# Routing Tutorial

小島 慎太郎 / @codeout

JANOG33, 2014/01/22

小島　慎太郎

- **NTT Communications**
- @codeout
- **http://about.me/codeout**

本日のスライドは
**https://speakerdeck.com/codeout/bgp-routing-tutorial**
にあります

# Agenda

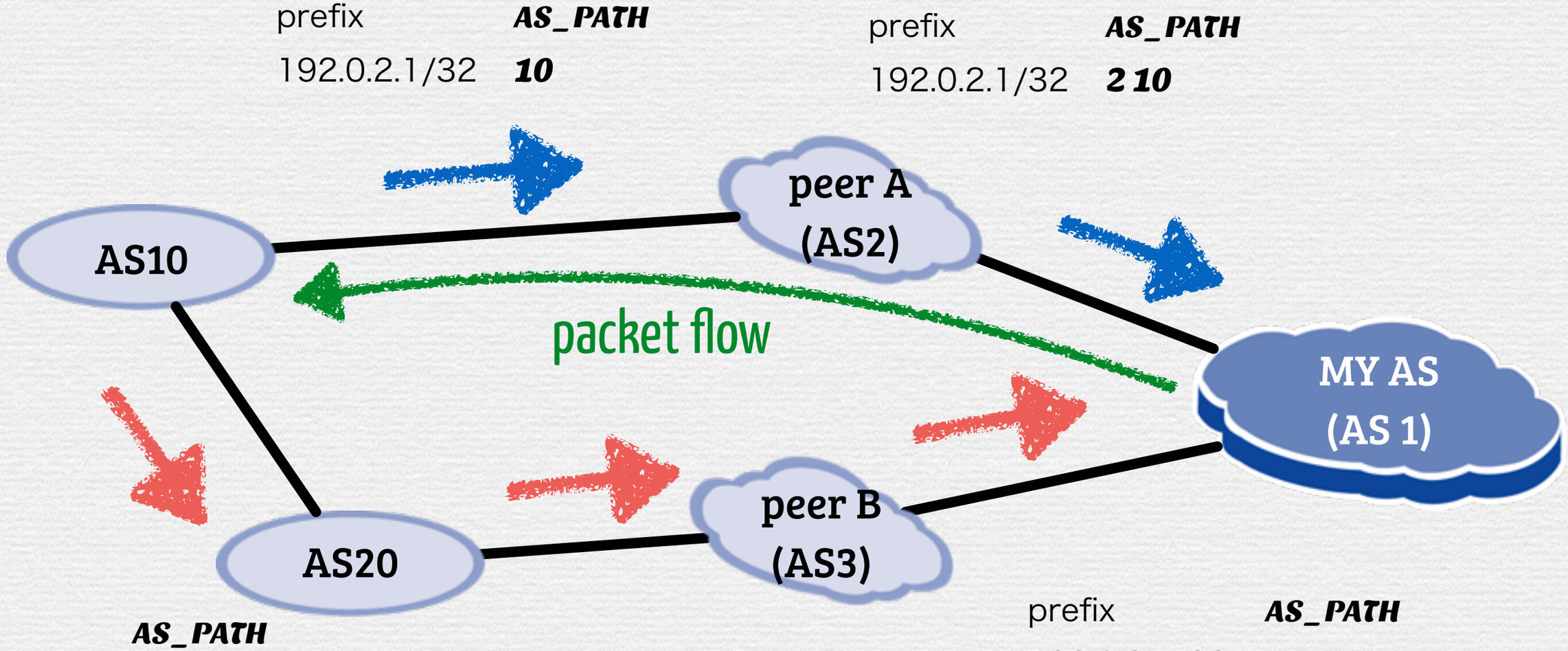
- BGP とは?
- BGP の設計を考えよう
- BGP の運用を考えよう

# BGPとは？

# BGP とは?

EGP の一種．**異なるAS間**でrouting情報を交換する

- AS接続グラフ(RIB)を構築し，packet forwardingに使う
- 1つのroute には複数のpathが含まれている
  - route: あるprefixに到達するためのpath群
  - path: 様々な属性(Path Attribute)の組
- routing loopを解消する仕組みがある
- AS内での**routing policyを実装**できる




**MY AS** routerの動作:

packetを転送しようとするたびに，RIB内からdst ip addressを検索し，宛先(nexthop)を特定する

- 正確にはRIBを展開したFIB内を検索する

# 異なるASには異なるrouting policyがあり，それが反映されたrouting情報を交換する点でBGPは難しいし，おもしろい



http://mrg.bz/WooRIg

# もう少し
# おさらいします

# IGPとの関係

- BGPで解決できるのは，**Internet上のある目的地に達するために，自AS内のどの出口から出ればいいか?**のみ

- **その出口にはどうやったら到達できるか?**はIGP頼み

- IGPの主な用途は，BGPのprotocol nexthopを解決すること
  - むやみにBGP経路をIGPに注入しないほうがいい

# A Border Gateway Protocol 4 (BGP-4)

- ***RFC1654*** (July 1994)
- ***RFC1771*** (March 1995)
- ***RFC4271*** (January 2006) 👍
- もちろん たくさん拡張されている
  - ***RFC1997*** BGP Communities Attribute
  - ***RFC2385*** Protection of BGP Sessions via the TCP MD5 Signature
  - ***RFC3065*** AS Confederations for BGP
  - ***RFC4451*** BGP MED Considerations
  - ***RFC4456*** BGP Route Reflection
  - ***RFC4360*** BGP Extended Communities Attribute
  - ***RFC5004*** Avoid BGP Best Path Transitions from One External to Another
  - ***RFC5668*** 4-Octet AS Specific BGP Extended Community
  - …

# BGP の経路選択

1. (最も高い **WEIGHT** を持つパスが優先されます．一部メーカーのみ)
2. **最も高い LOCAL_PREF を持つパスが優先されます**
3. network または aggregate BGP サブコマンドによって，あるいは IGP からの再配布を通じて，ローカルで発信されたパスが優先されます
4. **最短の AS_PATH を持つパスが優先されます**
5. 最小のオリジン タイプを持つパスが優先されます
6. **最小の Multi-Exit Discriminator (MED) を持つパスが優先されます**
   - ✎ **MED** は remote AS が同じ場合のみ評価される
7. iBGP パスよりも eBGP パスの方が優先されます
8. **BGP ネクストホップへの最小の IGP メトリックを持つパスが優先されます**
9. 両方のパスが外部のときは，先に受信したパス（最も古いパス）が優先されます
10. 最小のルータ ID を持つ BGP ルータから送られたルートが優先されます
11. 発信元 ID またはルータ ID が複数のパスで同じ場合は，最小のクラスタリスト長を持つパスが優先されます
12. 最小の隣接ルータ アドレスから送られたパスが優先されます



http://www.cisco.com/cisco/web/support/JP/100/1008/1008556_25-j.html

# 今日話すこと

- **BGPの設計を決める際に考えないといけないこと** について話します

- “BGP sessionの先にいるAS (顧客 / peering partner / transit 提供者) にも個別のrouting policyがある” という環境で, **どうやって自分の思うようにtraffic controlするか?** を理解するために

・ 私が使っている手法の紹介
・ その解説

もします

# BGP の設計を考えよう

- eBGP policy / 設計
  - 経路広告
  - 経路受信
- iBGP policy / 設計

# BGP 設計の前に

network policyってありますか?

- 可用性
- 品質
- 運用性
- 拡張性
- security
- cost

# BGP 設計の前に

network policy

routing policy

**物理設計**

**論理設計**

networkの物理的なこと
- POP 配置 / DC 選定
- cable path
- 収容設計
- 機器選定

どうpacketを流すか?
(通常serviceごとにpolicyが変わる)
- どのような情報を流すか?
- 顧客にどのようなnetwork service を提供するか?

routing policyを満たすよう，routingを設計する
- protocolは?
- route reflector(RR) or confederation?
- どの情報(経路)をどこに流す?

# eBGP policy (基本)

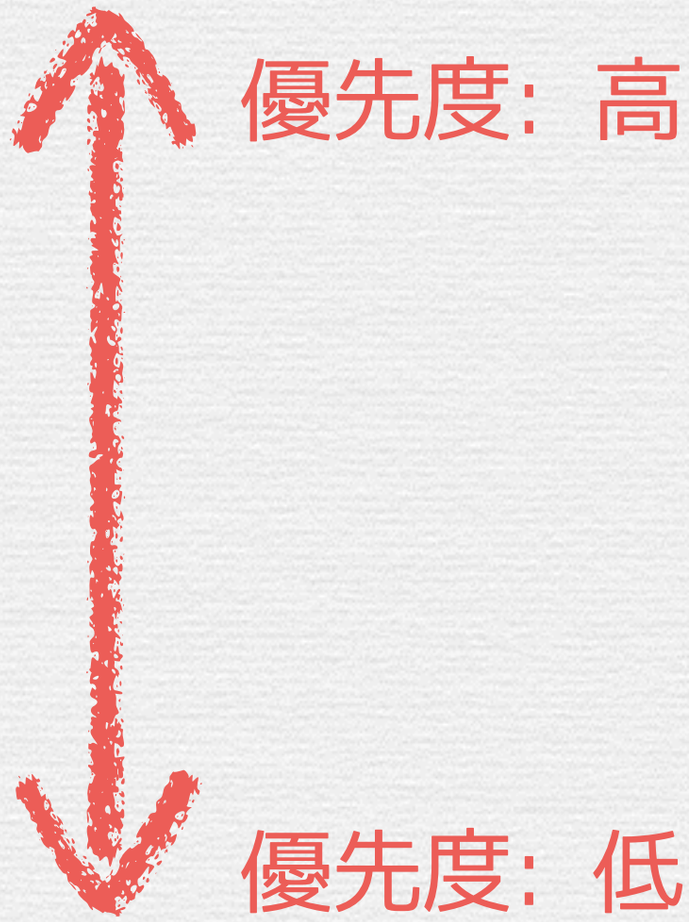
# eBGP Policy を考えるポイント

- どんな経路を
  - どんなeBGP session からもらうか? (もらわないか?)
    - その際の優先度は?
  - どんなeBGP session へ広告するか? (広告しないか?)
    - その際の優先度は?
- 経路の各path attributeは誰のものか?

  - full routeを持てないrouterはある?
    - default routeを利用: よく考えないと危険
      簡単にrouting loopが起きる
  - BGPではなく，別のprotocol(OSPF など) にredistributeしないといけない，とかある?

# 経路の種別

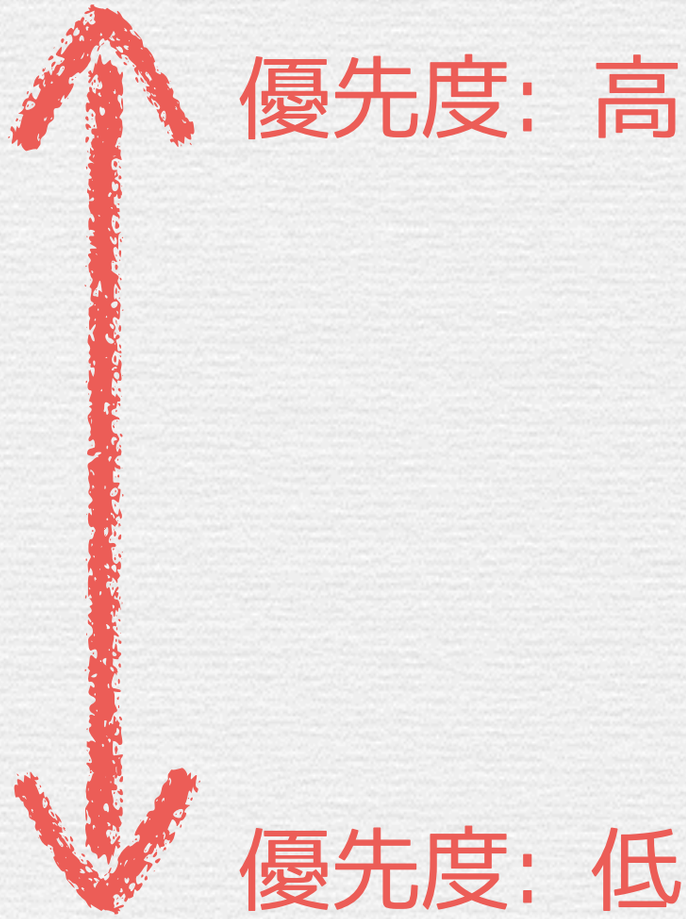
- 顧客の経路
  - BGP 顧客
  - static 顧客 (実際はPAに集約)
- 自ASの経路
  - PA/PI経路
- peering partnerの経路
- transit providerの経路
- 不要な経路




ビジネス上の観点から，基本的には上記の順に優先する
(優先 = なるべくその経路に従ってpacketを流したい / 流してほしい)

# eBGPセッションの種別

- 顧客
- peer
  - paid peer（収益を得ている）
  - private peer
  - public peer (IX)
  - paid peer（費用を払っている）
- transit




同様に，基本的には上記の順に優先する
(優先 = なるべくその接続/回線にpacketを流したい)

# eBGP Policy の基本

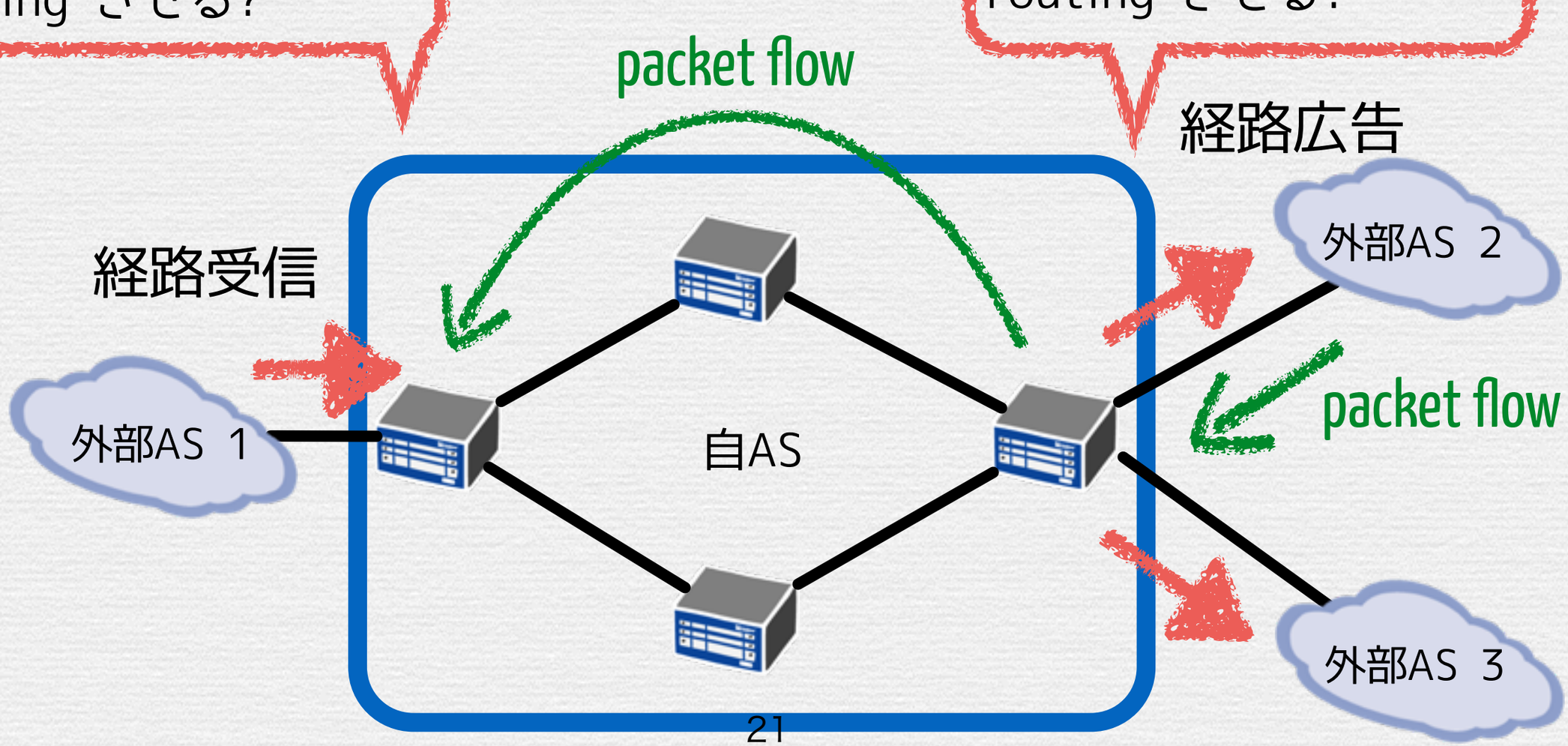
受信 / 広告Policyは何に影響する？

**受信Policy**

自AS**網内**でどのように routing させる?

**広告Policy**

自AS**網外**でどのように routing させる?



