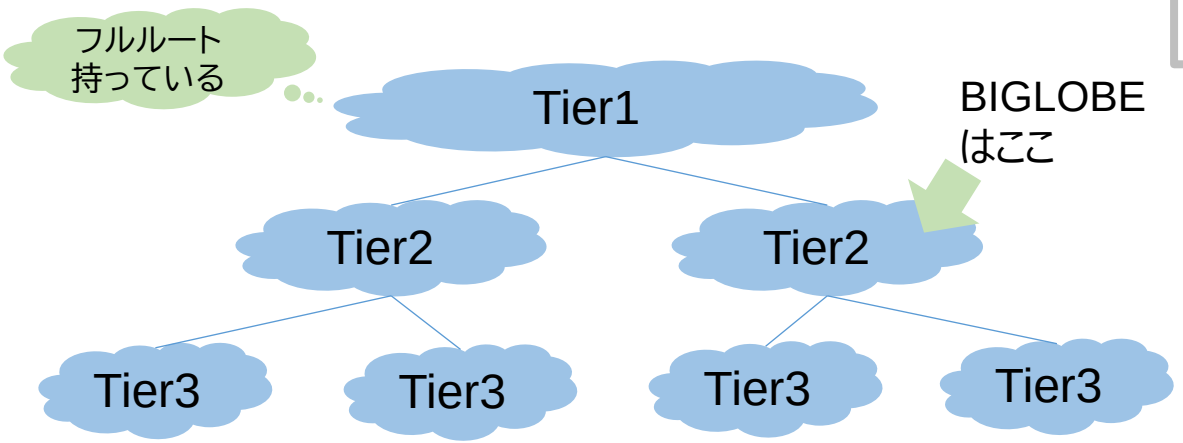
# BIGLOBE AS2518を まるごと仮想環境へ”コピー”してみた BIGLOBEパート

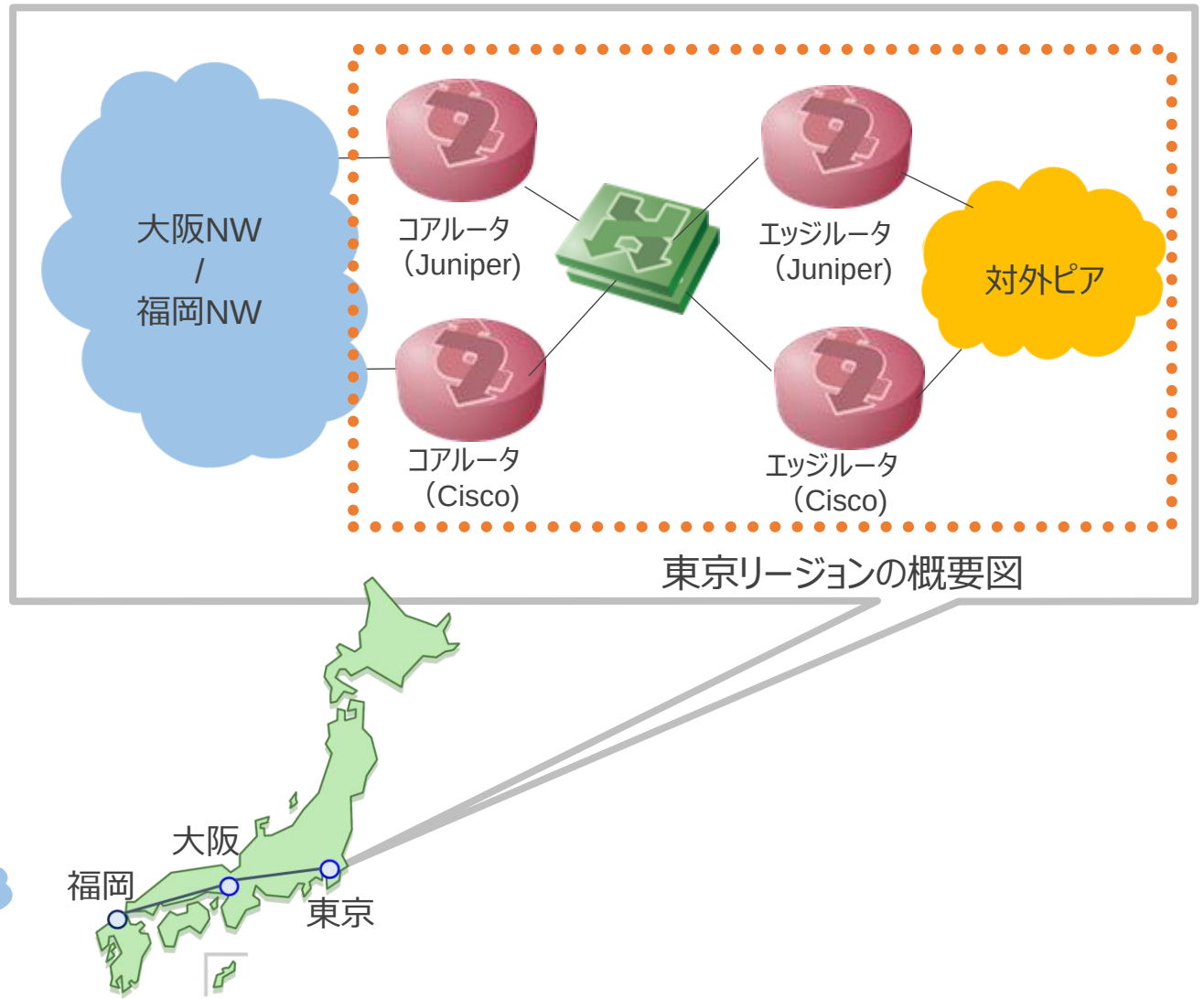
JANOG 53　2024年1月19日

ビッグローブ株式会社
プロダクト技術本部 ネットワーク技術部 開発G
滝口敏行

# BIGLOBE 商用環境

- BIGLOBE ISPバックボーンの規模感
  - インターネット階層：Tier2相当
  - 国内コア拠点：東京、大阪、福岡
  - DCの数：10拠点
  - ネットワーク装置（L2SW/Router）数：50台以上
  - JuniperやCiscoなどのマルチベンダーで構成
  - 総トラフィック：3.2Tbps以上（2023年度6月時点）
  - ピア数：220以上のASと相互接続

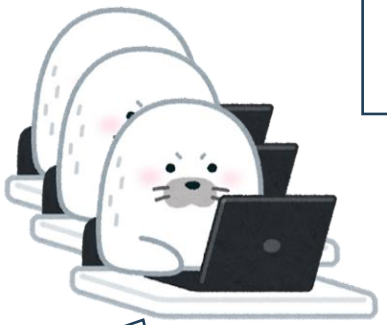


# バックボーン運用の課題

BIGLOBEでは作業品質を高めるためにレビューポイントを多く設定している。
しかし設定内容の正当性判断がレビュワーの経験に依存しており、ベテラン社員に作業レビューが集中してしまっている課題がある

作業者
（業務委託、
社員）
3名～ 体制



手順書
手順書
手順書

レビュー依頼

1次レビュー
ベテラン社員
1-2名体制

人力チェックなので、チェック漏れがないように細心の注意が必要

承認依頼

2次レビュー
マネージャー

作業概要：PNI or Publicピアの追加や廃止、
Transitの回線増速や廃止、
トラフィックコントロールなど
頻度：　毎月20件程度発生

レビューに時間がとられて他の業務が回らない

作業リスクによっては部長による3次レビューも

# バックボーン運用の課題

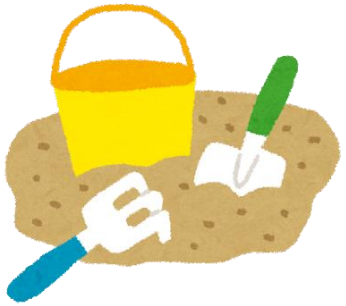
BIGLOBEのバックボーンを熟知したベテラン社員を増やしたいが、、

全体のトポロジー図を定期的にメンテする余裕がなく、情報の鮮度を保ててない。
全体のトポロジーと動作を把握するには商用のNWコンフィグを読んで、
理解できるスキルと経験が必要となってしまうジレンマ。

商用環境相当のStaging環境を用意することが難しいので（コスト面でも工数面でも）、
バックボーン全体の動作理解ができる環境も用意ができない。
地道に商用環境での運用経験を積むことでしか全体動作イメージを持てない。

