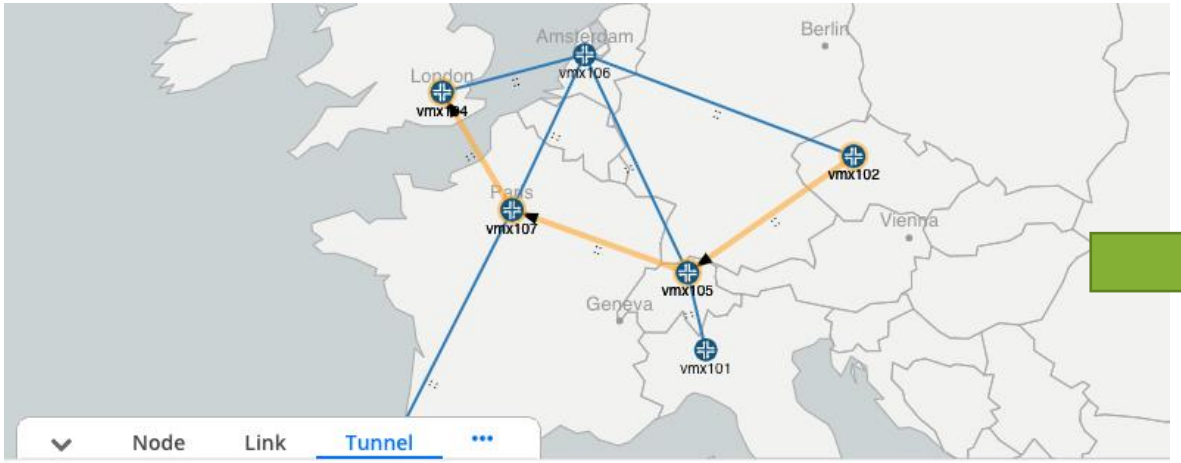
Tunnels 1 selected

Name	Node A	Node Z	Type	Op Status	Path Selection
LL_VMX107VMX102	vmx107	vmx102	SR	▲ Active	dynamic

- 測定された遅延値を収集し、トポロジー上の各リンクについて遅延値を把握
- 収集した遅延値を元に、遅延が最小となるようなパスを作成

- 特定のリンクの遅延値が変化したとき、条件を満たすパスを再計算し、再度パスをプロビジョニングすることも可能

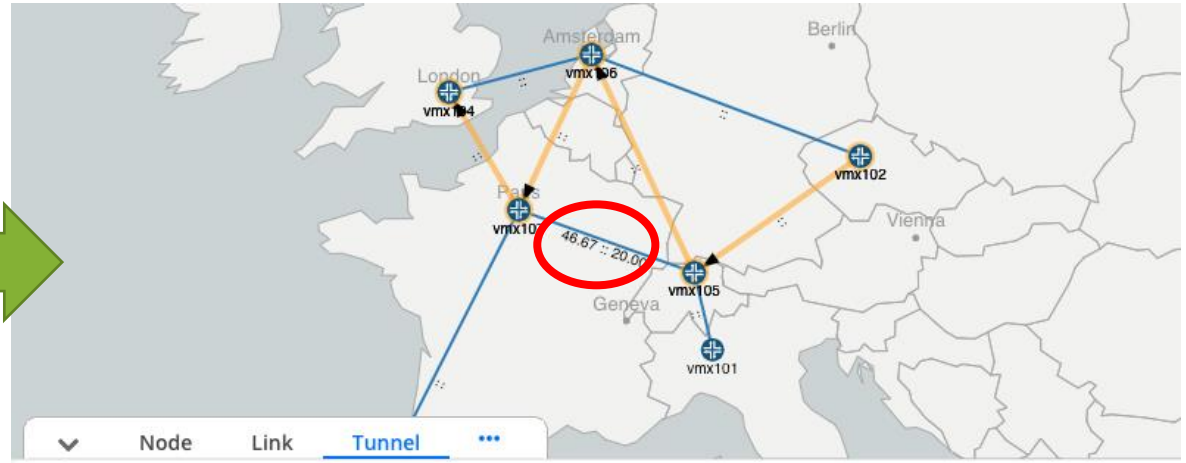
パケット損失に基づくパスの最適化



Tunnels 1 selected

Name	Node A	Node Z	Type	Op Status	Path Selection
VMX103-104	vmx102	vmx104	SR	▲ Active	dynamic

