The Internet Ecosystem and Evolution

Contents

- The Border Gateway Protocol (BGP)
  - architecture, process model
  - BGP messages, types, and attributes
  - BGP best path selection
- Policy routing with BGP
  - valley-free routing by configuring BGP import and export filters
  - BGP communities

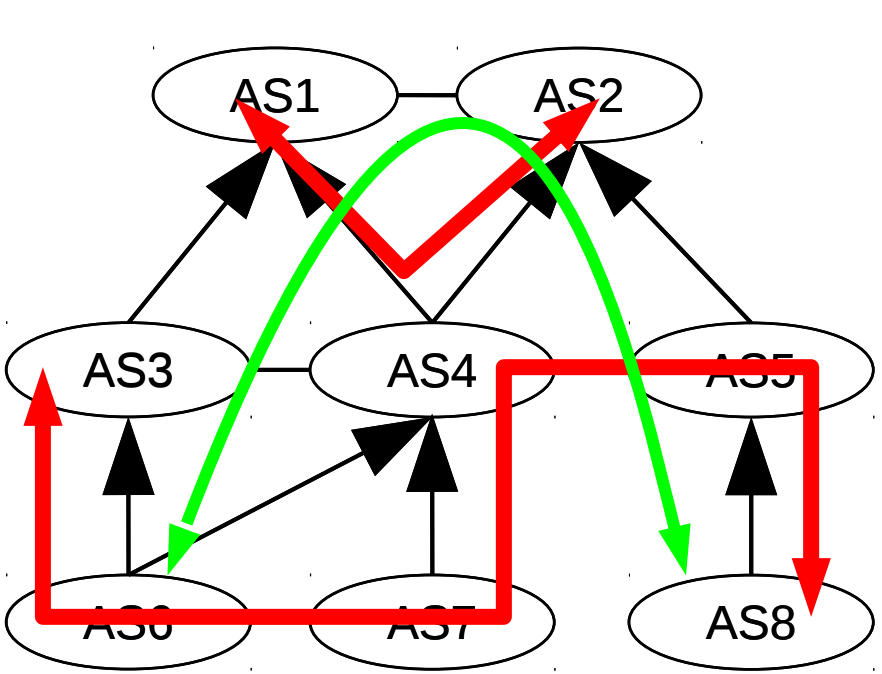
The Border Gateway Protocol

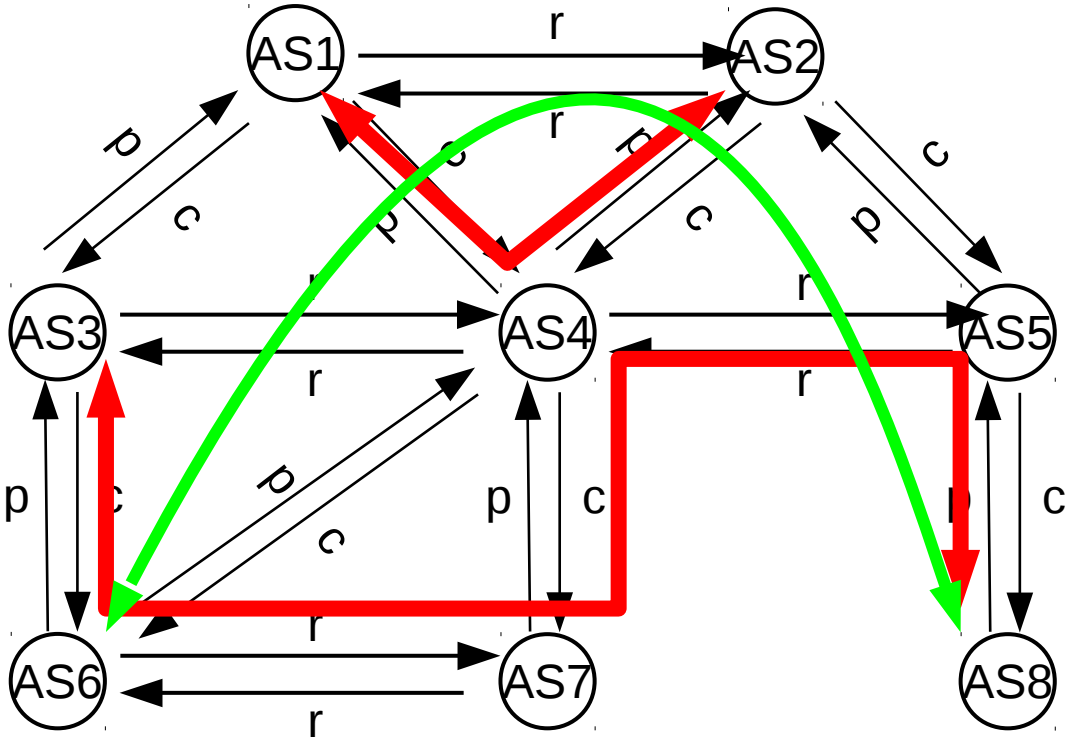
AS-AS business relationships

- **Recall:** two ASes typically establish a transit or a peering relationship between one another
  - **transit:** global Internet access for a fee
  - **peer:** „free” traffic exchange between to ASes and between any of their customers
- Internet traffic follows the cash-flow
- **Valley-free routing:** a model to understand the structure of paths that align with the business interests of ASes (**feasible paths**) and those that violate them (**forbidden paths**)

Valley-free routing

- The AS-level Internet and graph representation





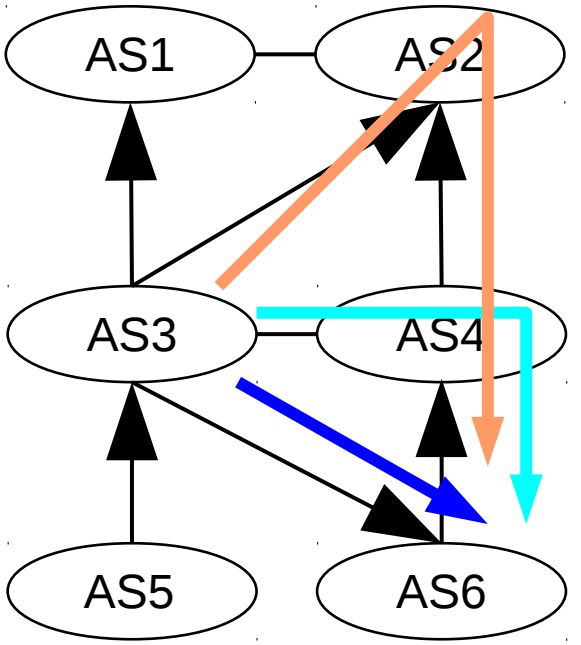
- A path is valley-free if the sequence of labels match the regular expression *p*r?c**

Path preference

- Paths via a customer AS are preferred over peer and provider paths: free of charge
- **Prefer customer rule:**

$$P_c < P_r < P_p$$

- Short paths imply small delay
- **Shortest AS path policy:** if multiple, equally-preferred paths are available, the one crossing fewer ASes is chosen



Inter-domain routing

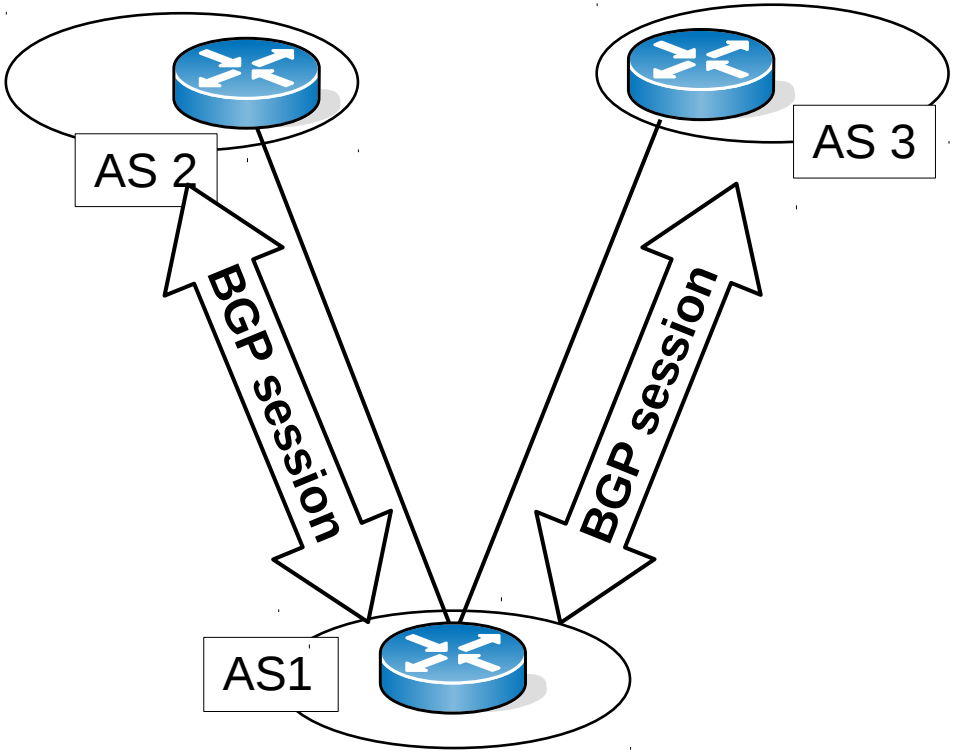
- **The graph representation is only a model:** Internet routers do not (and cannot) use it (the AS-level graph is not known, let alone business relationships)
- Internet paths are shaped by the autonomous stakeholders' interests: **distributed routing**
- ASes sovereignly pursue their business interests by picking and establishing specific forwarding paths via a **path-vector protocol**