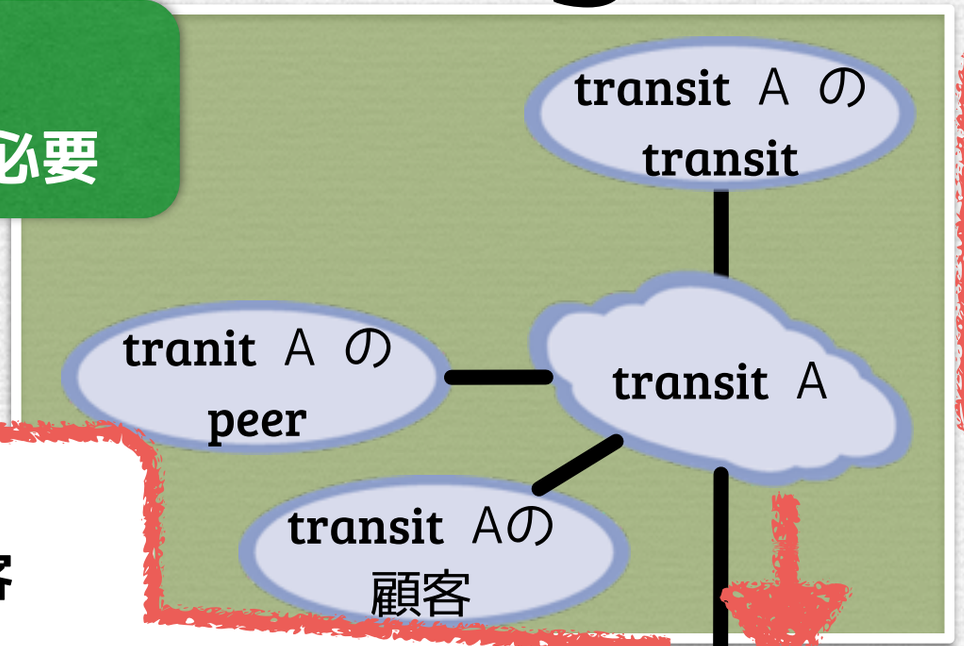
# eBGP

# 経路受信

# (ちょっと細かく)

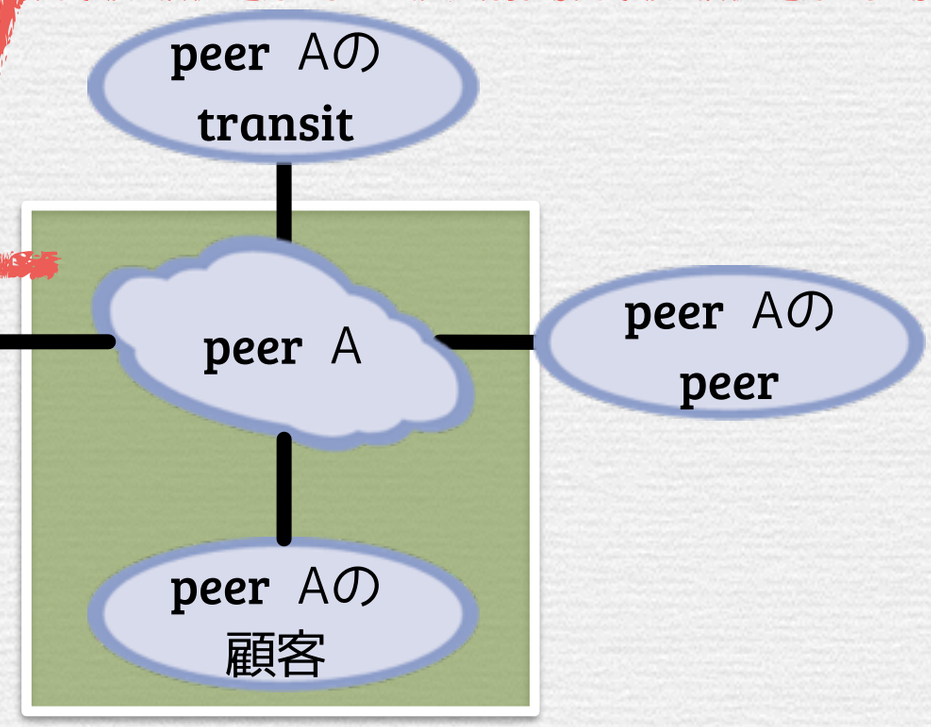
# eBGP Policy の基本（受信）

**bogon filter はすべてに必要**

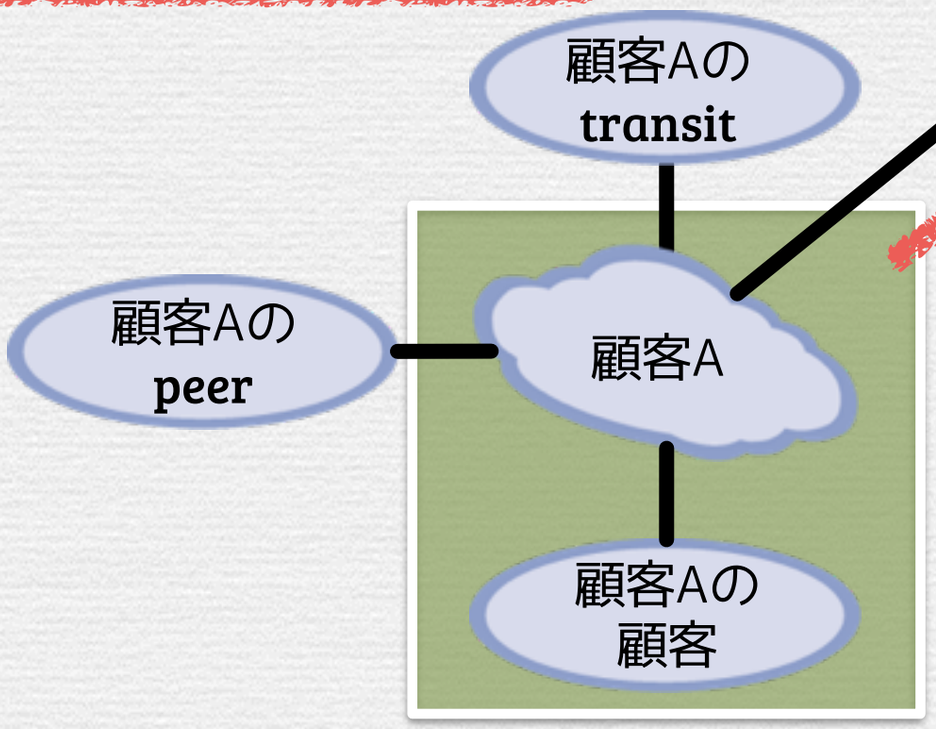



**すべて受信**
- transit A の顧客
- transit A 自身
- transit A のpeer
- transit A のtransit

**すべて受信（経路filterあり）**
- peer A の顧客
- peer A 自身
- peer A のpeer （通常流れてこない）
- peer A のtransit （通常流れてこない）



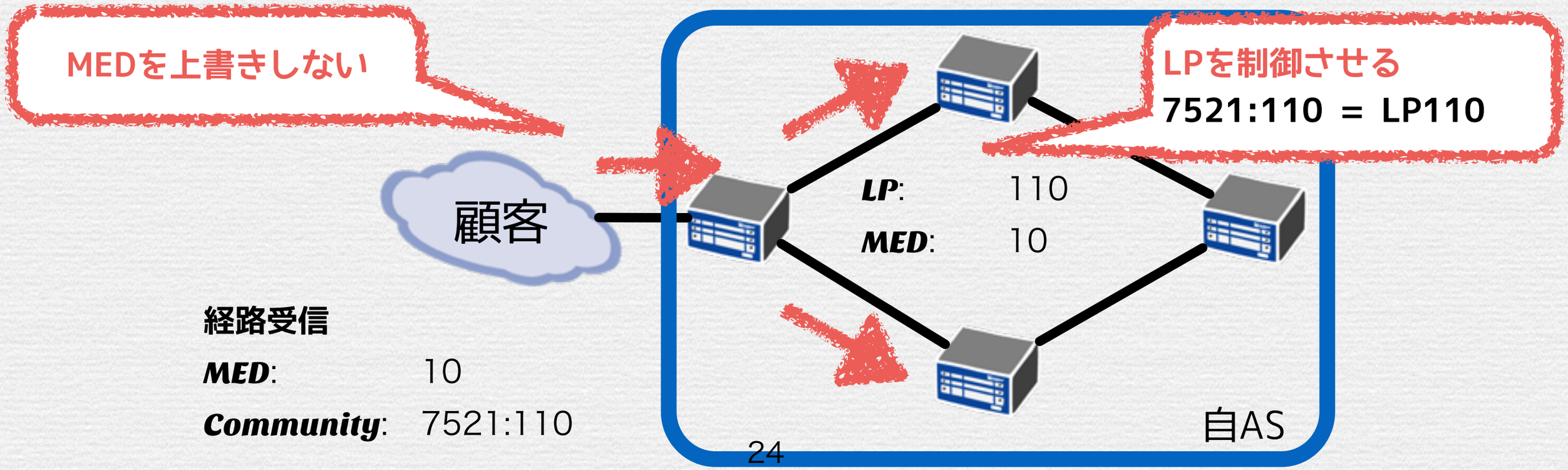





**自身の経路を除き，すべて受信（経路filterあり）**
- 顧客A の顧客
- 顧客A 自身
- 顧客A のpeer （通常流れてこない）
- 顧客A のtransit （通常流れてこない

# 経路の各path attributeは誰のもの?

- 以下のような機能を提供しているISPも
  http://onesc.net/communities/
  - 顧客経路の**MEDを上書きしない**
  - 自AS内での**local preference(LP)を制御する**
    **BGP community**を顧客向けに提供する



# 受信経路を扱うときに気にするべきポイント3つ

1. LPとMEDの値
2. 経路Filter
3. Path Attribute は本来の用途に使おう

# 1. LPとMEDの値

- **基本的に，受け取った経路は使う**
  - 例えば peer から“peerのpeer経路”を受信したとして，(相手の設定ミスの可能性大だが) 拒否する理由は特にない
  - 使いたくない理由があれば別 (優先度を下げる or 拒否する)
    - 輻輳しそう，など

| 優先度 | 経路種別 | LP | MED |
|---|---|---|---|
| 1 | 顧客 | 150~300 (*) | 上書きしない |
| 2 | 自AS | 200 | なし |
| 3 | peer | 200 | 上書き (十分大きな値) |
| 4 | transit | 120 | 上書き |

(*) 200をまたいで数段階 (default: 300)

- 幅に余裕をもって数値を設定
- LPのdefault値が100の実装が多いので，
  安全を期すならLPの値は≠100

# － 解説 －

顧客の明確な意図がない限り最優先
常にRIBに保持

経路を指定して他の顧客やpeerに迂回させることができる

peerの200と比較して，必ず次のAS_PATHで優先度が決定

顧客のTE選択肢を増やす

| 優先度 | 経路種別 | LP | MED |
|---|---|---|---|
| 1 | 顧客 | 150~300 (*) | 上書きしない |
| 2 | 自AS | 200 | なし |
| 3 | peer | 200 | 上書き (十分大きな値) |
| 4 | transit | 120 | 上書き |

- AS_PATHが同じならIGPベースのclosest exitを強制
- always-compare-med や peer&顧客AS対策でMEDを高めに
- 2^32 – 1 でもいいかも

private / public peer で2段階あるISPもある

一応>100

AS_PATHが同じならIGPベースのclosest exitを強制

(*) 200をまたいで数段階 (default:300)

# 2. 経路Filter

顧客経路の優先度が非常に高いため，細かくチェックする必要がある

- **IRRベース が理想的**
  - 頻繁にメンテナンスできるのであれば，手動でも問題ないかもしれない
- filter方法の候補
  - exact match な prefix filter
  - exact match な prefix filter & origin AS filter

- **bogon prefix，自ASのprefixは経路filterで除外**
- BGP communityは透過的に扱う

# peer経路も手放しでは危ない

- **経路hijackの可能性**
- 細かいfilterを設定してしまうと**メンテナンスされなくなるかもしれない**
  - “AS_PATHをメールで伝え合う” 仕組みが動いていた
  - やはり限界があるので，できるのであれば顧客経路同様IRRベースがよい
- 一方，他の国では以下の組み合わせが多い
  - **maximum-prefix (prefix-limit) filter**
    - 実績ベース（昨日から20%増えたらアウト，など)
    - peering dbベース
  - **prefix lengthによるfilter**
    - ipv4: le /24
    - ipv6: le /48 など

# Maximum-Prefix (Prefix-Limit) Filter

⚠ transitに設定すると危険な場合がある

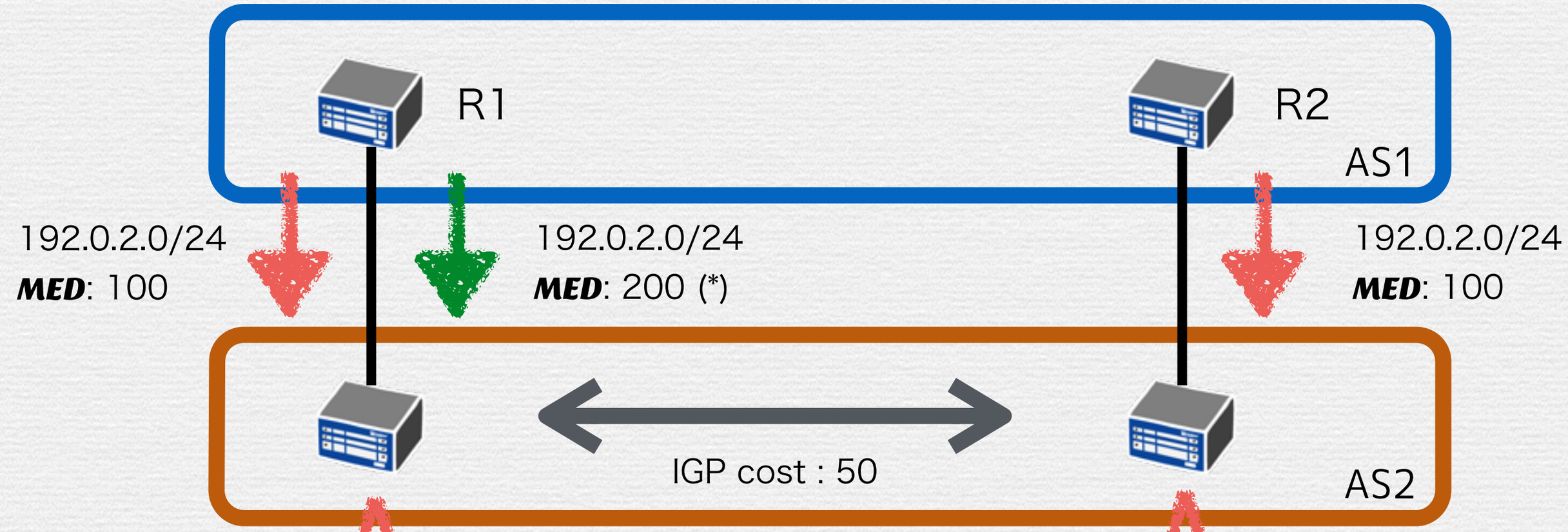
- network上のeventにより急にprefix数が増えると，transitが全断するリスクがある
  - internetから遮断されることになる

# 経路Filterに関する参考情報

JANOG Comment JC1000~1003 に
大変丁寧にまとめられている
http://www.janog.gr.jp/doc/janog-comment/

# 3. Path Attribute は本来の用途に使おう

## – 例えば closest exit –




```
  prefix          Next Hop   MED     IGP
> 192.0.2.0/24    R1         100     0
  192.0.2.0/24    R2         100     50
```

```
  prefix          Next Hop   MED     IGP
  192.0.2.0/24    R1         100     50
> 192.0.2.0/24    R2         100     0
```

IGPによる制御

(*) MEDが異なる経路でもclosest exitしたい場合はMEDを上書きする必要がある

IGPを用いる代わりに，MEDを加算することでも一応実現できる

| | |
|---|---|
| **Juniper**: | metric add 500 |
| **Cisco**: | set metric +500 |

× router間でMED加算して回る必要がある

△ こちらもMEDの上書きが必要

(“壁”になっている 500 という数値が十分かどうかはpeer partnerの広告policy次第)

・ 本来のMEDの用途とは異なる

# 受信経路に手を入れる
# ＝ 経路制御する
# 方法を決めておく
# (運用設計)

# 経路制御の選択肢

- transit / peer 経路
  - LPの微調整
  - AS_PATH prepend
  - MED微調整
  - passive IGP
  - 経路を止める

- 顧客経路
  - 顧客からの依頼に基づく場合を除き，操作しない

- LP / MED などの数値はなるべく **弱くなる** 方に制御する
  - ヘンにtrafficを吸い込むのを防ぐ
  - MED: 増やす
    LP: 減らす (多くの実装でdefault: 100)
- AS_PATH prepend
  prepend された経路がRIBに残ってしまう可能性があるので，なるべく避ける
  - transitを売っている場合など，全体として経路が遠く見えてしまうのは良くない
- passive IGPは経路ごとの制御ができない

- LP / MED / AS_PATH 操作
  - **なるべくメンテナンス頻度を下げる**
    - peer AS
    - nexthop
    - origin AS
    - AS_PATH
    - prefix

  の順で操作

  (例えばprefixは時間が経つと変化する可能性が高い)

```
protocols {
    bgp {
        neighbor x.x.x.x {
            import [ regular irregular finalize ];
        }
    }
}

policy-options {
    policy-statement regular {
        from { ... }
        then {
            # 通常のpolicy
            next policy;
        }
    }

    #
    # remove me soon !!
    #
    policy-statement irregular {
        from { ... }
        then {
            # irregular なpolicy
            next policy;
        }
    }
    policy-statement finalize {
        then accept;
    }
}

# Juniperの例
```

```
no route-map route-filter
route-map route-filter permit 10
  ! 通常のpolicy
route-map route-filter permit 20
  ! 通常のpolicy
route-map route-filter permit 30
  ! 通常のpolicy

!
! remove me soon !!
!
route-map route-filter permit 25
  ! irregular なpolicy

! Cisco の例
```

# irregularなことは目立つように / 消しやすく

# よく使ったのは次の3種類くらい

# 経路制御の選択肢（受信）

# － LP 制御 －

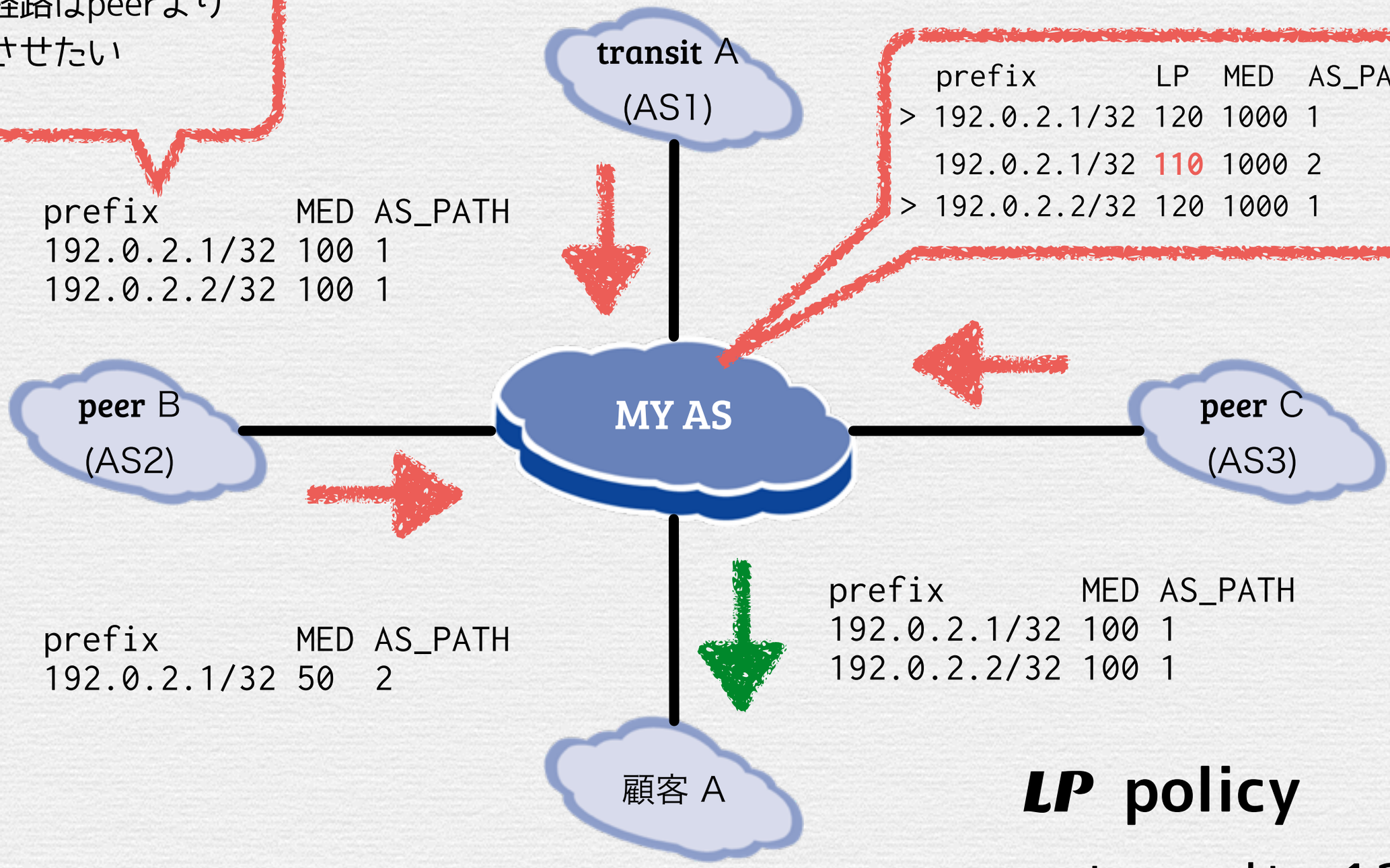
一部経路はpeerより
優先させたい

```
prefix          MED AS_PATH
192.0.2.1/32    100 1
192.0.2.2/32    100 1
```

```
prefix          MED AS_PATH
192.0.2.1/32    50  2
```

```
prefix          MED AS_PATH
192.0.2.1/32    100 1
192.0.2.2/32    100 1
```

```
   prefix          LP  MED  AS_PATH
>  192.0.2.1/32    120 1000 1
   192.0.2.1/32    110 1000 2
>  192.0.2.2/32    120 1000 1
```




