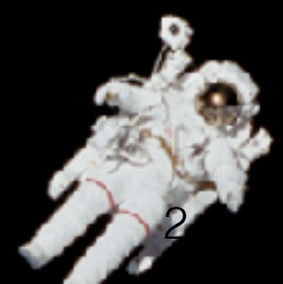
BGP経路問題発生時の行動を考えよう

AS？ なくても大丈夫だ

Shintaro Kojima コーダンス / @codeout

フリーランスの
ネットワークエンジニアです
とくに特定の組織に属してない



8/28、ブログ書きました

LGTM

Looks Good To Me

2017-08-28

パブリックデータから経路リークを探る

networking

2017/08/25 12:30 (JST) ごろ、日本国内で大規模な通信障害が観測されました。

通信障害の内容について、とても詳細にまとめられている記事があります。

8月25日に発生した大規模通信障害をまとめてみた - piyolog
2017年8月25日12時過ぎより、Webサイトにつながらない等の接続障害とみられる事象が複数発生しました。ま...

d.hatena.ne.jp ★☆25 77 users

d.hatena.ne.jp

障害の内容はさておき、このエントリでは障害のしくみについて探ってみようと思います。

プロフィール

ic:codeout

+ 読者になる 21

そこそこ読まれた

2017/08/28 06:45

itpro.nikkeibp.co.jp

あとで読む　google

ネットワーク　network

337 USERS

Linux コンテナの内部を知って ワンランク上のコンテナ遣いを目指そう / JTF20...

2017/08/27 12:54

speakerdeck.com

あとで読む　linux　docker

コンテナ

171 USERS

2017/08/27 22:47

choichoi.hatenablog.com

あとで読む　AI　Zoh　ネタ

1397 USERS

入社からの半年間でコードレビューで指摘されたことのまとめ - 30歳からのプロ...

2017/08/26 09:57

numb86-tech.hatenablog.c...

あとで読む　プログラミング

programming　開発

109 USERS

2017/08/27 16:04

magazine.rubyist.net

ruby　あとで読む　Rails

Hanami

急成長するサービスを支えるDevOps 戦略と組織変革へのアプローチ / Approach...

2017/08/27 15:56

speakerdeck.com

あとで読む　DevOps

monitoring　slides

132 USERS

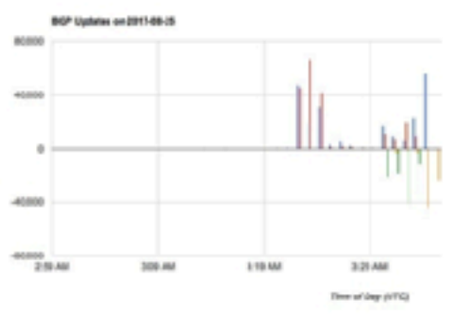
2017/08/28 00:18

www.publickey1.jp

あとで読む　Google　security

170 USERS

パブリックデータから経路リークを探る - LGTM



2017/08/28 01:16

codeout.hatenablog.com

ネットワーク　あとで読む

network　BGP

80 USERS

http://b.hatena.ne.jp/hotentry/it

格のちがい

http://b.hatena.ne.jp/hotentry/it

インターネットを米国視点でみてたり、障害を真横で観測できてるあたりがすごい

- "広報された宛先向けの通信が米国経由になった"
- "大量の経路広報を受信した"

https://www.attn.jp/maz/p/t/pdf/20170825-routeleakage.pdf

持たざる者も戦える

- 経路を記録してない
- 国際回線がない

場合でも、公開されている情報から障害を考え、対策することができるはず

モチベーション

大なり小なり経路障害の影響を受けた。

障害の原因を知って、次は止めたい。

という話をします

- 経路障害、どうやって調べる ？
- 8/25 のケーススタディ

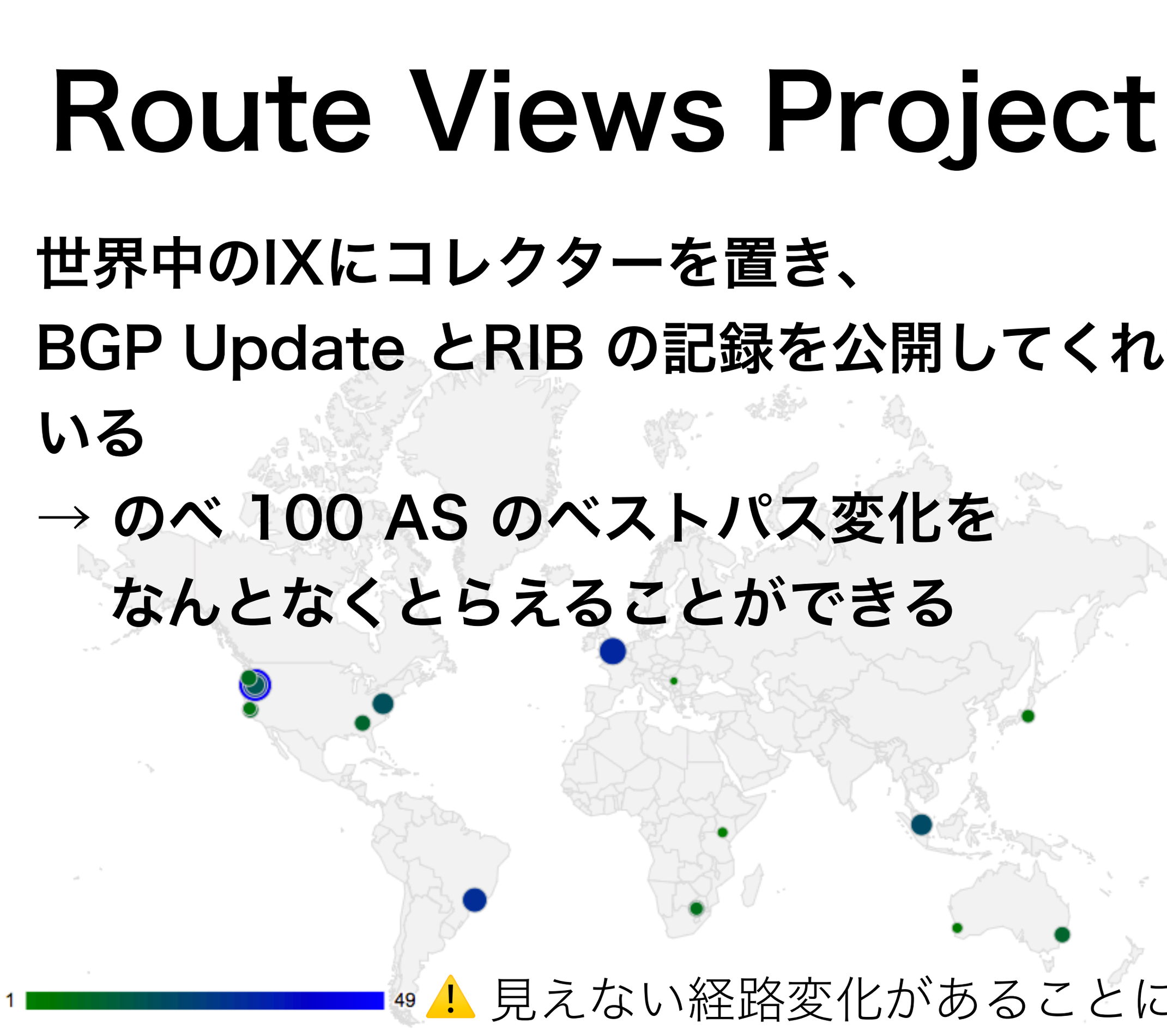
経路障害、どうやって調べる ？

データソース

- MRT Dump
  - RouteViews Project
  - RIPE RIS
- Looking Glass
- RIPEstat BGPlay
- AS 分析データ
  - caida AS Relationships

Route Views Project

世界中のIXにコレクターを置き、
BGP Update とRIB の記録を公開してくれている
→ のべ 100 AS のベストパス変化を
なんとなくとらえることができる



⚠️ 見えない経路変化があることに注意

1. MRT アーカイブを取ってきてPostgreSQL に入れる

というプログラムを書く

→ サンプル

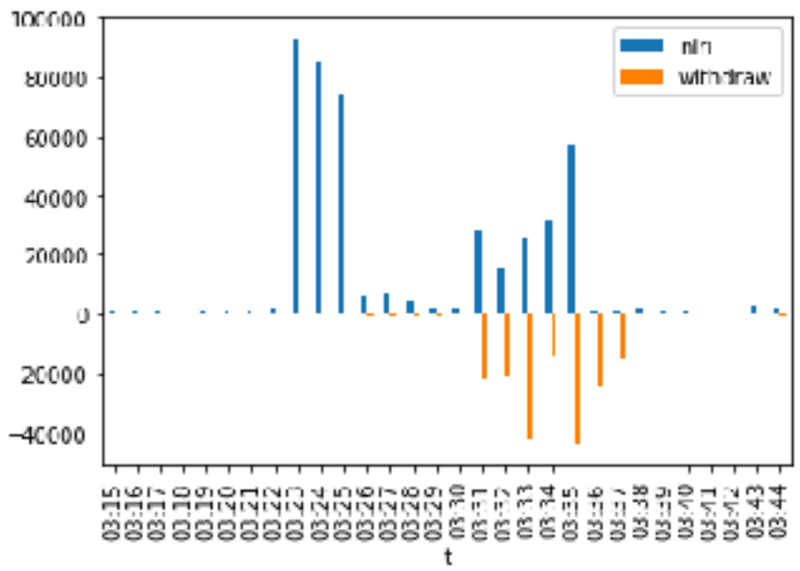
```
createdb -E UTF8 -T template0 route_leak
ruby route_views.rb migrate route_leak

for i in 0300 0315 0330 0345; \
  ruby route_views.rb update download 20170825.$i
ruby route_views.rb update load route_leak

ruby route_views.rb rib download 20170825.0200
ruby route_views.rb rib load route_leak
```