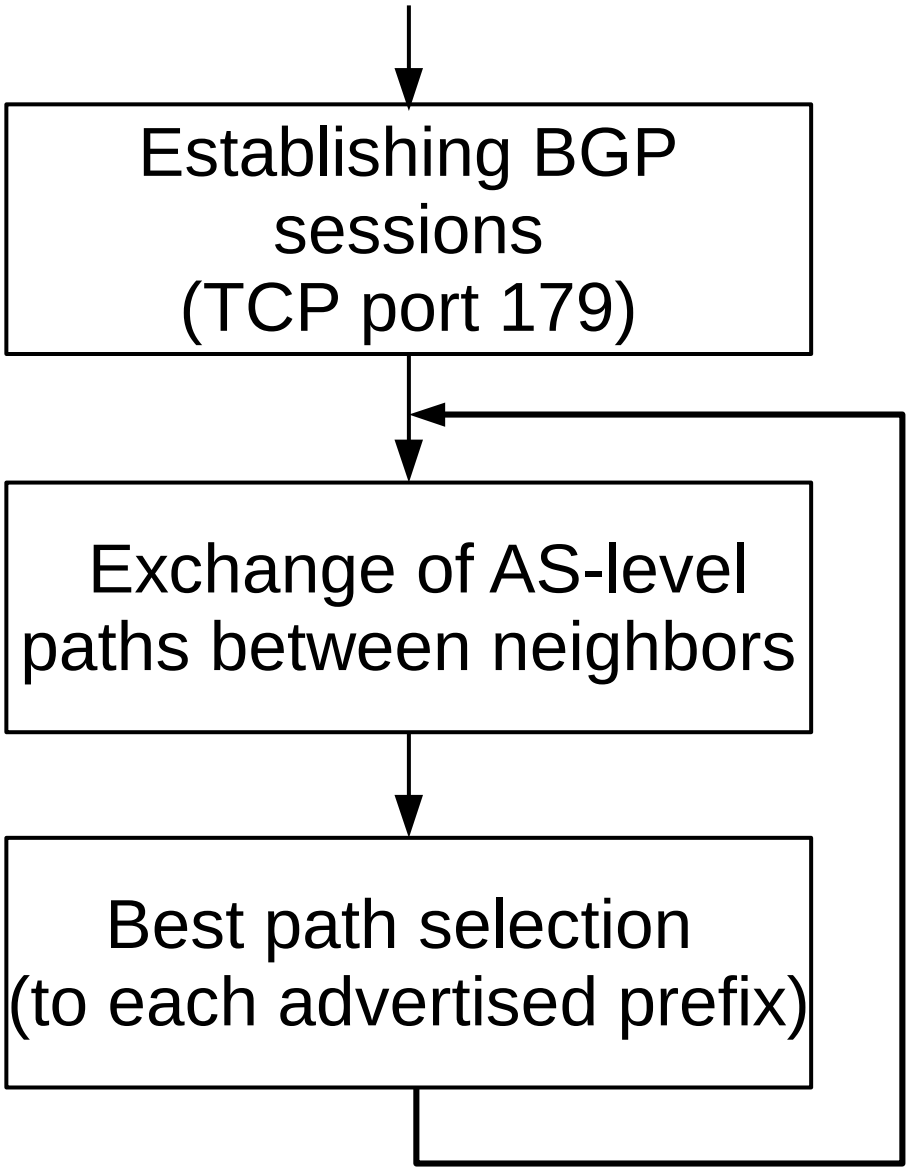
BGP: a path-vector EGP

- **Border Gateway Protocol version 4: BGP**
  - a path-vector policy routing protocol
  - de facto standard Internet EGP
  - the protocol is simple, configuration is hard
- Result of 15 years of evolution:
  - 1989 : BGP-1 [RFC 1105]
  - 1995 : BGP-4 [RFC 1771]: CIDR support
  - minimal modification since then

BGP: Process model

- Neighboring routers establish a **BGP session** between each other





BGP: a path-vector EGP

- The destinations in BGP are IP subnet prefixes
  - one prefix → one AS-level path
  - same path is used to each IP address in the destination prefix
- Routing based on **AS-level paths**
  - nodes along the paths are ASes, links are AS–AS interconnections
  - the business relations (transit/peer) are not known to BGP (business secret!)
  - thus the labels (*p*, *c*, *r*) are not distributed
  - only business relations to neighbor ASes is known

BGP: a path-vector EGP

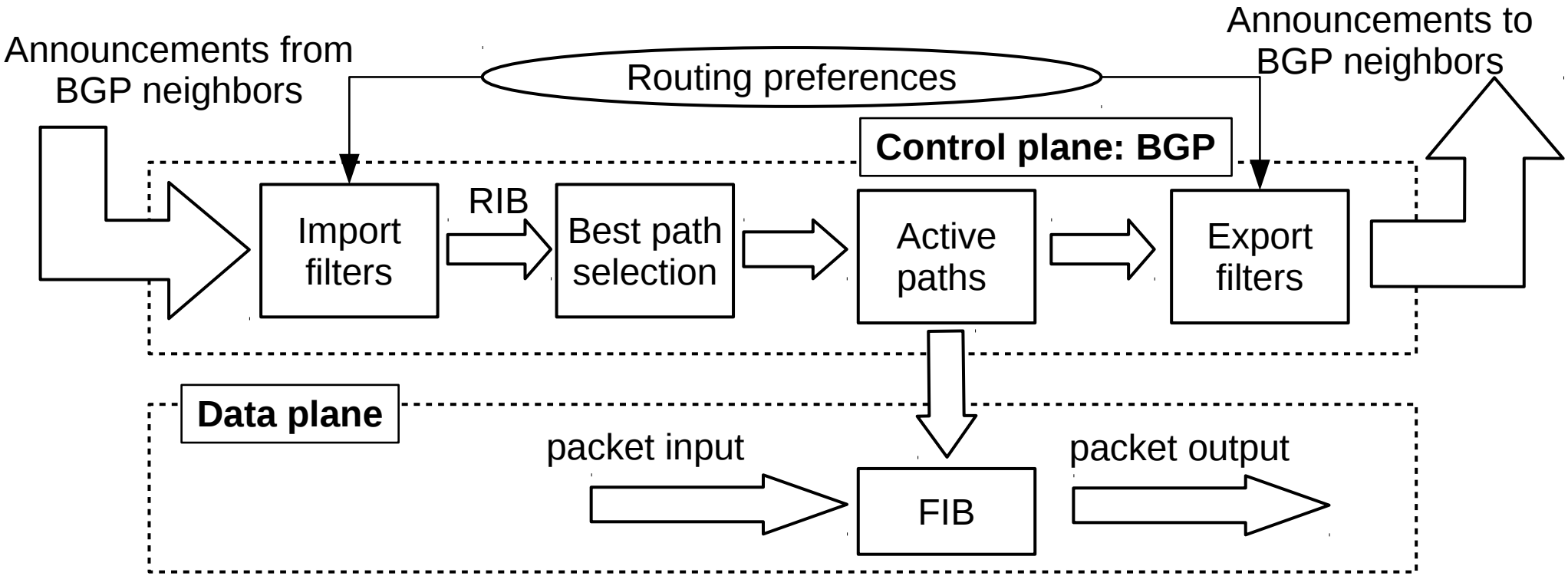
- ASes advertise the best paths towards each prefix to neighboring ASes: **BGP announcement**
  - the sequence of ASes to the destination prefix
  - plus some further attributes
- Paths advertised to neighbor ASes are subjected to **export filters**
  - routers can withdraw/rewrite/suppress paths
- Paths received from neighbors can be subjected to **import filters**
  - accepting a path is not mandatory

BGP: a path-vector EGP

- BGP announcements, after subjected to import filters, go into an AS-path database: **BGP RIB**
- From this database **the best path is selected** to each advertised prefix: **active path**
  - based on the AS's own routing policies
  - e.g., “valley-free routing” + „prefer customer” + “shortest AS path”
  - practically any routing policy can be realized
  - by properly configuring import/export filters

The BGP routing process

- **BGP configuration:** BGP session config + announced prefixes + import/export filters
- Routing can be influenced through the filters



BGP: Messages

- **Open:** initiate a BGP session between two routers
  - two routers participating in a BGP session are usually physically connected (same link)
  - but BGP sessions can also be set up through arbitrary number of intermediate routers
- **Keep Alive:** periodic session refresh
- **Notification:** close a BGP session
- **Update:** send/recv network reachability information
  - **Announce:** new path is avialable to a prefix
  - **Withdraw:** revoke a formerly announced path

BGP: Messages

- **BGP announcement = prefix + attributes**
- Important BGP attributes:

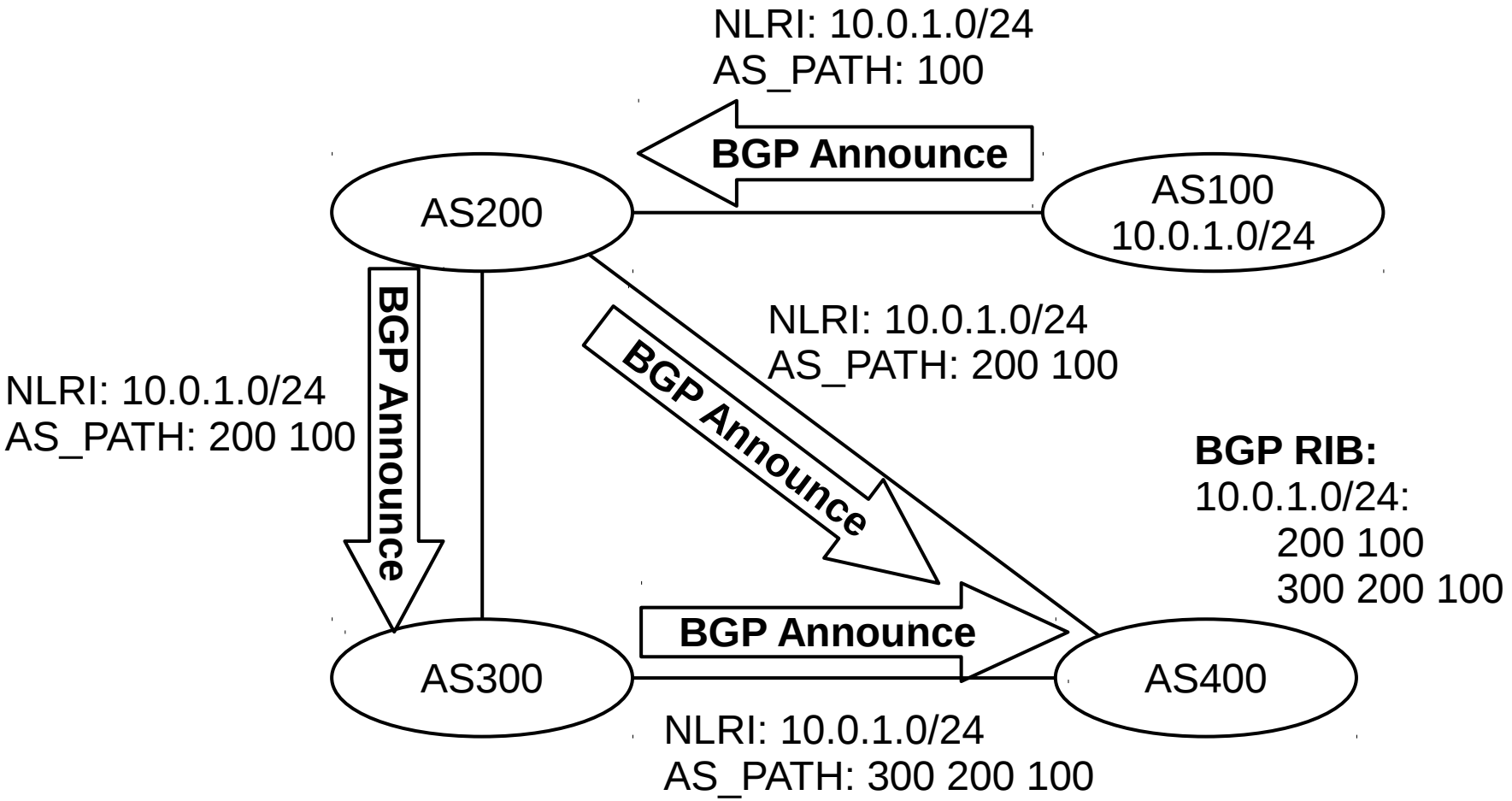
Value	Code	Reference
...		
2	AS_PATH	[RFC1771]
3	NEXT_HOP	[RFC1771]
...		
5	LOCAL_PREF	[RFC1771]
...		
8	COMMUNITY	[RFC1997]
...		
16	EXTENDED COMMUNITIES	[RFC4360]
...		
255	reserved for development	

BGP announcement: NLRI

- **Network Layer Reachability Information (NLRI)**: the prefix whose reachability is communicated/refreshed in the BGP announcement
  - usually just a single IP prefix: e.g., `10.0.1.0/24`
  - but more prefixes can also be specified in a single announcement
  - in this case the attributes in the announcement describe each advertised prefix
  - can also encode an IPv6 address, or multicast address, MPLS label, etc.