→現状では**ベテラン社員を増やすことは容易ではない。**

鮮度が
大事！



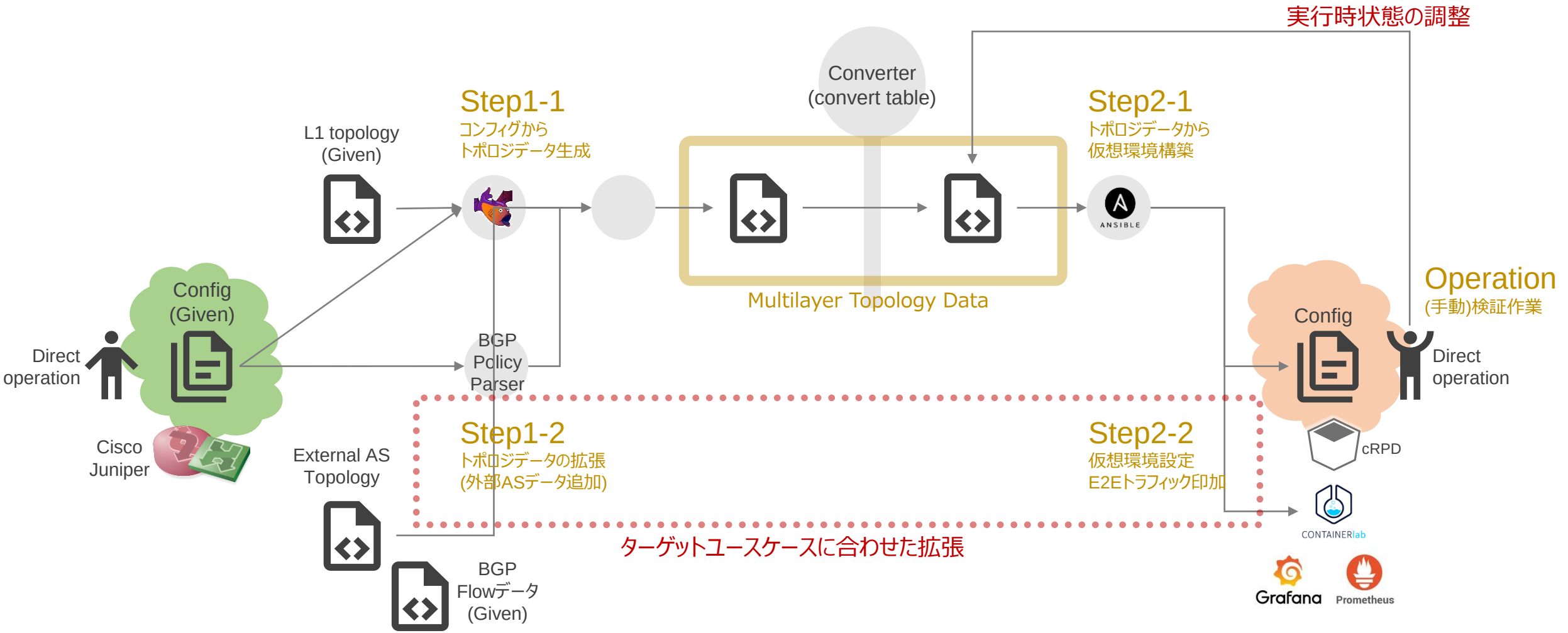
商用環境
に近い
砂場が
欲しい

ベテラン以外でも、商用に近い環境を素早く作成して
自由に動作検証できる”コピー環境”が欲しい

その環境を使うことで、商用環境の動作を理解を促し
ベテラン社員にしかできなかった作業内容の正当性判断に関して、
ベテラン社員以外でも対応できるようにしたい

# ”コピーする”システムを
# BIGLOBEの商用環境で試してみた

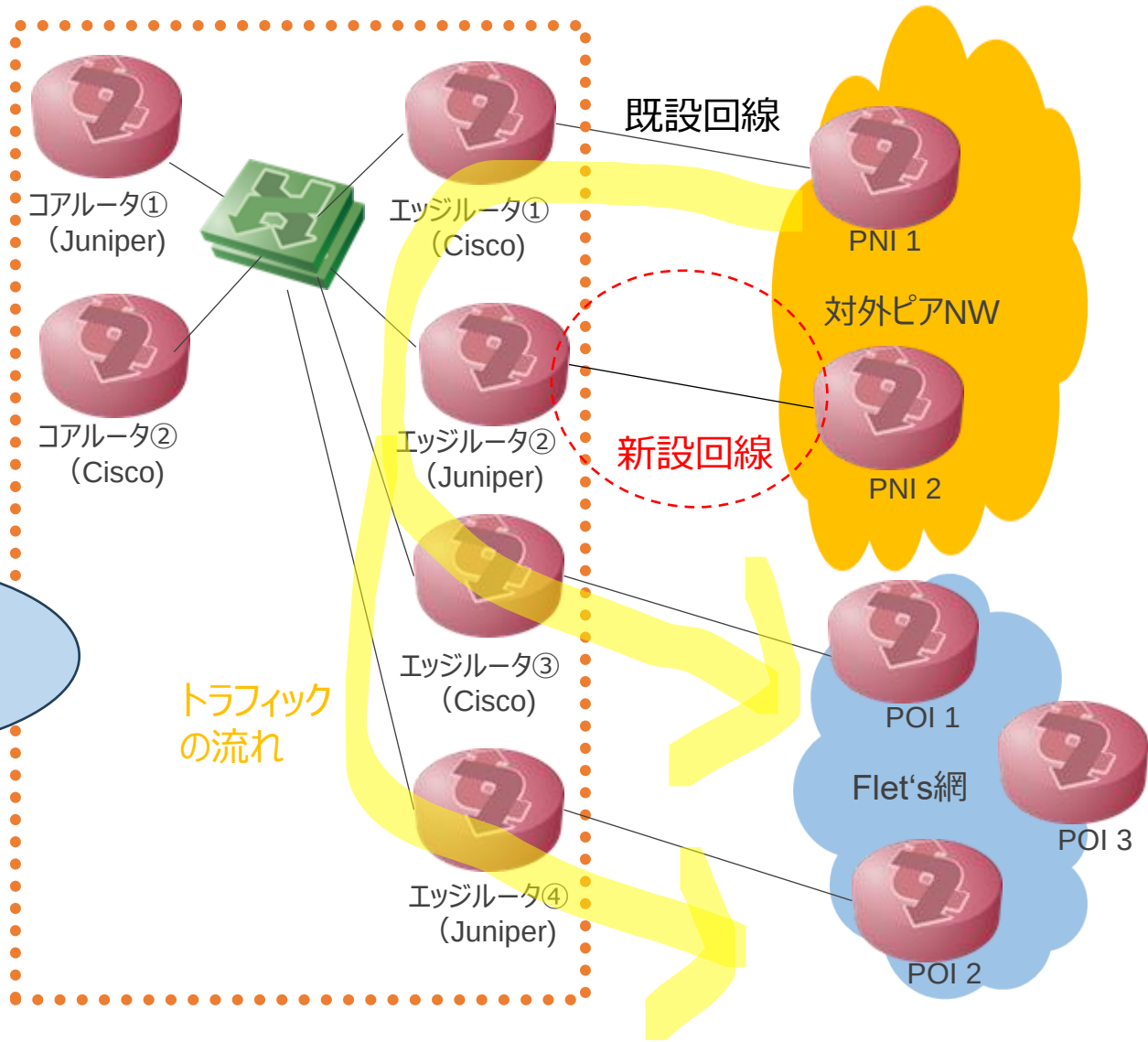
# 商用環境を”コピーする”システム構成



# BIGLOBEでコピーした構成とオペレーション評価の概要

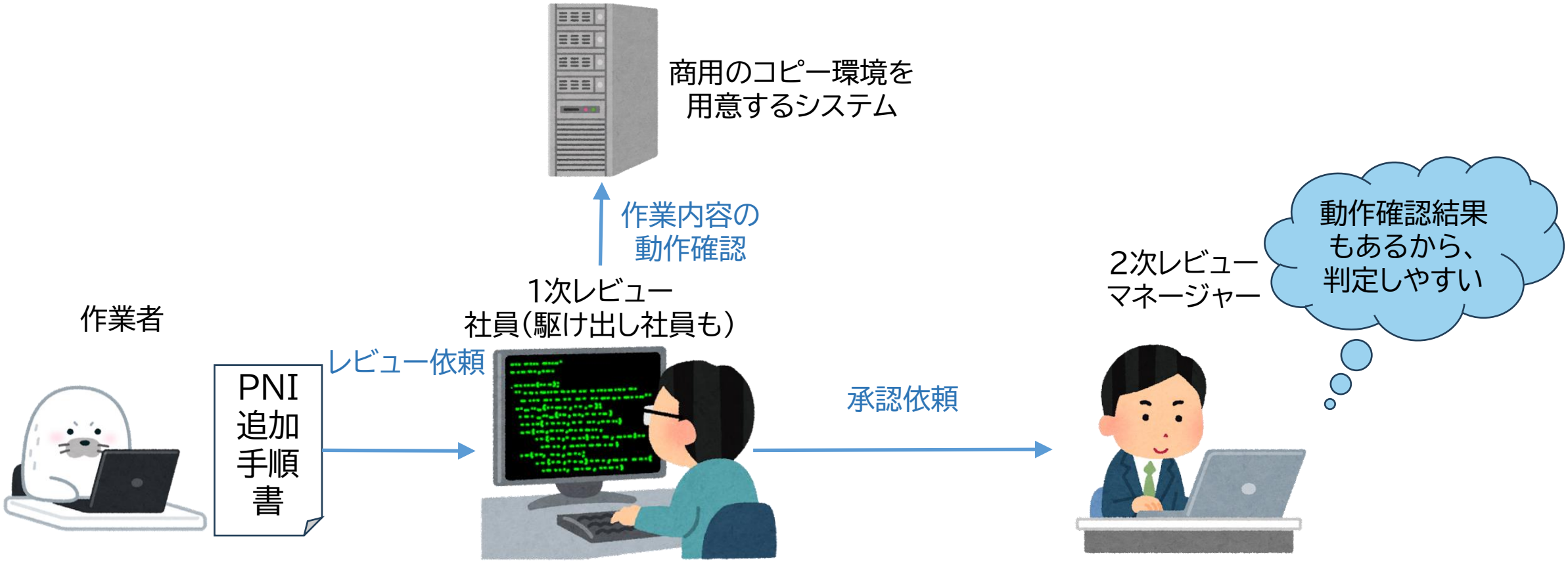
- オペレーション評価
  - スコープ:
    - 東京リージョンのコアルータ、エッジルータの本番Config
  - 確認内容:
    - 本番Configからモデルデータ/トポロジ生成が可能か
    - モデルデータから構築した検証環境が本番と同等か
  - ユースケース:
    - 1st STEP: Privateピア(PNI)の増設作業
    - 2nd STEP: Public(IX)ピアの増設作業
    - 3rd STEP: Transitピアの増設作業

PNI作業は全体の約20%を占め、一番シンプルなケースなため、1st STEPに選定




大阪と福岡も概ね同じ構成,Config設計のため、
東京でのオペレーション評価ができれば
他の地域にも適用できると判断して検証

# ユースケースの流れ


商用のコピー環境を
用意するシステム
作業内容の
動作確認
1次レビュー
社員(駆け出し社員も)
作業者
レビュー依頼
PNI
追加
手順
書
承認依頼
2次レビュー
マネージャー
動作確認結果
もあるから、
判定しやすい