peer B からの経路について，LP を下げる

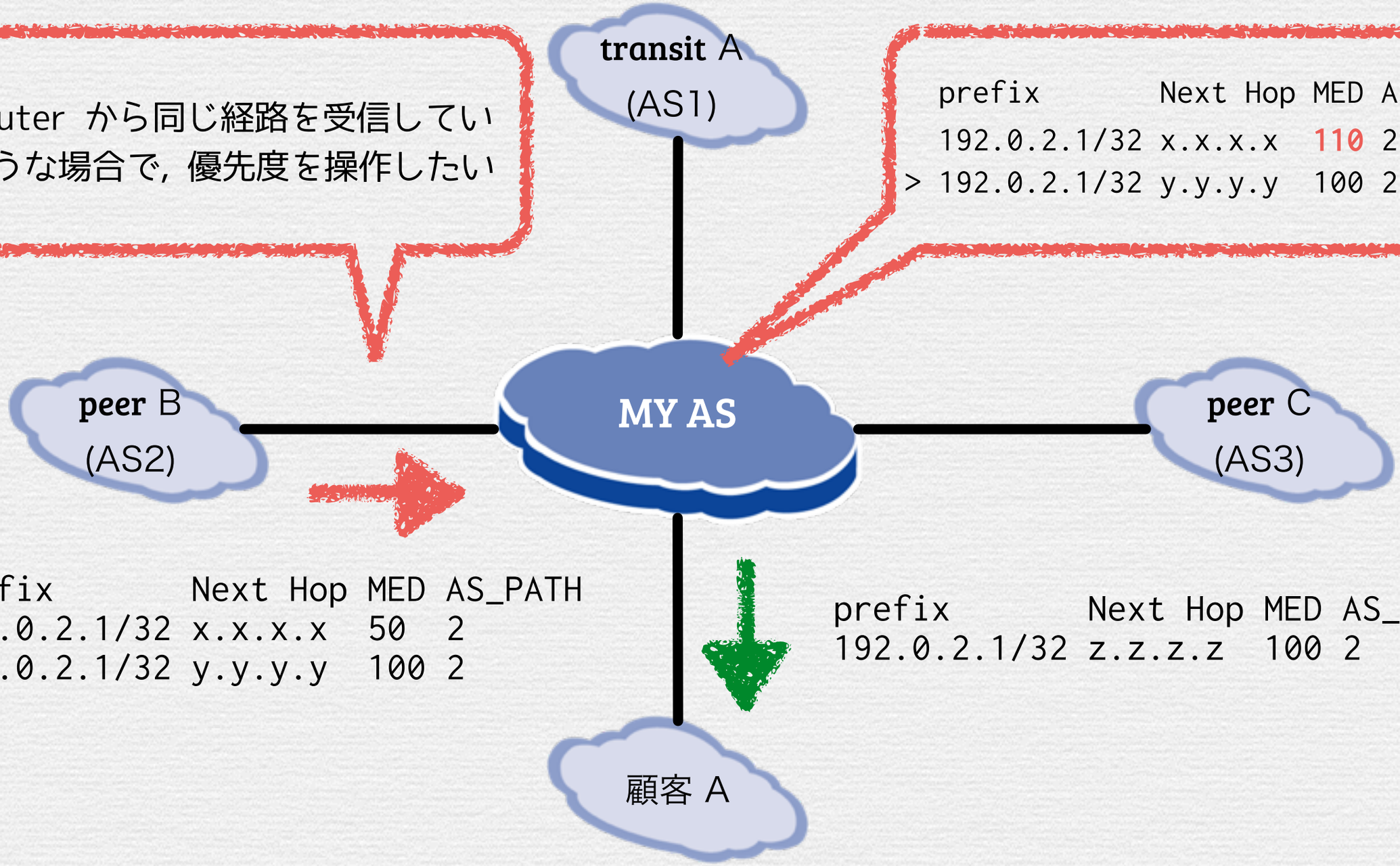
***LP* policy**

- transit 120
- peer 200

# 経路制御の選択肢（受信）

# － MED 制御 －

2 router から同じ経路を受信しているような場合で，優先度を操作したい




```
  prefix          Next Hop MED AS_PATH
  192.0.2.1/32    x.x.x.x  110 2
> 192.0.2.1/32    y.y.y.y  100 2
```

```
prefix          Next Hop MED AS_PATH
192.0.2.1/32    x.x.x.x  50  2
192.0.2.1/32    y.y.y.y  100 2
```

```
prefix          Next Hop MED AS_PATH
192.0.2.1/32    z.z.z.z  100 2
```

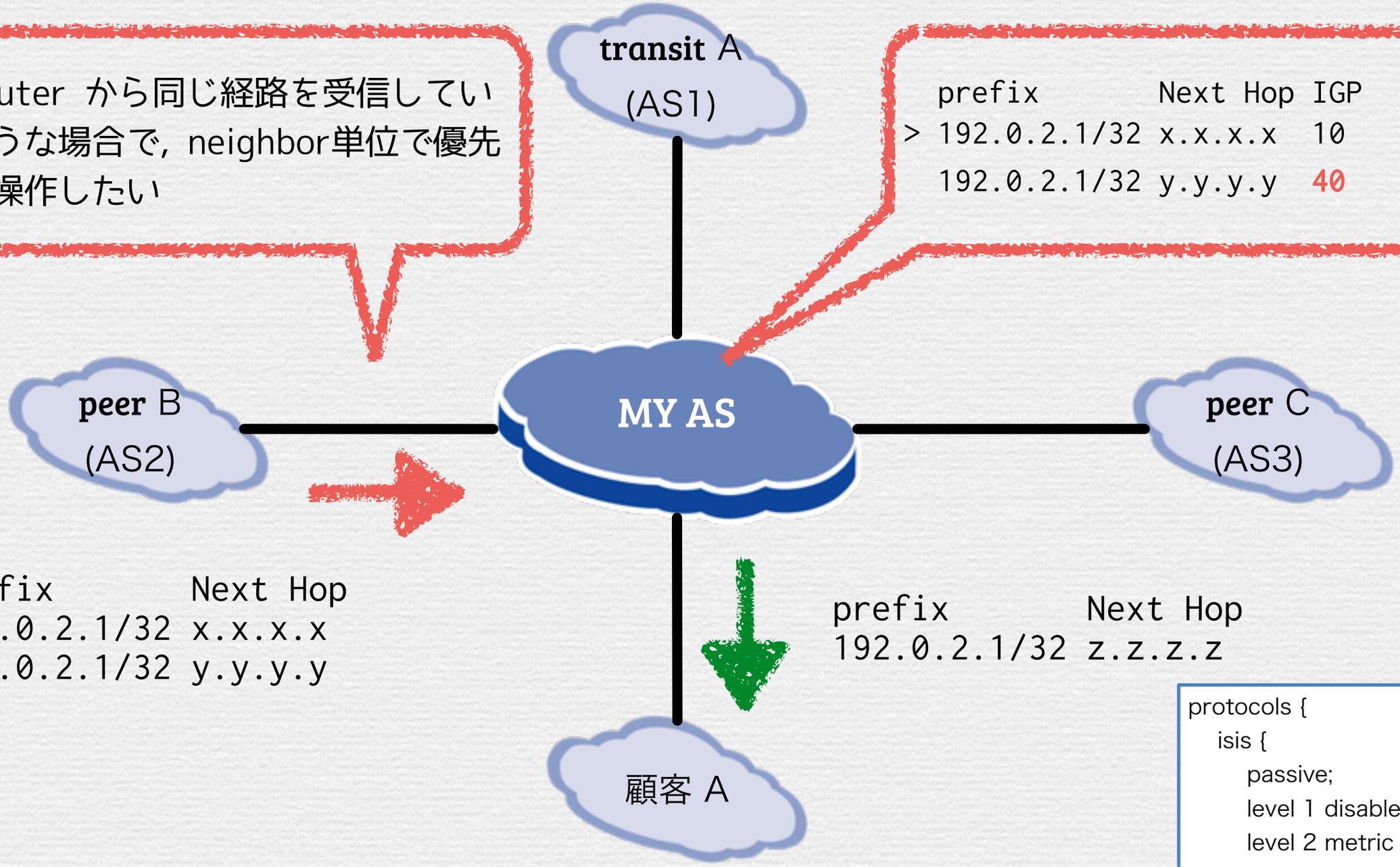
peer B からの経路について，一方のMEDを大きくする
(大きな値をセットする or **加算して大きくする**)

# 経路制御の選択肢（受信）

# – IGP 制御 –

2 router から同じ経路を受信しているような場合で，neighbor単位で優先度を操作したい

```
  prefix          Next Hop IGP
> 192.0.2.1/32    x.x.x.x  10
  192.0.2.1/32    y.y.y.y  40
```




```
prefix          Next Hop
192.0.2.1/32    x.x.x.x
192.0.2.1/32    y.y.y.y
```

```
prefix          Next Hop
192.0.2.1/32    z.z.z.z
```

通常だと，MY ASから見て x.x.x.x，y.y.y.y へのIGP costは両方 10 だが，一方だけ 40にする

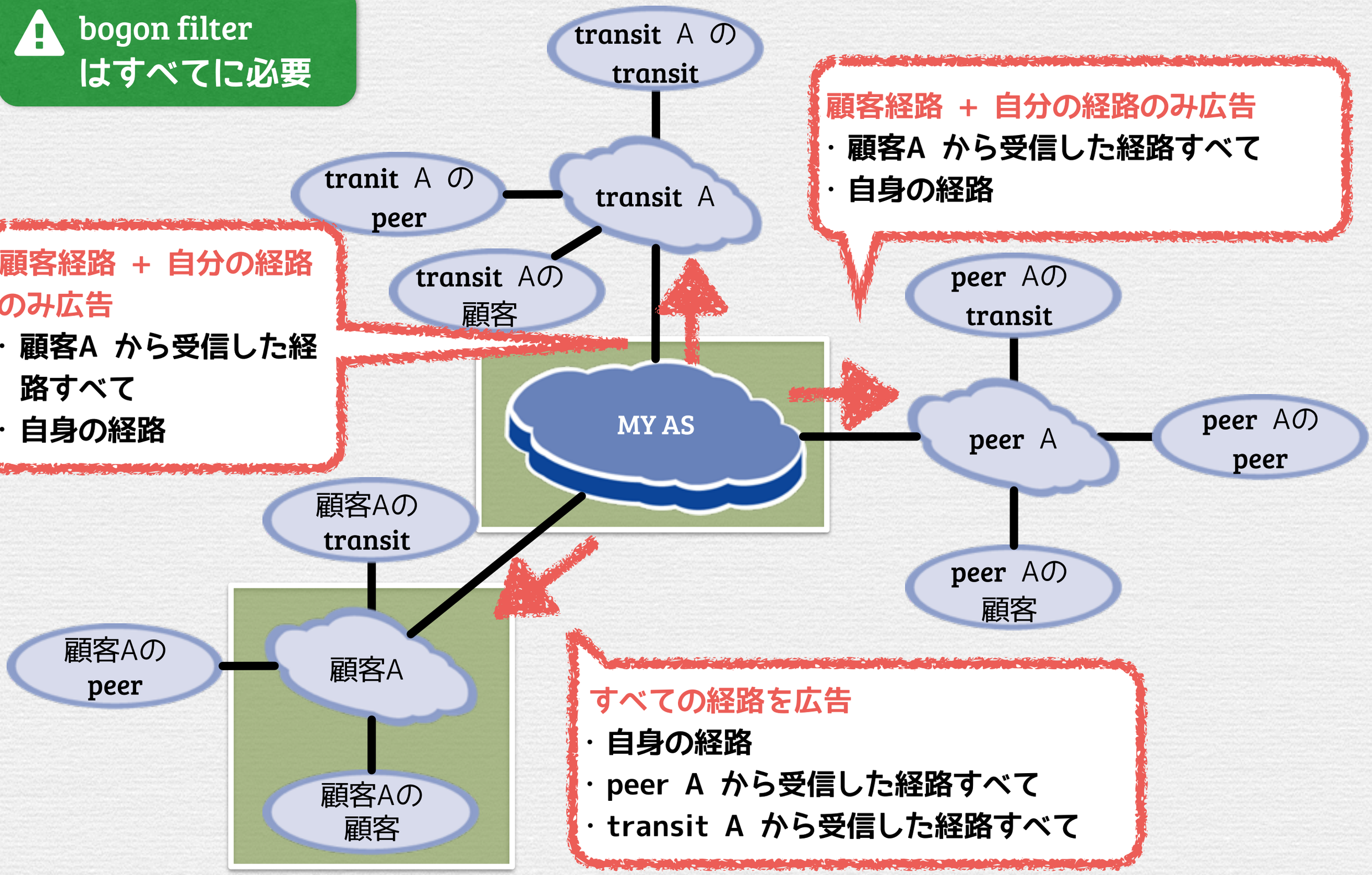
```
protocols {
    isis {
        passive;
        level 1 disable;
        level 2 metric 30;
    }
}

# Juniper の例
```

# eBGP

# 経路広告

# eBGP Policy の基本（広告）

**bogon filter はすべてに必要**




**顧客経路 + 自分の経路のみ広告**

- 顧客A から受信した経路すべて
- 自身の経路

**顧客経路 + 自分の経路のみ広告**

- 顧客A から受信した経路すべて
- 自身の経路

**すべての経路を広告**

- 自身の経路
- peer A から受信した経路すべて
- transit A から受信した経路すべて

# BGP Community 便利

・ 経路の種別による”マーク”をBGP communityとして付与すると顧客は便利に使える

・ 外部ASから，**どの地域**で受け取ったか?
・ 外部ASとは，**transit**か? **peer**か? **顧客**か?

# prepend community とかどうですか?

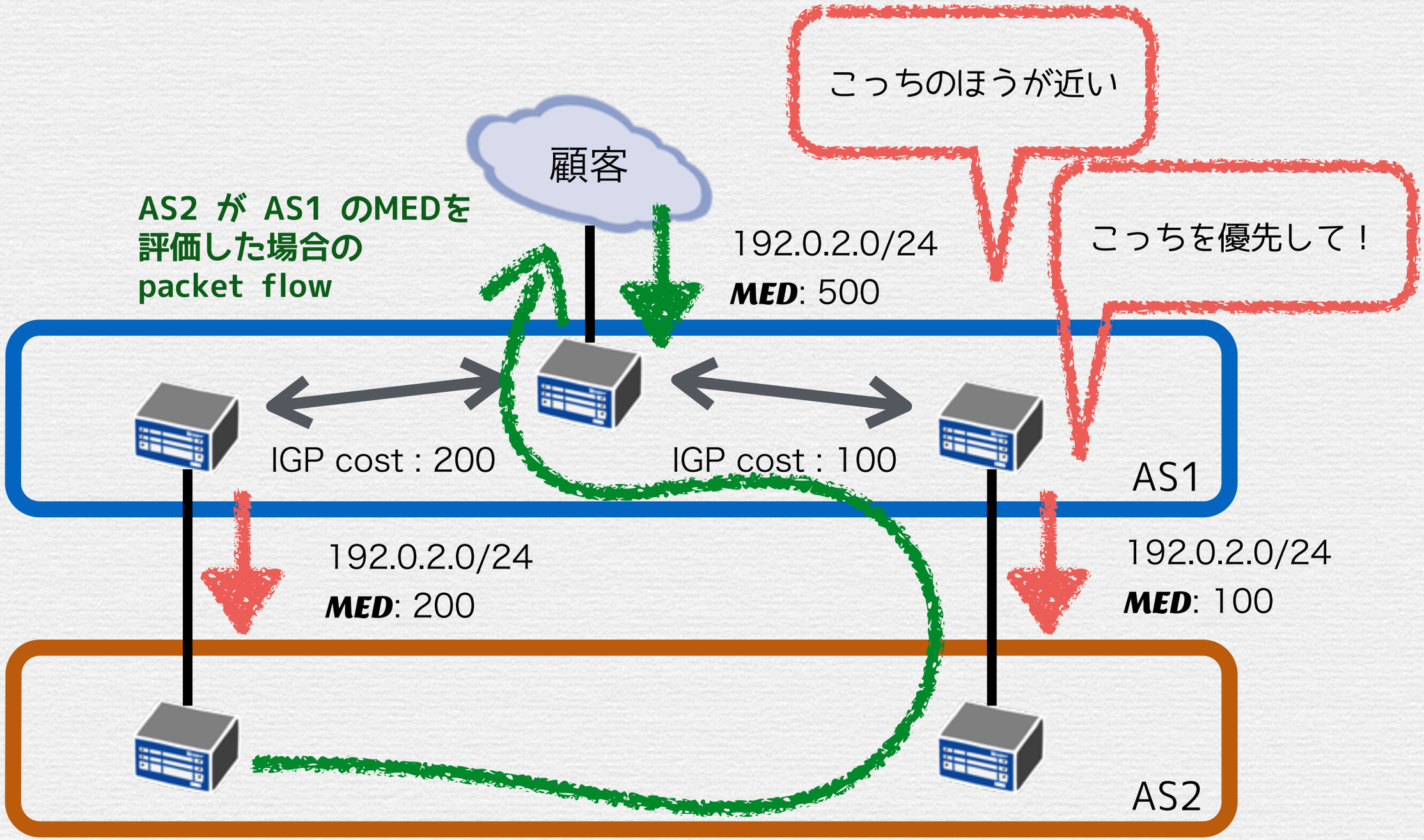
**顧客に**，自AS ➔ 外部ASへの広告時の
AS_PATHを制御させる


| | | | |
|---|---|---|---|
| community: | 7521:102 | community: | N/A |
| ***AS_PATH***: | 1 | ***AS_PATH***: | 7521 7521 7521 1 |

# 広告経路を扱うときに気にするべきポイント3つ

1. MEDをちゃんと付ける
2. 経路Filter
3. ASPATH prepend ってちょっと強い

# 1. MEDをちゃんと付ける



- metric-out igp が便利
  - IGP costを広告時のMEDとして利用

```
protocols {
    bgp {
        group ebgp {
            metric-out igp;
            neighbor x.x.x.x { ... }
        }
    }
}

# Juniper の例
```

```
router bgp xxxx
  ...
  neighbor y.y.y.y route-map ebgp
route-map ebgp
  ...
  set metric-type internal

! Cisco の例
```

⚠ 外部ASが常にMEDを評価してくれるとは限らない
- ダメもとでも広告しておく価値あり
- 評価してもらいたいなら，交渉

# 2. 経路Filter

- 内部の細かい経路は広告しない
  - connected(direct)経路はBGPにredistributeしないなど
    - するときにはno-export communityを付けておく
- bogonその他，internetに流すべきでない経路が含まれないかを再確認
- **remove-privateしておく**（private ASは削って広告）