2. ひたすらSELECT

```
SELECT masklen(prefix) AS len, count(distinct prefix) \
FROM updates WHERE \
  ix='wide' AND neighbor_as=2497 AND aspath ='2497 701 15169 4713' AND \
  time > '2017-08-25 03:23'::TIMESTAMP AND \
  time < '2017-08-25 03:35'::TIMESTAMP \
GROUP BY len ORDER BY count DESC LIMIT 10;

 len | count
-----+-------
  24 | 16594
  22 |  3035
  23 |  2432
  21 |  1764
  20 |   868
  19 |    79
  16 |    29
  18 |    15
  17 |    10
  15 |     3
(10 rows)
```

実際は Jupyter Notebook とかで

In [6]:
```
import pandas as pd
import psycopg2
import matplotlib.pyplot as pp

conn = psycopg2.connect('dbname=route_leak')

df = pd.read_sql(sql="SELECT \
        SUBSTRING(CAST(time AS VARCHAR) ,12 ,5) AS t, count(*) AS NLRI \
        FROM updates \
        WHERE time >= '2017-08-25 03:15'::TIMESTAMP AND  time < '2017-08-25 03:45'::TIMESTAMP \
        AND ix = 'wide' AND withdraw IS NOT TRUE \
        GROUP BY t ORDER BY t', con=conn, index_col='t')
df['withdraw'] = pd.read_sql(sql="SELECT \
        SUBSTRING(CAST(time AS VARCHAR) ,12 ,5) AS t, count(*) AS NLRI \
        FROM updates \
        WHERE time >= '2017-08-25 03:15'::TIMESTAMP AND  time < '2017-08-25 03:45'::TIMESTAMP \
        AND ix = 'wide' AND withdraw IS TRUE \
        GROUP BY t ORDER BY t', con=conn, index_col='t')
df.plot.bar()
pp.show()
```

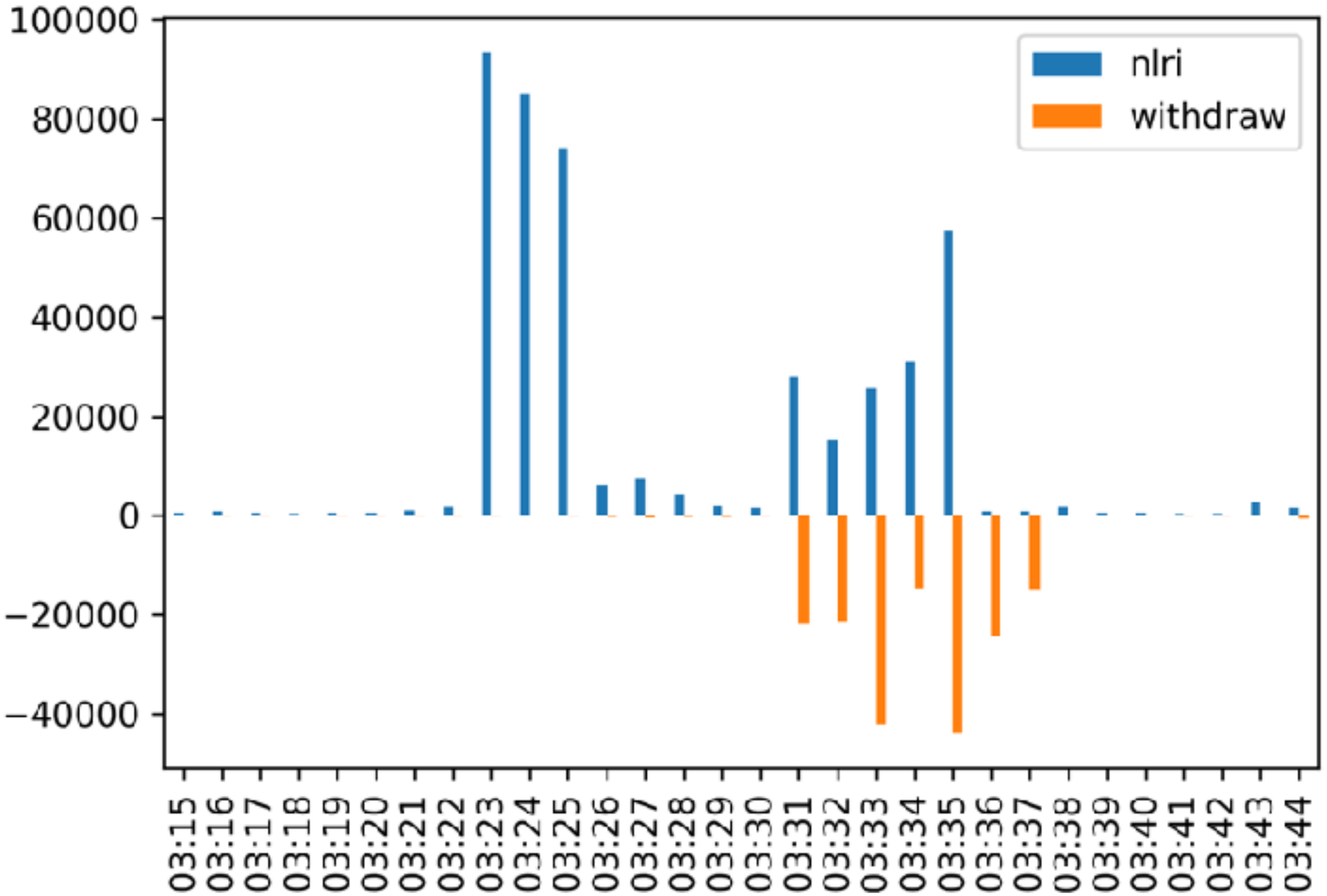


In [ ]:

持たざる者 視点でみえたこと

8/25 のケーススタディ

dix-ie で見えたアップデート数



03:23 ~ 03:35 (UTC) BGP Update が急増

BGP アップデートの中身 - Prefix数

```
route_leak=# SELECT count(distinct prefix) FROM updates
WHERE
  time >= '2017-08-25 03:23'::TIMESTAMP AND
  time < '2017-08-25 03:35'::TIMESTAMP AND
  ix = 'wide' AND withdraw IS NOT TRUE;
 count
--------
 122891
(1 row)

route_leak=# SELECT distinct count(distinct prefix) FROM updates
JOIN rib USING (prefix)
WHERE
  updates.time >= '2017-08-25 03:23'::TIMESTAMP AND
  updates.time < '2017-08-25 03:35'::TIMESTAMP AND
  updates.ix = 'wide' AND rib.ix = 'wide' AND withdraw IS NOT TRUE;
 count
-------
 30972
(1 row)
```