BGP attributes: AS_PATH

- AS-path to a prefix: list of AS-numbers to target

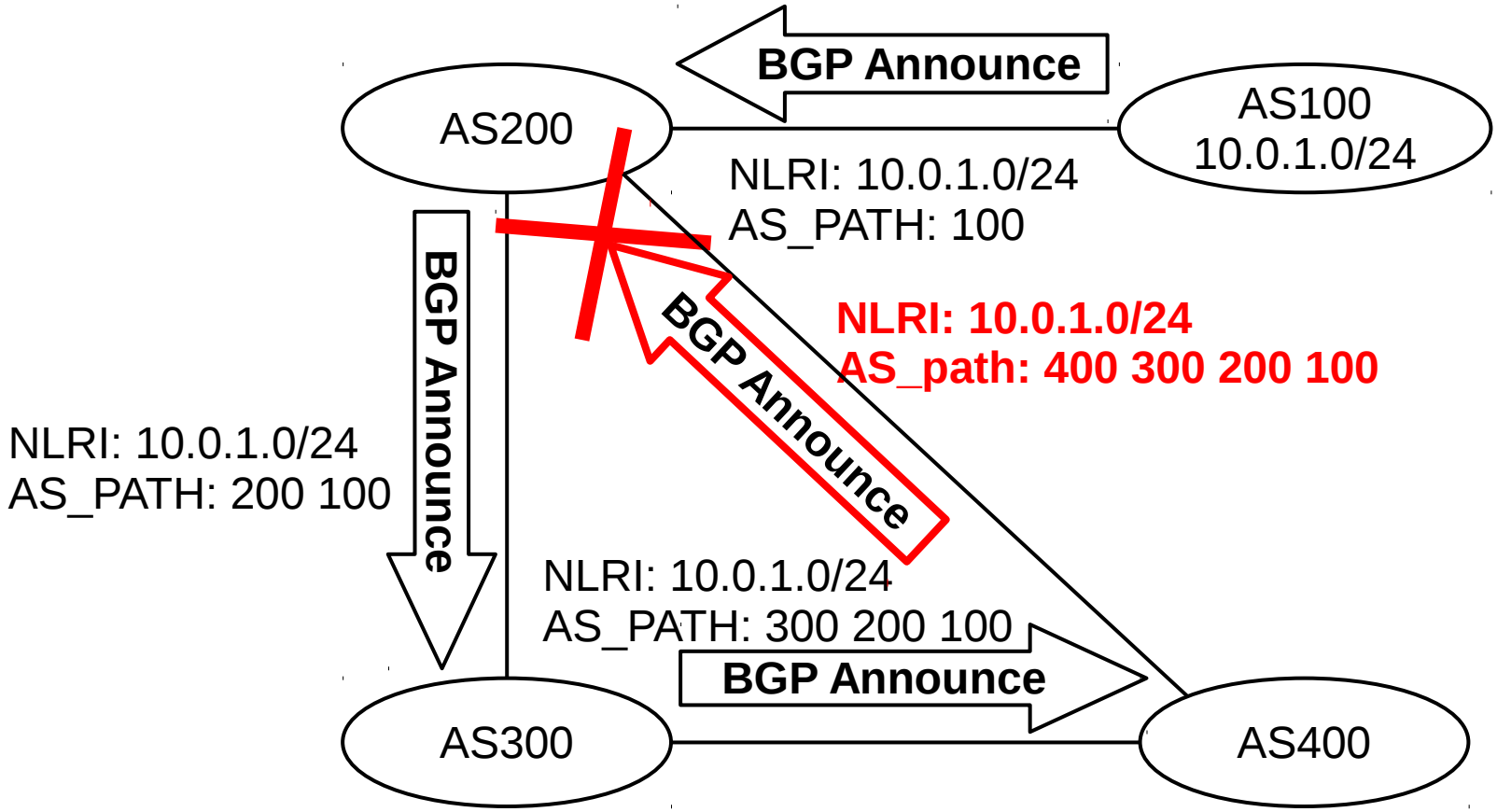


BGP announcements: Rules

1) **Announcing a prefix into BGP:** the AS must testify that it is the valid owner/user of that prefix
   - otherwise, it is a prefix hijacking: an invalid announcement may deflect traffic from the legitimate owner and the CIA may soon knock on your door
   - no one checks this!
2) **Forwarding a BGP announcement:** the forwarder AS guarantees that it will handle traffic from the receiver AS to the prefix, should that AS pick this announcement as the best path
   - preferably without wiretapping/tampering with traffic
   - otherwise, it is a traffic black-hole (recall, CIA!)

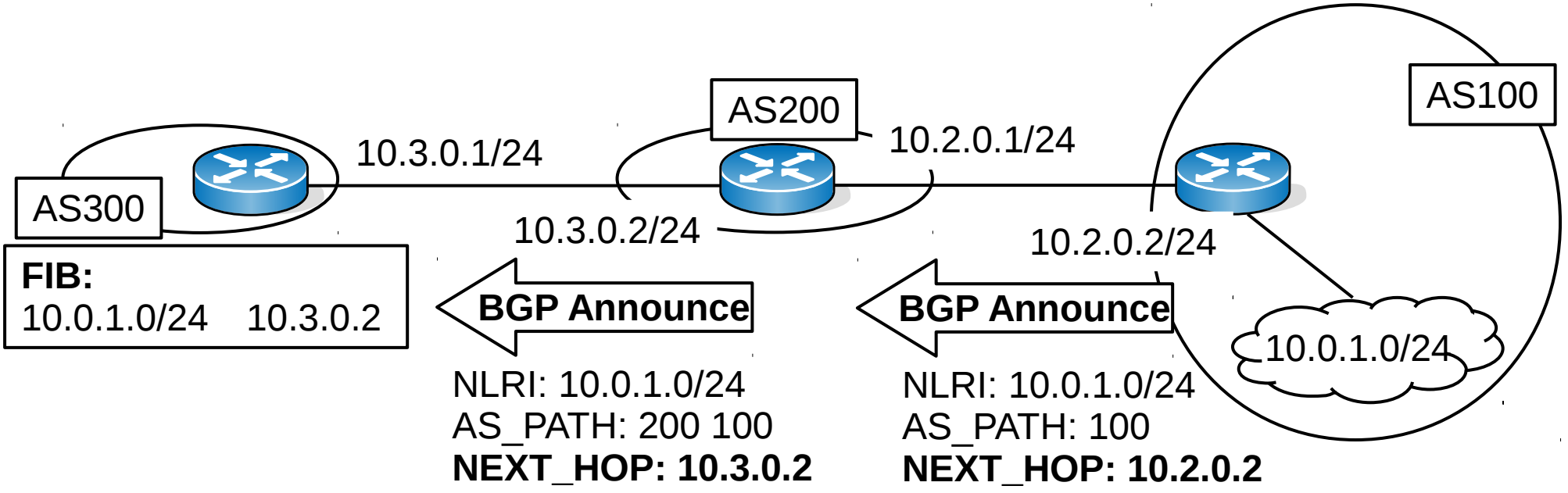
BGP: Loop avoidance

- **Rule:** a BGP router does not accept a BGP announcement that contains its own AS number in the AS_PATH attribute (would induce a loop)



BGP attributes: NEXT_HOP

- Specifies a next-hop IP address for the advertised path
  - this next-hop will be installed into the FIB for the prefix, should the router accept this announcement
  - routers set their own IP address in the NEXT_HOP attribute upon forwarding the announcement



Further BGP attributes

- **LOCAL_PREF (Local Preference)**:
  - one of the most important attributes
  - plays a crucial role in best path selection
  - the higher the LOCAL_PREF the more preferred the AS path
- Used for encoding routing policies into BGP configuration (see later)
- **COMMUNITY/EXTENDED COMMUNITIES:** annotate announcements with opaque “labels”

BGP announcement: Example

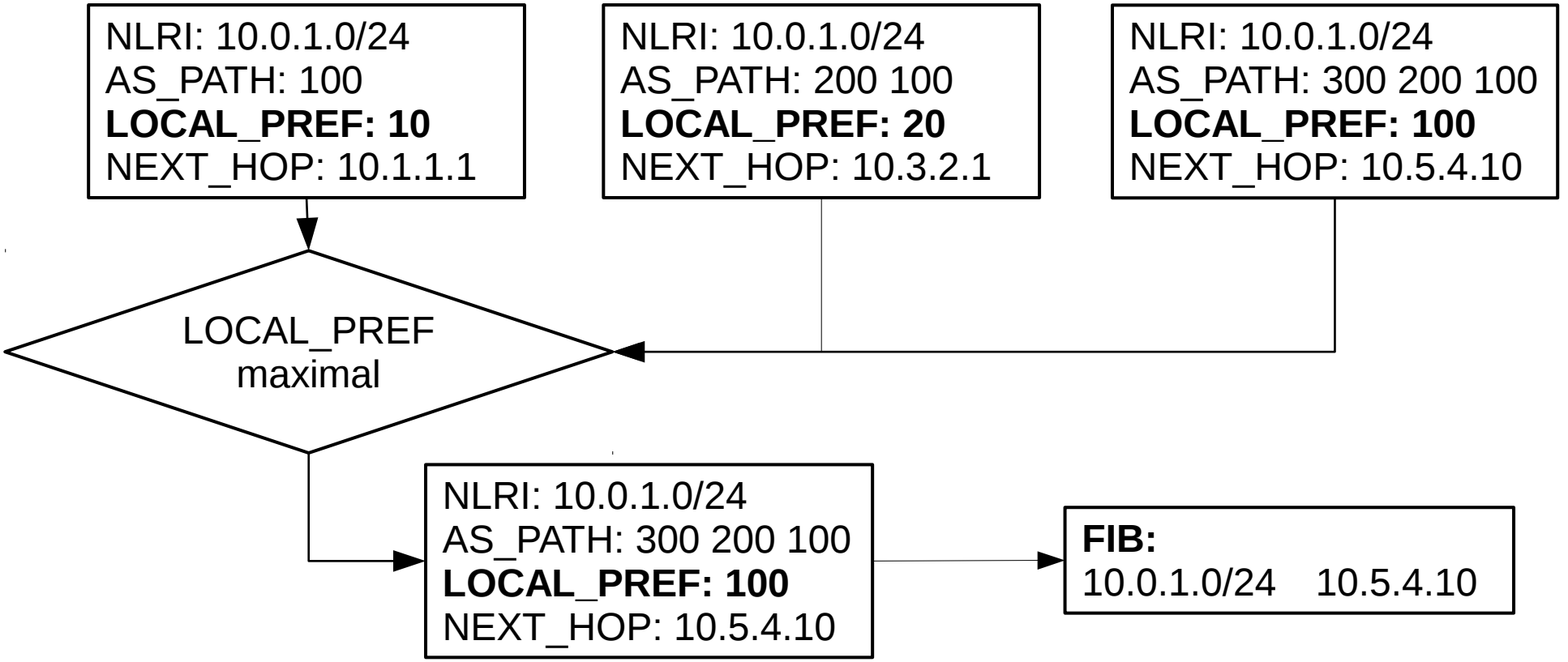
```
Internet Protocol Version 4, Src: … Dst: …
Transmission Control Protocol, Src Port: 58463
(58463), Dst Port: 179 (179), Seq: 84, Ack: 84,
Len: 52
Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 52
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 25
    Path attributes
        Path Attribute - ORIGIN: IGP
        Path Attribute - AS_PATH: 200 100
        Path Attribute - NEXT_HOP: 10.3.0.2
        Path Attribute - MULTI_EXIT_DISC: 0
    Network Layer Reachability Information (NLRI)
        10.0.1.0/24
```

The best path selection mechanism

- A router may receive multiple advertisements with respect to a prefix (one from each neighbor)
- The **best path selection process** decides which one will take effect and go into the FIB:
  - **input:** all announcements for a prefix ← BGP RIB
  - **output:** the active path to the prefix → FIB
  - the decision is made based on processing and comparing the attributes of the announcements
- The **decision process is fixed**, we can only enforce our routing policies indirectly, through **configuring the import/export filters**!