# 3. AS_PATH prepend

- 制御できるが，比較的制御が強い
  - “設計”に含めるのは，やりすぎなイメージ
  - どちらかというとtrouble shootなどの暫定対処用

# 広告経路に手を入れる
# ＝ 経路制御する
# 方法を決めておく
# (運用設計)

# 経路制御の選択肢（広告）

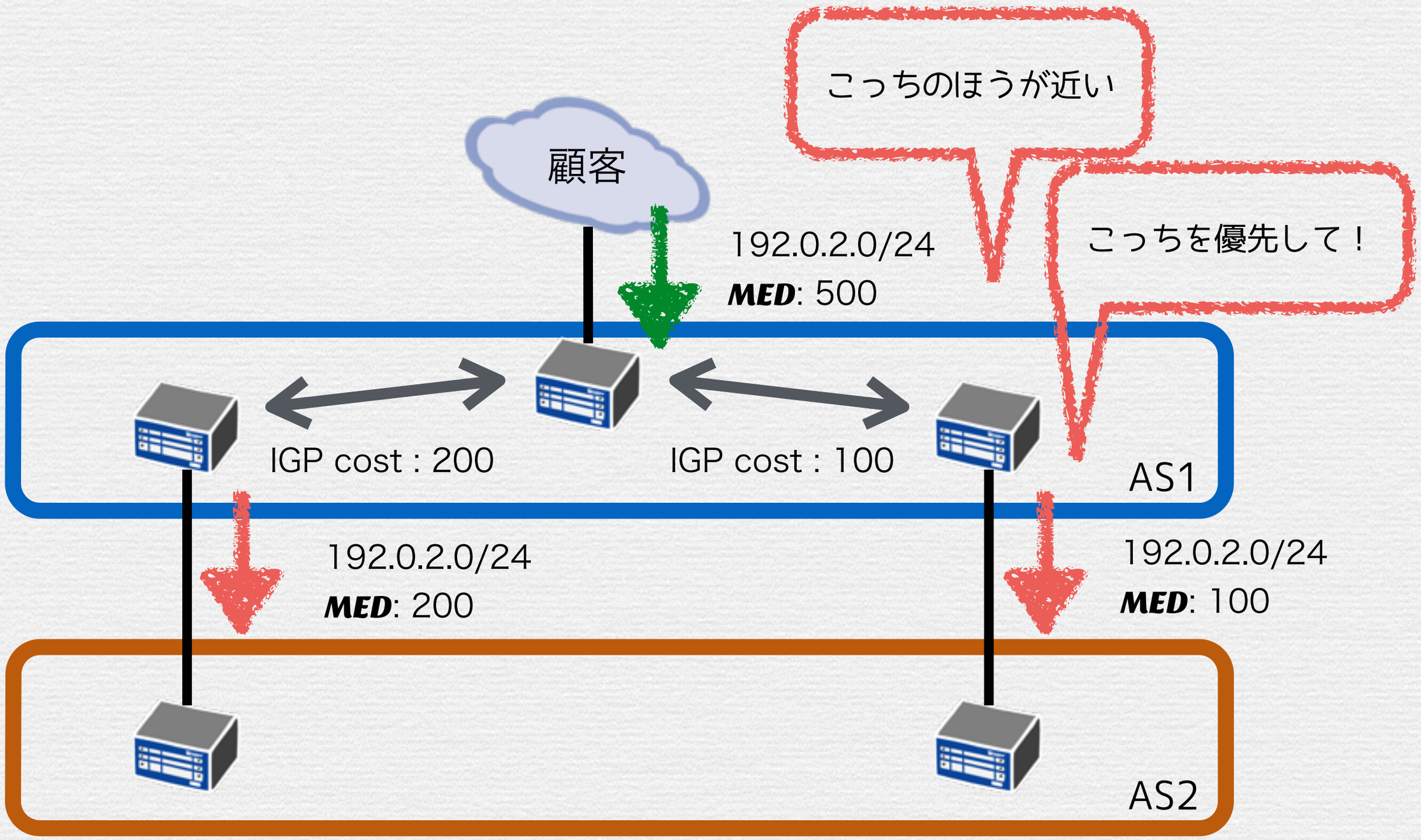
- 誰に対して制御するかについて
  - transit / peer
    - AS_PATH prepend
    - MED 微調整
  - BGP community（一部transitのみ）
    - transit AS 内のLP を制御
    - transit AS ➔ 他AS 経路広告を制御（広告しない / prepend）
  - 経路広告を止める

- 顧客
  - 顧客からの依頼に基づく場合を除き，操作しない

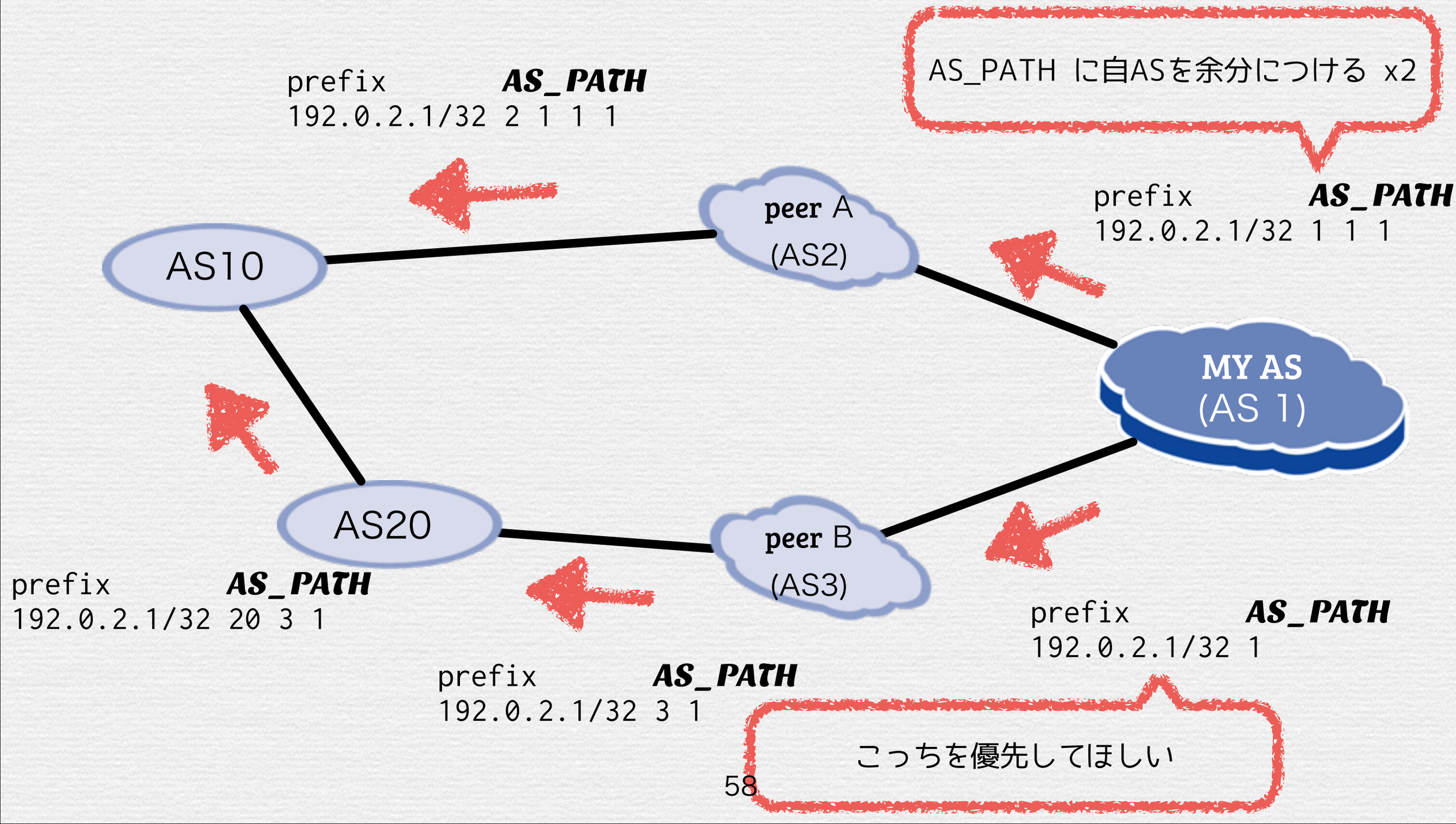
# あまり

# 手がない

# 経路制御の選択肢（広告）
## － MED 制御 －



# 経路制御の選択肢（広告）
## － AS_PATH 制御 －



# 経路制御の選択肢（広告）

- 広告経路の制御が効かない場合も多々ある
  - 顧客 / peering partner / transit提供者にも彼らなりのpolicyがあるので，やむを得ない
  - **接続しているtransit / peer のrouting設計を理解することがすごく重要**
    - MEDは効くか?
    - AS_PATH prepend可能か?
    - best pathはどのpath attributeで決まっているか?
    - 制御系BGP communityはあるか?
    - **そのような設計になっているのはなぜか?**

- 直接答えてもらえればラッキー
- 推測の蓄積でも十分役立つ

以下の選択肢は高い確率で効果が期待できる

- BGP Community

奥の手！

- 経路広告を止める
  - 多くの場合，冗長性を損なう
  - more specificな経路にする(/20 = 2x /21)
    - 管理が煩雑になる

# シンプルに
# わかりやすく



http://www.flickr.com/photos/techsavvyed/5926978939/

# iBGP policy / 設計

あまりたくさん
話しませんが，
気に留めておくと
良さそうなこと2つ

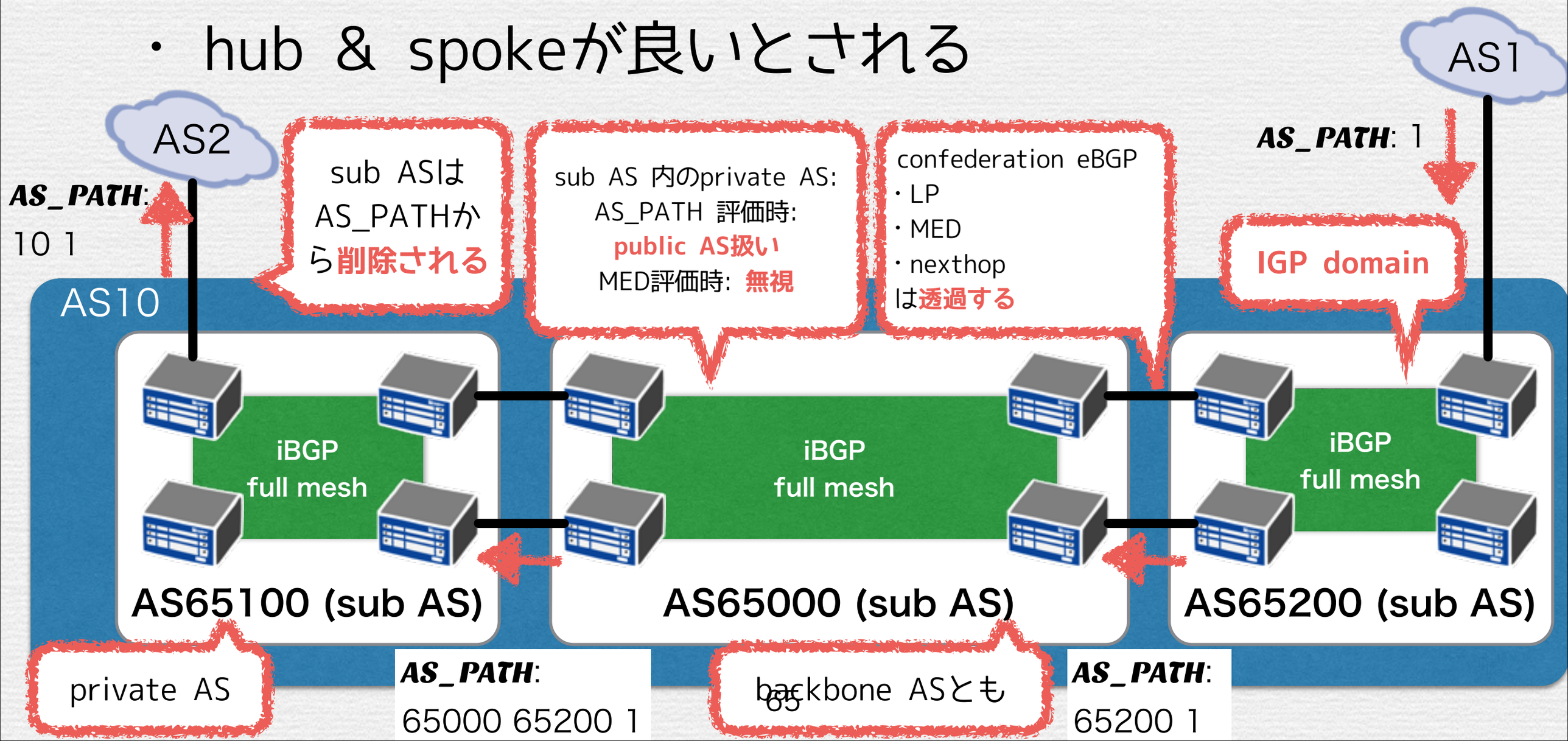
1. BGP Confederation

2. next-hop self

# 1. BGP Confederation

Route Reflector(RR)同様，BGPスケーラビリティを向上させる技術

- 複数のsub ASに分割し，sub AS間をeBGP接続
- hub & spokeが良いとされる



- 利点
  - IGP domainを分割できる
    - 大規模になると不安定になりがちなIGPへの対策
  - sub AS単位でBGP policyを分けられる
    - サービス別/国別/エリア別などで運営母体を分けるときにぴったりハマる
    - **M&Aなどにより統合したnetworkを，1ASに移行するステップとして**
- 欠点
  - sub ASの規模が大きくなるとiBGP session数 / それに伴う経路数が負荷になる
    - (bestではない)経路が多い ➔ route convergence timeが長い

sub ASが大きくなってきたら，sub AS内へのRR導入もOK
(BGP ConfederationとRRの併用)

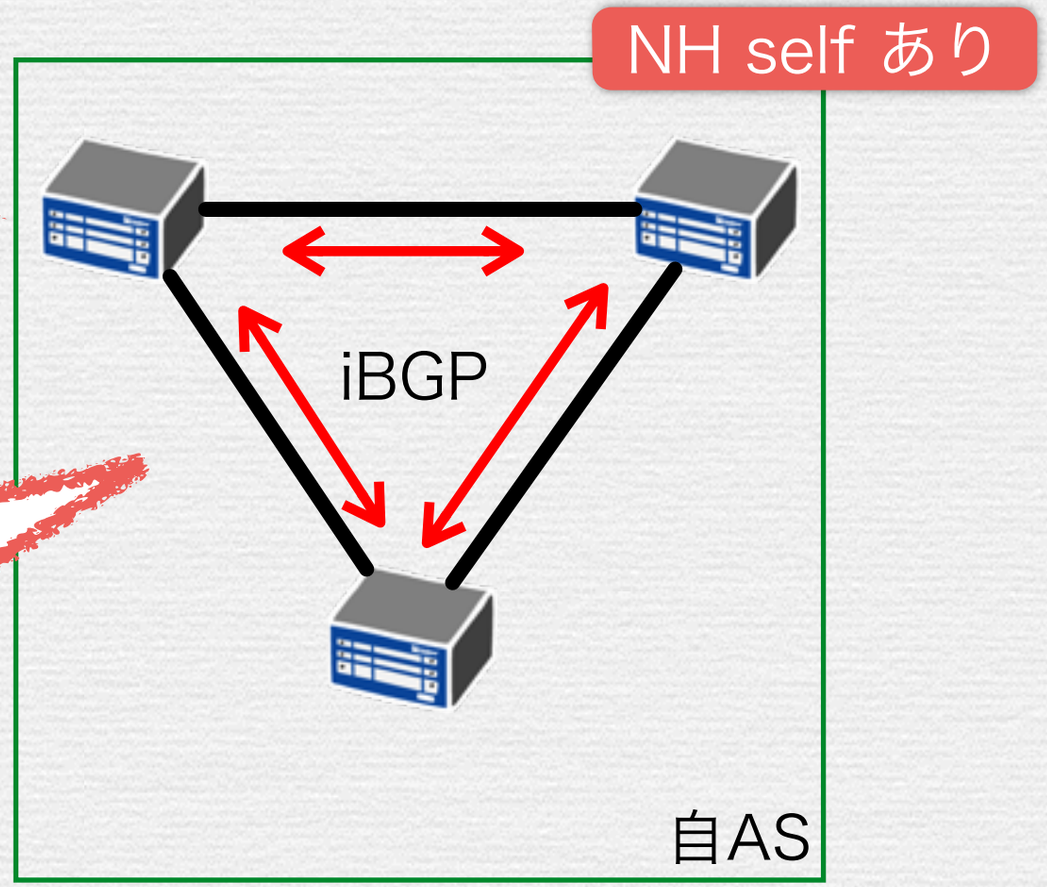
⚠ 意外と重要

# 2. next-hop self

- 自AS内のtrafficを細かく制御したい場合はnext-hop selfしないほうがいい
  - **経路制御の選択肢を1つ失う**
- したほうがいい場合もある（後述）

| Prefix | NH |
|---|---|
| 192.0.2.0/24 | x.x.x.x |
| 198.51.100.0/24 | y.y.y.y |

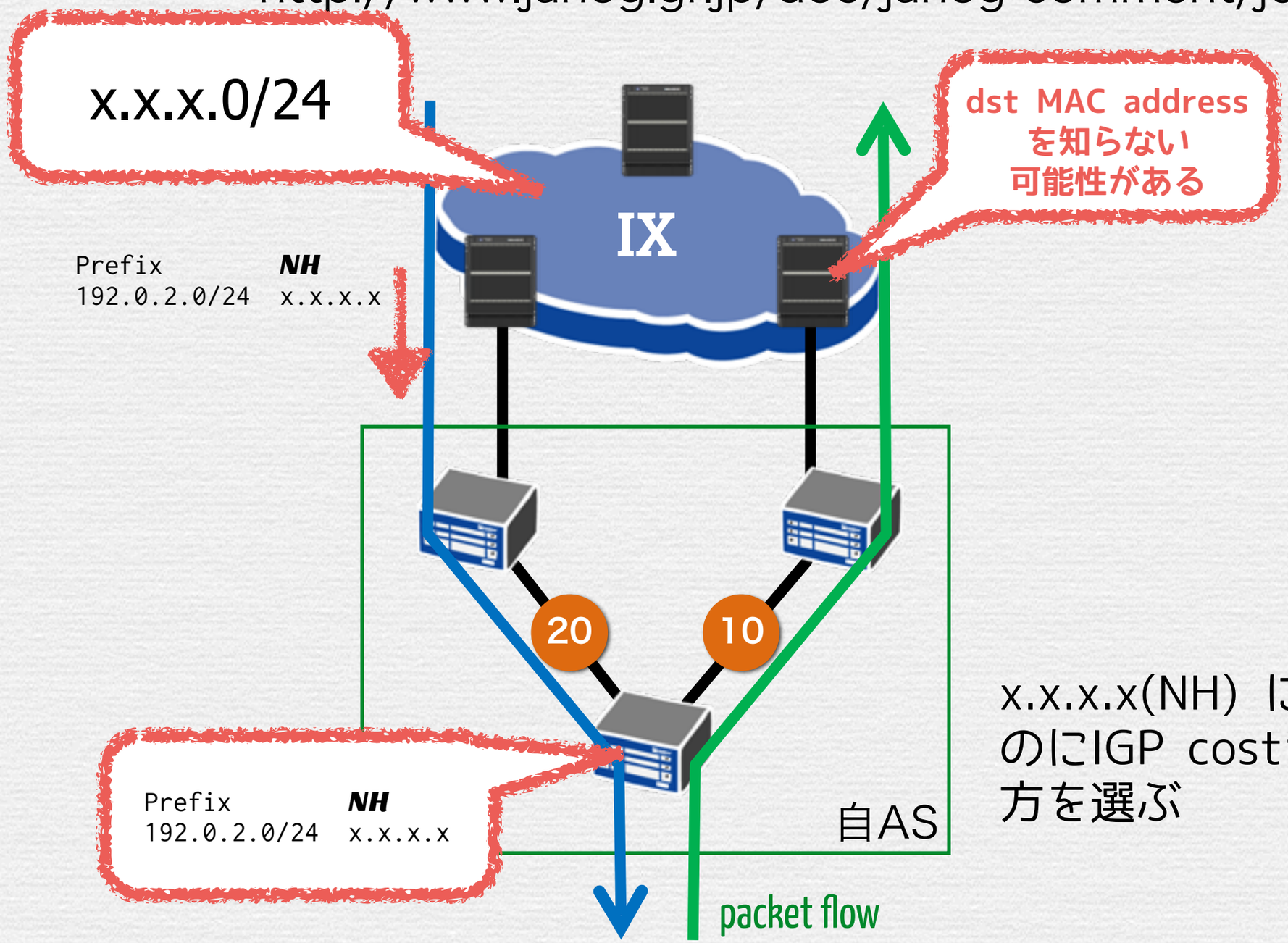
| Prefix | NH |
|---|---|
| 192.0.2.0/24 | z.z.z.z |
| 198.51.100.0/24 | z.z.z.z |