# 過去のPNIの事例をもとに
# ユースケース評価やってみた

このシステムができたのが直前で、ちょうどいいタイミングのPNI案件がなかったので過去事例をなぞってみました。。。

# まずは、作業前の商用環境をコピーする

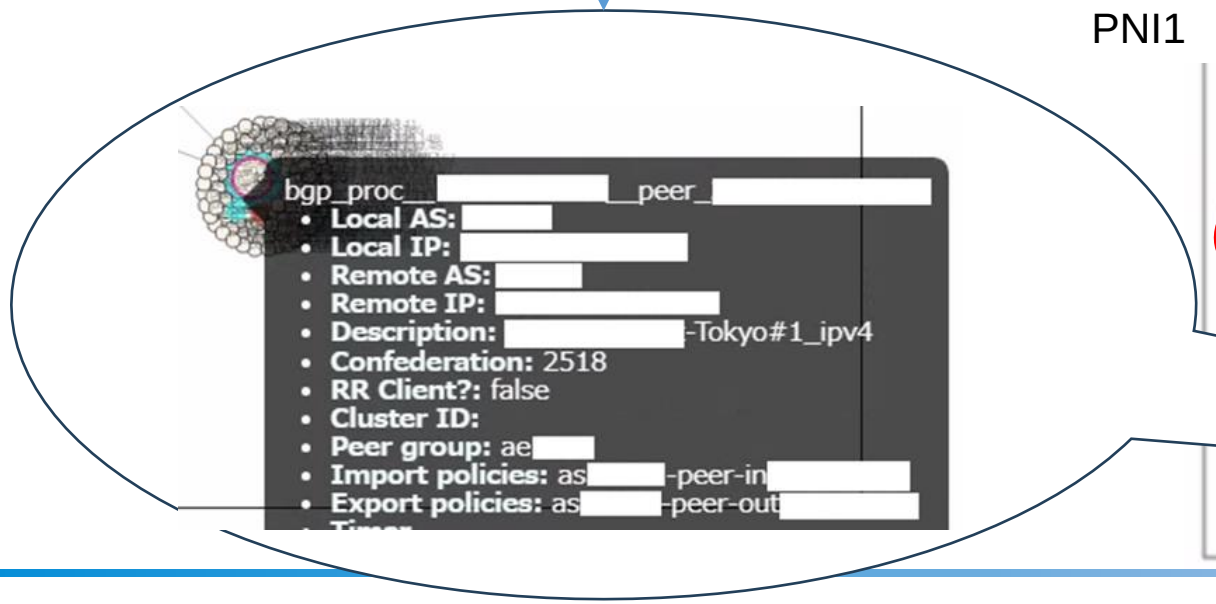
コピーする操作の流れはAppendixを参照してください。

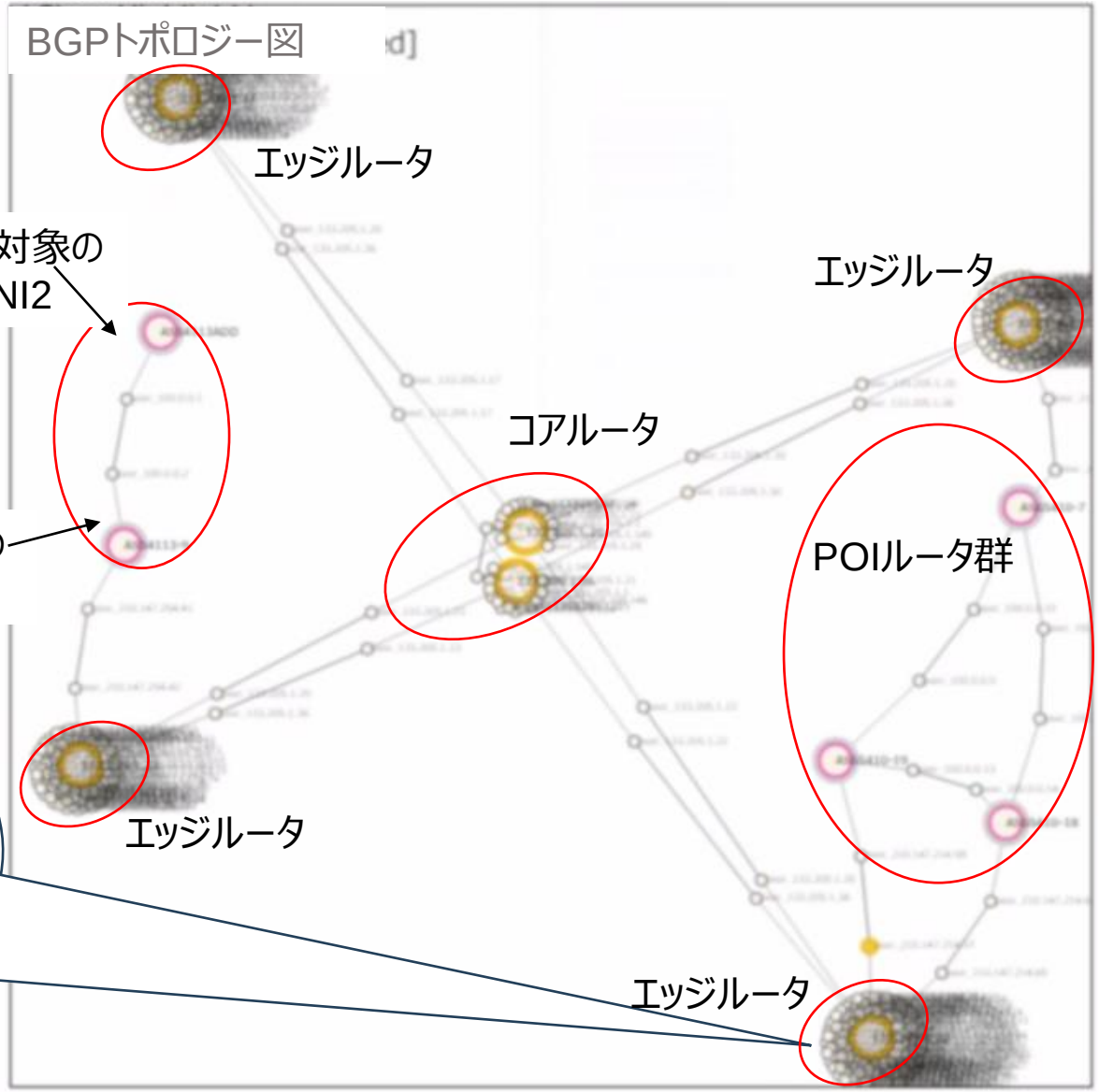
# 商用環境の”コピー”できたトポロジー図(BGP)

PNIユースケースとして
PNIのAS番号とPOIのAS番号を指定することで
外部ASの自動補完機能が動作する。

指定したAS番号のeBGP構成を含んだ
BGPトポロジーが自動生成される。

エッジルータ部分の拡大図
…eBGPネイバーの端点を
選択すると、属性情報
が参照できる



# 商用環境の”コピー”できたトポロジー図(Layer3)

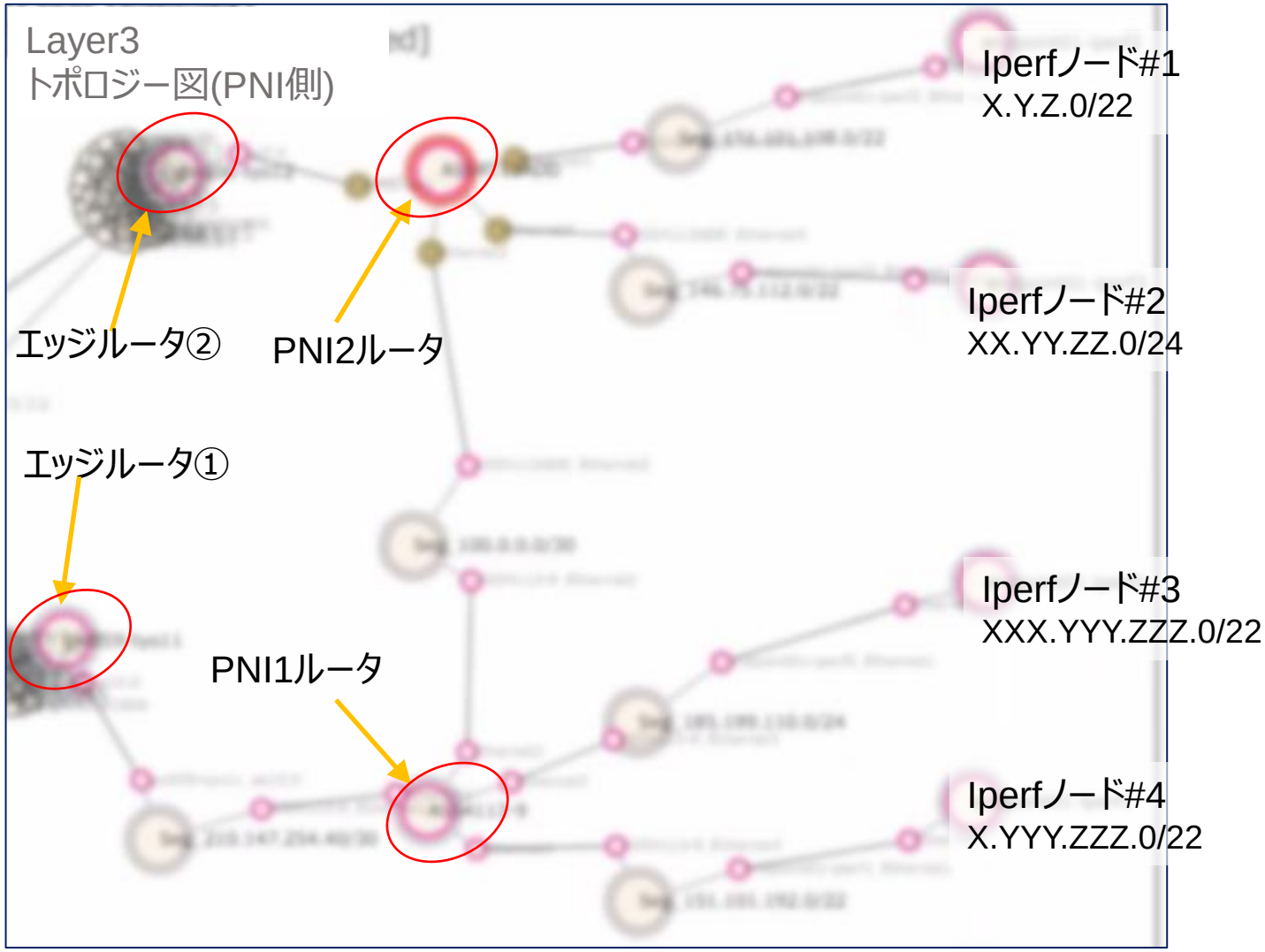
Layer3のトポロジー図では
ユースケーステストで必要となるE2Eトラフィックを
印加するIperfノードの構成も生成される。
(Flowデータをもとに生成される)