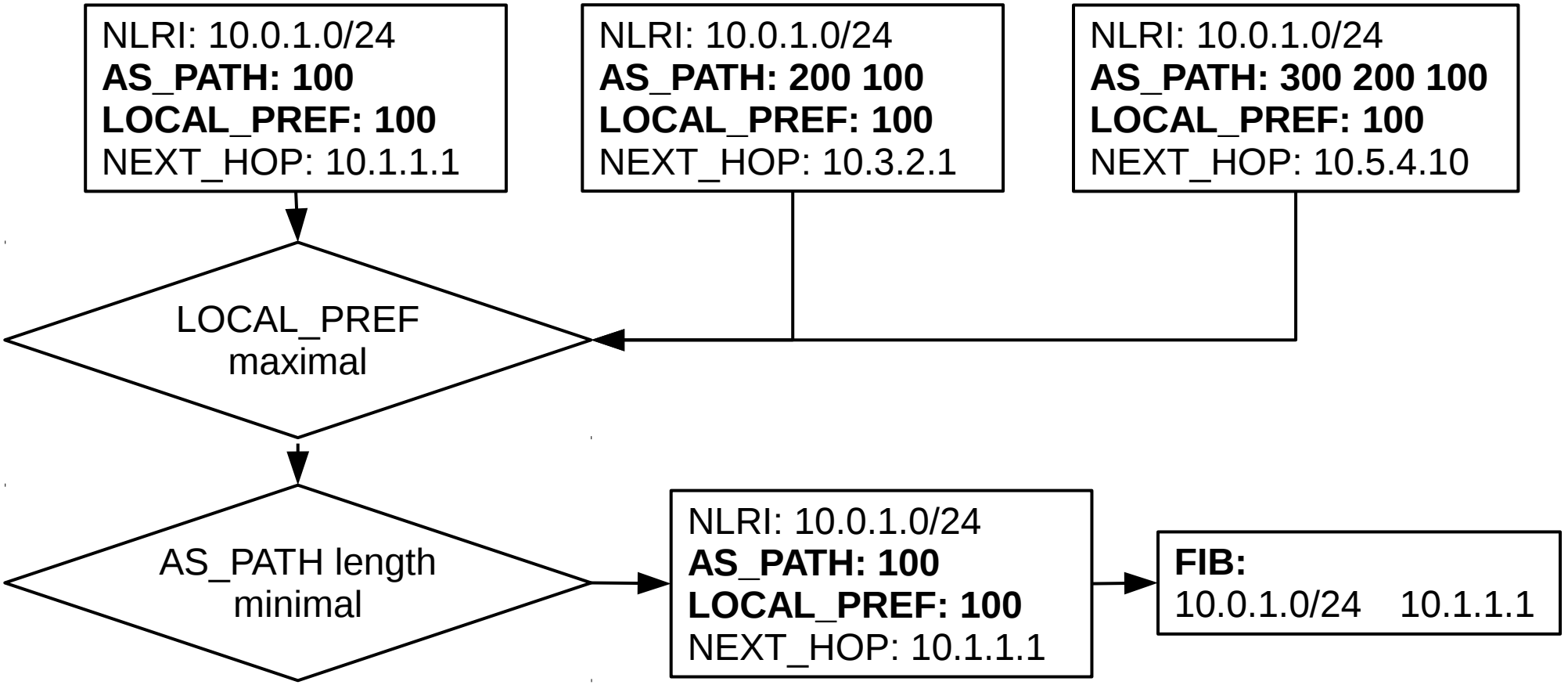
The best path selection mechanism

- Choose the announcement with the **largest LOCAL_PREF attribute**
- Even if the AS path is longer!



The best path selection mechanism

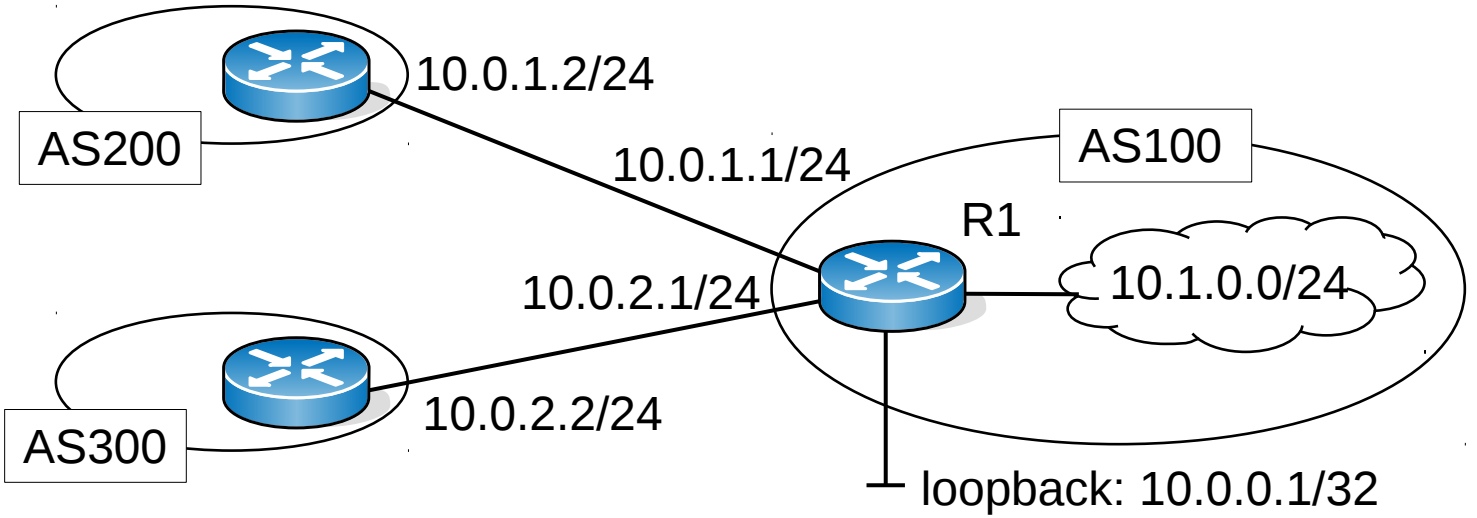
- If multiple announcements have the same local preference: **choose shorter AS path length**
- If still a tie, consider further attributes...



BGP configuration: Basics

- We are given the topology in the figure, with IP addresses already configured for the interfaces
- Fire up BGP on router `R1` and start BGP configuration: `router bgp <AS-number>`

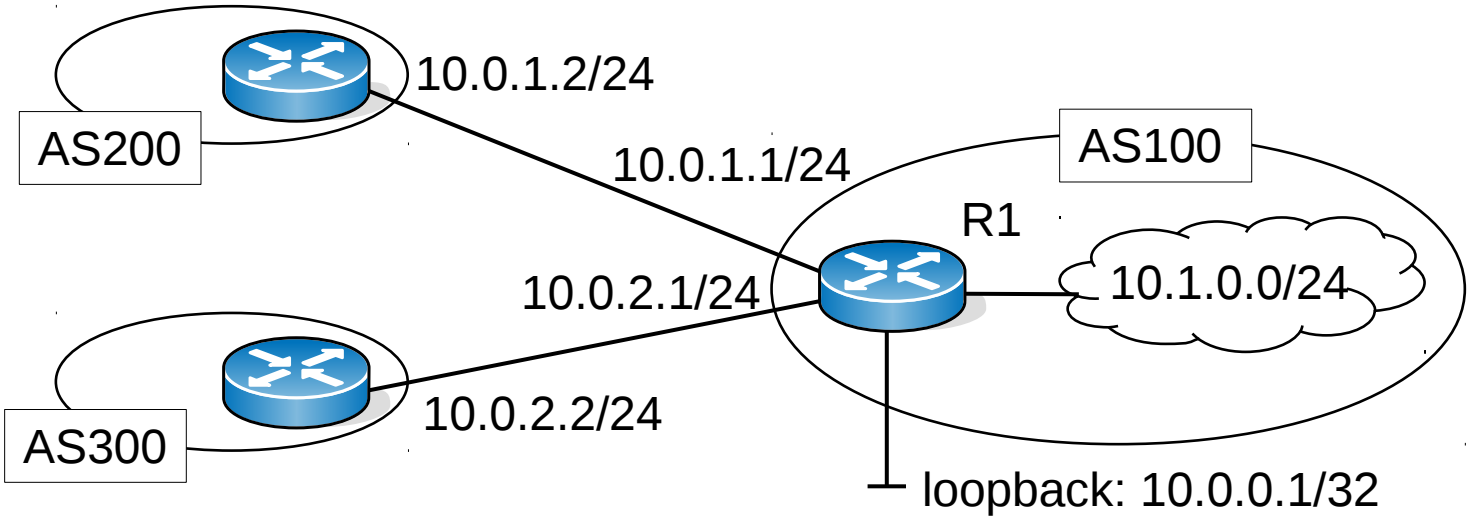
  `router bgp 100`



BGP configuration: Basics

- It is worth assigning a unique identifier to each BGP router (otherwise, there is no well-defined IP address we can use to reach the router)
- Usually, the **router-id** is a routable IP address configured to the `loopback` interface

```
bgp router-id 10.0.0.1
```

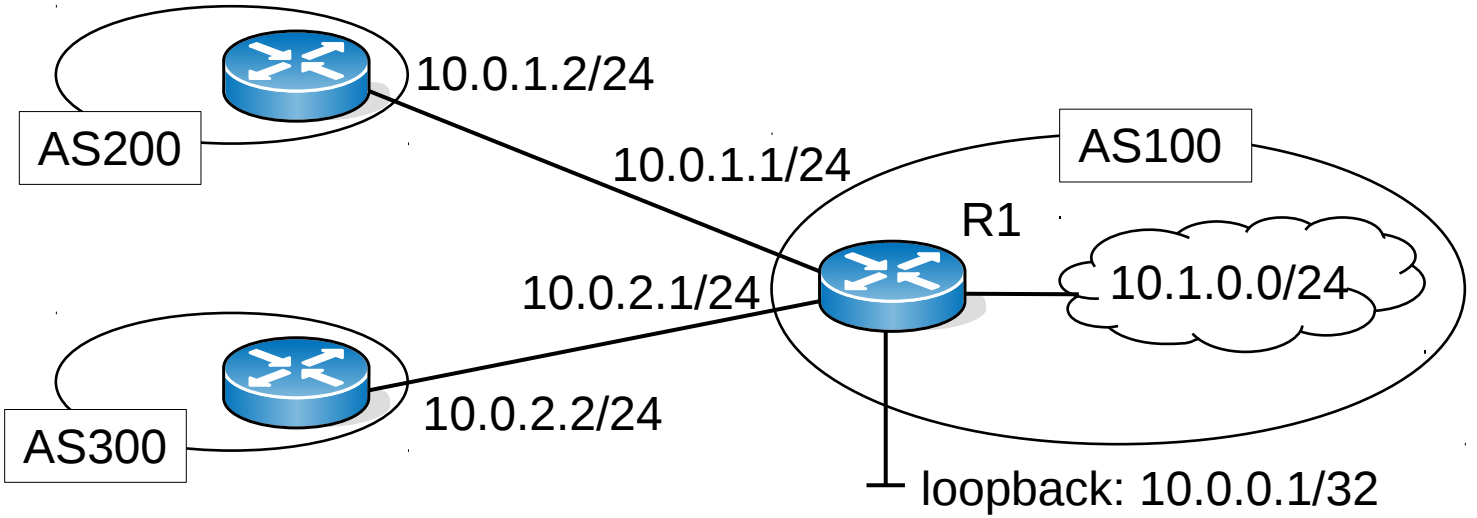


BGP configuration: Basics

- `R1` announces the prefix `10.1.0.0/24`:

  `network 10.1.0.0/24`

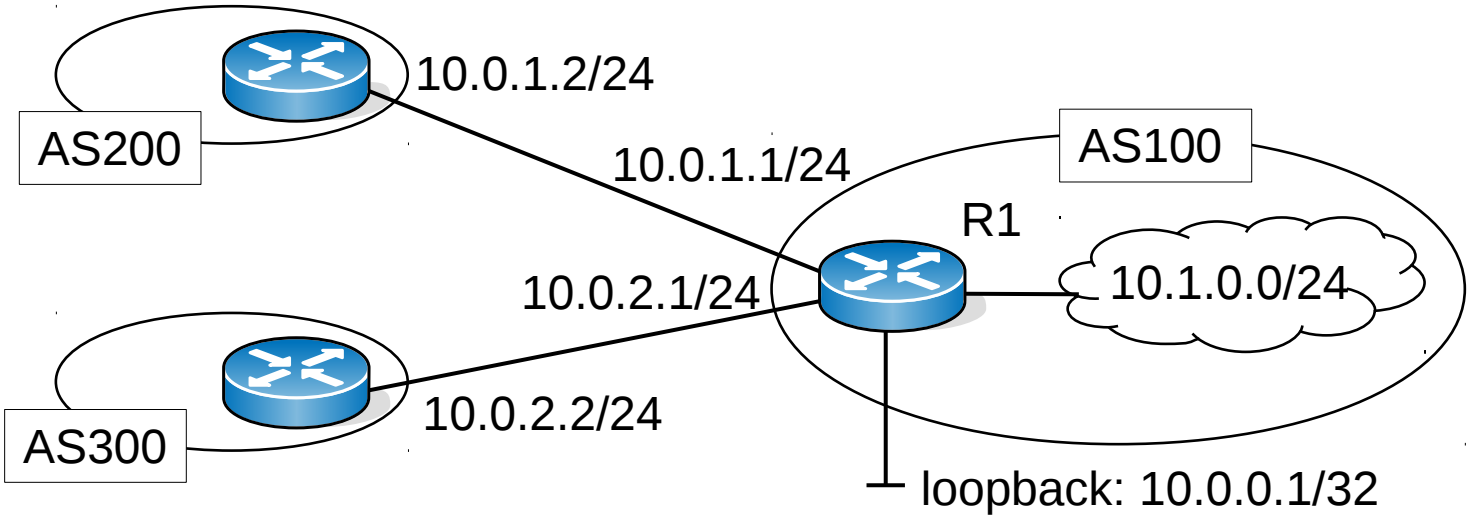
- Recall, if an AS announces a prefix, it must be a valid owner/user of that prefix!



BGP configuration: Basics

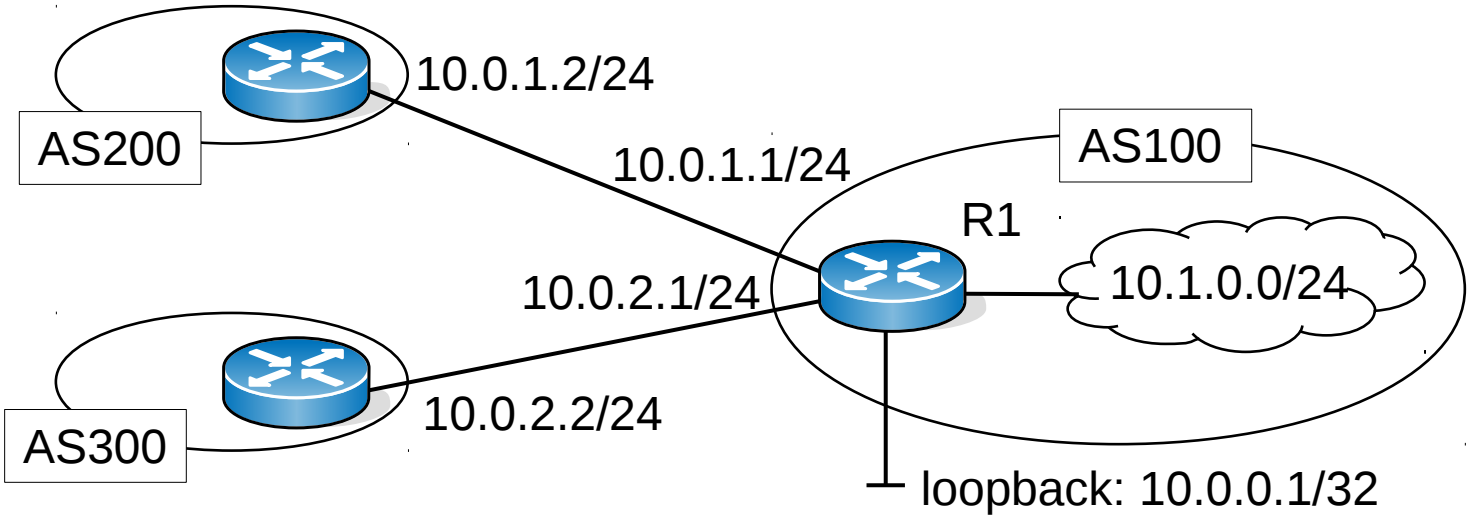
- Set up the BGP sessions: specify IP addresses and the remote AS number for the “other side”:

```
neighbor 10.0.1.2 remote-as 200
neighbor 10.0.2.2 remote-as 300
```



BGP configuration: Basics

```
router bgp 100
  bgp router-id 10.0.0.1
  network 10.1.0.0/24
  neighbor 10.0.1.2 remote-as 200
  neighbor 10.0.2.2 remote-as 300
```



Policy routing with BGP

Policy routing with BGP

How to make BGP choose the forwarding paths that best align with our business interests?

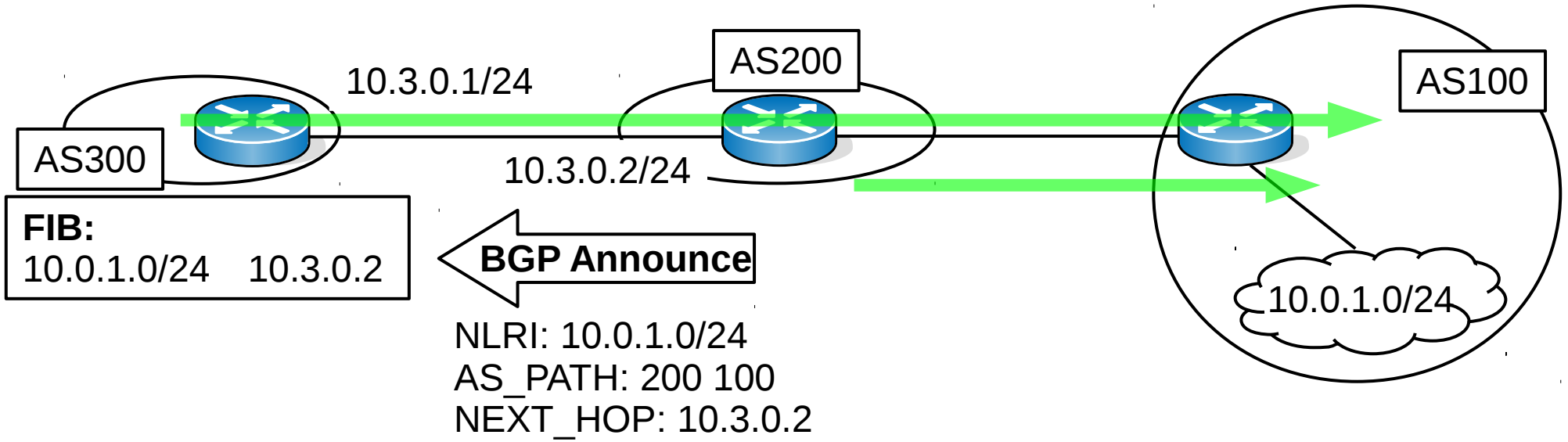
- Since valley-free routing is not hard-coded into the path selection mechanism of BGP, it will not select valley-free paths without explicit configuration
- But if BGP diverges from the optimal routing policies, that would cause enormous profit loss
- We must achieve that BGP block any path between, say, any two of our provider ASes...

Valley-free with BGP

- First, we ask how to configure pure valley-free routing into BGP, later we shall extend this setup to finer-grained policies (prefer-customer, etc.)
- We exploit the fact that an AS can only pick a path that was actually announced to it in the first place
- If we do not announce a path, the neighbor AS will not be aware of it and will not use it for routing traffic (“security through obscurity”)
- **BGP configuration for valley-free routing:** configure import/export filters so that only feasible paths are announced to neighbors

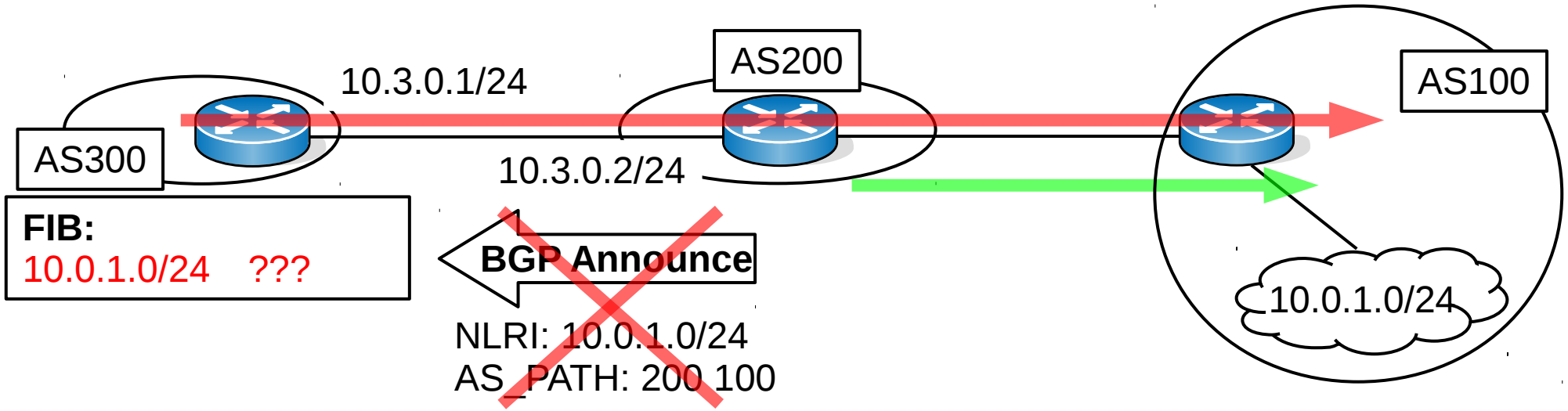
BGP announcement = transit

- An AS announces a path to another = declares that it is willing to route traffic along that path
- **Example:** `AS200` announces to `AS300` the path to prefix `10.0.1.0/24` = it will forward traffic from `AS300` on path `AS300→ AS200 → AS100`