# next-hop self したほうがいい場合

IX経由でもらった経路をiBGPに流すとき

http://www.janog.gr.jp/doc/janog-comment/jc1005.txt

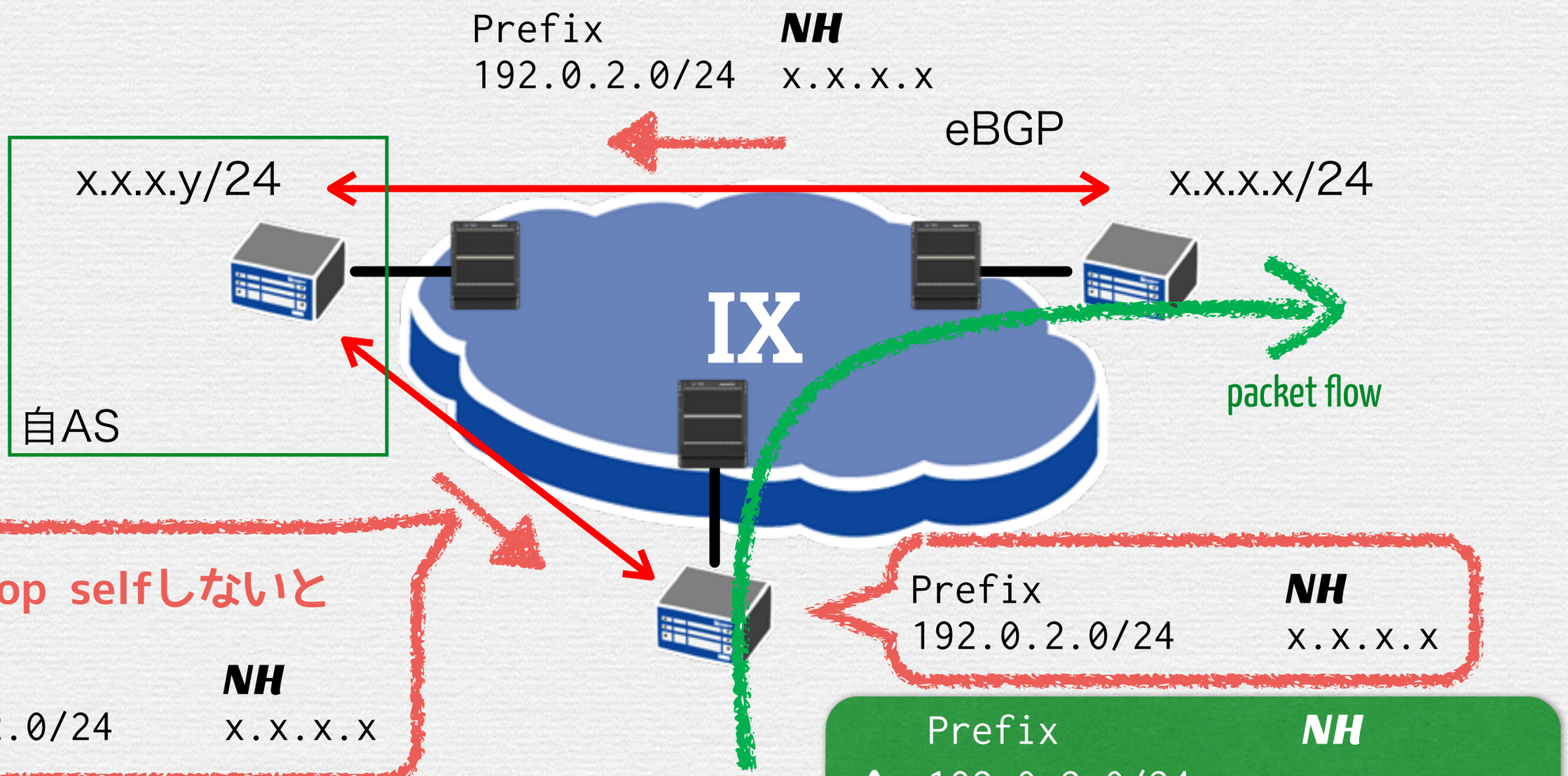



x.x.x.x(NH) に到達するのにIGP costが小さい方を選ぶ

# next-hop self したほうがいい場合

IX経由でもらった経路を同じIX上の別のeBGPの流すとき

http://www.janog.gr.jp/doc/janog-comment/jc1005.txt



# BGP の運用を考えよう

- Traffic Engineering

- その他

# Traffic Engineering

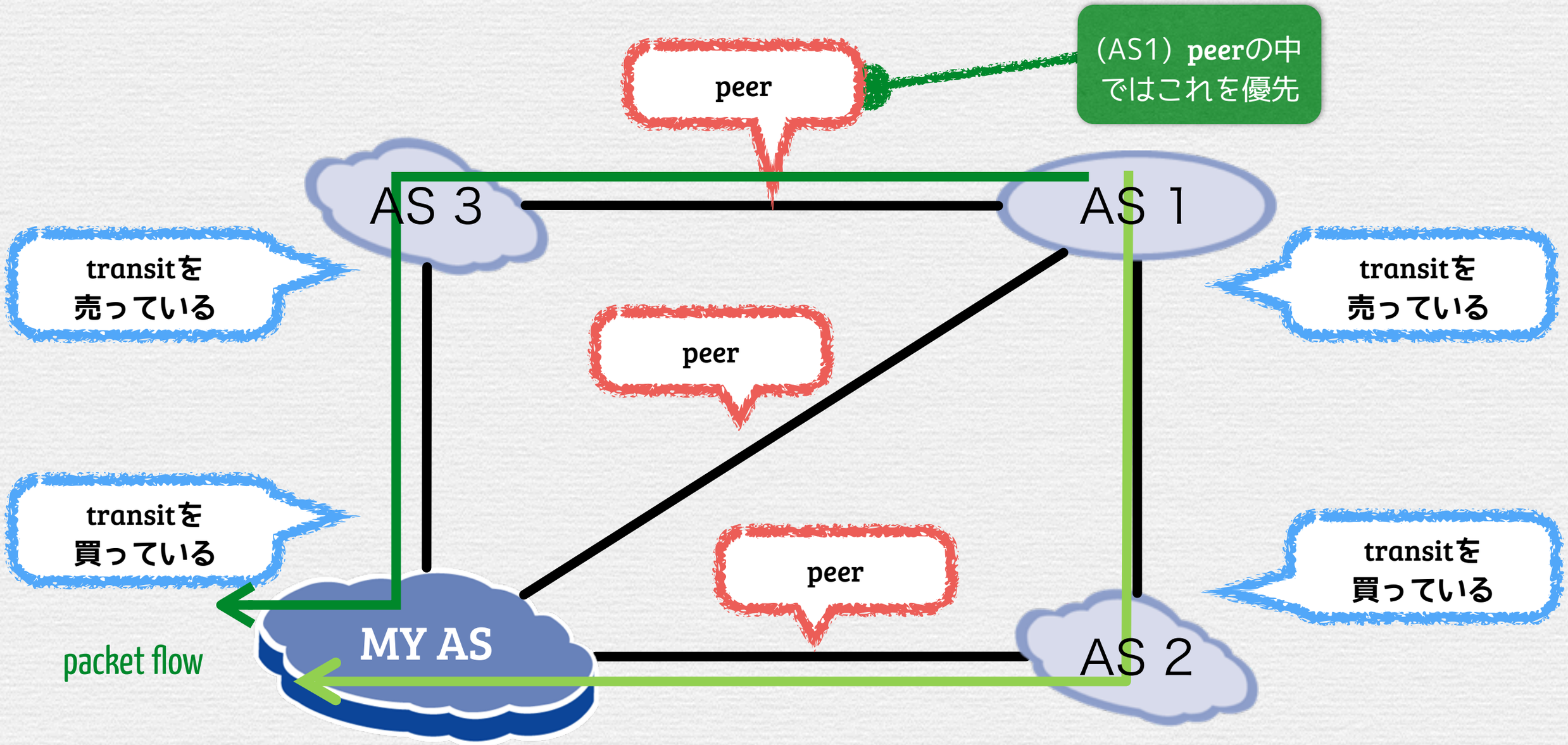
# これから，運用上問題になったケースをいくつか挙げます

みなさんも

“自分ならどうするか”

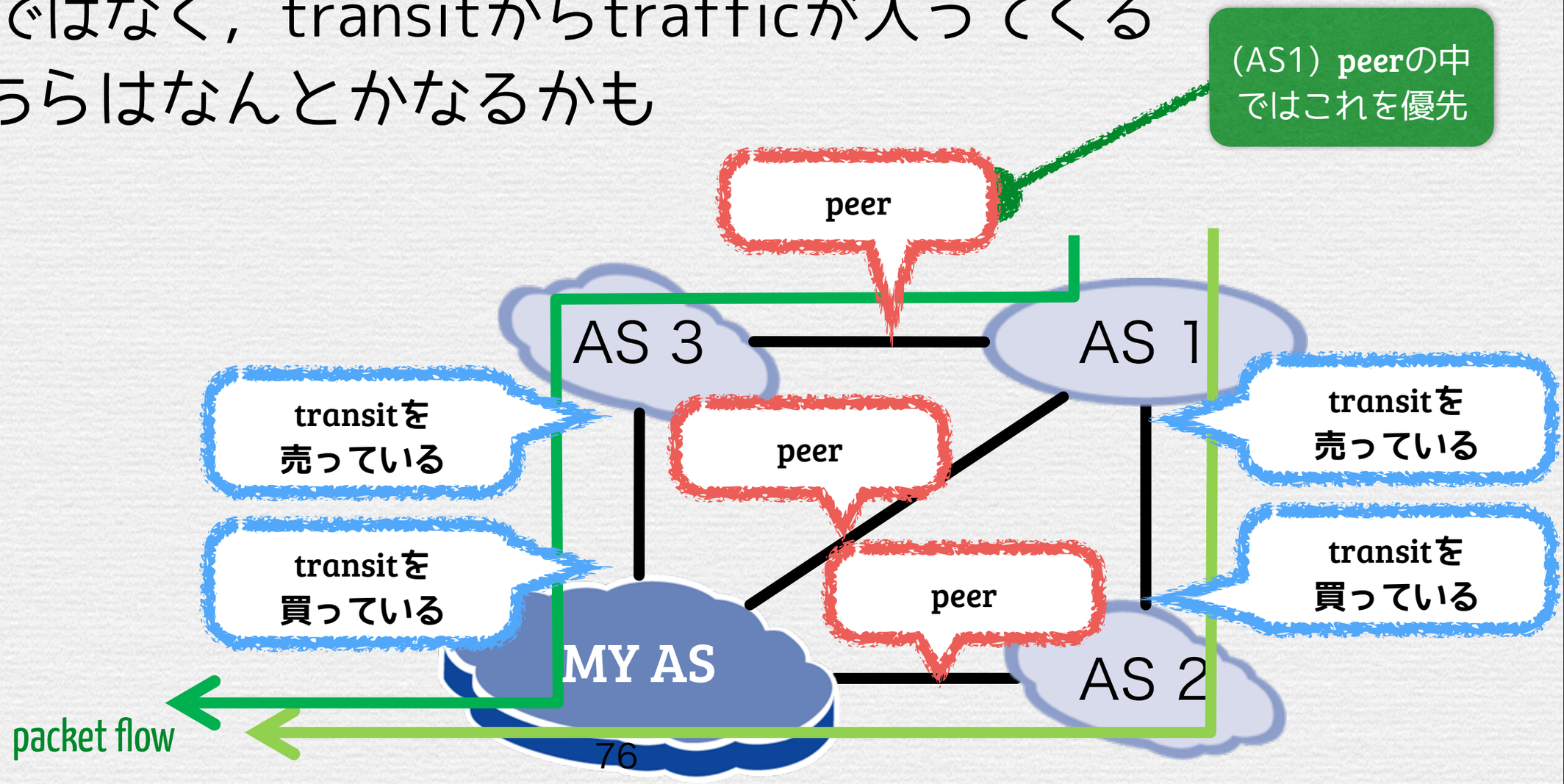
考えてみてください

# 問題1：直接peerしているのに trafficが流れてこない



# 問題1：直接peerしているのにtrafficが流れてこない

1. AS_PATH的には近いのに，別のpeerからtrafficが入ってくる
   - 図のようなケースで AS1内のroutingを変えることは困難
2. peerではなく，transitからtrafficが入ってくる
   - こちらはなんとかなるかも



# 答え：直接peerしているのに trafficが流れてこない

**○ AS1にメールする（またはAS3にメールする）**

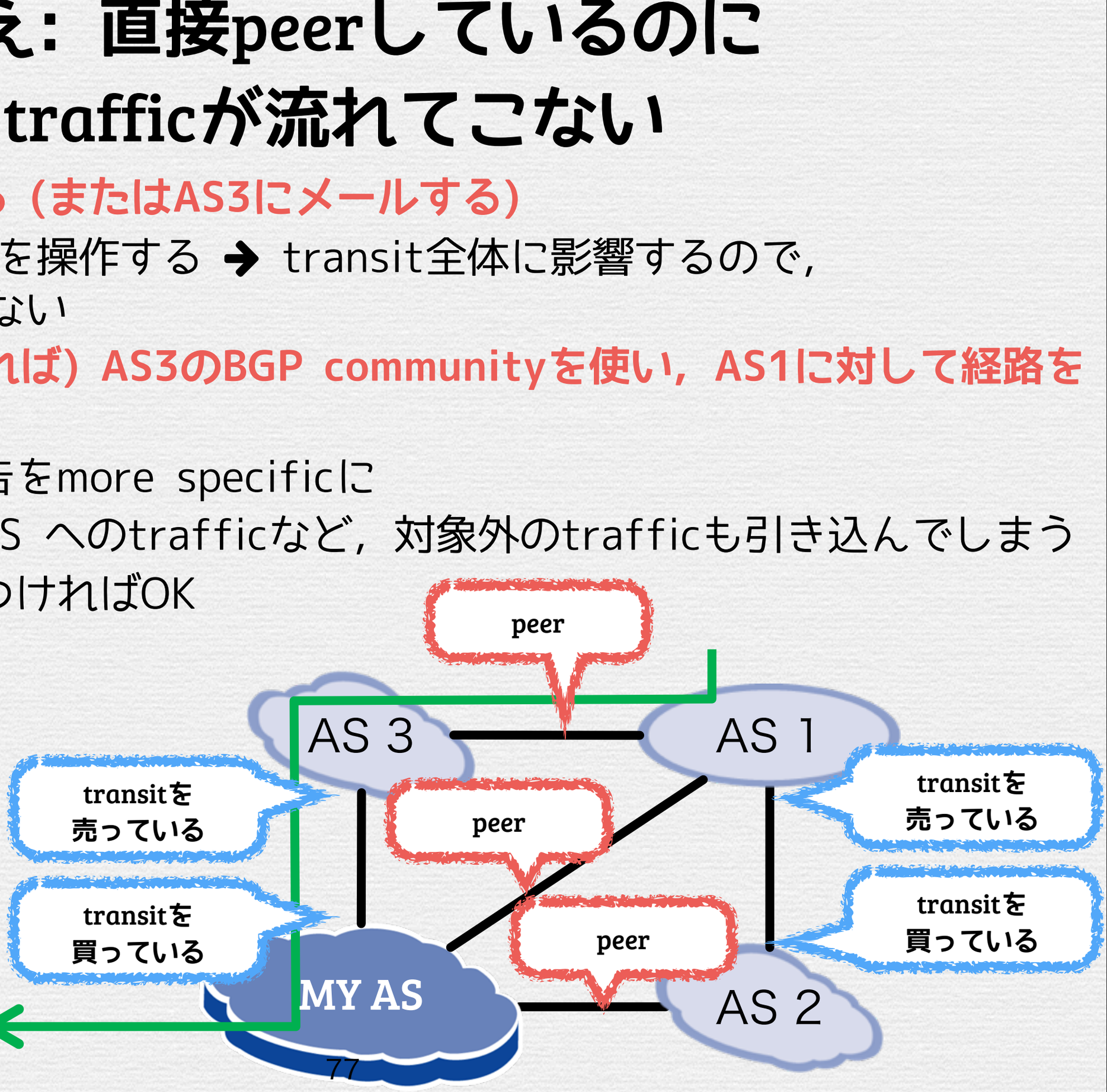
✖ AS3への経路広告を操作する ➔ transit全体に影響するので，
基本的には良くない