122,891 - 30,972 = 91,919 新規 Prefix

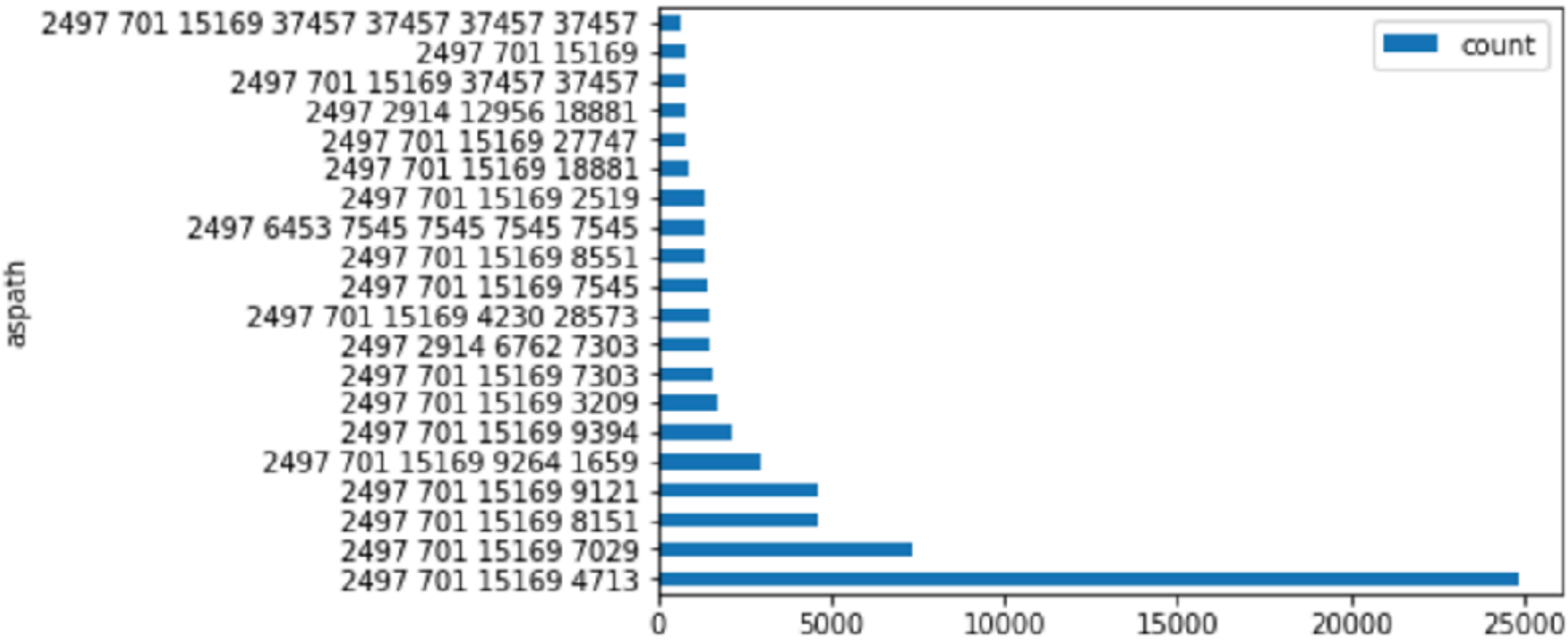
観測 1

・日本近辺で、+9万経路ふえた

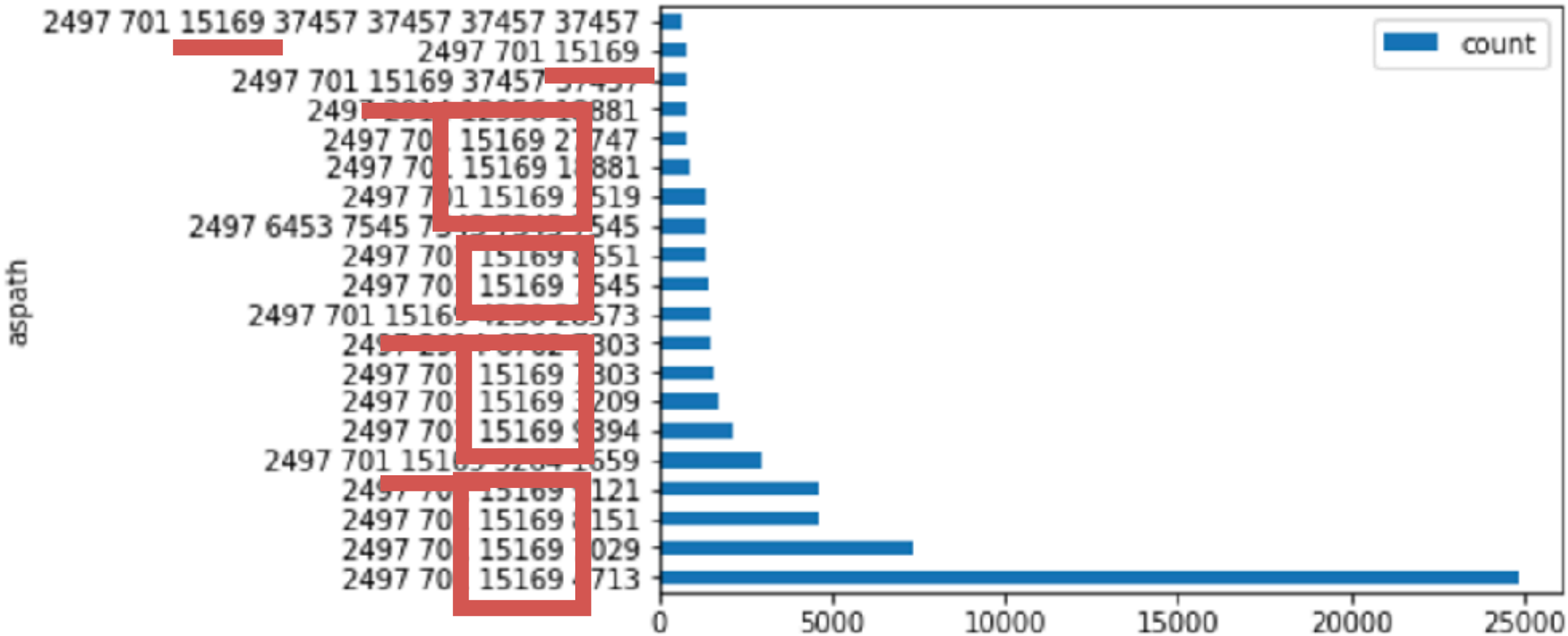
起こったであろうこと:

- 経路急増による負荷
- 経路急増によるRIB / FIB あふれ

dix-ie (AS2497経由)で見えた、AS_PATH あたりのPrefix数

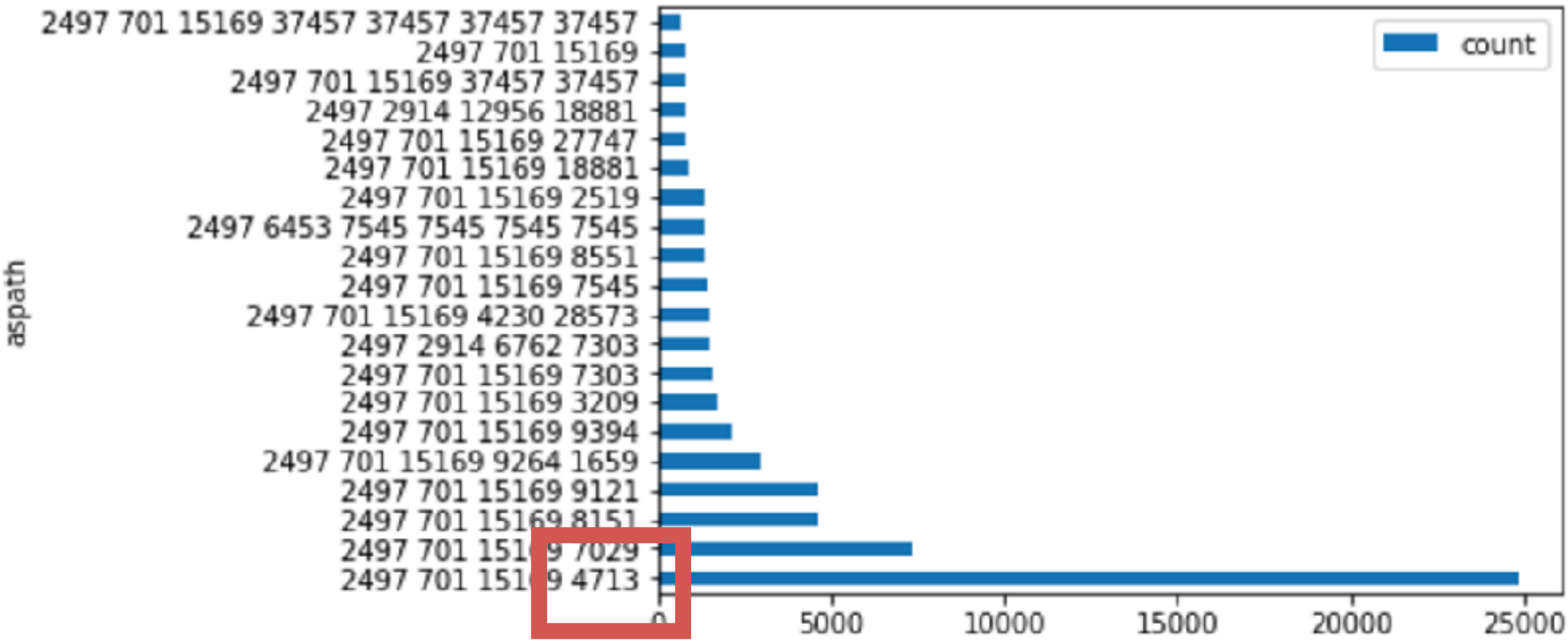


dix-ie (AS2497経由)で見えた、AS_PATH あたりのPrefix数



なんとなく、AS15169 が挟まっている？

dix-ie (AS2497経由)で見えた、AS_PATH あたりのPrefix数



圧倒的 4713 😱

2497 701 15169 4713

- Google(15169) がOCN(4713) をトランジットしているのはおかしい
- そのほかの AS関係はわからない 🤔

→ 重要なことなので、推測したい

2497 701 15169 4713

```
route_leak=# SELECT aspath, count(aspath) FROM updates WHERE aspath ~ '701
15169' GROUP BY aspath ORDER BY count DESC;
                          aspath                          | count
----------------------------------------------------------+--------
 286 701 15169 4713                                       | 105228
 2497 701 15169 4713                                      | 100706
 7500 2516 701 15169 4713                                 |  49684
 34288 15576 8220 5511 701 15169 4713                     |  49662
 286 701 15169 7029                                       |  41958
 286 701 15169 9121                                       |  33838
```