Withholding path = path is blocked

- If the AS withholds an announcement, then neighbors will not be able to use that path
- **Example:** `AS200` does not further-announce prefix `10.0.1.0/24` to `AS300` = path `AS300` → `AS200` → `AS100` is blocked by `AS200`

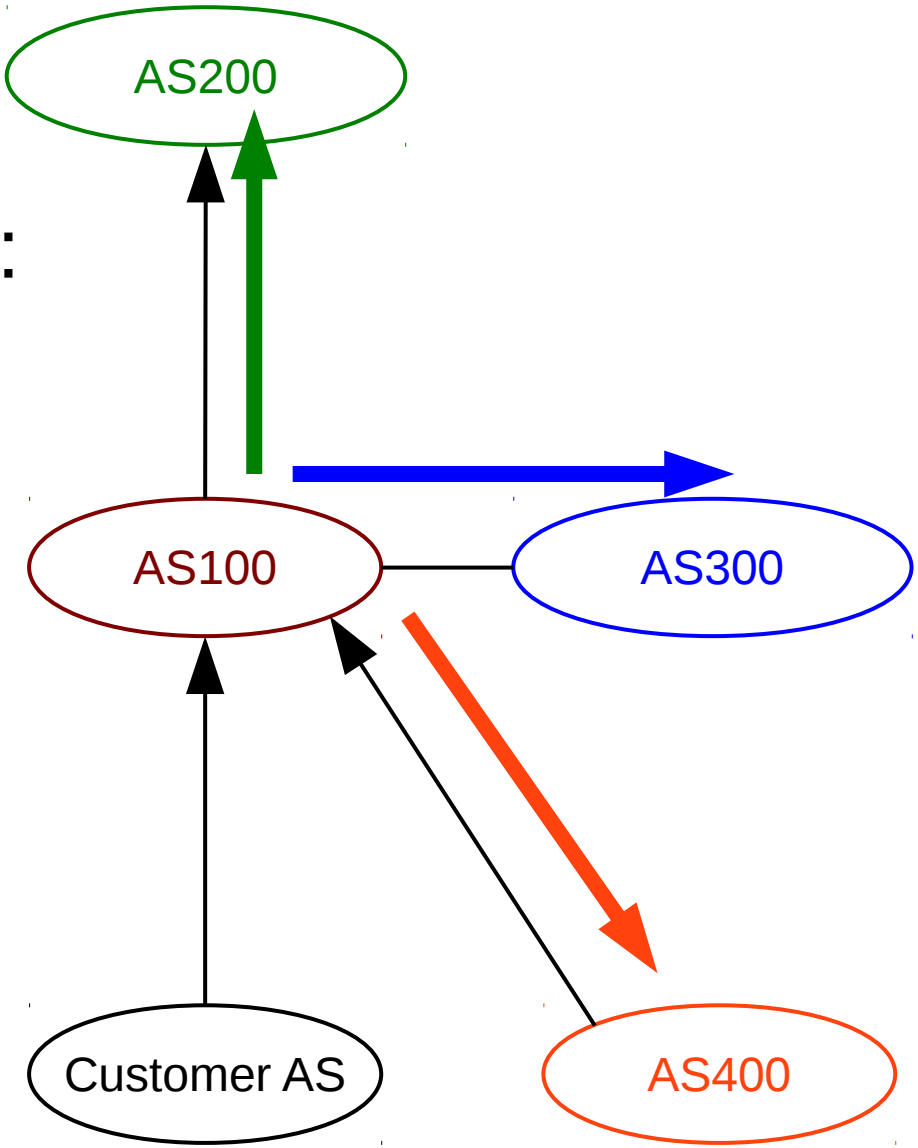


BGP filters

- The task is now to configure BGP filters so that the neighbor will receive valley-free paths only
- Paths must be filtered based on
  - which type of AS–AS link they were received on (from customer, from peer, or from provider), and
  - through which type of AS–AS link we are about to announce it to a neighbor (towards a customer, peer, or provider)

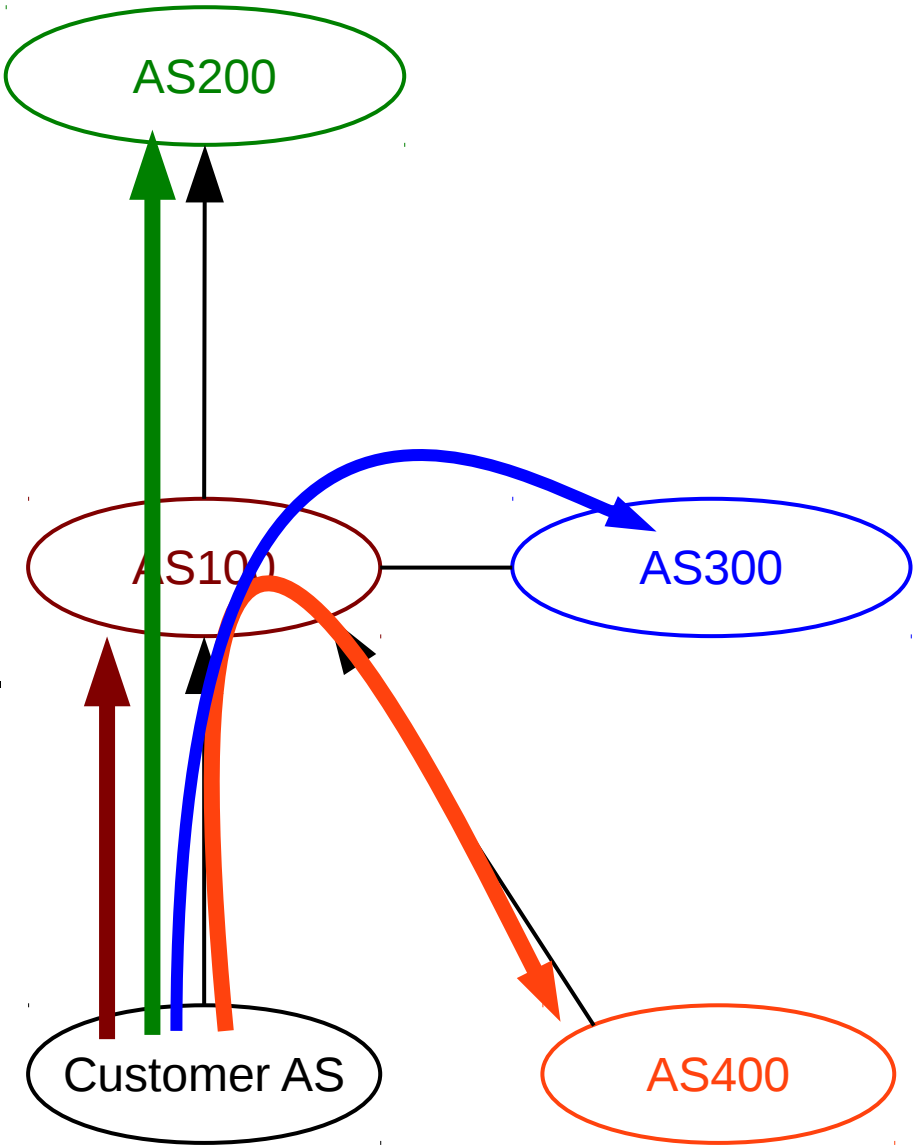
BGP filters to/from customers

- Paths of `AS100` can be divided into 4 categories based on the first link type:
  - path via a customer
  - path via a peer
  - path via a provider
  - path to own prefix
- Which ones to export to `Customer AS`?



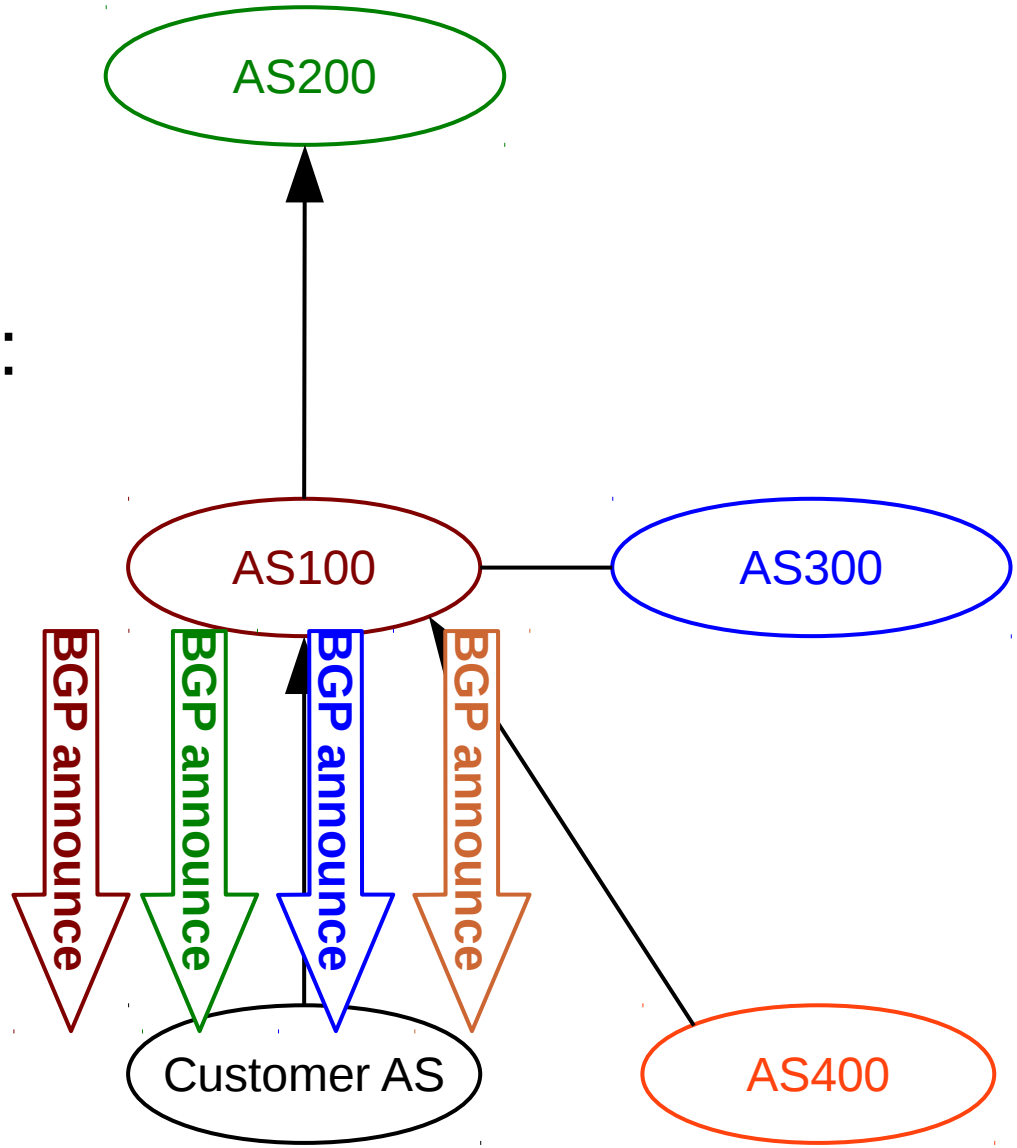
BGP filters to/from customers

- Can we obtain forbidden path if we prepend the paths of `AS100` with the `Customer AS`→`AS100` link?
- **Observation:** a path beginning with a customer-to-provider link is always feasible