**○(もし提供していれば) AS3のBGP communityを使い，AS1に対して経路を止める**

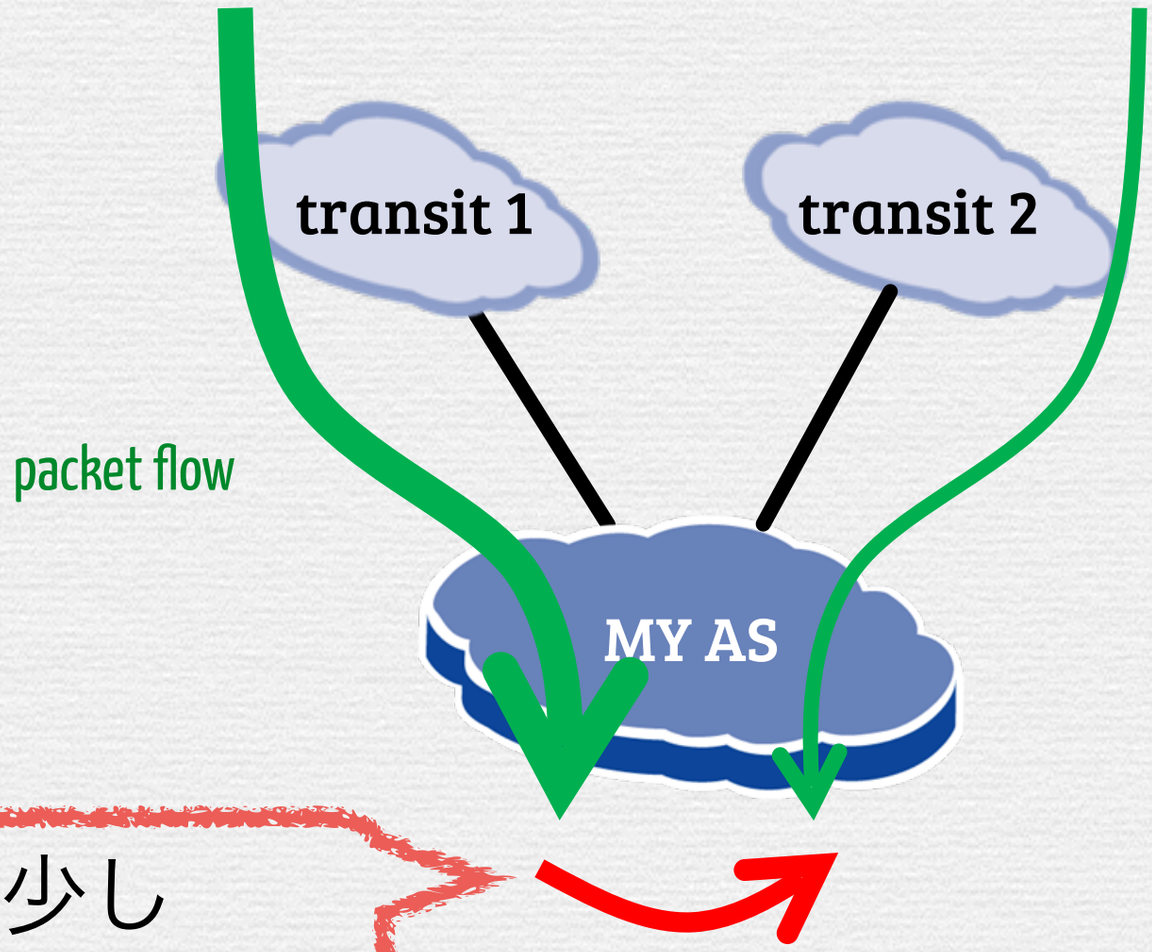
△ AS1への経路広告をmore specificに

- AS3 ➔ MY AS へのtrafficなど，対象外のtrafficも引き込んでしまう
- no-exportをつければOK

(もしAS1 が提供していれば)ダメもとでAS1のBGP communityを付与してAS3に経路広告してみるのも一手



# 問題2：transit間で trafficを動かしたい



# 答え：transit間で trafficを動かしたい

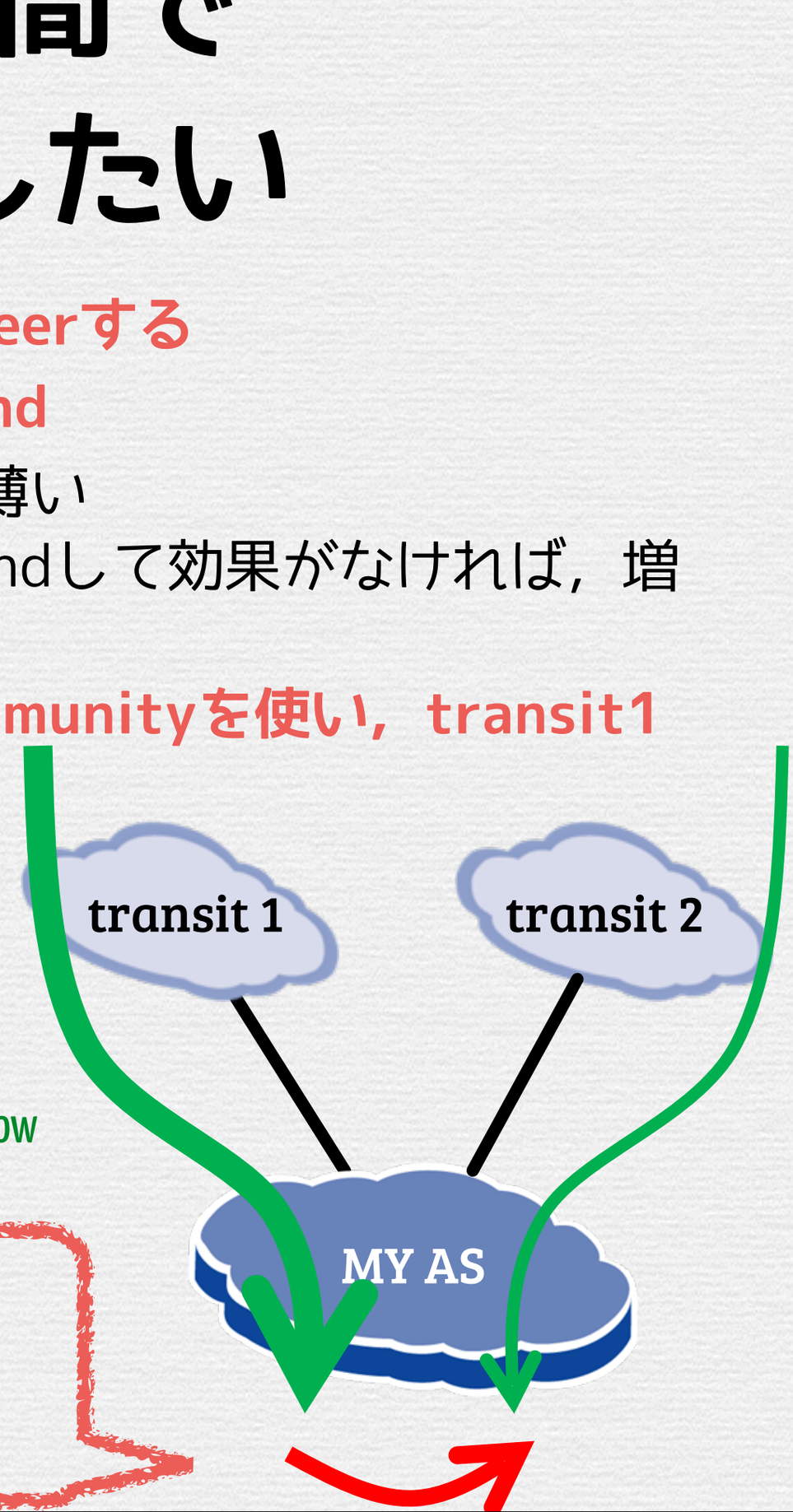
**○ (特定ASからのtrafficが大きい場合) 直接peerする**

**○ transit1への経路広告時にAS_PATH prepend**

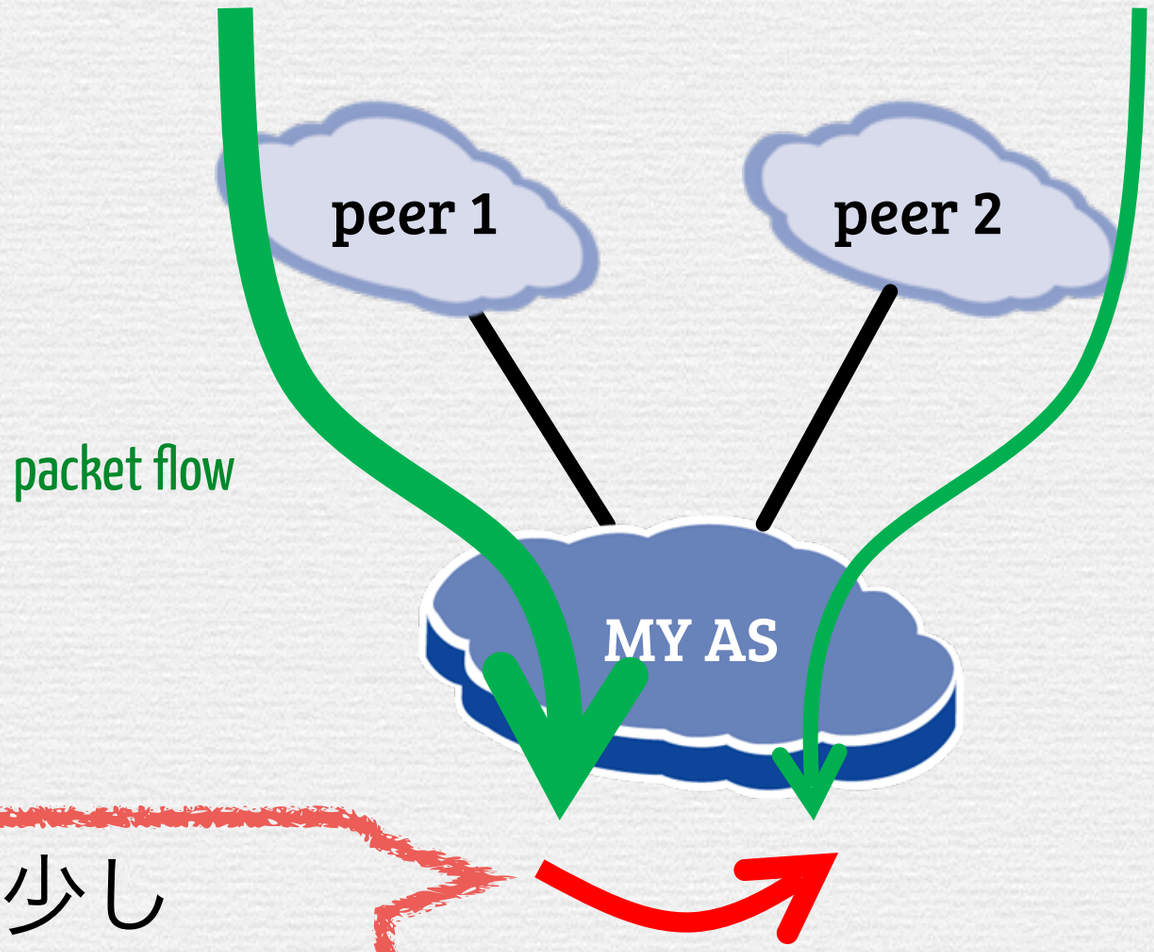
- 経験的にいくつもprependしても効果は薄い
  - 経験的には限界は+3程度．+3 prependして効果がなければ，増やしてもたぶん同じ

**○ (もし提供していれば) transit1のBGP communityを使い，transit1内でのLPを下げる**

△ 丁度いいvolumeの経路について，
- transit2への広告をmore specificに
- transit1への経路広告を止める



# 問題3: peer間でtrafficを動かしたい



# 答え：peer間でtrafficを動かしたい

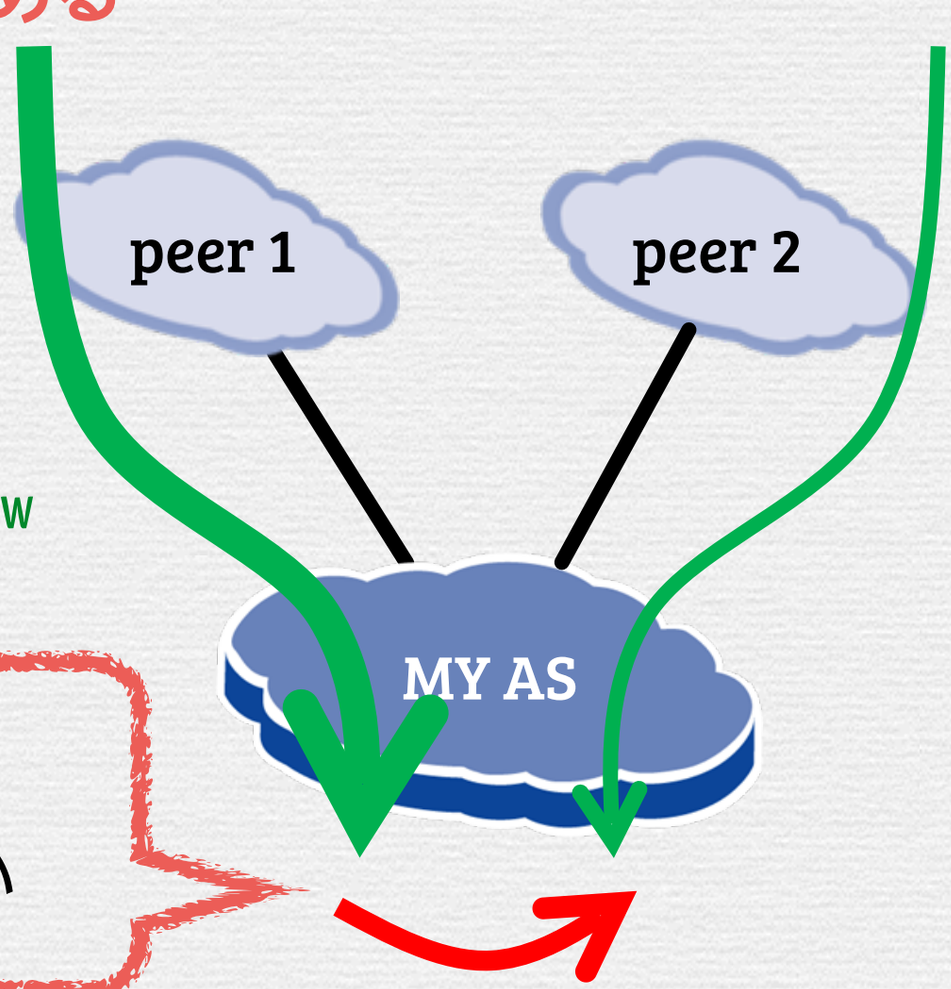
**○ peer1への経路広告時にAS_PATH prepend**

- 経験的にいくつもprependしても効果は薄い
  - 経験的には限界は+3程度．+3 prependして効果がなければ，増やしてもたぶん同じ

**○（もしpeer1と複数peerしているなら）trafficをどけたいpeer1とのeBGP sessionでのみ特定の経路広告を止める**

△ 丁度いいvolume の経路について，
- peer2への広告をmore specificに
- peer1への経路広告を止める

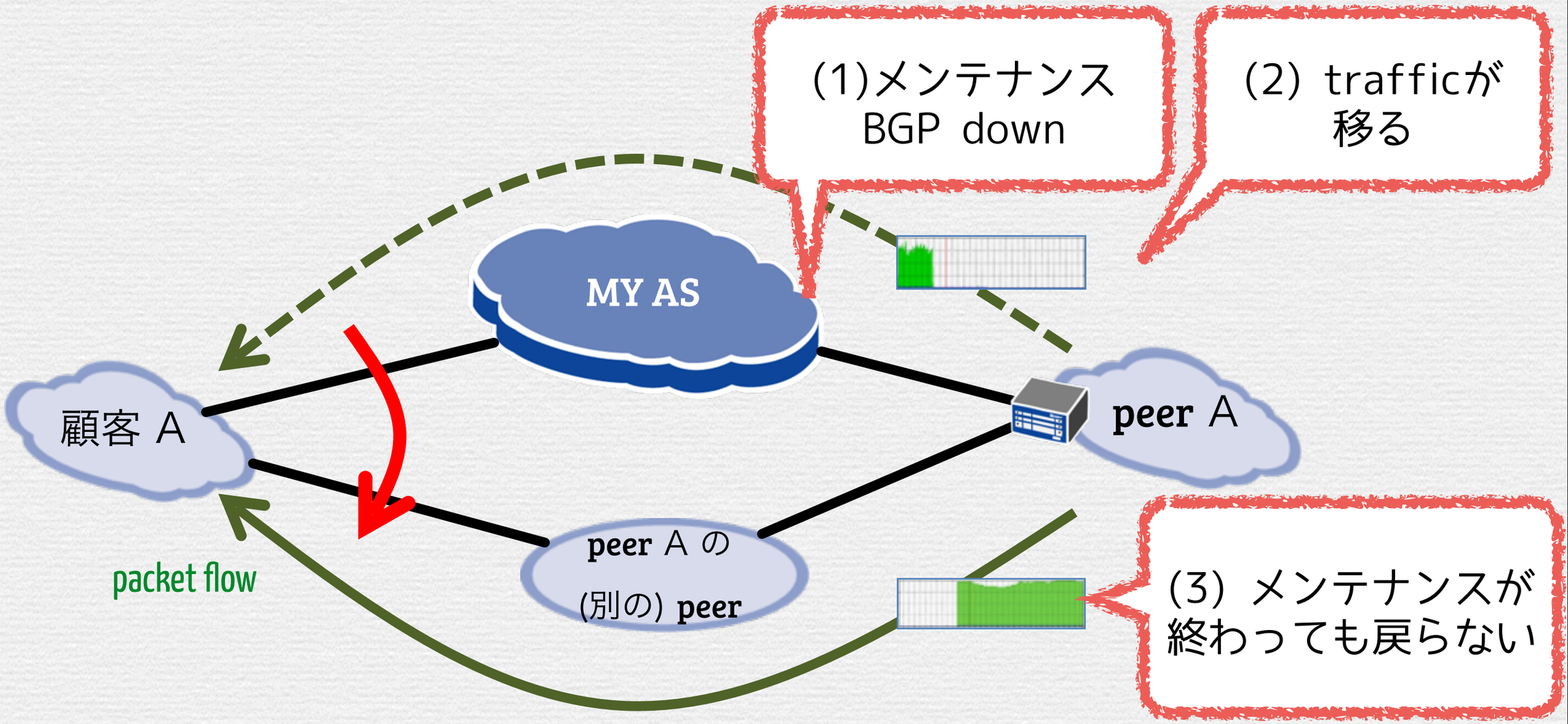
**peer1，peer2 のASNが違うので，MED操作が効かない !!**



# 補足：peer間でtrafficを動かしたい

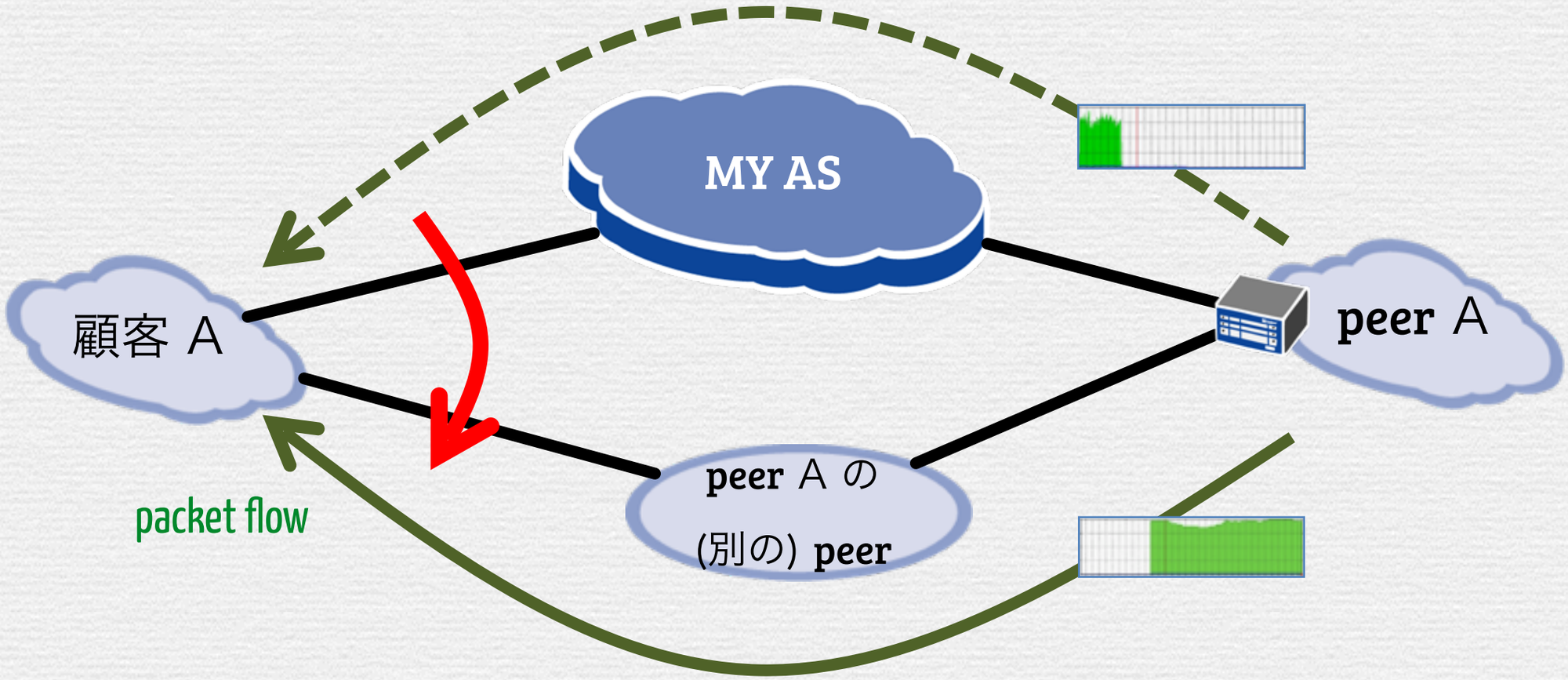
- peer間でTEしたい理由
  - 品質が悪いから
  - 時間帯により輻輳する
  - latencyが大きい
- paid peerだから
- なぜかは分からないが顧客に依頼されたから