このユースケースでは、
エッジルータ②とPNI2ルータのインターフェース間の
接続は済んでいるため、トポロジー図としては
Layer3のリンクが存在している。

POI側の構成も同様に、
Iperfノードが自動生成されている。

BGP Flowデータのサンプル

| 送信元Prefix | 送信先Prefix | rate[Mbps] |
|---|---|---|
| X.Y.Z.0/22 | A.B.C.0/24 | 85793 |
| XX.YY.ZZ.0/24 | A.D.C.0/24 | 2738 |
| XXX.YYY.ZZZ.0/22 | A.B.E.0/24 | 21955 |
| X.YYY.ZZZ.0/22 | A.C.D.0/24 | 1044 |



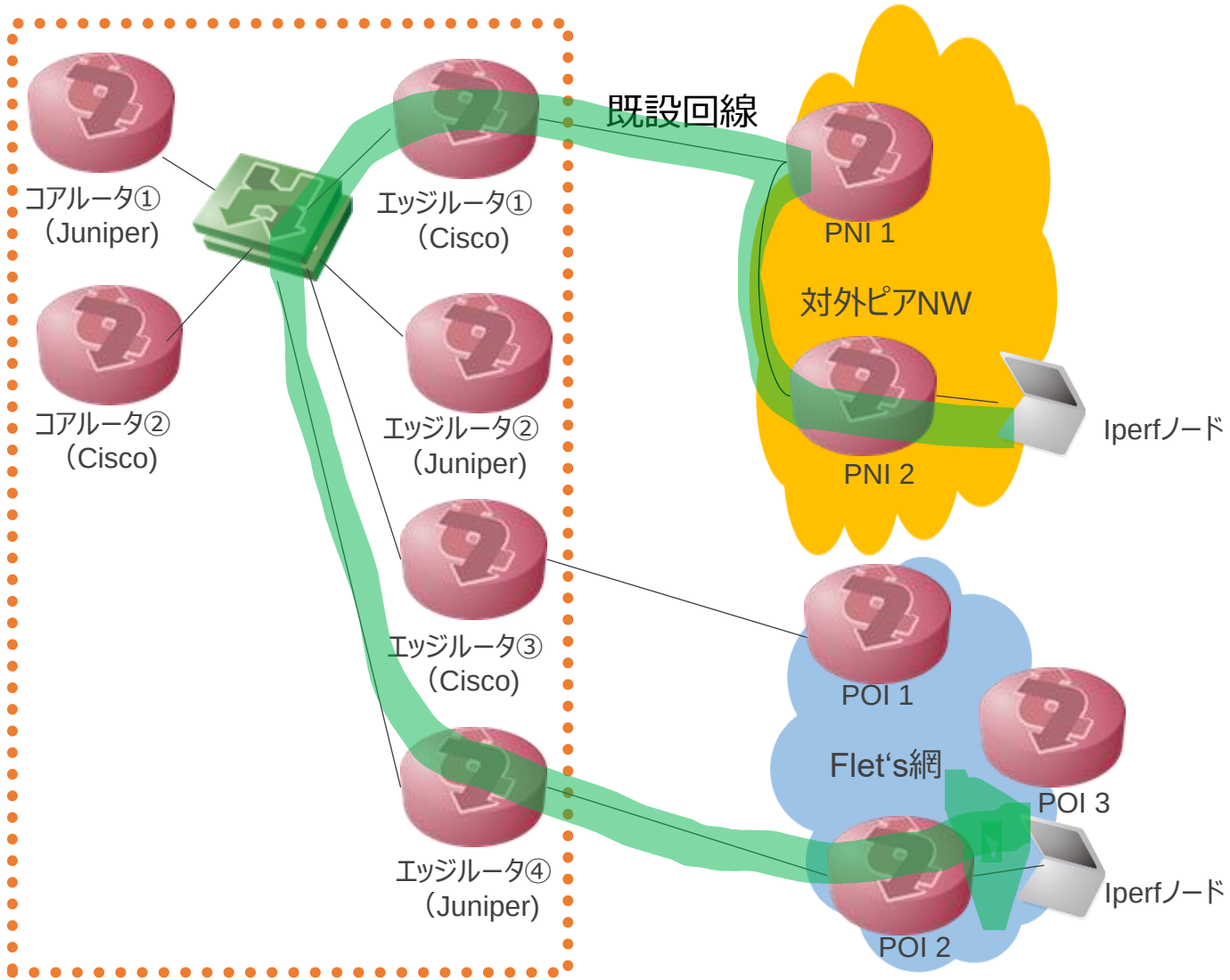
# 作業前の商用環境を”コピー”した構成の動作確認

・PNI 2のIperfノードから
POI 2のIperfノードへのTraceroute 結果

```
1  PNI 2 0.064 ms  0.012 ms  0.010 ms
2  PNI 1 0.024 ms  0.015 ms  0.013 ms
3  エッジルータ① 0.035 ms  0.018 ms  0.017 ms
4  エッジルータ④ 0.035 ms  0.021 ms  0.021 ms
5  POI 2 0.082 ms  0.029 ms  0.028 ms
6  POIのIperfノード 0.044 ms  0.164 ms  0.056 ms
```

・IperfのE2Eトラフィックも欠損なく転送されている。





# ”コピー環境”を操作して作業手順を確認する

**手順レビュー内容の抜粋**

・PNI 2向けのBGPネイバーの設定はないことを確認

```
$ docker exec -it clab-emulated-エッジルータ② cli
root@エッジルータ②> show bgp summary | match PNI 2のアドレス
root@エッジルータ②>
```

・エッジルータ②へ設定投入
  ・BGP Policyの設定
  ・BGP neighborの設定

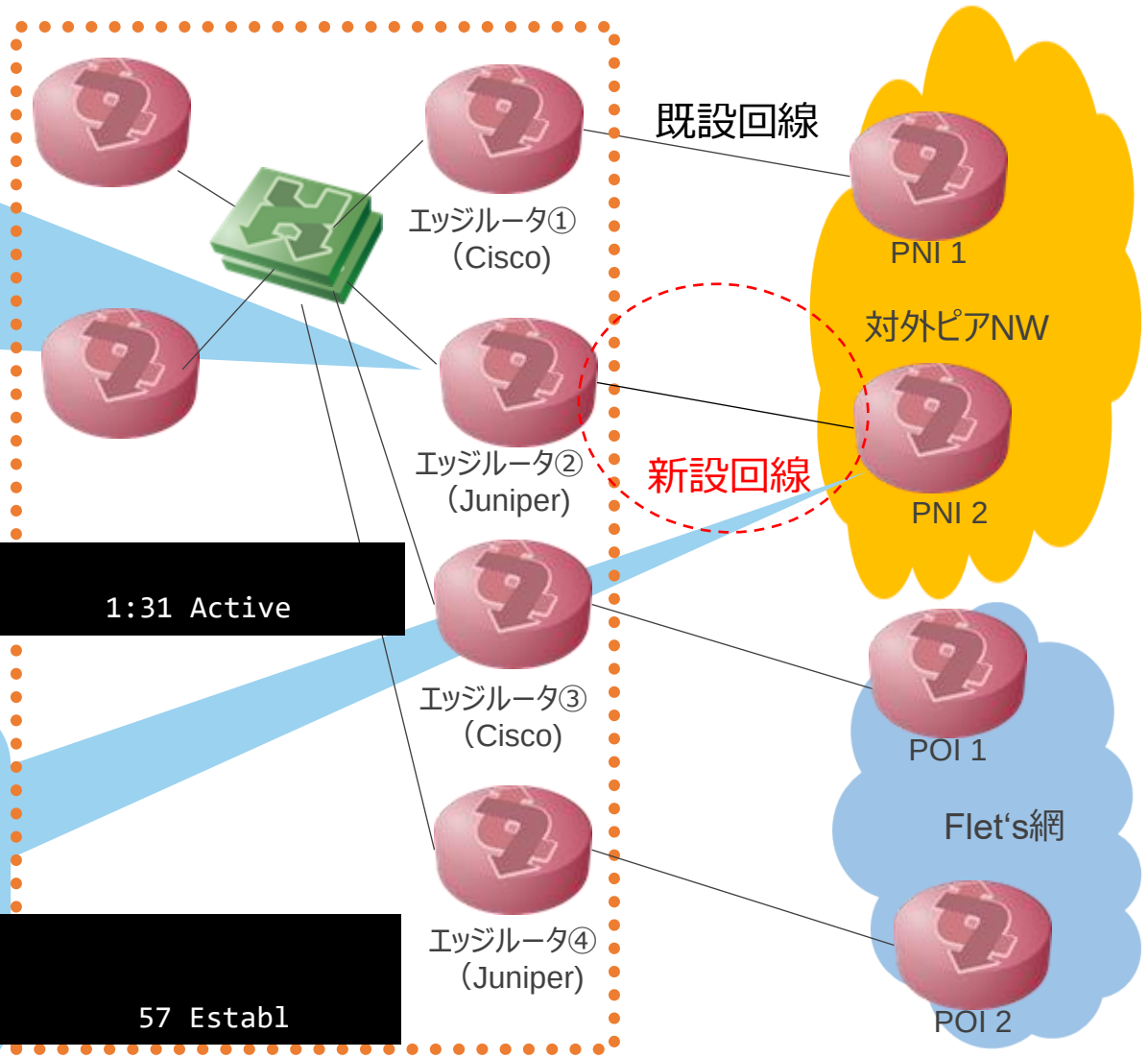
・ピアはActive状態

```
root@エッジルータ②> show bgp summary | match PNI 2のアドレス
PNI 2のアドレス          PNIのAS番号          1          0       0       0        1:31 Active
```

・PNI 2ルータへ設定投入
  ・BGP Policyの設定
  ・BGP neighborの設定

・ピアがEstablになっていることを確認

```
docker exec -it clab-emulated-エッジルータ② cli
root@エッジルータ②> show bgp summary | match PNI 2のアドレス
PNI 2のアドレス          PNIのAS番号          6          5       0       0          57 Establ
```