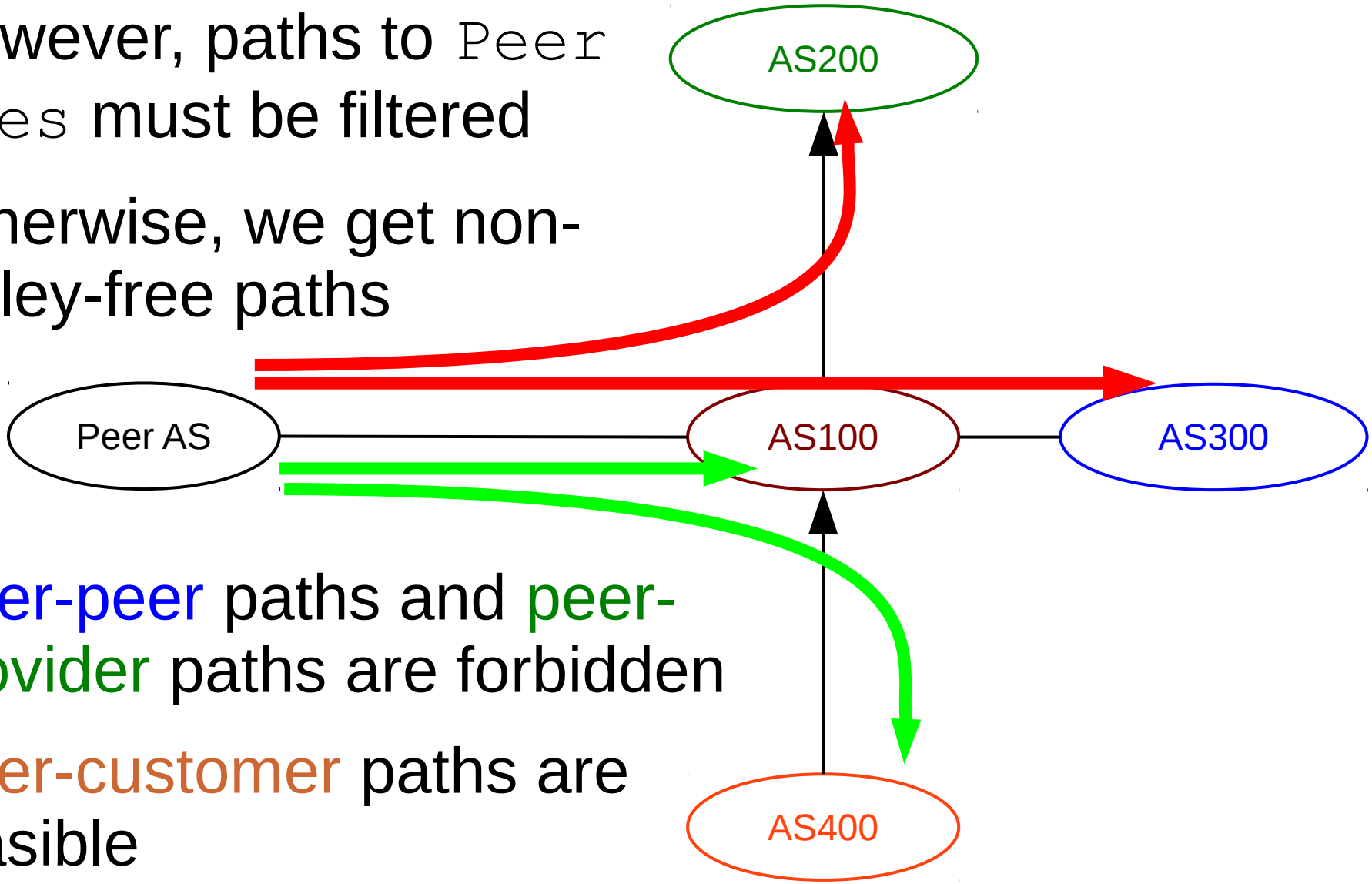
BGP filters to/from customers

- **Rule: export all paths to customer ASes**
- Exporting as many paths to customers as possible: customer traffic brings profit by transit fees
- What to export to a customer: every path from providers, peers, and other customers
- `AS100` announces its own prefixes as well



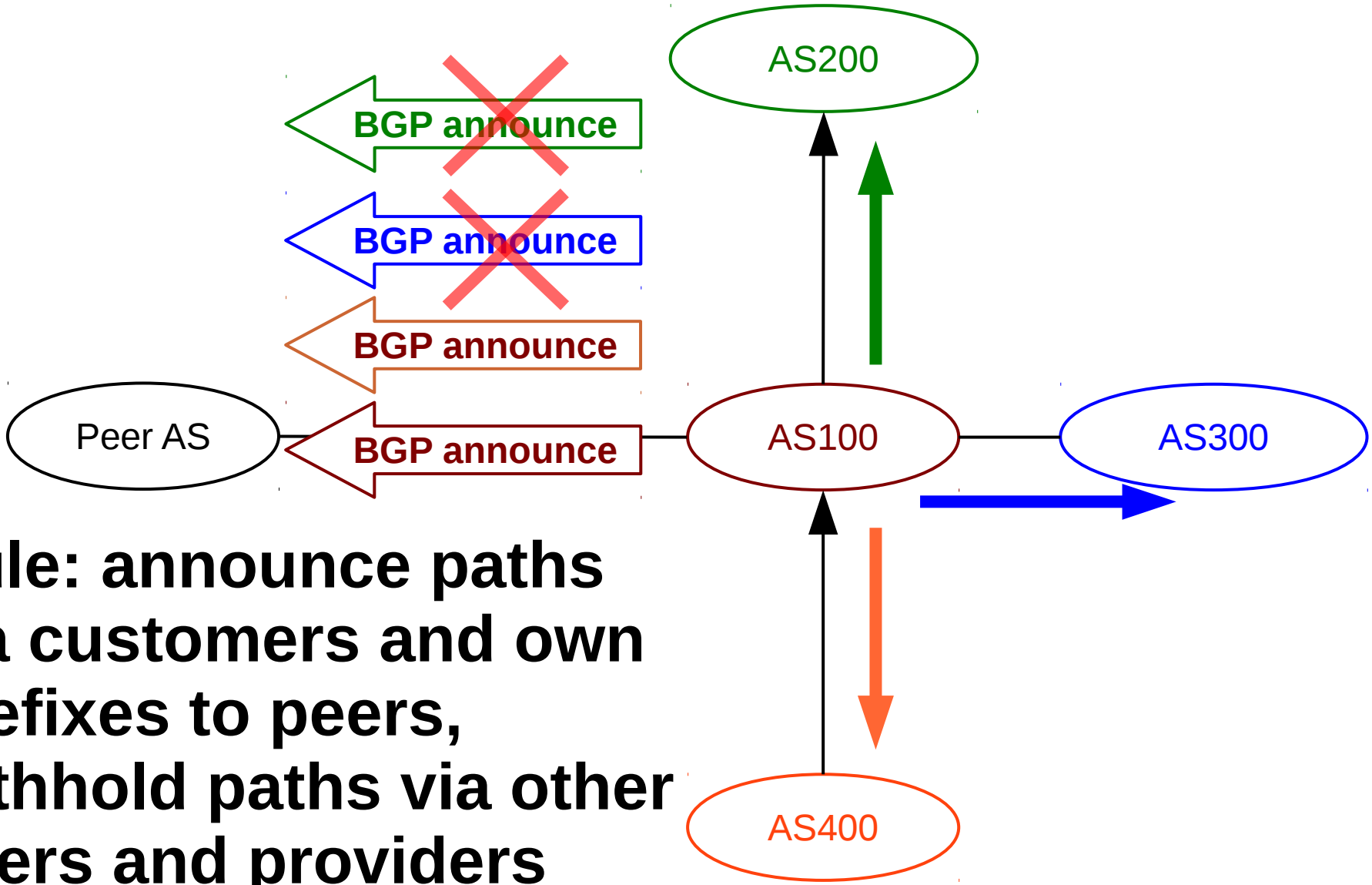
BGP filters to/from peers

- However, paths to `Peer ASes` must be filtered
- Otherwise, we get non-valley-free paths



- Peer-peer paths and peer-provider paths are forbidden
- Peer-customer paths are feasible

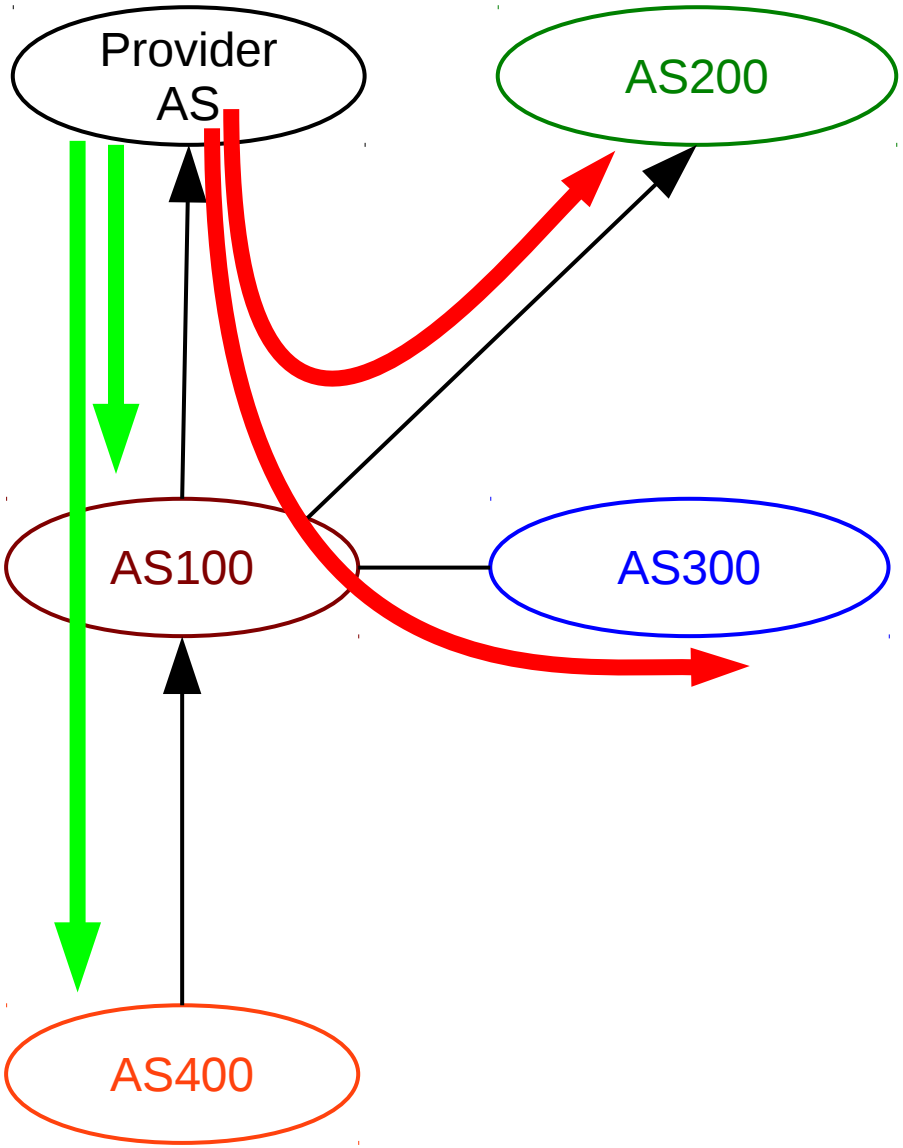
BGP filters to/from peers



- **Rule: announce paths via customers and own prefixes to peers, withhold paths via other peers and providers**