# 問題4： peer間でよくある traffic移動



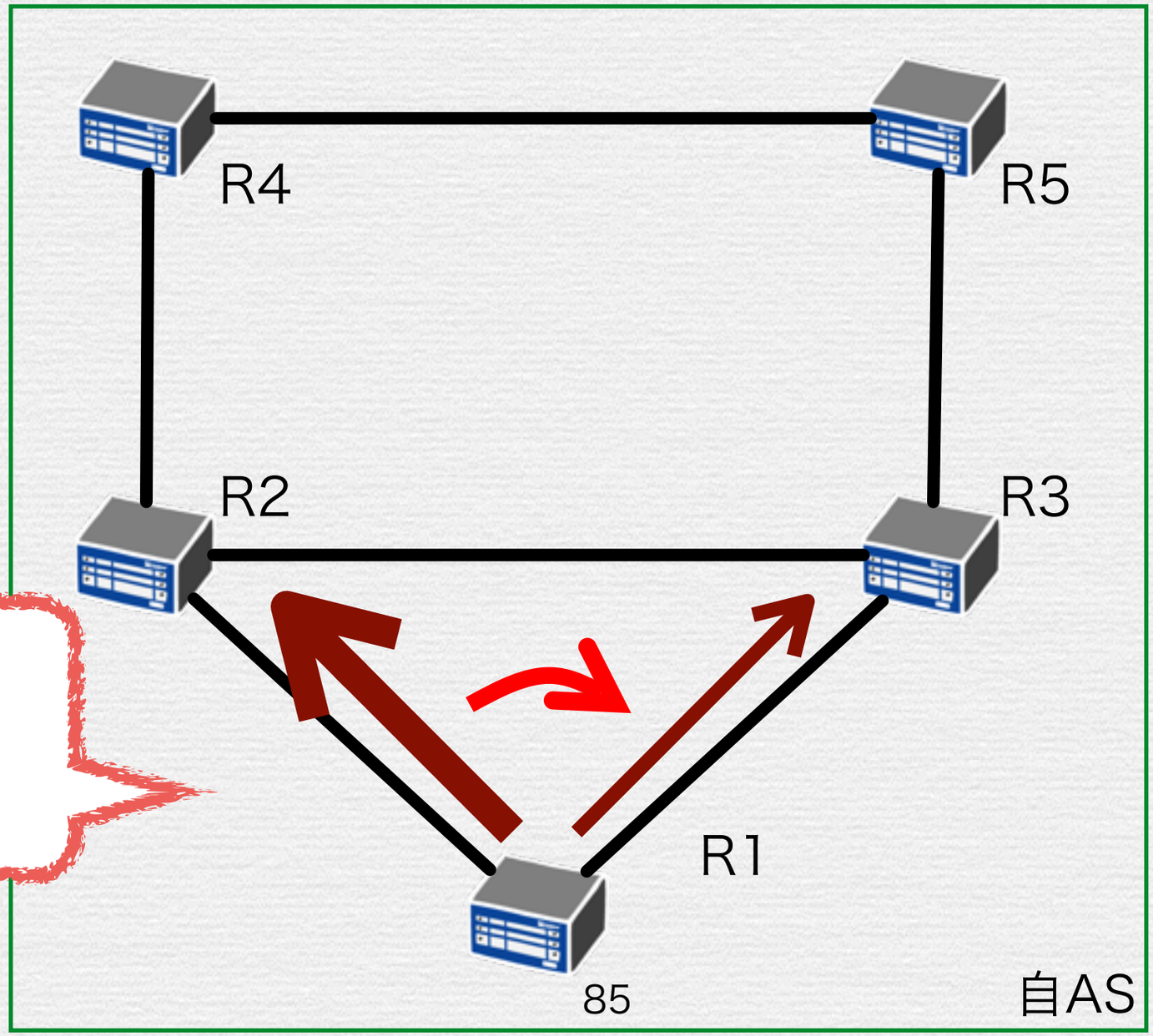
# 答え?: peer間などでよくあるtraffic移動

- 戻すのは困難




- eBGP間のbest path selectionはrouter IDではなく“経路の生存期間”で決定する場合が多い

# 問題5：自AS網内の
# traffic balanceを変えたい




少し
移したい

# 答え：自AS網内の traffic balanceを変えたい

まず R4から出るtrafficの一部をR5に移せないか考える

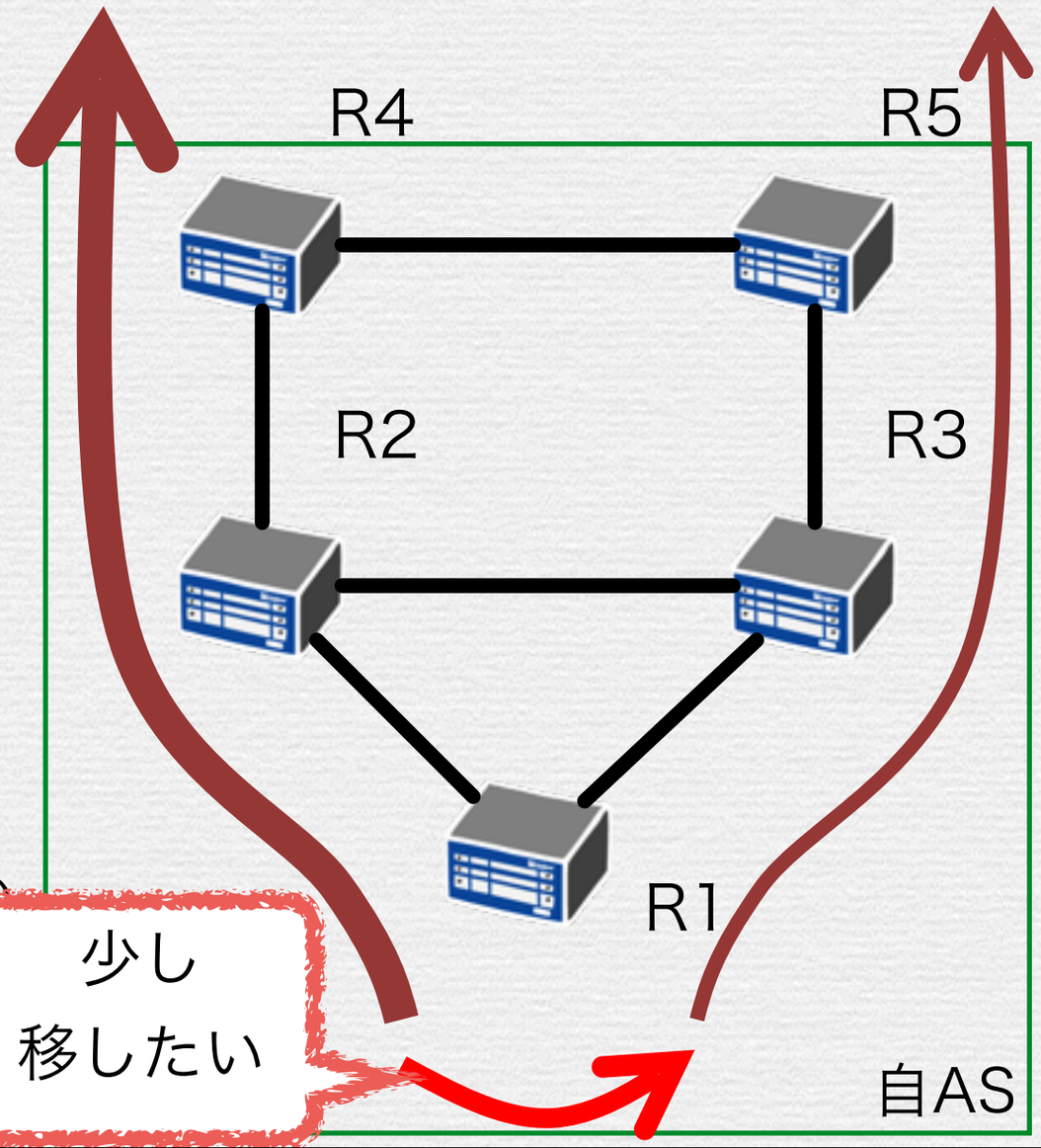
- R4，R5両方でpeerしていて，顧客ではないASがあれば，R4での経路受信時に特定経路について

✖ LP を下げる（制御が強すぎる – AS_PATHが効かなくなる）

△ AS_PATH prepend（制御がまだ少々強い．自AS内全域に影響を与える）

**○ MED を追加する**

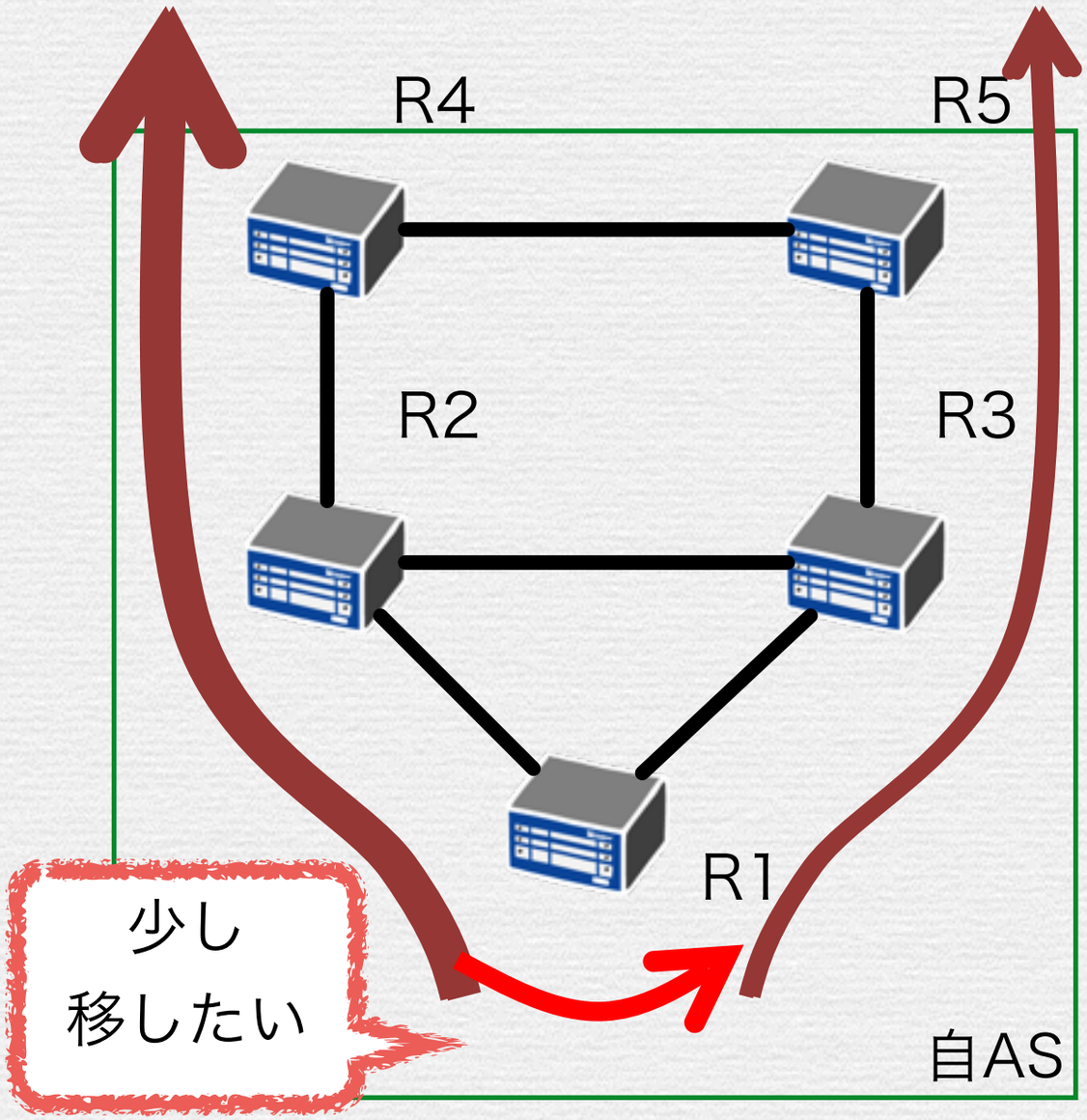
**○(MED を制御に使えない場合)**
**passive IGP costを付ける**
(そのBGP session全体に影響し，
topologyによっては劇的に変化するので注意)



# ちょうどいい移動対象trafficがなかったら

△ **R1 → R2 → R4 trafficをR1 → R3 → R5 → R4 にできないか考える**

- R2-R4のIGP costを上げる (影響範囲が大きいので注意)
- 一時的な対応には使えるかも

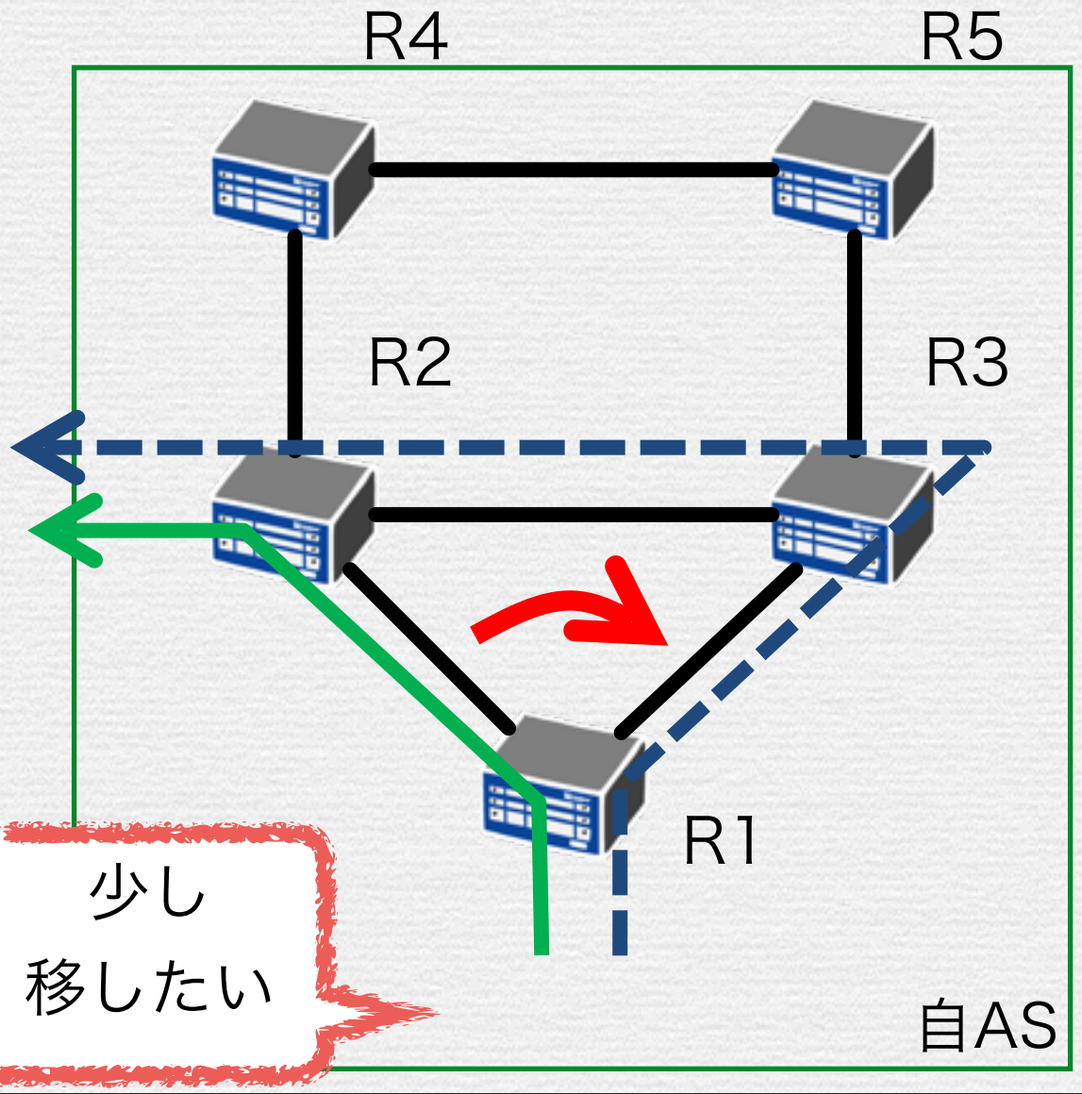


# ちょうどいい移動対象trafficがなかったら

**○ R1 → R2 traffic をR1 → R3 → R2 にできないか考える**

- (R2が持っているeBGP sessionのうち, session単位のtraffic合計で丁度いいものがあれば) next-hopになっているconnected prefix (ipv4なら/30など) へのstatic 経路をR1でR3向けに設定する.