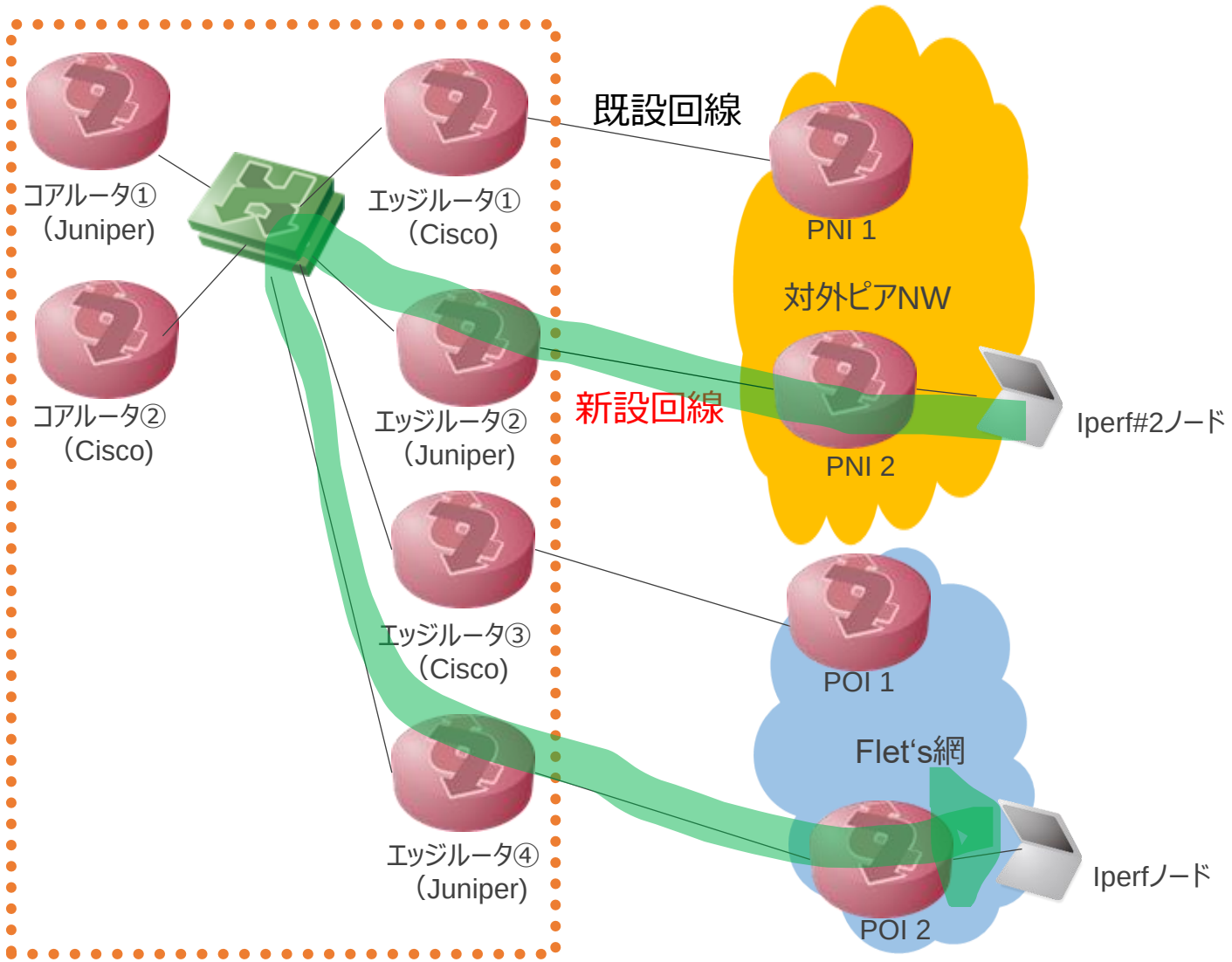
# ”コピー環境”を操作して作業手順を確認する

設定作業後の経路確認

・PNI 2のIperfノードから
POI 2のIperfノードへのTraceroute 結果

```
1  PNI2  0.100 ms  0.018 ms  0.014 ms
2  エッジルータ②  0.030 ms  0.015 ms  0.014 ms
3  エッジルータ④  0.032 ms  0.020 ms  0.018 ms
4  POIのIperfノード  0.038 ms  0.024 ms  0.030 ms
```

エッジルータ①経由からエッジルータ②経由で
トラフィックの経路が変わっていることを確認



# 操作後の状態確認してレビュー結果の判定

作業手順における作業後の事後確認のチェック項目が“コピー環境”ですべて確認できた

# 対象BGPネイバーの状態を確認
show bgp summary | match A.A.A.A
☑BGP sessionが確立していることを確認
☑AS番号(左から2番目の数字)がAAAAであることを確認

# 送信経路を確認
show bgp neighbor A.A.A.A | match "Advertised prefixes“
☑フルルートを送信していないことを確認

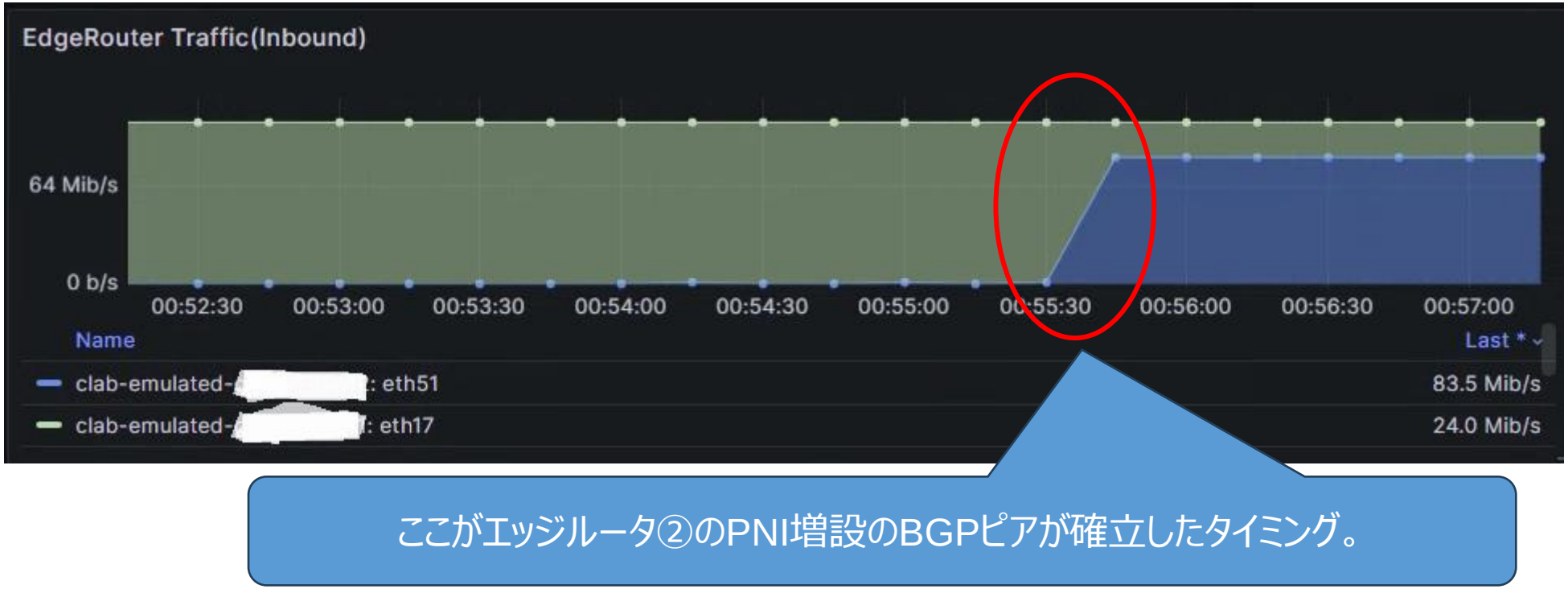
# 受信経路を確認
show bgp neighbor A.A.A.A | match "Active prefixes"
☑フルルートを受信していないことを確認

show route source-gateway A.A.A.A terse
☑as-pathが AAAA I のprefixを確認
　☑ネイバーASのOrigin経路のLPがX00、MEDがZ00であることを確認
☑as-pathが AAAA AAAA以外のAS I のprefixを確認
　☑ネイバーAS配下のASの経路のLPがY00、MEDがX00であることを確認

# 対象ネイバーとのBGP設定を確認
show bgp neighbor A.A.A.A | match "Description|Export|Import"
☑Description: PNI#2_ipv4
☑Export: [ asAAAA-peer-out ] Import: [ asAAAA-peer-in]

# 操作後の状態確認してレビュー結果の判定

経路切り替えの際にもトラフィック欠損が発生しないことも確認できた




**→”コピー環境”上で、**
**作業手順の正当性を判定できるチェックポイントをすべて満たすことができた**

# 評価結果のまとめ

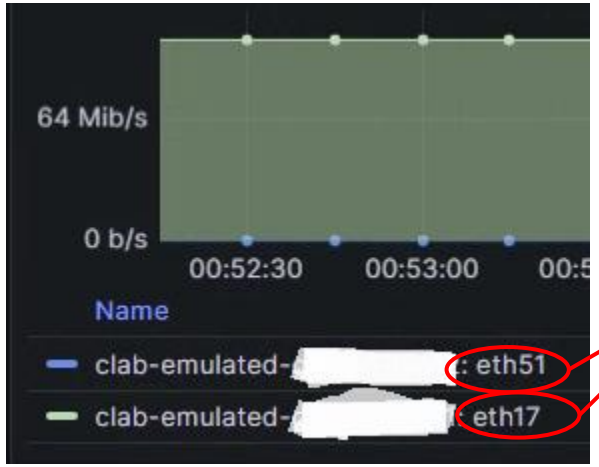
# バックボーンチームメンバーからのコメント

- コンフィグおよび再現試験にて機械的にチェックできるようになるためレビューの正確性は上がる。
  また、人の経験に依存しないチェックができるので、経験が浅い人でもpeer作業できるようになりそう。
  ただ、手順書作成がまだ手動なので時間短縮にはならないと思う。
- 対向からどう動くかの振る舞いも見れるので(変な経路漏らしていないか、など)、
  そのチェック項目が作れて作業品質がさらに向上できる。
- 実機は評価環境しかなくStaging用途のものはないから、
  実環境を再現できるのはこの”コピー環境”がBIGLOBEとしては唯一なので、
  このシステムがあるだけでも非常に有用。
- Prefixを削って他PNIやIXピアへトラフィックを移植する場合は、現状はベテランの感覚値になっている。
  これがロジカルにシミュレートできるので、フルルートを扱った時の振る舞いにも対応してくれるのを期待してる
- BGPポリシーの妥当性判断として、障害試験して冗長できているのかが確認もできるので便利。
- 非定型作業の動作シミュレーションして本番手順を作るのにも使えそう。