- コントローラが各リンクのパケットロスに関するデータを収集

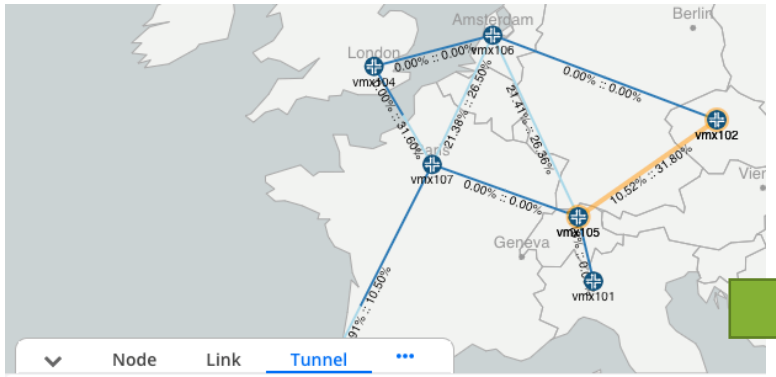


Tunnels 1 selected

Name	Node A	Node Z	Type	Op Status	Path Selection
VMX103-104	vmx102	vmx104	SR	▲ Active	dynamic

- コントローラがパケット損失のしきい値を超えたときに、パケット損失の高いリンクを迂回するようにLSPを最適化

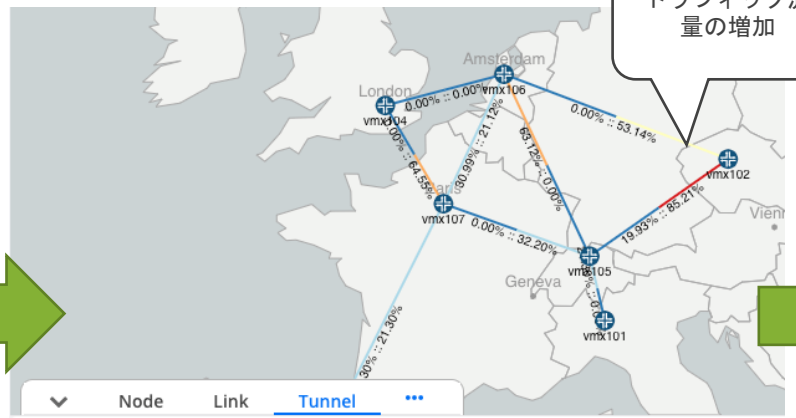
リンクの輻輳回避



Tunnels - on link

Name	Node A	Node Z	Type	Op Status
VMX103-104	vmx102	vmx104	SR	Active
rsvp-102-105	vmx102	vmx105	RSVP	Active
Silver-102-101	vmx102	vmx101	RSVP	Active
Silver-102-103	vmx102	vmx103	RSVP	Active

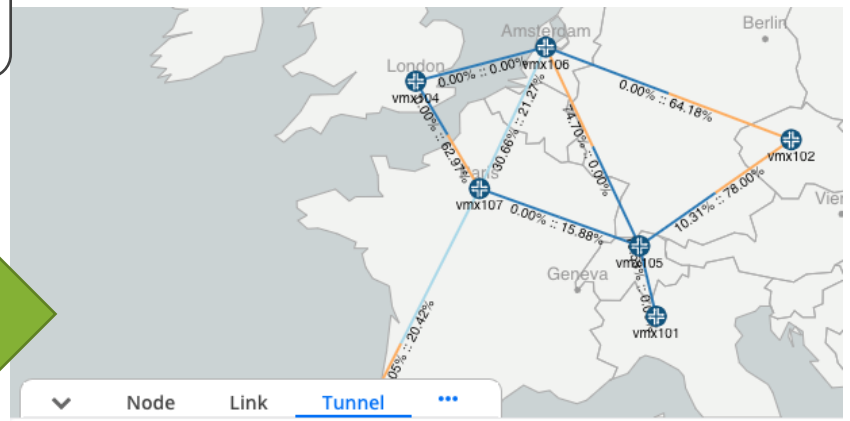
14 items



Tunnels - on link

Name	Node A	Node Z	Type	Op Status
VMX103-104	vmx102	vmx104	SR	Active
rsvp-102-105	vmx102	vmx105	RSVP	Active
Silver-102-101	vmx102	vmx101	RSVP	Active

12 items



Tunnels - on link

Name	Node A	Node Z	Type	Op Status
VMX103-104	vmx102	vmx104	SR	Active
rsvp-102-105	vmx102	vmx105	RSVP	Active
ios-xr9_t101	ios-xr9	vmx101	RSVP	Up

11 items

- コントローラがトポロジー上の各リンクについてリンク使用率を測定
- リンクが設定したしきい値を超えると、コントローラが輻輳しそうなリンクを迂回して、輻輳の発生を軽減する

Agenda

- Segment Routing と コントローラ
- コントローラを用いたSR-TE
- コントローラを使わないSR-TE
- TEのユースケース
- **まとめ**