- **8/25 の経路をよくみると…** 286 701 15169...
- 286 / 5511 ⇔ 701 = **ピアと思われる**

  → 701 ⇔ 15169 **はトランジットと思われる**

⚠️ 701がミスってなければ、という前提

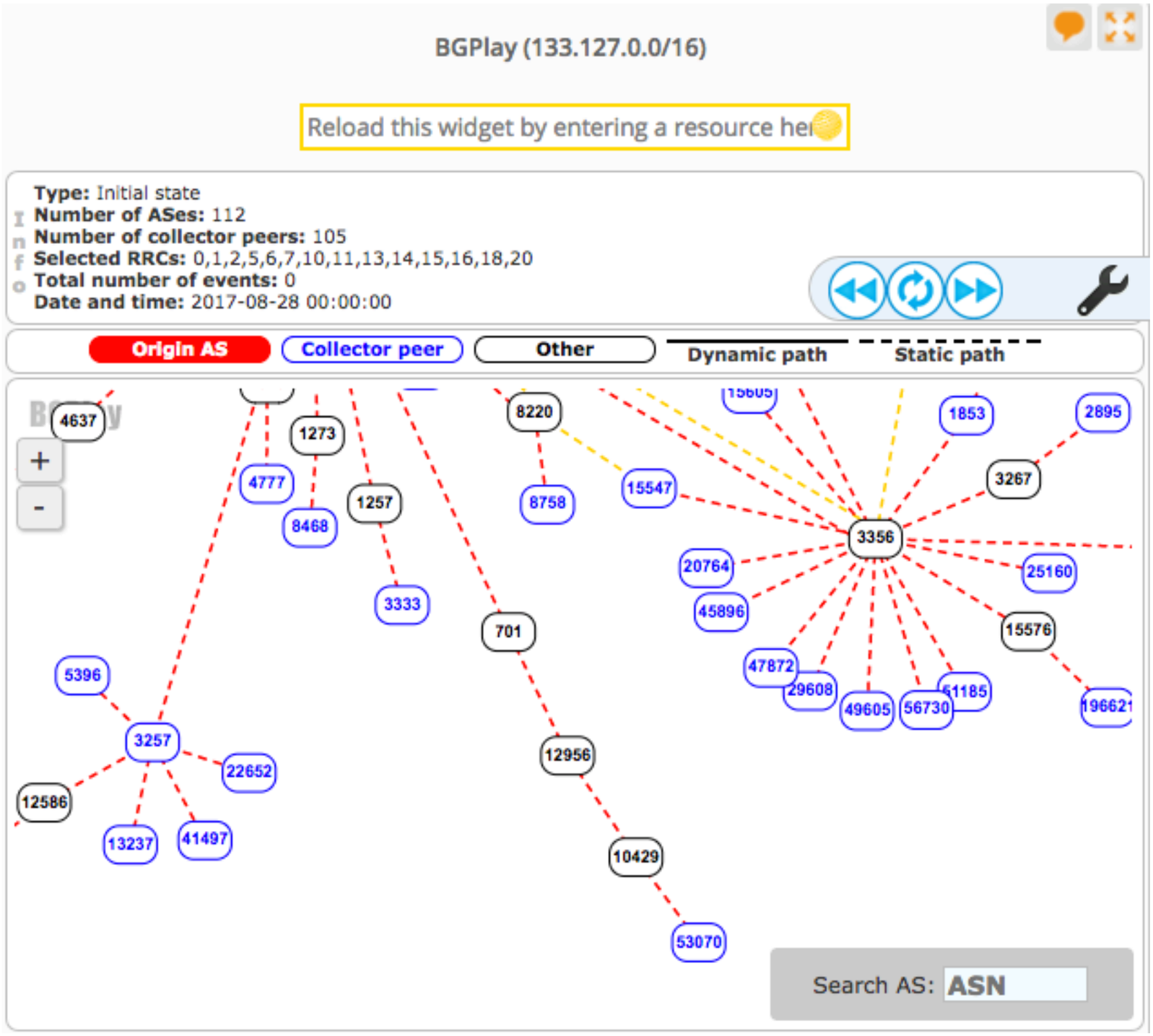
2497 701 15169 4713

2497 の経路を適当に引く

→ 12956 701 2497 という経路が見える

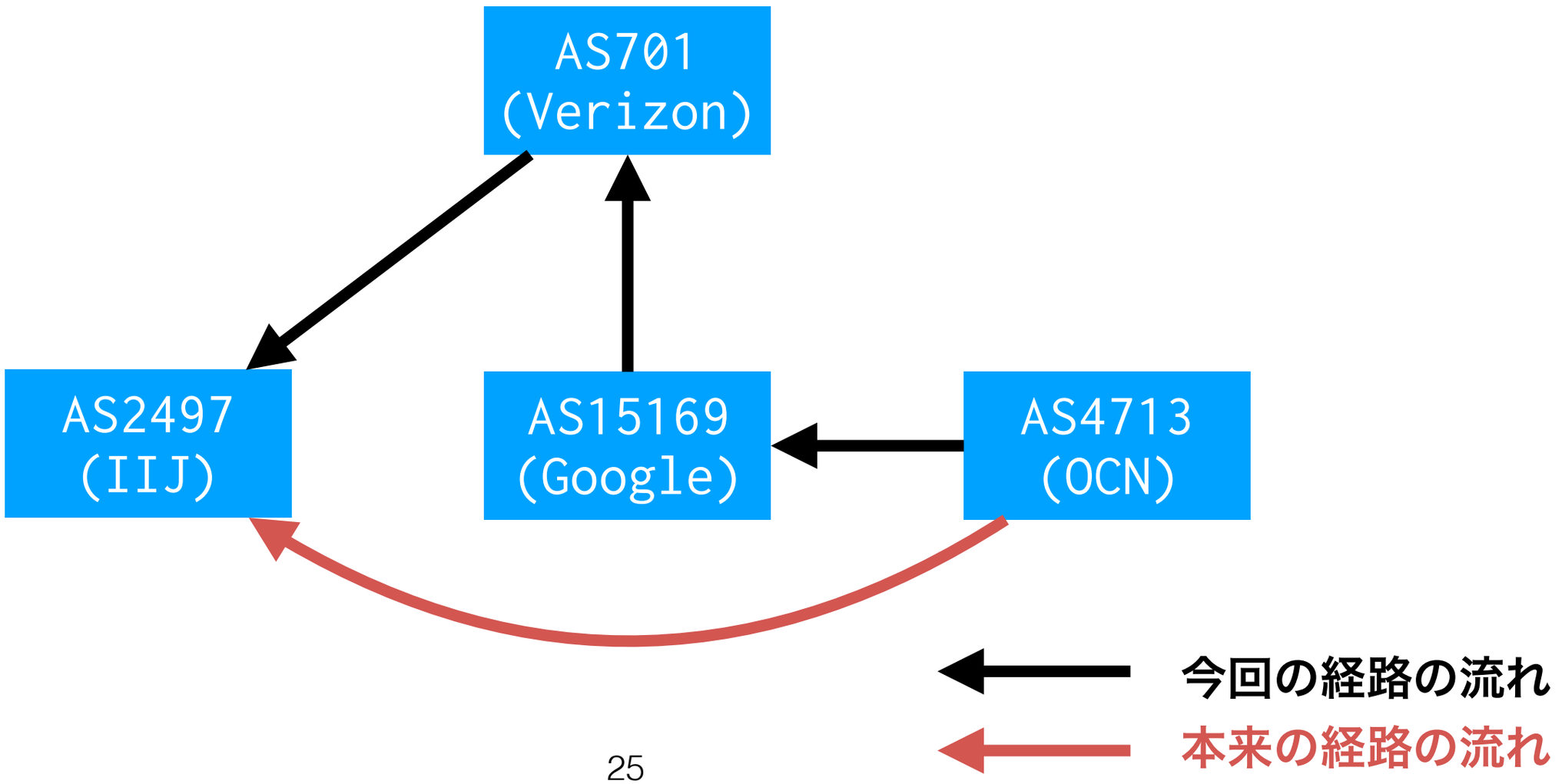
→ 12956 ↔ 701 はピアと思われる

→ 701 ↔ 2497 は<u>トランジットと思われる</u>