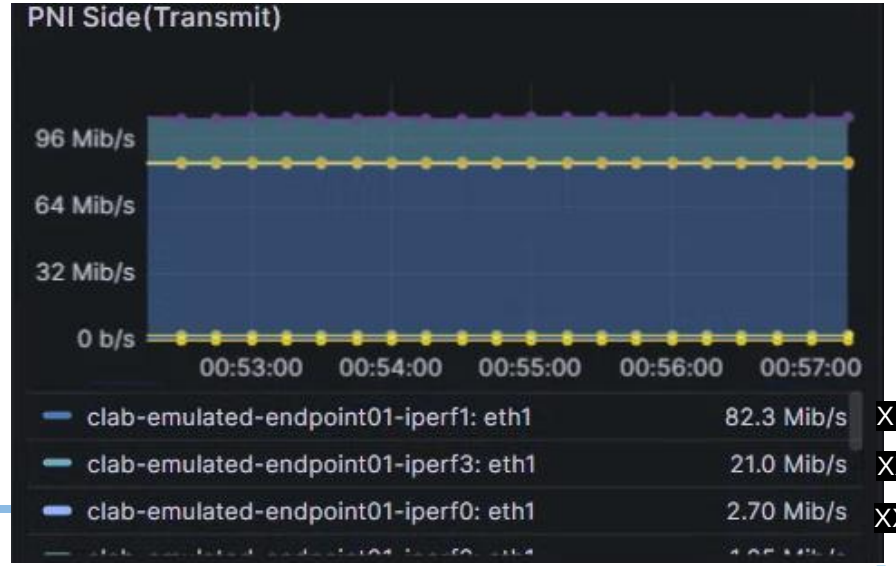
バックボーンメンバーからは1st STEPとしてのPNIユースケースの評価は良かった
2nd STEPでPublicピア, 3rd STEPにてTransitピアのユースケースに
進めても効果は十分期待できるとわかった。

# できなかったこと

- IOSXRのポリシーをすべてJunosポリシーに変換できなかった。
  …PNI向けのシンプルなポリシー構造のみフォーカスしてコンバートさせる形となった
- 外部AS補完する際に、本番だとPNIだけでなくPublic(IX)ピアもあり、
  まだシステムとしてPublicピアの1対多のネイバー関係が対応していないため、
  Publicピアは丸ごと除外してしまっている。
- “コピー環境”はJuniperのコンテナルータ(cRPD)に統一しているが、ほかのルータOSも対応して
  OS固有の振る舞いも再現できるようにしたかった
  …現状はciscoが対象だった場合は作業内容確認時の”コピー環境”への設定コマンドをjunos向けに変換する処理を作って対応
- “コピー環境”のGrafanaのトラフィックグラフにおいて、IF名がコンテナルータのIF名となっていて、
  OriginalのIF名にはできなかった。
  IperfノードはNWアドレスも見れるようにして、どのPrefixがどれだけのトラフィック量を持っているかわかるようにしたい。



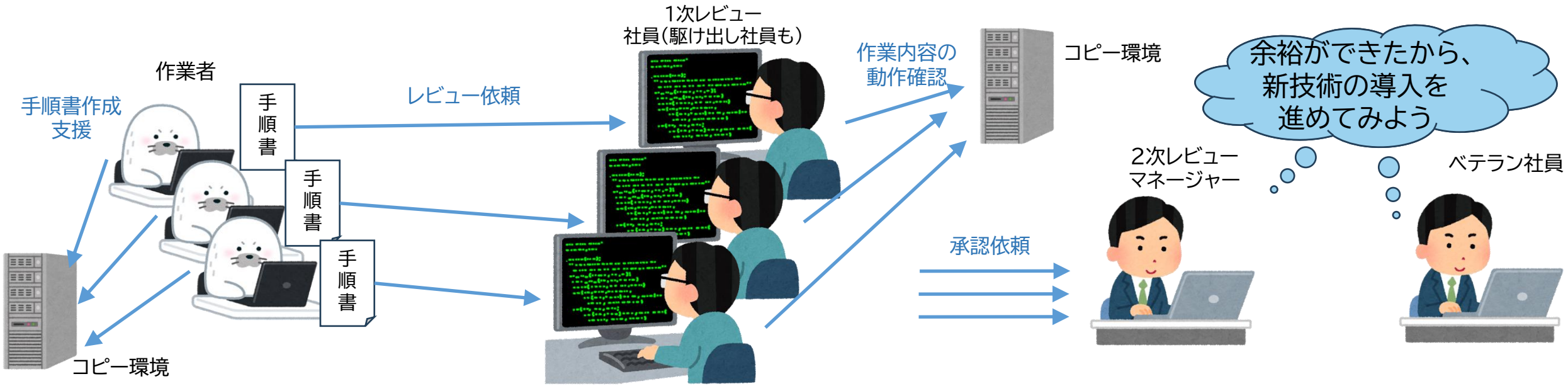

コピー環境では
cRPDのIF名ethXに
なっており、Original
なIF名でないとわかり
にくい…

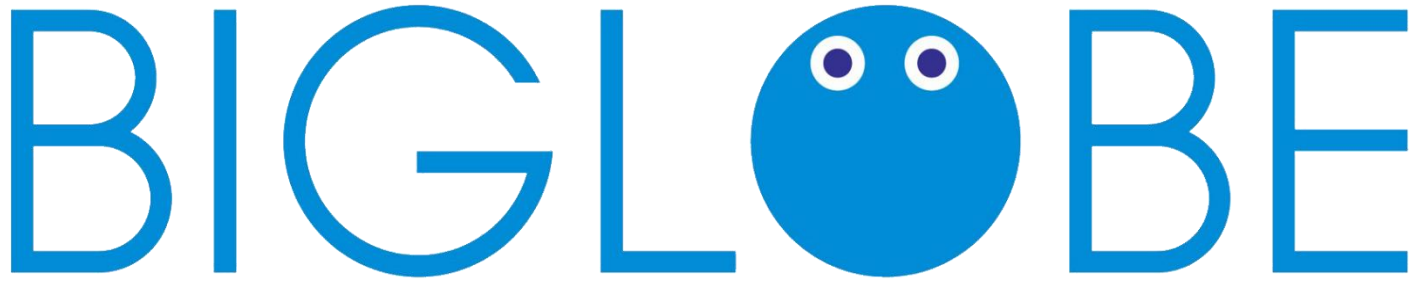


# これからやっていきたいこと

- Public(IX)ピア/Transitピアの対応
  …PNIよりも複雑なケースが多く、非定型作業になることがおおいため、対応できればベテランへの負荷集中をより削減できる。
- Junos/Cisco以外のネットワークベンダーの拡張
  …2社以外のNWベンダーのルータの導入も進んでおり、運用に合わせて対応ベンダーも増やせる形にしたい。
- “コピー環境”でドライランした結果を商用環境向けに設定コンフィグを提案する機能の実装。

→これらを実現して、対応できるユースケースを拡大していき
人も業務もスケールできるISPバックボーン業務を作り上げていきたい。




BIGLOBE

# Appendix-1 コピーの流れ

# 事前準備

トライアル向けの環境変数の情報を設定する
- SOURCE_ASはPNIのAS番号
- DEST_ASはPOIのAS番号

それ以外の項目は以下URLの
本システムのGithub参照
https://github.com/ool-mddo/playground/blob/main/demo/copy to emulated env/doc/provision.md

```
cat demo_vars
<省略>

# all steps: target network name
NETWORK_NAME="biglobe_deform"
NETWORK_INDEX="${NETWORK_NAME}_index.json"

# step2-2, preffered peer parameter (use original_asis
node/interface name)
PREFERRED_NODE="edge-tk01"
PREFERRED_INTERFACE="ge-0/0/3.0"
EXTERNAL_ASN=65550

# step1-2
SOURCE_AS=XXXX
DEST_AS=YYYY
```

トライアルのシステムを起動する

```
$ docker compose up -d
[+] Running 9/9
✓ Network playground_default                Created     0.1s
✓ Container playground-batfish-1            Started     0.1s
✓ Container playground-bgp-policy-parser-1  Started     0.1s
✓ Container playground-netoviz-1            Started     0.1s
✓ Container playground-batfish-wrapper-1    Started     0.0s
✓ Container playground-fish-tracer-1        Started     0.0s
✓ Container playground-netomox-exp-1        Started     0.0s
✓ Container playground-model-conductor-1    Started     0.0s
✓ Container playground-api-proxy-1          Started     0.0s
```