コントローラを使わないTEとコントローラを用いたTE

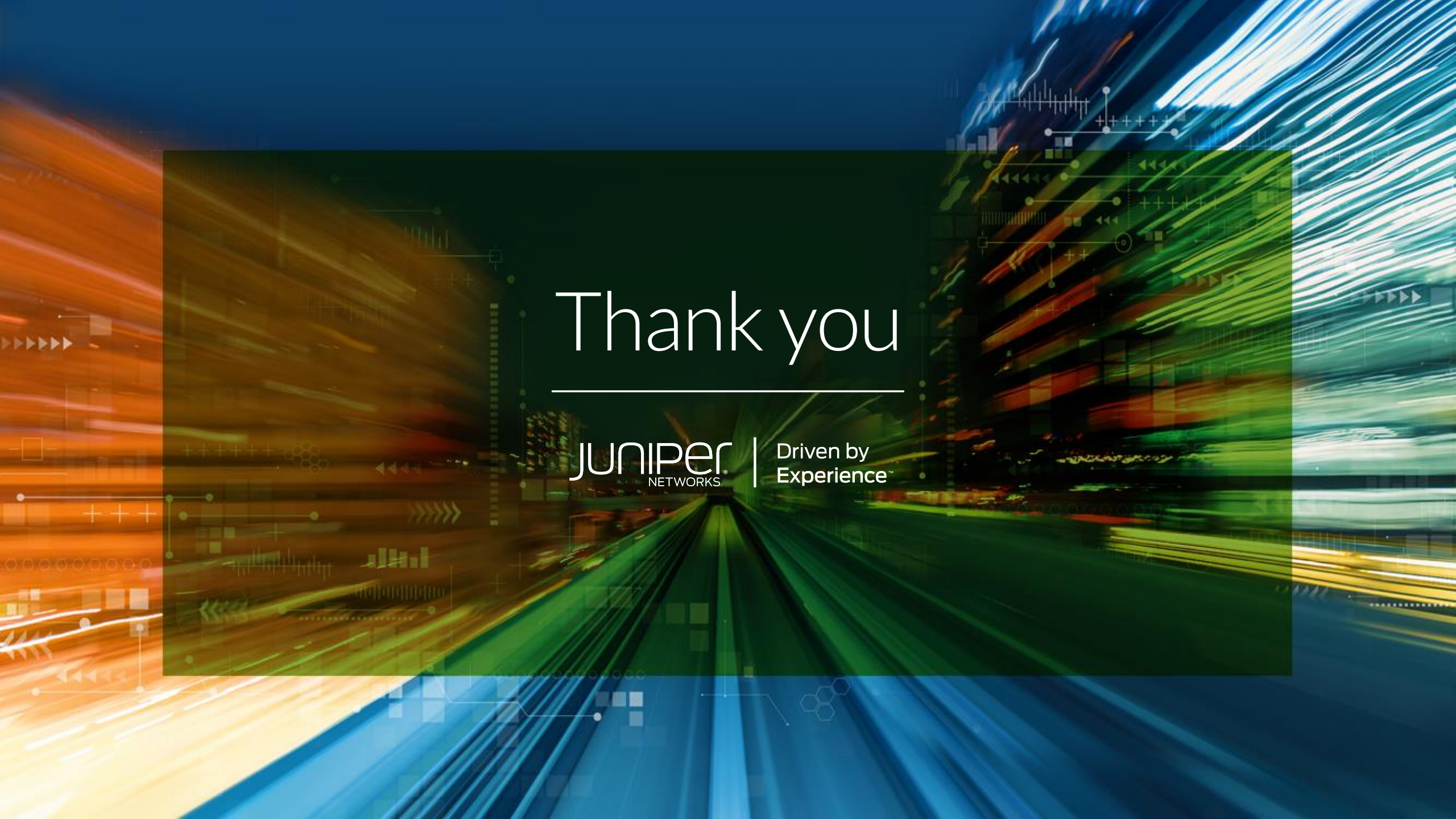
明示的な SR-TE の逐次設定	→	技術的には可能だが、大規模化すると構成が複雑に…。
Head End での SR-TE Path の計算	→	Head-endがトポロジ全体を関知している前提。簡単なTEならいけそう？
Controller による SR-TE Path の管理	→	細かい粒度でのTEはできそうだが、そこまでのTEは必要か？

コントローラとSR-TEに関する所感:

- ✓ Segment Routing自体はステートレスなプロトコル
  - ルータ単体でパスを計算する技術は登場しつつあるが、ルータ単体のみでの細かいTEは(アーキテクチャ的に)難しそう
  - ステートを保持するという観点ではステートフルPCE/コントローラとの相性は良さそう
    - Telemetryの活用先の一つとしても
- ✓ 要件に応じたネットワークを提供するうえで、ネットワークスライス、TEパスがたくさん作られる未来になるという仮定に立つと、トランスポートコントローラに対する関心は高まりそう
  - その一方でコントローラの議論はこれまでに十分なされていない(気がする)

終わりに: お伺いしたいこと

- Segment Routing の導入/運用にあたってコントローラを使用/検討していますか？
- 検討中の場合は、どういった点が気になりますか？
- 検討したがやめたという場合は、なぜ使わないという判断に至ったのでしょうか？
- 実際にコントローラを使ってSRを運用してみて、あるいはコントローラなしでSRを運用してみて苦労したこと、困ったこと、工夫して解決したことなどがあればご意見をお伺いさせてください。


Thank you
JUNIPER NETWORKS
Driven by Experience™