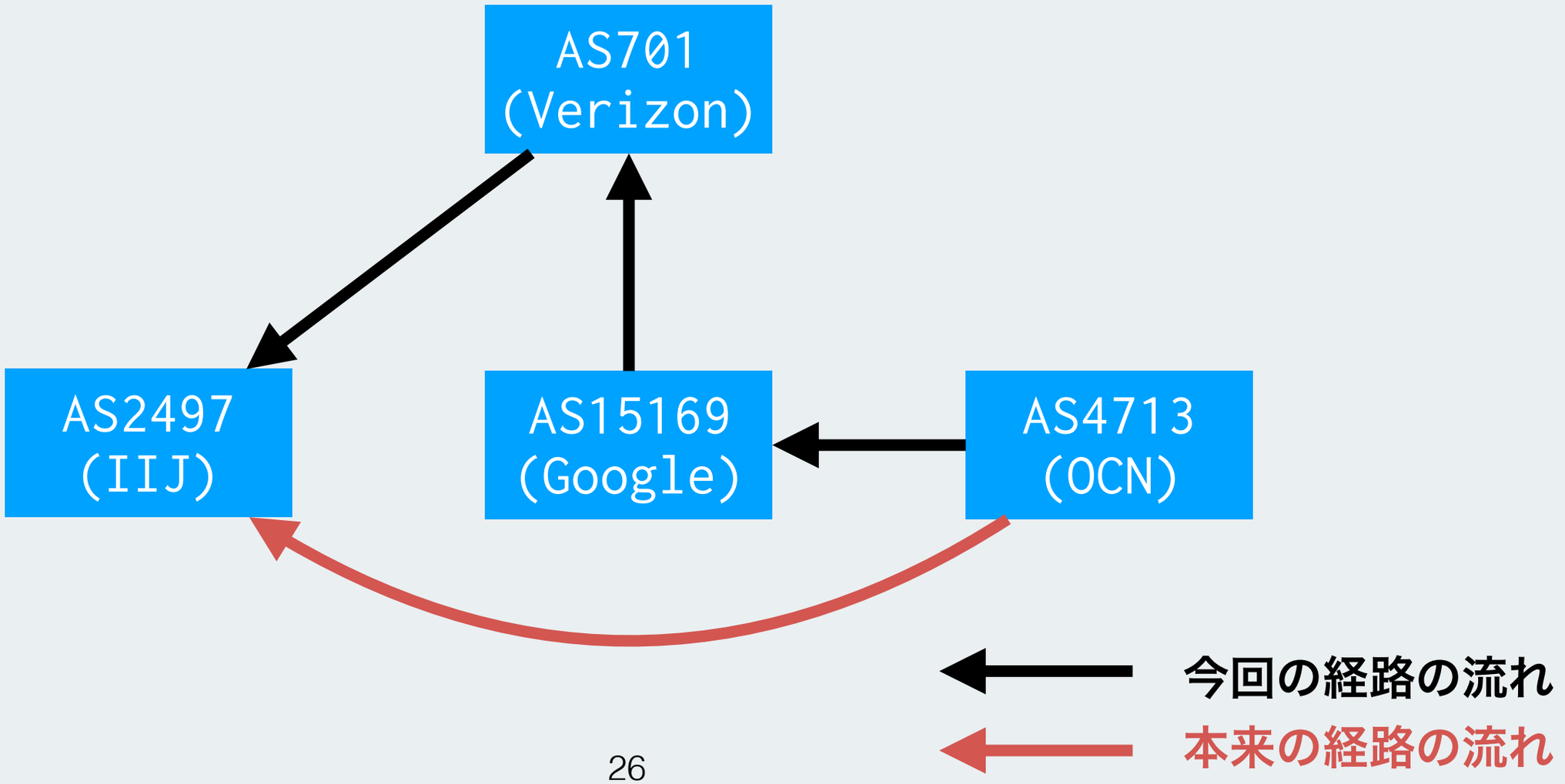
2497 701 15169 4713

まとめると、たぶんこう

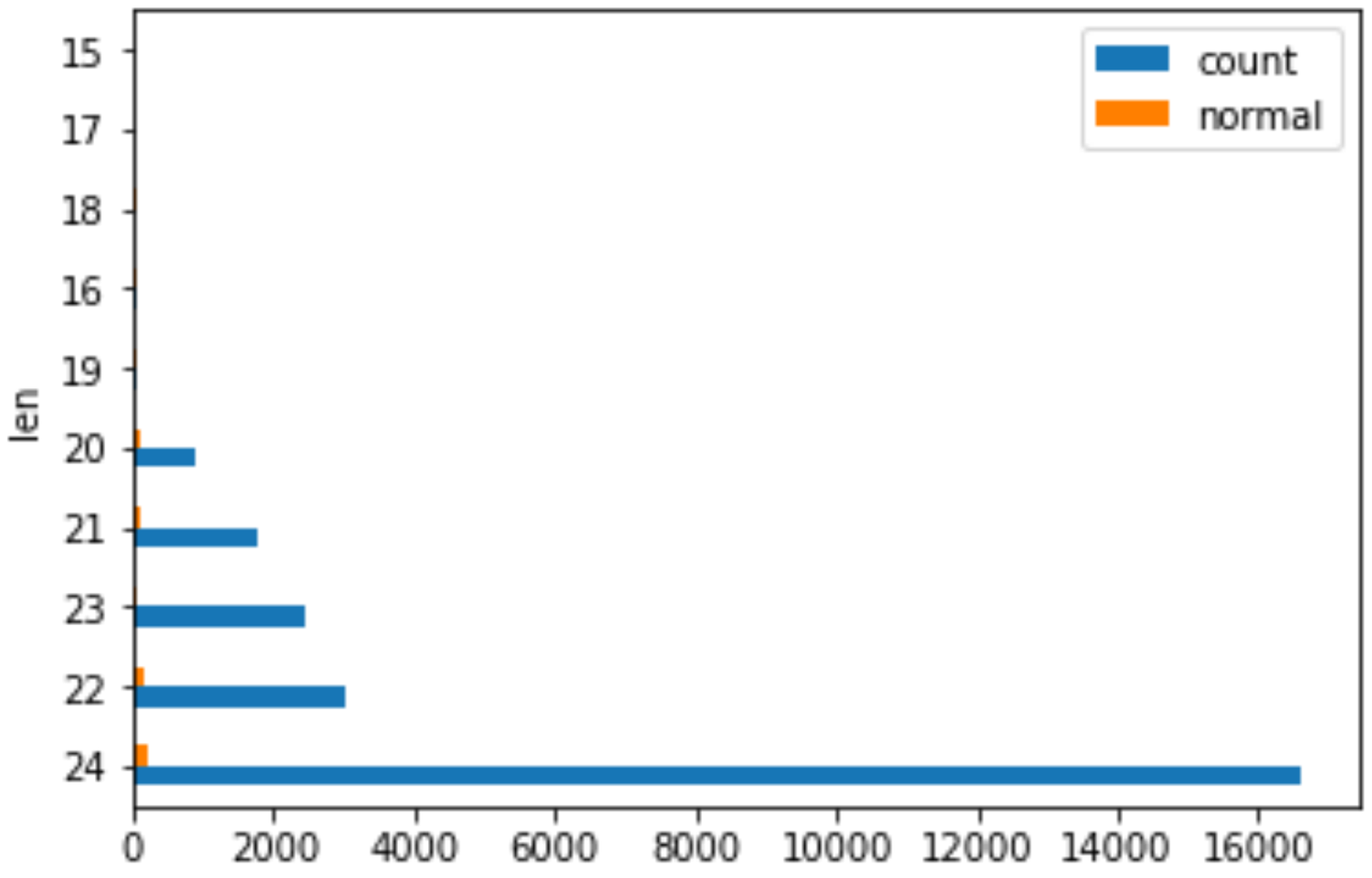


疑問

この経路がベストになったのはなぜか？



dix-ie (AS2497経由)で見えた、Prefix長 あたりのPrefix数 (2497 701 15169 4713 限定)



細かい経路に吸い込まれた様子

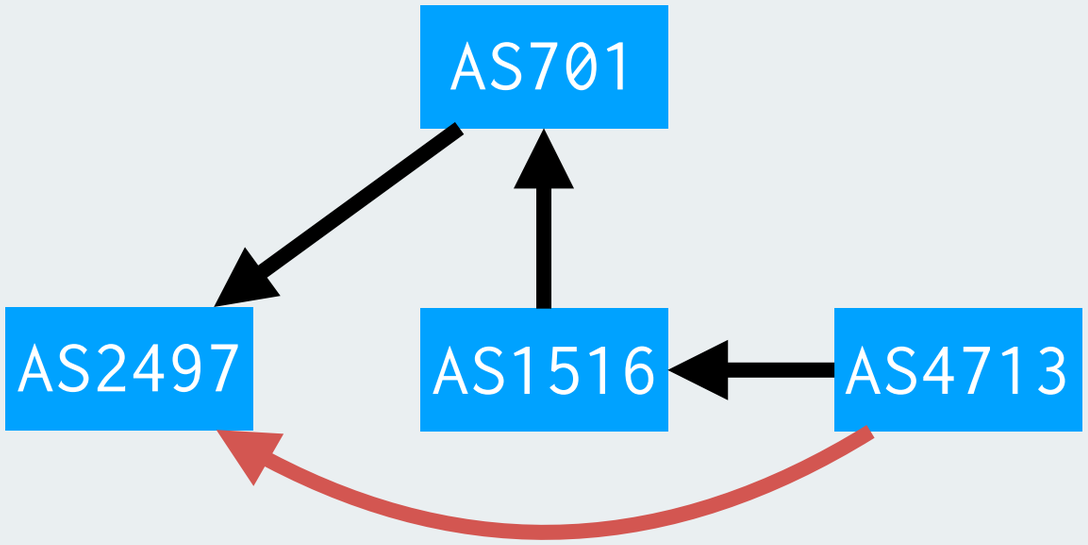
観測 2

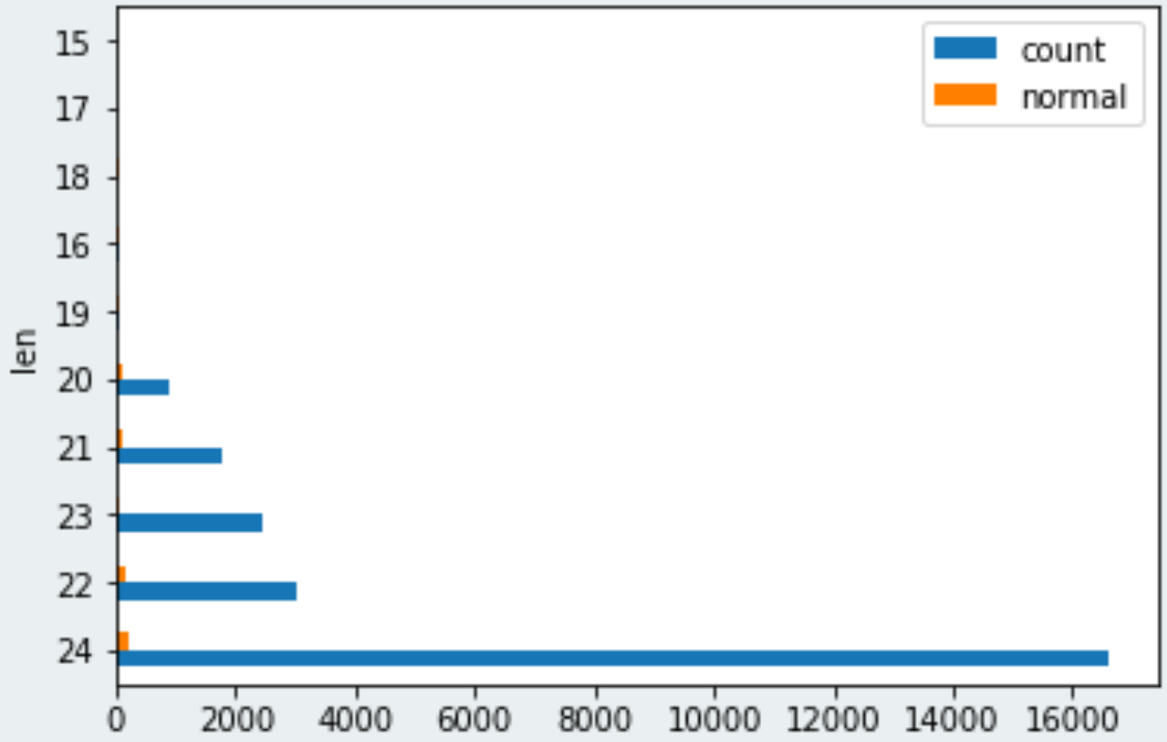
- /24 をはじめとする細かい経路が増えた

起こったであろうこと:

- Longest Match により自AS宛てのパケットが、または自AS発のパケットが吸い込まれた → Drop Rate は未確認

観測 2





新たな疑問:

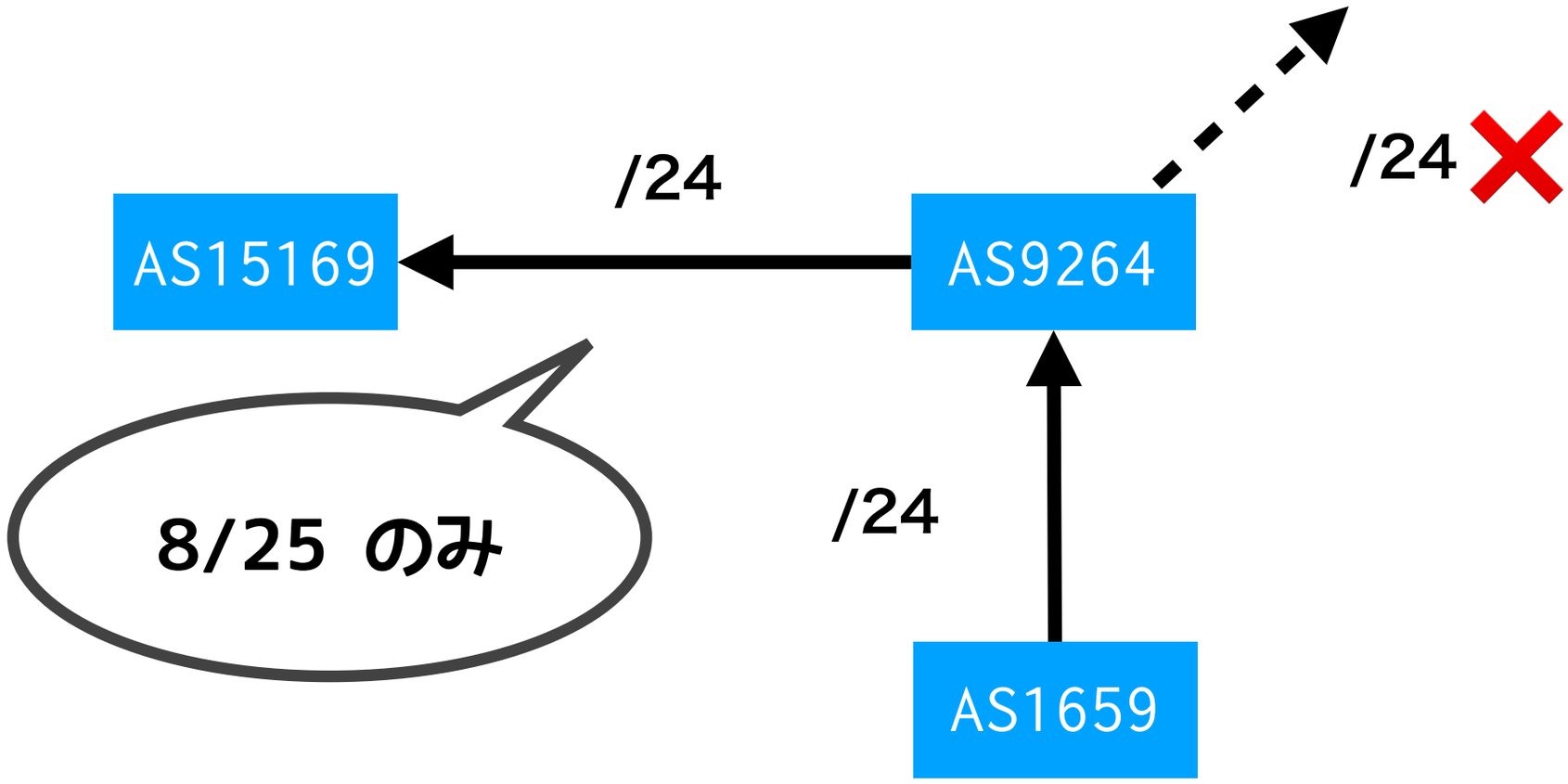
- 細かい経路はどこから来た？
- 701 の入り口で止まらなかったのはなぜか？

疑問: 細かい経路はどこから来た？

1. 4713 (所有者) がオリジネートしていた
2. AS PATH の途中のAS が分割していた
3. 全く別のAS が広告していた

→ 2 の可能性がありそう (推測)

2497 701 15169 9264 1659

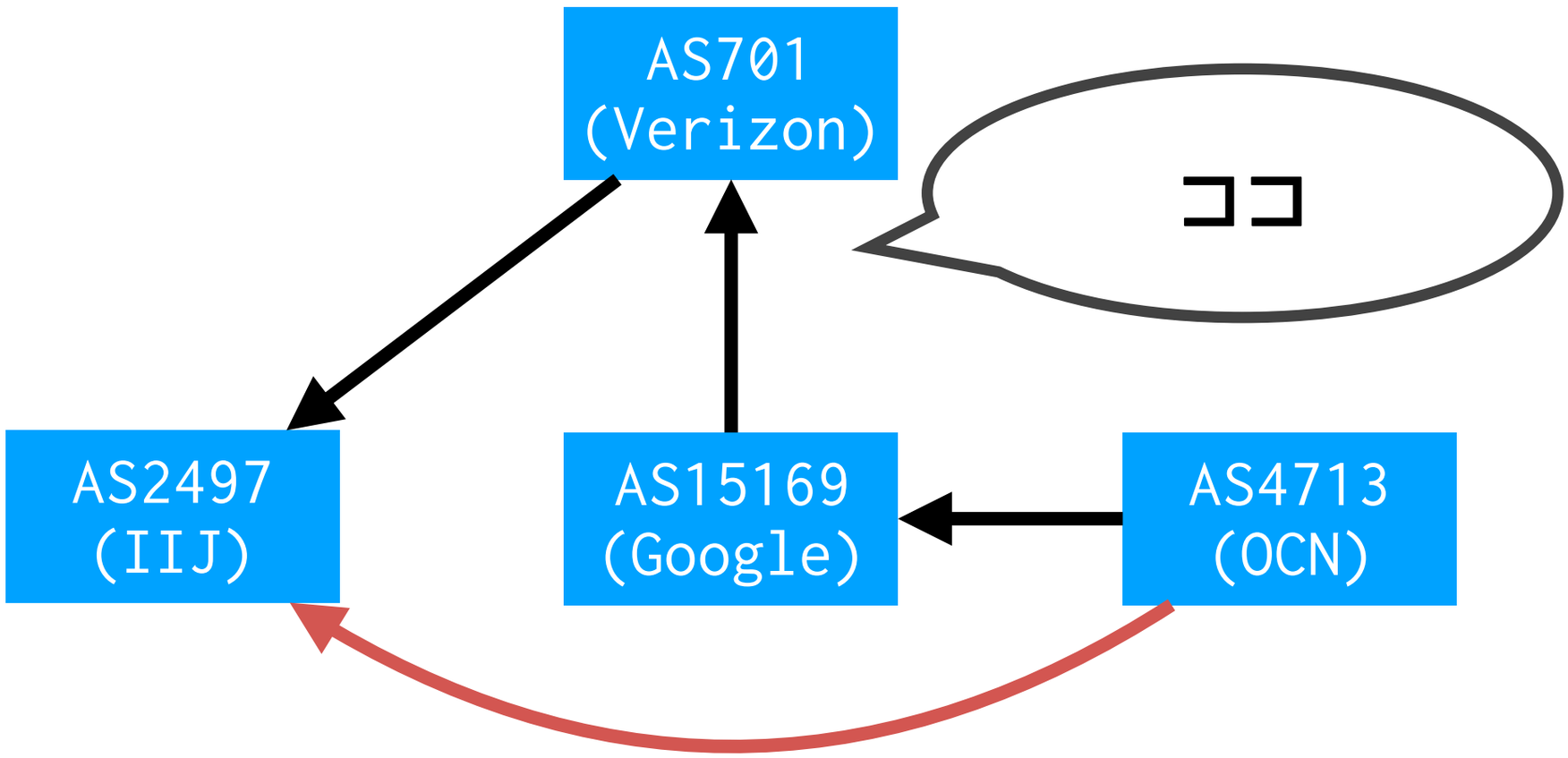


普通、こうはならない。8/25に観測された経路のうち、

- 15169をAS_PATH に含む & AS_PATH長 > 2 の経路数→ 30,700
- そのうち、このようなタイプの経路数 → 20,457
- 特別アレンジそんなに多い？ → 仲介者がオリジネートしているのでは？