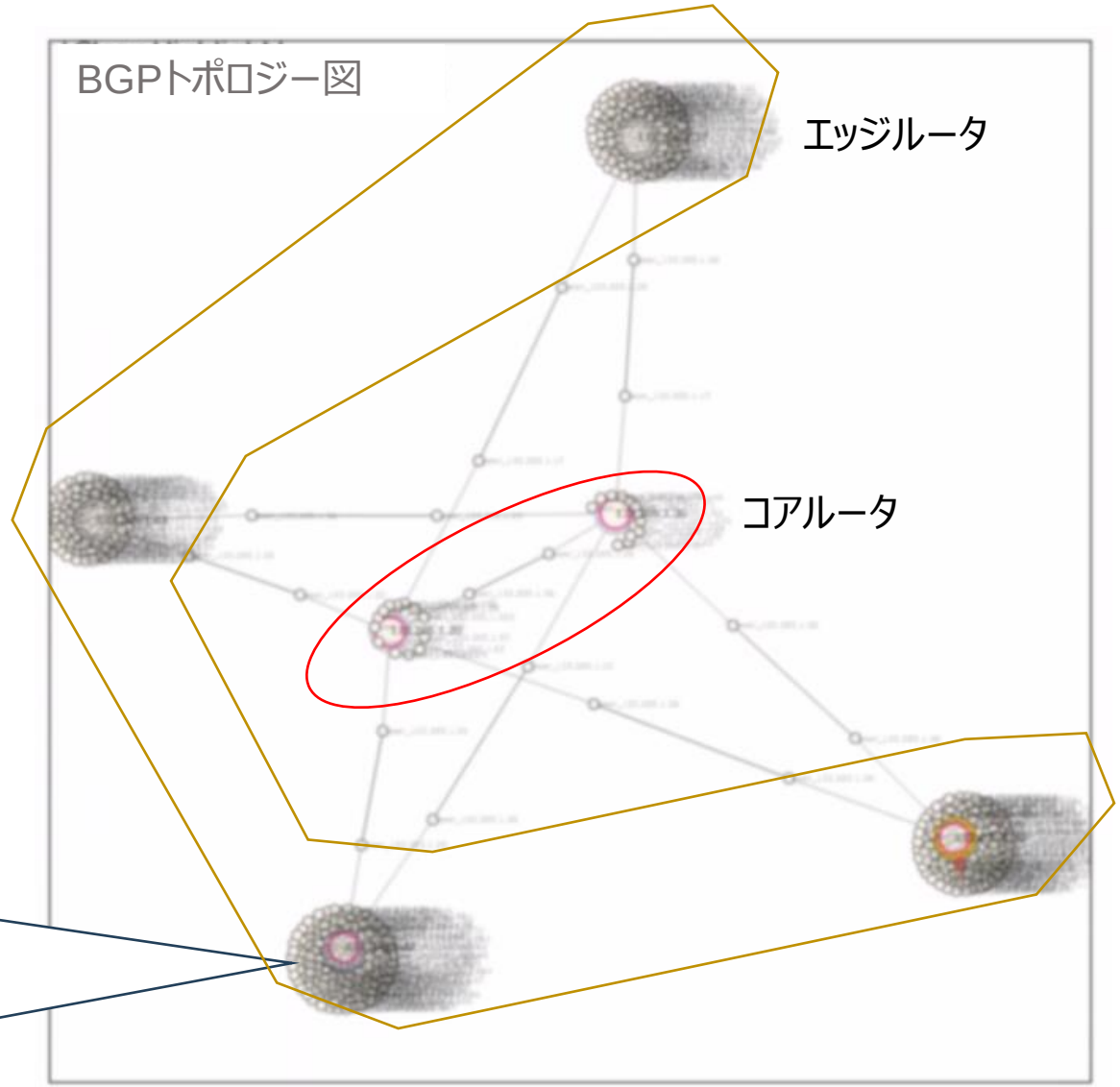
# 作業前の本番環境を再現 – Step1-1

**①本番コンフィグを本システム内に配置し、**
**BIGLOBEのトポロジーモデルを生成するスクリプトを実行**

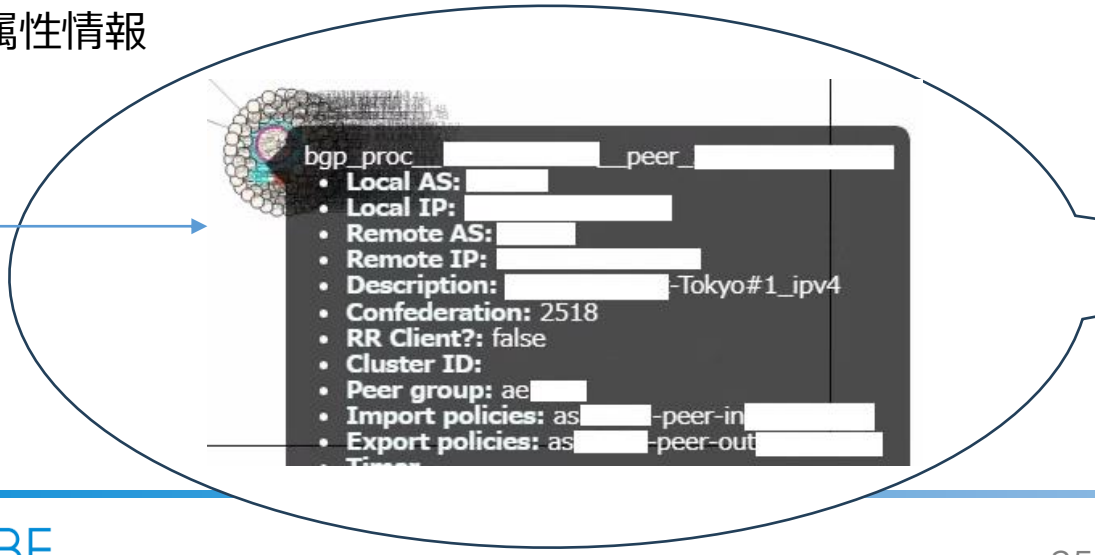
```
$ bash demo_step1-1.sh
```

生成された本番構成のトポロジーを確認してみると

- BGPのトポロジー図はiBGPトポロジのみが生成されている
- 外部ASに関しては、対向機器（他社機器）のコンフィグがないので、エッジルータ側のeBGPネイバーの端点のみが表示される




エッジルータ部分の拡大図
…eBGPネイバーの端点を選択すると、属性情報が参照できる



# 作業前の本番環境を再現 - Step1-2

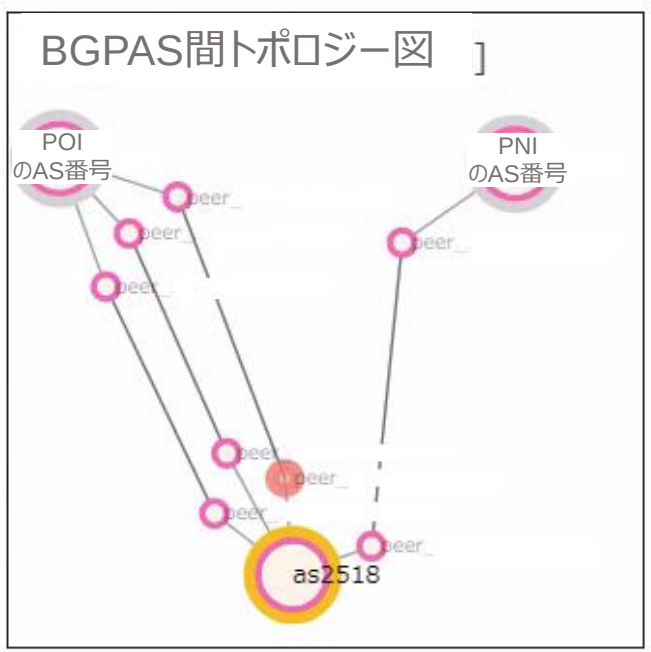
**②次に外部ASの情報を補完**
**　補完の際に以下の情報を追加する**

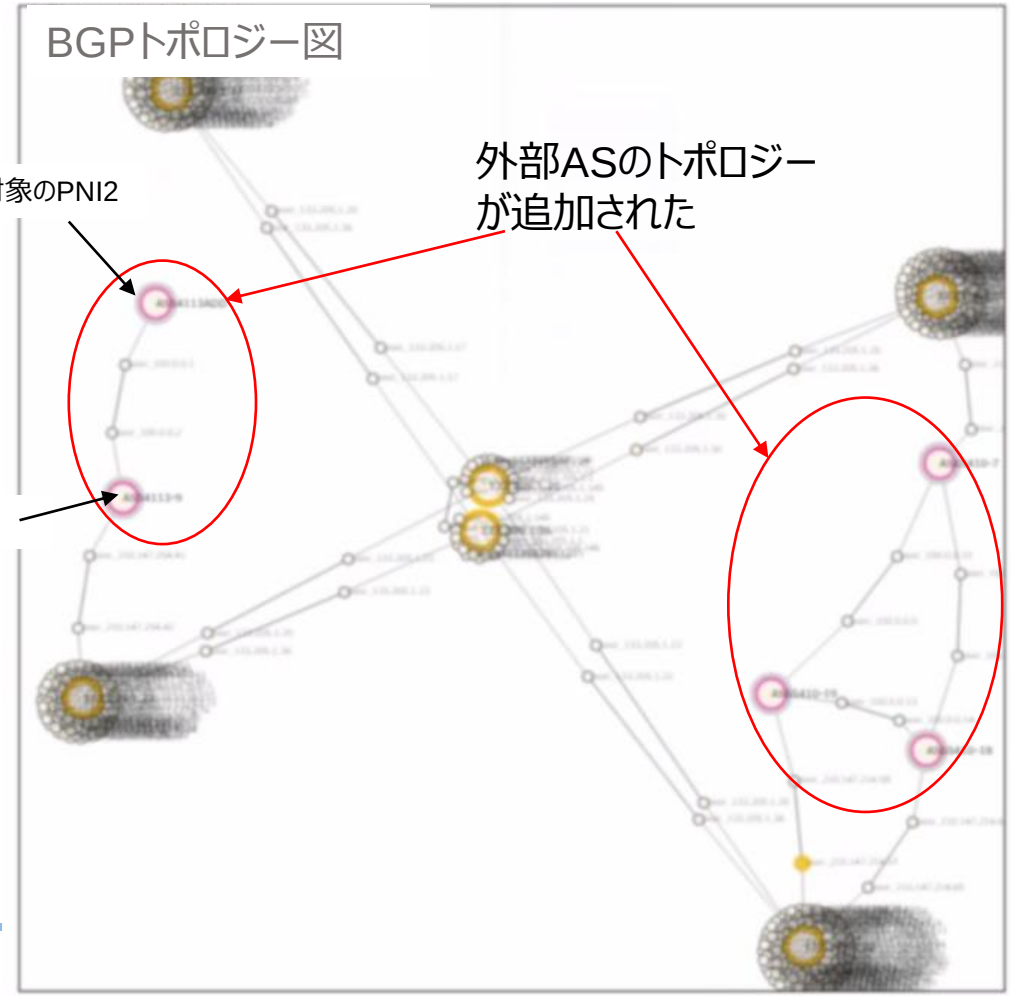
- BGPのFlowデータ収集ツールから移設対象の外部ASにおける送信元・送信先Prefixごとの通信量データを収集したリスト
- 増設するPNIの情報（IF情報、対向BGPのネイバーアドレス）の定義リスト
- 除外したいBGPネイバーを定義（理由は後述）したリスト