**(iBGPでnext-hop selfしていない場合に限る. また構成によってはrouting loopが起きる可能性があるので注意)**

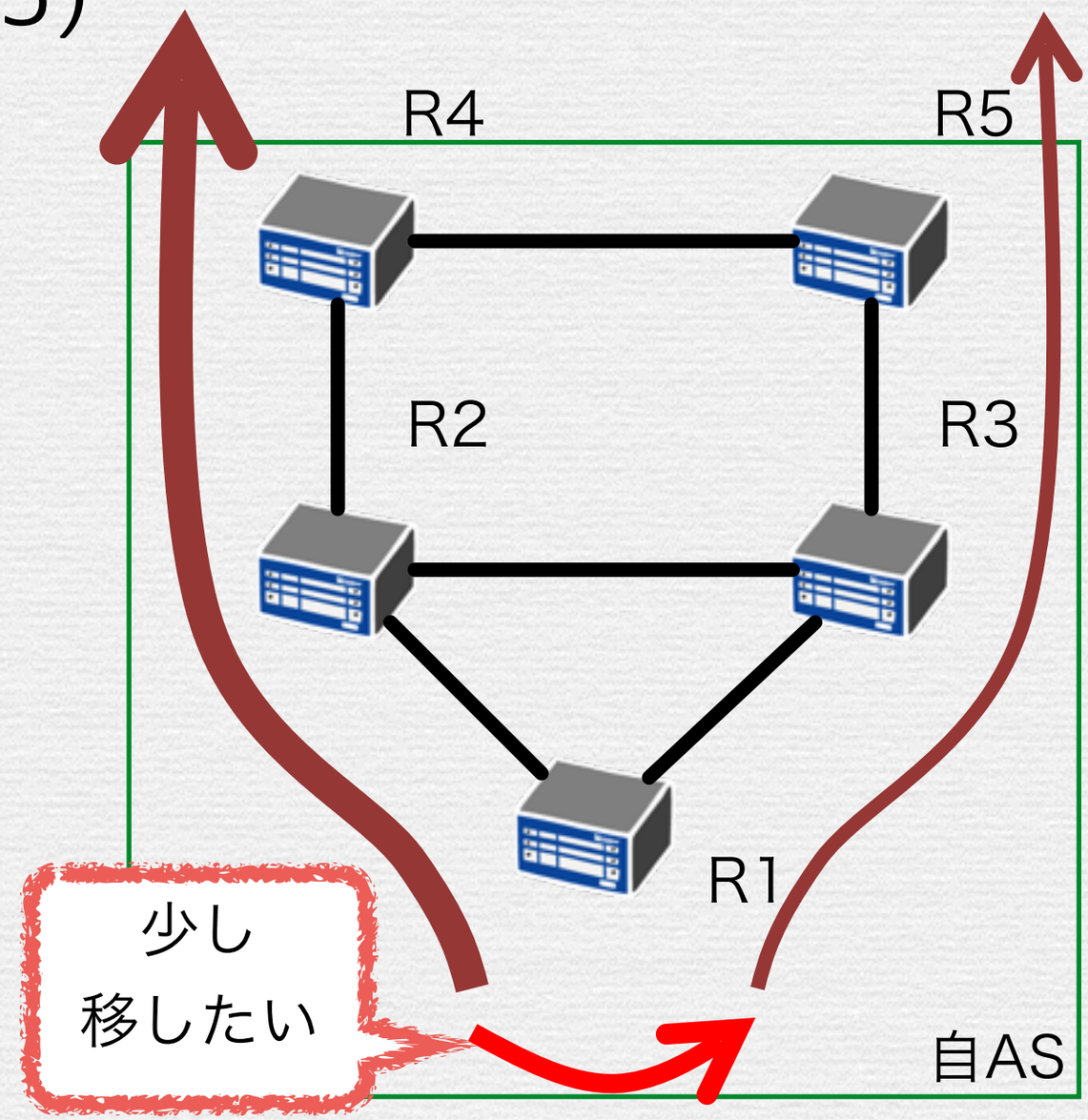


△ 物理的な変更を伴うアイデア

- R3-R4(R2-R5 も）にlinkを追加
  - R1-R3-R4 / R1-R2-R4 でECMPを効かす
    - R1➔R2➔R4 trafficの半分がR1➔R3➔R4 に移る
- 収容を変える（R4 ➔ R5）

△ その他

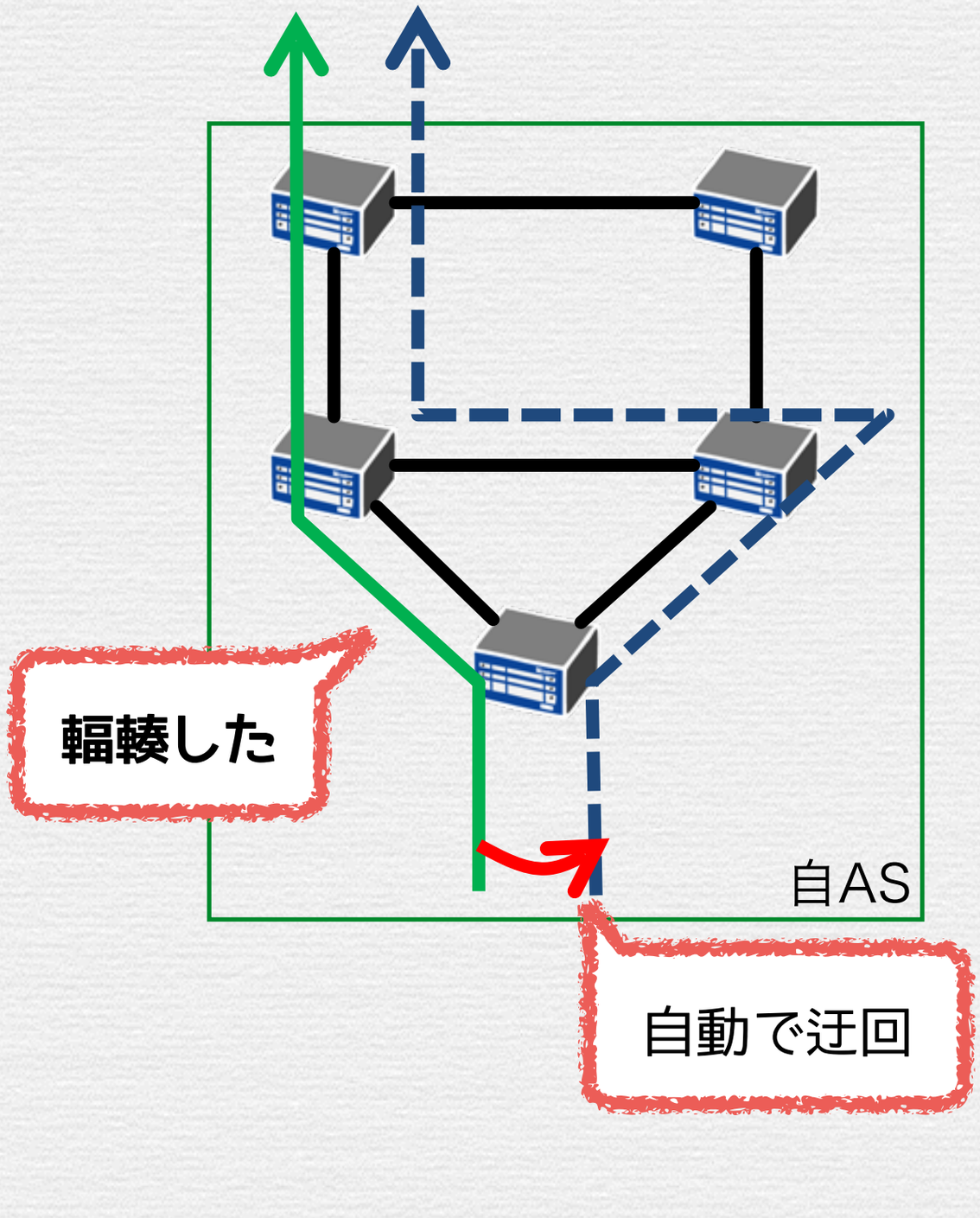
- R4, R5が同じIXに接続しているなら
  - iBGP のnext-hop self をやめる
    - R1➔R2➔R4,
      R1➔R3➔R5がバランス



# MPLS-TE

- 自AS網内のtraffic balanceを自動で制御する技術
  - 空き帯域を自動で探し，そこへrouting
  - QoSも可能
- MPLS(Multi-Protocol Label Switching) + RSVP(ReSerVation Protocol)
- IP Packetにlabelをつけてカプセル化
  - 仮想的なトンネル(LSP: Label Switching Path)をつくる

trafficのend-pointは変わらず，rerouteする



# なんだか便利そうですよね?

JANOG33 には,

**“MPLSトラフィックエンジニアリング チュートリアル”**

もありました!

http://www.janog.gr.jp/meeting/janog33/tutorial/mpls.html

# MPLS-TE

- 利点
  - 手動TEからの開放
  - 帯域設計がラク
    - 最悪rerouteしてくれる
- 欠点
  - overlay model
    - label overhead
  - LSP sizeを設定するため，日々LSP毎のtraffic量をカウントする必要がある
    - automatic bandwidth に期待

# その他

# peering 判断の基本

transitコストを抑えることが主な目的

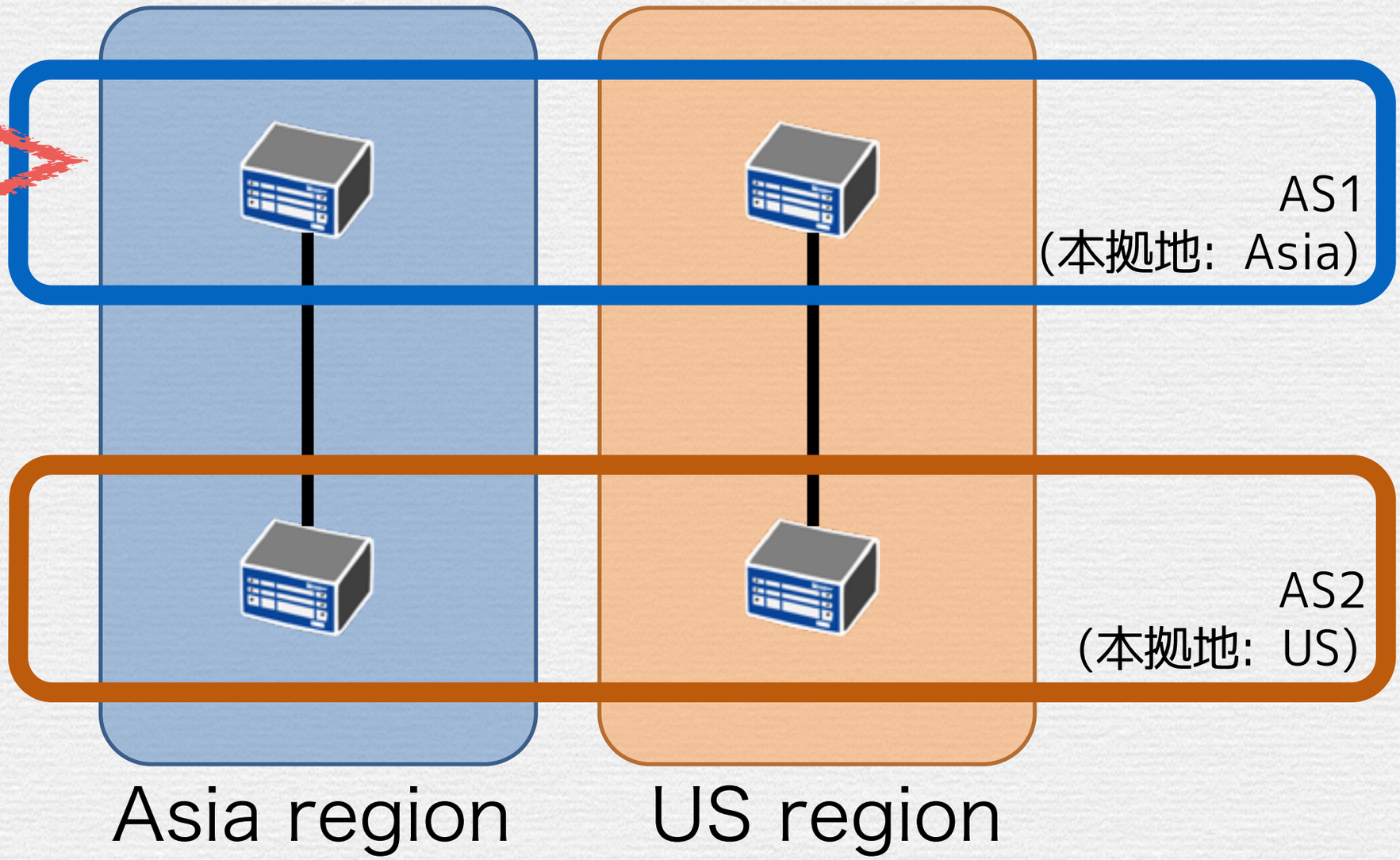
- peerをするASホルダー同士の力関係にもよるが
  - peerすることでコストメリットがある
  - お互いのビジネスを侵害しないなど，両者のpeering policyを満たして初めてpeeringを行う

- “コストメリットがある” ことの目安として ”trafficが???Mbps 出ること” という条件がある場合や，(続く)

・“お互いのビジネスを侵害しない” 条件として複数リージョン / 複数POPでの接続を条件とするISPもいる

自分の経路を，自分のビジネス拠点の近くで渡したくない
(peering partnerはその経路でビジネスができる可能性がある)

“networkの規模が同じくらい” という目安にもなる



・AS ホルダー同士の力関係により
  ・paid peer
  ・“ある地域でtransitを買えば，別の地域でpeerできる”
    というバーター
のような形態もある

・www.peeringdb.com にある程度まとめられている．”General Policy” が “Open” なASホルダーはpeeringに応じてくれる可能性大
  ・web or mysqlで一覧が取得できる

```
$ mysql -r -hpeeringdb.com -upeeringdb -ppeeringdb Peering
mysql> SELECT asn, name, aka, policy_locations, policy_ratio FROM peerParticipants
WHERE policy_general IN ('Open') ORDER BY asn;
```

# peering 判断の基本

- ICPにとって，ISPとのpeeringにより ”ビジネスが侵害” される可能性は低いため，単純に ”コストメリットがあるか?” がpeering判断の大きな評価軸になる場合が多い

- 一方，ICPは他のICPとpeeringするメリットはある?
  - アプリ/ビジネスプラットフォームの連携
  - 他社APIの利用

などは増えてくる見込み?

# 生の声はこちらで

**“Peering in 2014” (1/23 16:00)**

http://www.janog.gr.jp/meeting/janog33/program/peering.html

# その他，ルーティングにとって重要なコンポーネント

# transit providerの filterを制御する

- どこかのInternet Routing Registry(IRR) に登録する必要がある
  - 日本でよく使われているもの
  - JPIRR (jpirr.nic.ad.jp)
    - JPNIC 会員のみ
  - RADB (whois.radb.net)
    - 有償 ($500/y)
  - NTTCOM (rr.ntt.net)
    - ntt.net ユーザーのみ

| | | このIRR の情報を持っているか? | | |
|---|---|---|---|---|
| | | JPIRR | RADB | NTTC |
| このIRRが | JPIRR | ○ | ○ | × |
| | RADB | ○ | ○ | ○ |
| | NTTCO | ○ | ○ | ○ |

お互いにmirrorしているため，どこか1箇所に登録すればよい

# IRR への登録

http://www.nic.ad.jp/doc/jpnic-01077.html
に丁寧な解説がある

- いくつかのobjectを登録する
  - Maintainer
  - Aut-num
  - Route
  - AS-Set

- 登録したAS-Set objectをtransit providerに伝える
- www.peeringdb.com にも登録しておく

# peering db

“細かいことはpeering dbを見てください”

ですむので便利

- www.peeringdb.com
- 情報閲覧はguestアカウントでOK
- 自ASの情報を登録するには
  ユーザー登録 ➔データベース運営者の承認が必要
  - mail addressのdomainを見られるので，ASとの関連が明らかなmail addressを使う

# peering dbには，いろいろ記載できる

- ASの基本情報
  - 名前
  - ASN
  - 種別(ISP / ICP など)
  - traffic level
  - looking glass / route server URL
  - ipv4 / multicast / ipv6
- **peering policy**
  - open / selective / restrictive / No
  - 複数拠点でのpeerを条件にするか?
  - in / outのtraffic比率を条件にするか?
- 連絡先
- 接続IX (public peer 用)
  - 名前
  - 帯域
- POP (private peer 用)
  - DC名
  - Interface 種別

# 経路奉行

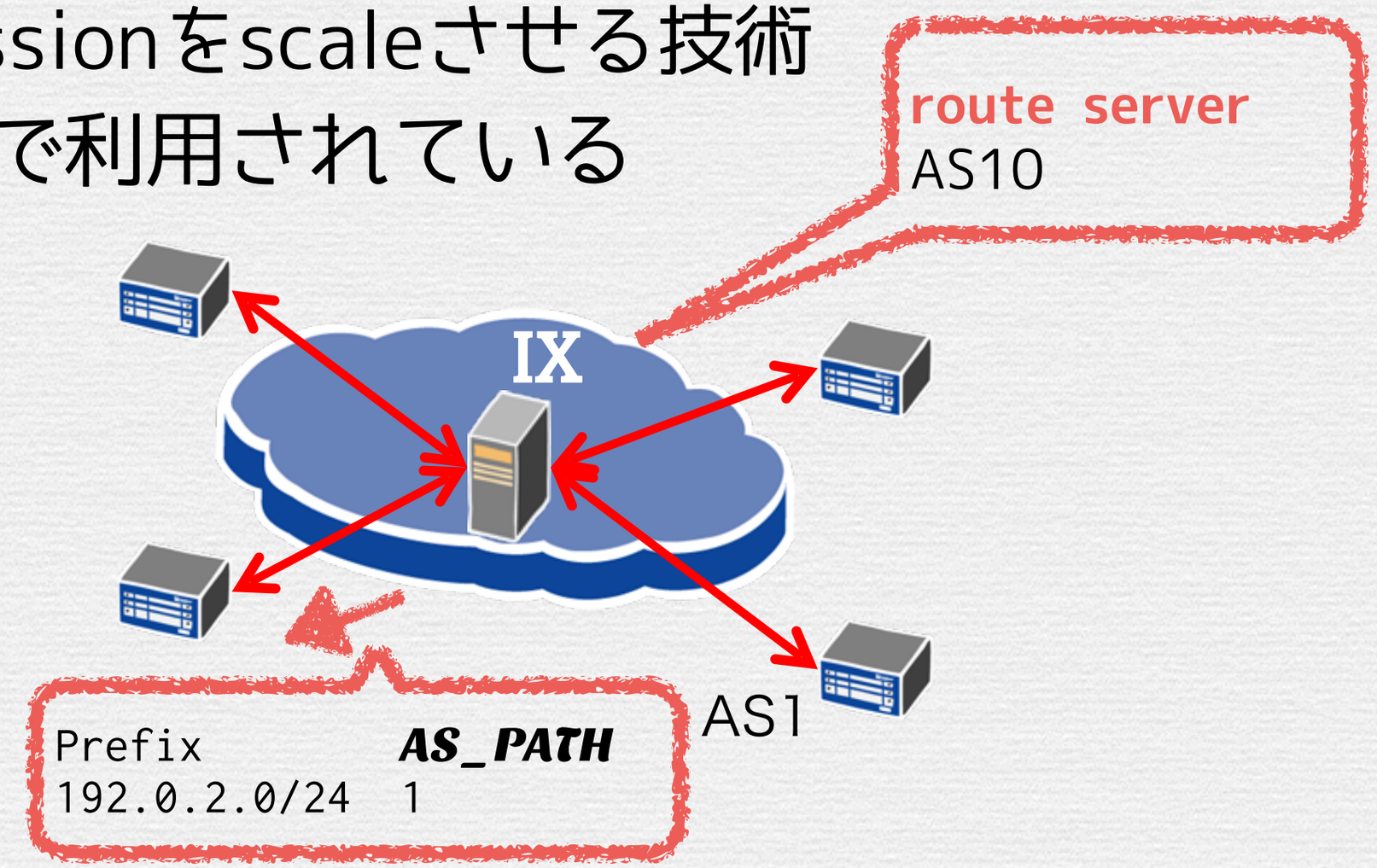

- JPIRRに登録するといいことがある
  - BGP route hijackingを検知 / 通知してくれる
    - http://www.nic.ad.jp/ja/ip/irr/jpirr_exp.html
  - ISAlarm, BGPMON (*) みたいなもの

```
route:          202.12.30.0/24
descr:          JPNICNET
                Japan Network Information Center
                Kokusai Kogyo Kanda Bldg. 6F
                2-3-4 Uchi-Kanda
                Chiyoda-ku, Tokyo 101-0047
                JAPAN
                X-Keiro: okadams@nic.ad.jp      <--追加記述
                X-Keiro: okadams-noc@nic.ad.jp  <--複数あて先に通知する場合記述
origin:         AS2515
admin-c:        SN3603JP
tech-c:         YK11438JP
tech-c:         MO5920JP
notify:         system@nic.ad.jp
mnt-by:         MAINT-AS2515
changed:        apnic-ftp@nic.ad.jp 20080116
source:         JPIRR
```

(*) ISAlarm:
http://www.ripe.net/is/alarms
BGPMON:
http://www.bgpmon.net/

# route server

- eBGP sessionをscaleさせる技術
  - 主にIXで利用されている




- 通常のpeeringの手法(bilateral)とroute server経由のpeeringの手法(multilateral)で，RIBの中身がほぼ同じ
  - **remote AS(この例ではAS10)はASPATHに載らない**
- **eBGP session数を減らすことで運用をラクに**
  - open policyのASホルダー向き

# まとめ

# 今日の目的

- **BGPの設計を決める際に考えないといけないこと** をつかむ

- “BGP sessionの先にいるAS (顧客 / peering partner / transit 提供者) にも個別のrouting policyがある” という環境で, **どうやって自分の思うようにtraffic control するか?** を理解する

# BGPのPolicyや設計を決めるために考えるべきこと

# NetworkやRoutingを考える際には常にPolicyを意識

# 常に意識できるよう
# Routing Policyは
# なるべくシンプルに

とは言っても，
BGPはASをまたぐので
いつもPolicy/設計通りに
実装できるとは限らない

# Policy/設計に反することは誰が見ても「反している」のが分かるように

技術的負債を返すため，目的を達成したら必ず戻す
その際に周囲に影響なく消せるよう，実装時から工夫することが大事