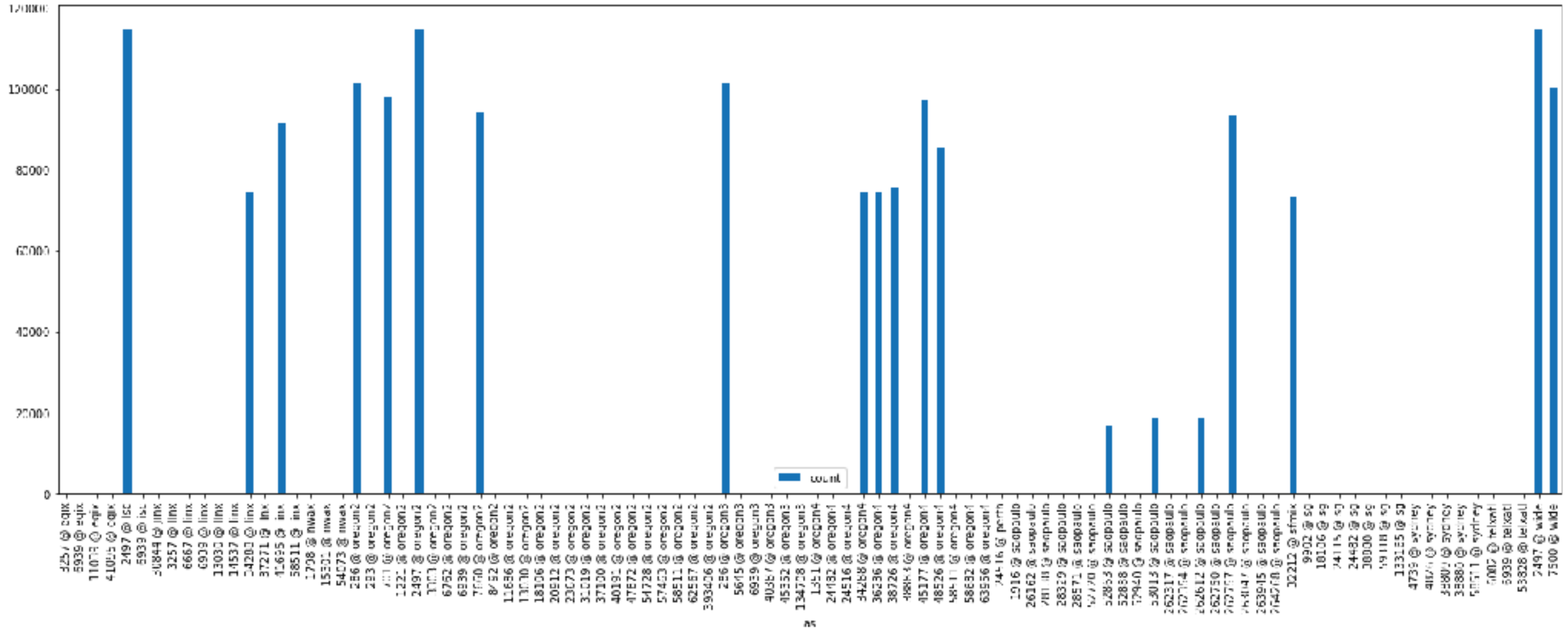
疑問:

701 の入り口で止まらなかったのはなぜか？



よく分からない。普通はフィルターされそう

海外のIXはどうか
(IX, AS) 別の、AS15169 Update数



Route Views と繋がっているのべ100 AS のうち、影響あったのは20 ASくらい

観測 3

- グローバルでみれば、影響あったASとなかったASがある

推測:

- 適切に経路フィルターが効いた？
- もしくは、そもそもリークがなかった？
- もしくは、ピアごと落ちた？ 🤕

観測と疑問まとめ

- 経路数が +9万 (観測1)
- その多くは/24 など、細かい経路 (観測2)
  - 普段はインターネットに存在しない

疑問 (議論したい点)

- 細かい経路はどこから来た？ 所有者が広告していたやつ？
- トランジットの経路フィルターで止まらなかったのはなぜ？ (止まったケースもありそうなのに)
- 自衛する方法はあるか？