**③外部AS補完スクリプトを実行する**

```
$ bash demo_step1-2.sh
```

→BGPのAS間トポロジーおよびeBGPのトポロジーが生成させる

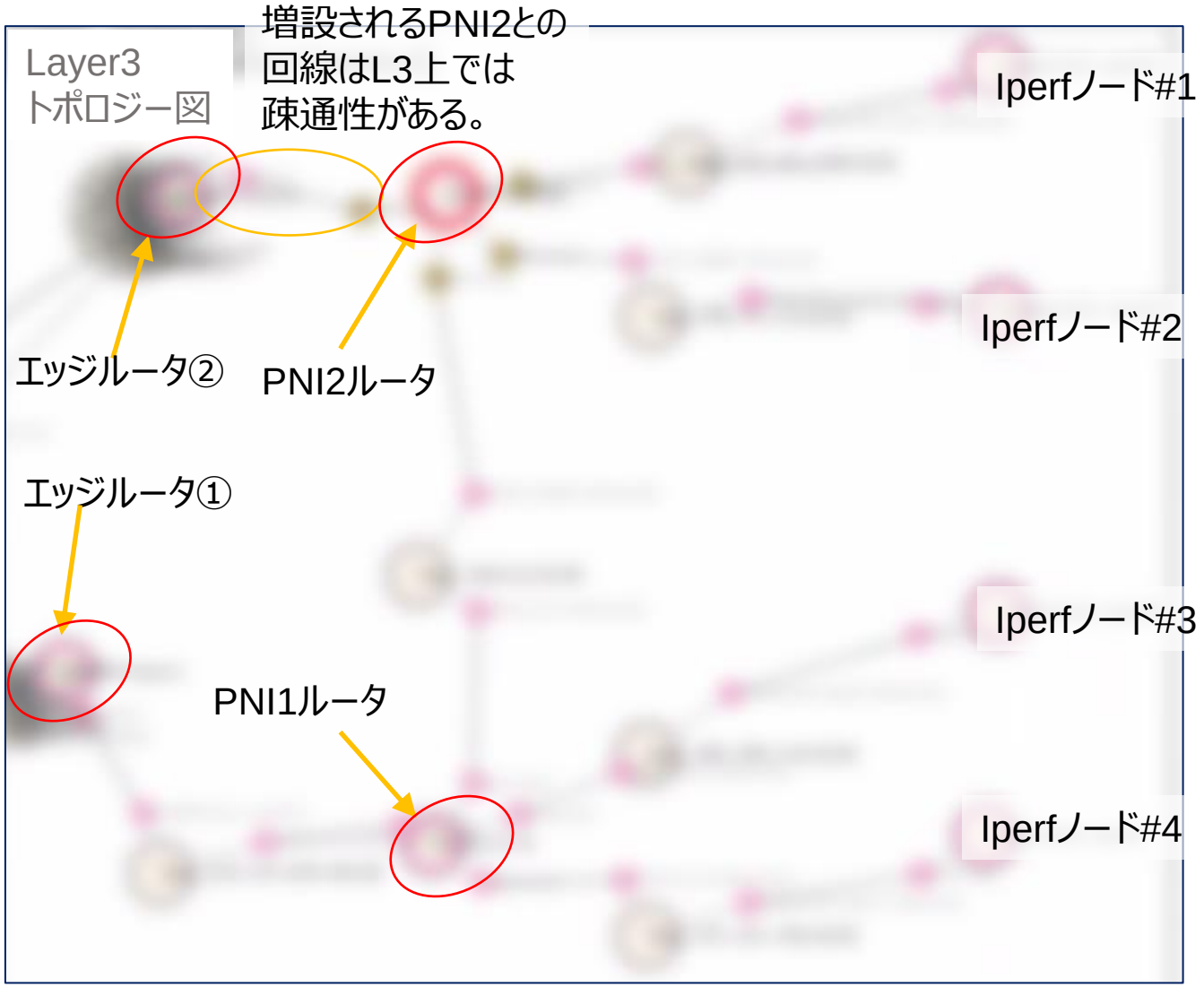


# 作業前の本番環境を再現 - Step1-2

・Layer3のトポロジー上は
増設するPNI2との回線はStep1-2スクリプトで
addl3.csvの情報をもとにエッジルータ②→PNI 2
へのL3のリンクが自動生成される。

・IperfノードはFlowデータの
送信元PrefixごとにIperfノードを各PNIへ、
および送信先PrefixごとにIperfノードを各POI
のルータへ自動で配置される



# Step1-2の追加情報ファイル

- 除外したいBGPネイバーを定義したリスト

```
# except.csv
except_peer
AAA.BBB.CCC.DDD
```

- 増設するPNIの情報（IF情報、対向BGPのネイバーアドレス）の定義リスト

```
# addl3.csv
srcrouter,srcif,srcaddress,peeraddress,netmask,peeras,srcas
Edge-RT#1,aeX.X,A.B.C.D,AA.BB.CC.DD,30,AAAAA,2518
```

- BGPのFlowデータ収集ツールから移設対象の外部ASにおける送信元・送信先Prefixごとの通信量データを収集したリスト

```
# flowdata.csv
source,dest,rate,dstAS
X.Y.Z.0/22,A.B.C.0/24,85793,AAAAA
XX.YY.ZZ.0/24,A.D.C.0/24,2738, AAAAA
XXX.YYY.ZZZ.0/22,A.B.E.0/24,21955, AAAAA
X.YYY.ZZZ.0/22,A.C.D.0/24,1044, AAAAA
```

# 作業前の本番環境を再現 – Step2-1

## ④生成したトポロジーデータからコンテナ基盤上にEmulated環境をデプロイする

```
$ bash demo_step2-1.sh
<省略>
PLAY RECAP *********************************************************************
docker-host                : ok=13   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
localhost                  : ok=55   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

## Containerlab上にEmulated環境の構成が起動されていることが確認できる

```
$ sudo containerlab inspect --topo clab/clab-topo.yaml
INFO[0000] Parsing & checking topology file: clab-topo.yaml
+----+------------------------------+--------------+--------------------------------+-------------+---------+-----------------+----------------------+
| #  |             Name             | Container ID |             Image              |     Kind    |  State  |  IPv4 Address   |     IPv6 Address     |
+----+------------------------------+--------------+--------------------------------+-------------+---------+-----------------+----------------------+
|  1 | clab-emulated-ASAAAAA-9      | b79a823289bb | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.3/24  | 2001:172:20:20::3/64  |
|  2 | clab-emulated-ASAAAAAADD     | c85862ed4a79 | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.12/24 | 2001:172:20:20::c/64  |
|  3 | clab-emulated-ASBBBBB-18     | 05368e9078d8 | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.14/24 | 2001:172:20:20::e/64  |
|  4 | clab-emulated-ASBBBBB-19     | 48ea314260ef | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.10/24 | 2001:172:20:20::a/64  |
|  5 | clab-emulated-ASBBBBB-7      | ed72383591df | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.9/24  | 2001:172:20:20::9/64  |
|  6 | clab-emulated-Core-01        | 468cf7bb9eec | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.15/24 | 2001:172:20:20::f/64  |
|  7 | clab-emulated-Core-02        | 147b0a655d8b | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.4/24  | 2001:172:20:20::4/64  |
|  8 | clab-emulated-endpoint01-iperf0 | 87e00a860fdd | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.5/24  | 2001:172:20:20::5/64  |
|  9 | clab-emulated-endpoint01-iperf1 | d951db70b8bc | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.11/24 | 2001:172:20:20::b/64  |
| 10 | clab-emulated-endpoint01-iperf2 | f12c2034b8d0 | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.2/24  | 2001:172:20:20::2/64  |
| 11 | clab-emulated-endpoint01-iperf3 | ad827e497a8e | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.16/24 | 2001:172:20:20::10/64 |
| 12 | clab-emulated-endpoint02-iperf0 | da4461baa5ae | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.8/24  | 2001:172:20:20::8/64  |
| 13 | clab-emulated-endpoint02-iperf1 | 4b4092e07a6d | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.7/24  | 2001:172:20:20::7/64  |
| 14 | clab-emulated-endpoint02-iperf2 | ed93bedea93e | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.19/24 | 2001:172:20:20::13/64 |
| 15 | clab-emulated-endpoint02-iperf3 | 80cc7d7aa8c5 | ghcr.io/ool-mddo/ool-iperf:main | linux       | running | 172.20.20.13/24 | 2001:172:20:20::d/64  |
| 16 | clab-emulated-Edge-01        | 405e94b03a9f | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.20/24 | 2001:172:20:20::14/64 |
| 17 | clab-emulated-Edge-02        | fcc1118cec96 | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.18/24 | 2001:172:20:20::12/64 |
| 18 | clab-emulated-Edge-03        | a33eba758d13 | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.6/24  | 2001:172:20:20::6/64  |
| 19 | clab-emulated-Edge-04        | f6a774bdc698 | crpd:22.1R1.10                 | juniper_crpd | running | 172.20.20.17/24 | 2001:172:20:20::11/64 |
+----+------------------------------+--------------+--------------------------------+-------------+---------+-----------------+----------------------+
```

# 作業前の本番環境を再現 – Step2-2

**⑤デプロイしたEmulated環境上で、Flowデータをダウンサイズした値でトラフィックの負荷をかける**

```
$ bash demo_step2-2.sh
<省略>
TASK [Iperf client command] ***************************************************
changed: [docker-host] => (item=[{'node': 'endpoint02-Iperf0'}, {'source': 'endpoint01-Iperf3', 'dest_address': 'A.A.A.100', 'number': 5201, 'rate': '21955'}])
changed: [docker-host] => (item=[{'node': 'endpoint02-Iperf1'}, {'source': 'endpoint01-Iperf0', 'dest_address': 'B.B.B.100', 'number': 5201, 'rate': '2738'}])
changed: [docker-host] => (item=[{'node': 'endpoint02-Iperf2'}, {'source': 'endpoint01-Iperf1', 'dest_address': 'C.C.C.100', 'number': 5201, 'rate': '1044'}])
changed: [docker-host] => (item=[{'node': 'endpoint02-Iperf3'}, {'source': 'endpoint01-Iperf2', 'dest_address': 'D.D.D.100', 'number': 5201, 'rate': '85793'}])
PLAY RECAP ********************************************************************
docker-host                : ok=6    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Prometheus/Grafanaも起動しており、各コンテナのトラフィック量を確認できる。

→PNIの外部ASルータからの送信トラフィック量とPOI側の受信トラフィック量はほぼ一致しているので、正常にE2E通信できている。