ポイント

パブリックなデータソースから
ここまで分かります。

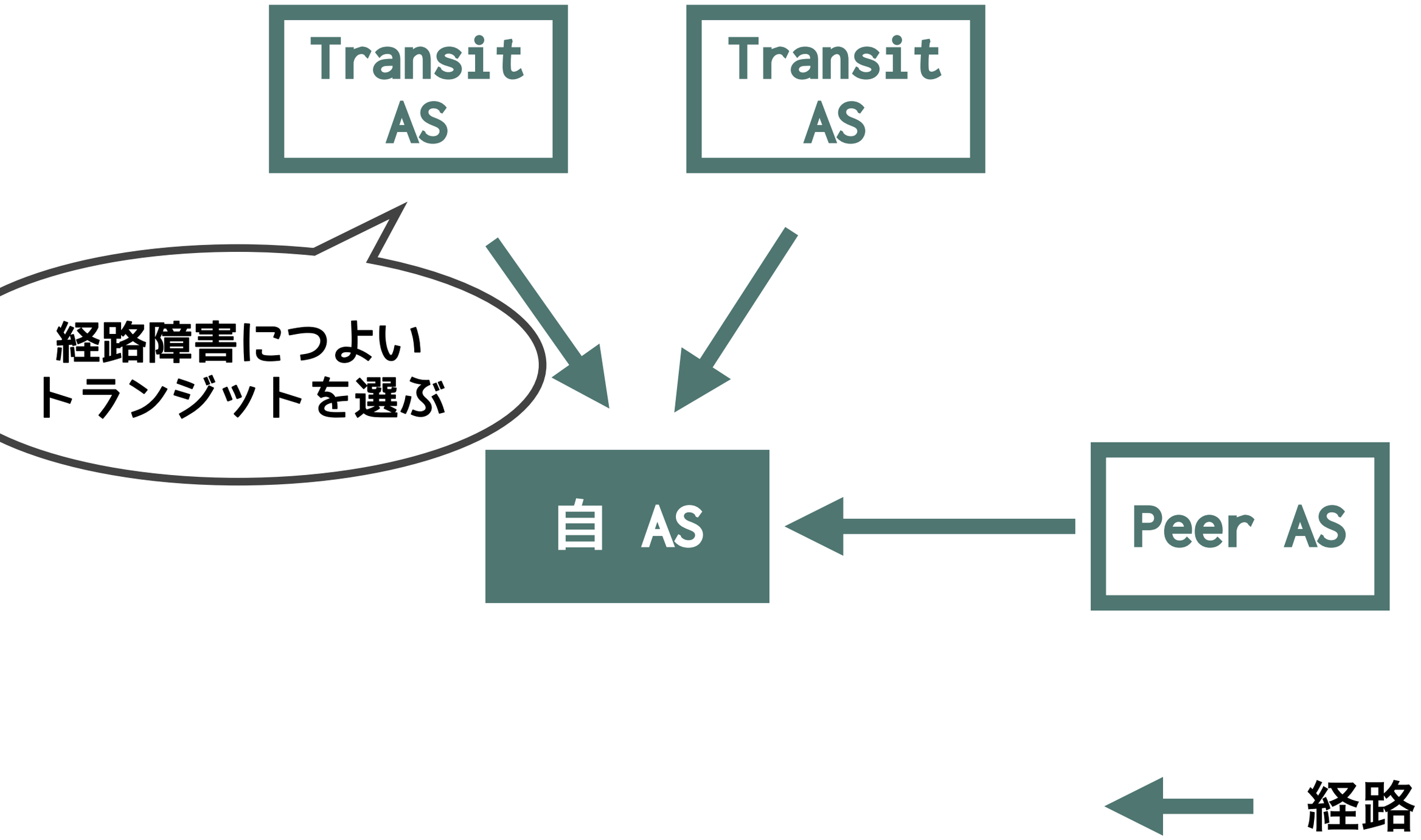
MRT Dump 感謝 🙏

対策

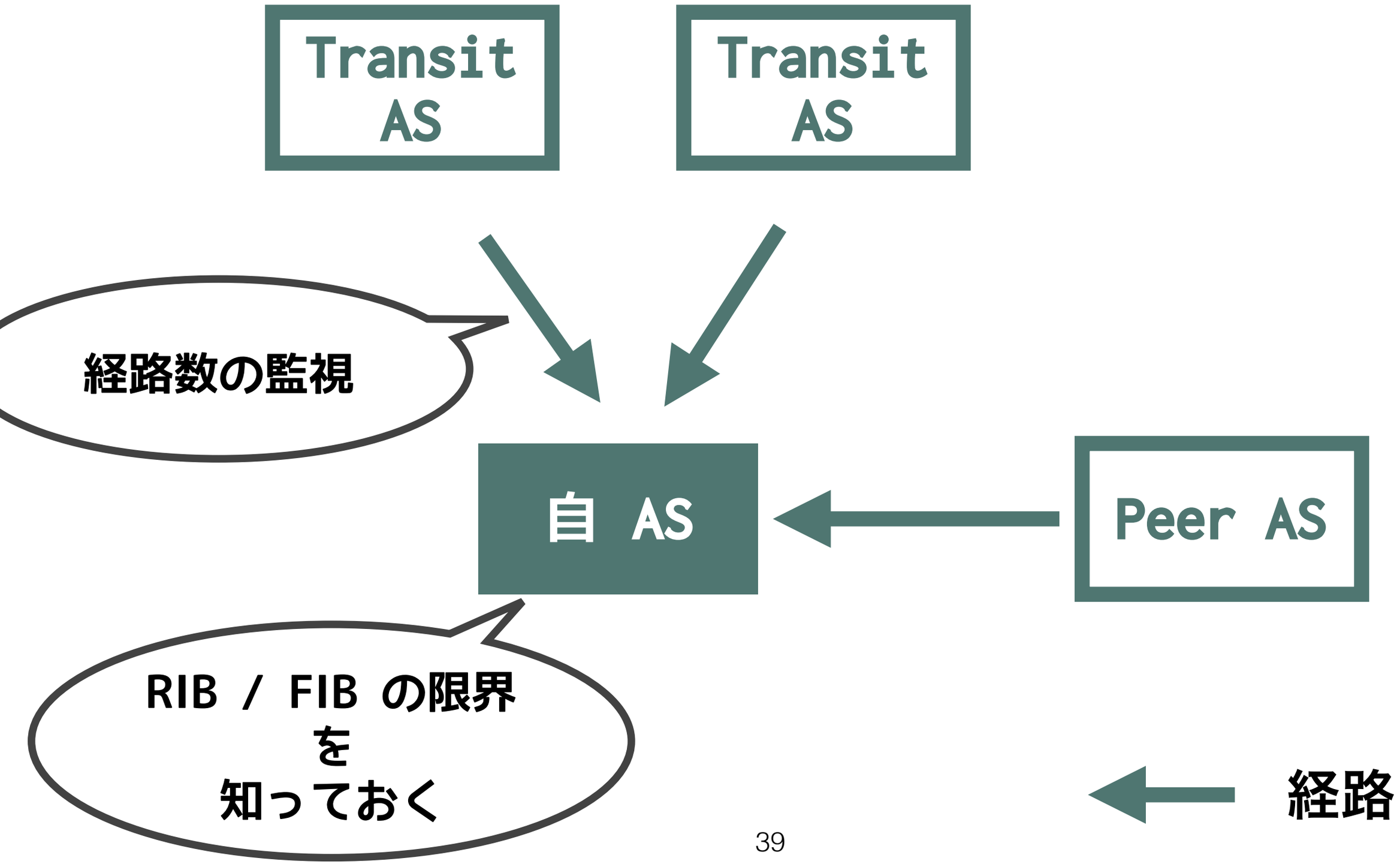
JANOG や NANOG などでも

様々な議論が

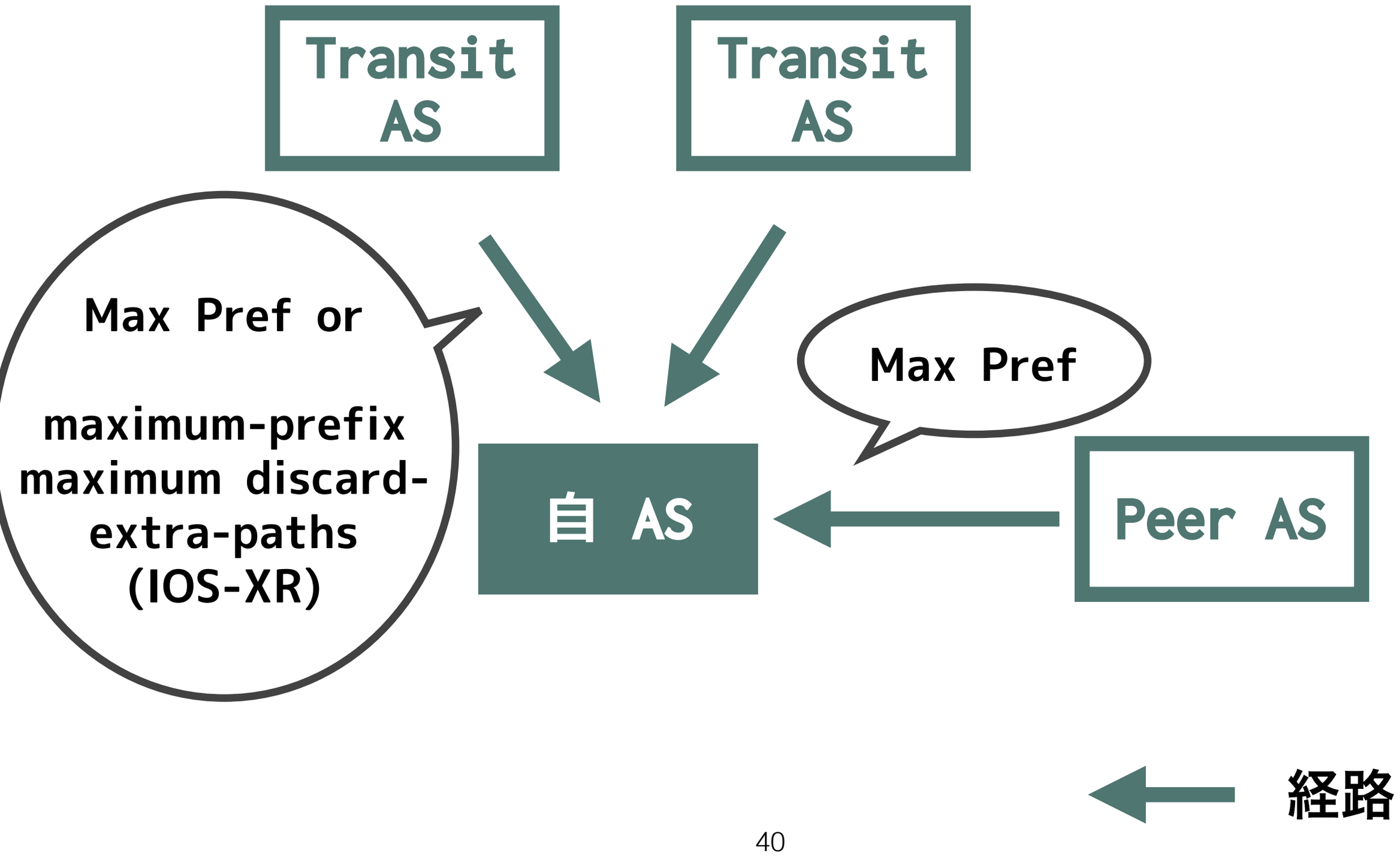
自衛したい



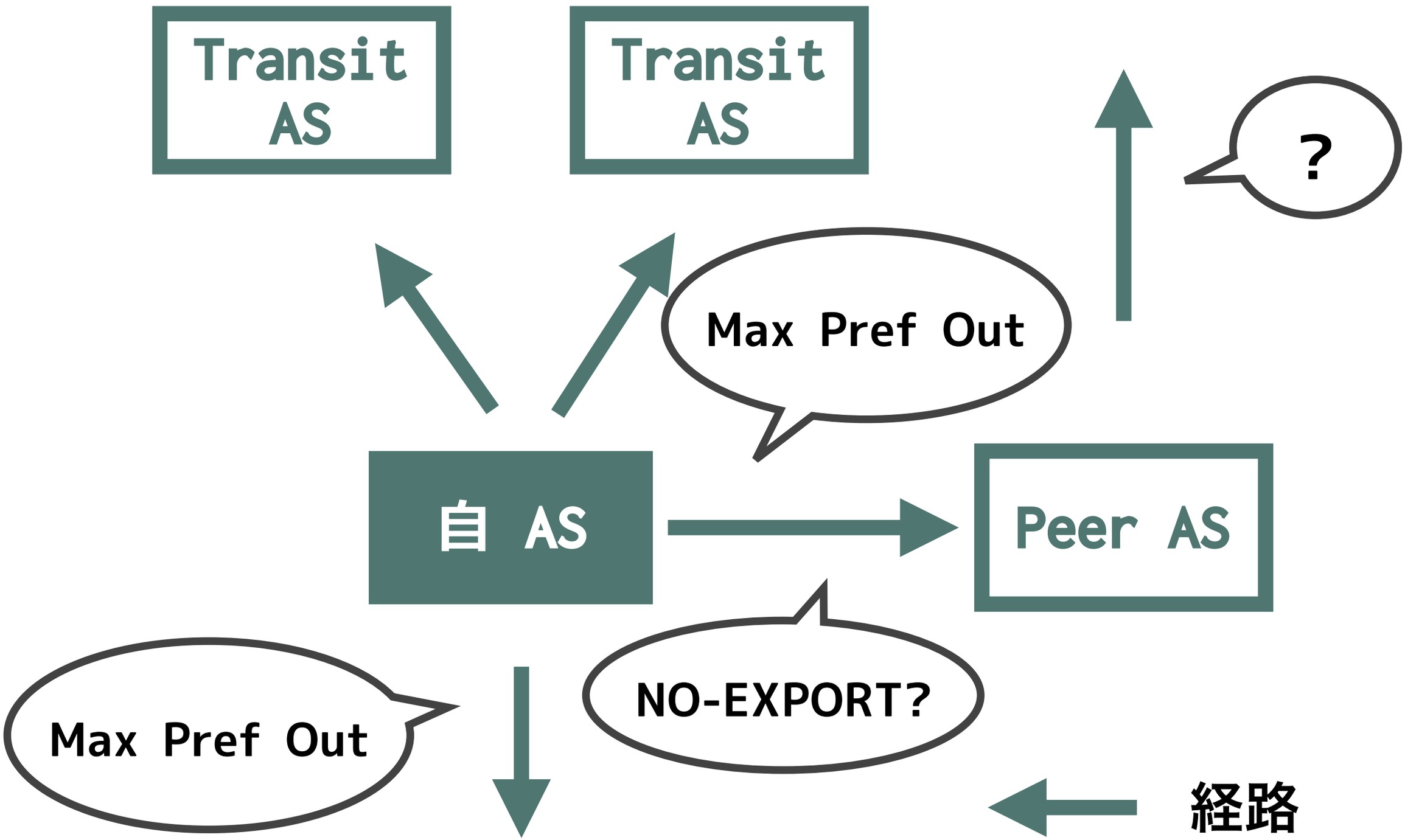
自衛したい


Transit
AS
Transit
AS
経路数の監視
自 AS
Peer AS
RIB / FIB の限界
を
知っておく
経路

自衛したい


Transit AS
Transit AS
Max Pref or
maximum-prefix
maximum discard-extra-paths
(IOS-XR)
Max Pref
自 AS
Peer AS
経路

自衛したい


Transit AS
Transit AS
?
Max Pref Out
自 AS
Peer AS
Max Pref Out
NO-EXPORT?
経路

モチベーション

大なり小なり経路障害の影響を受けた。

障害の原因を知って、次は止めたい。

→ いいアイデアありませんか？