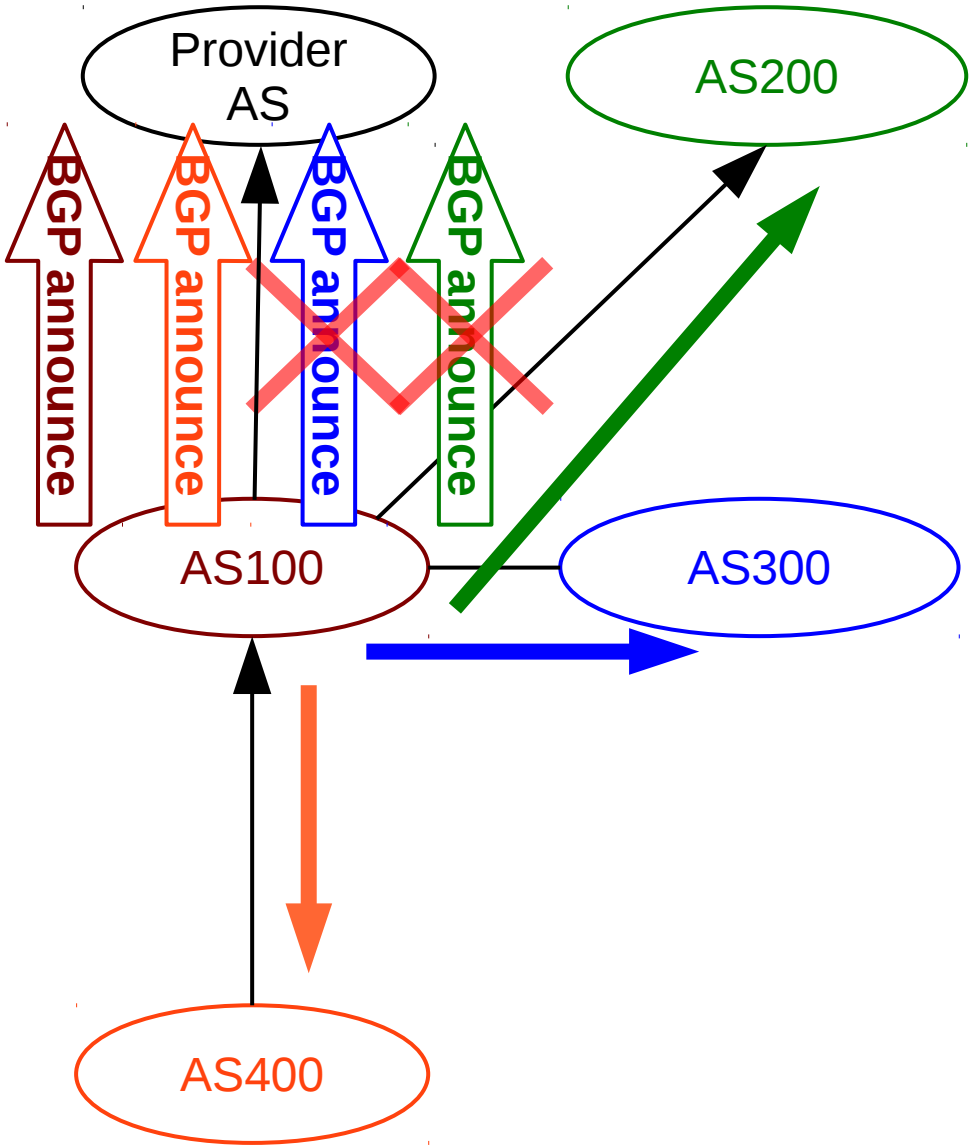
BGP filters to/from providers

- `AS100` filters provider paths received as well
- Blocks all traffic between any two providers, and provider → peer paths are also forbidden
- But provider → customer and provider → `AS100` paths are feasible

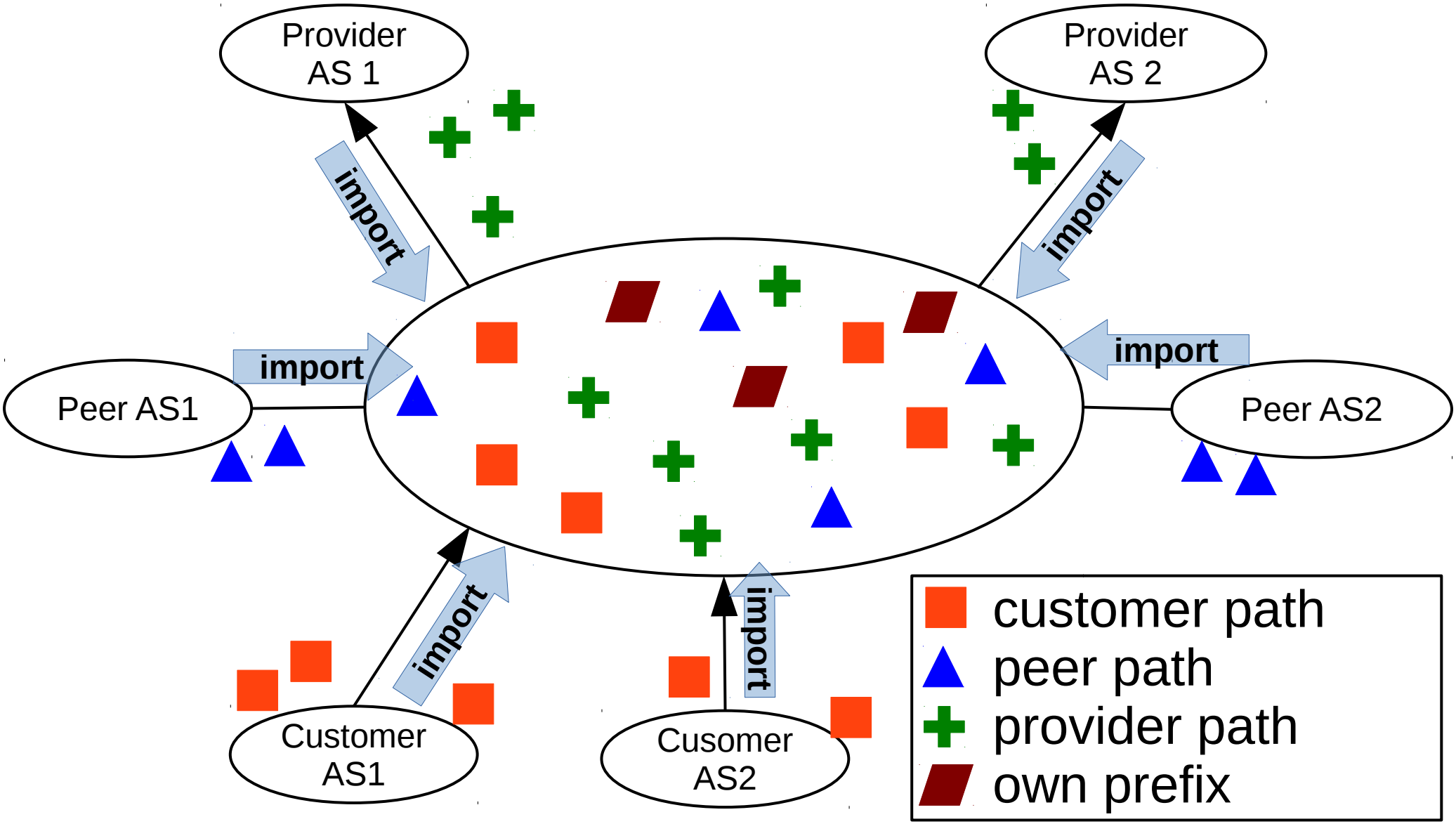


BGP filters to/from providers

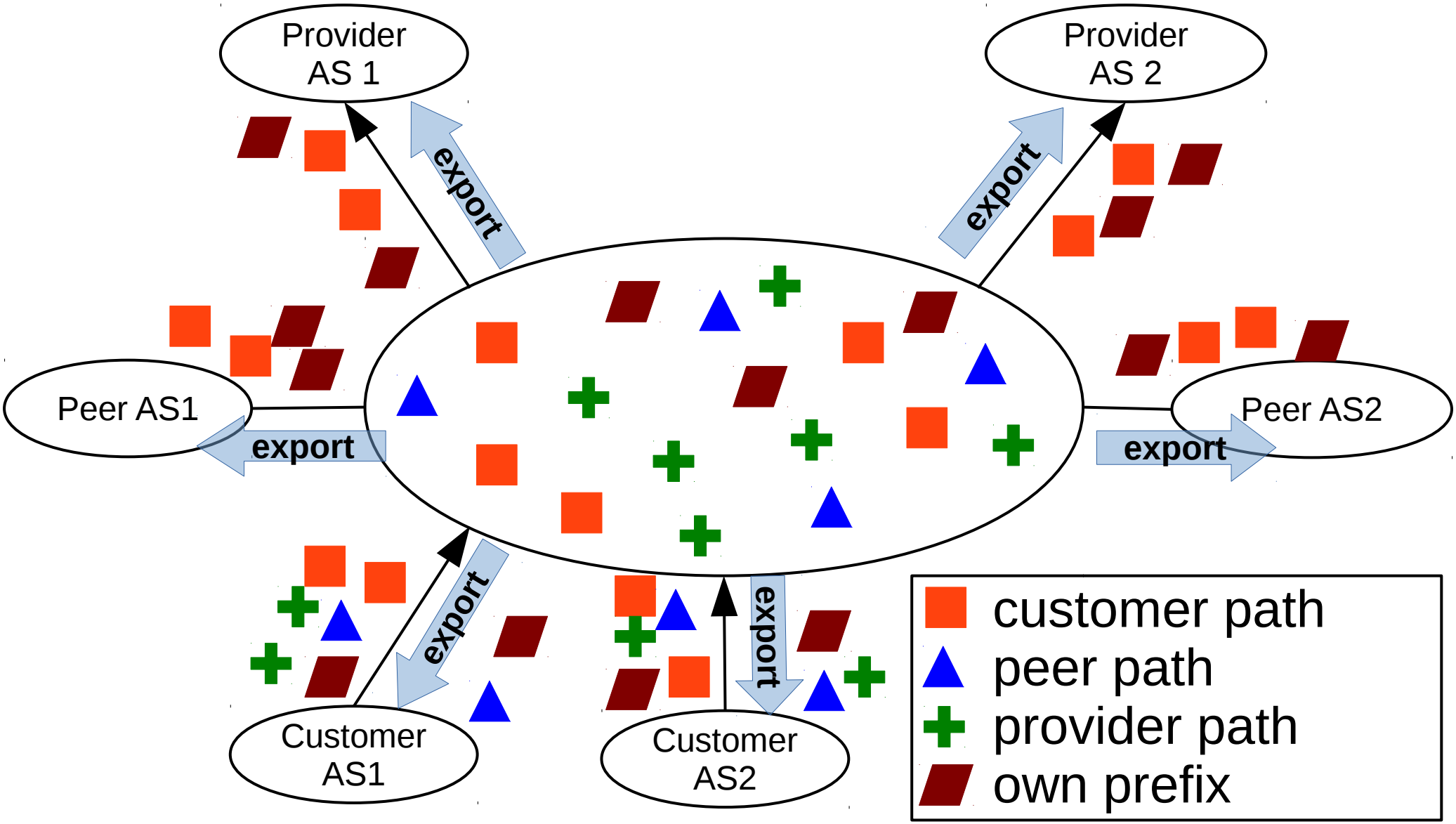
- **Rule: announce paths through customers and own prefixes to providers, block everything else (peer paths and path via other providers)**
- Same filtering rule as for peers



Valley-free routing: import



Valley-free routing: export



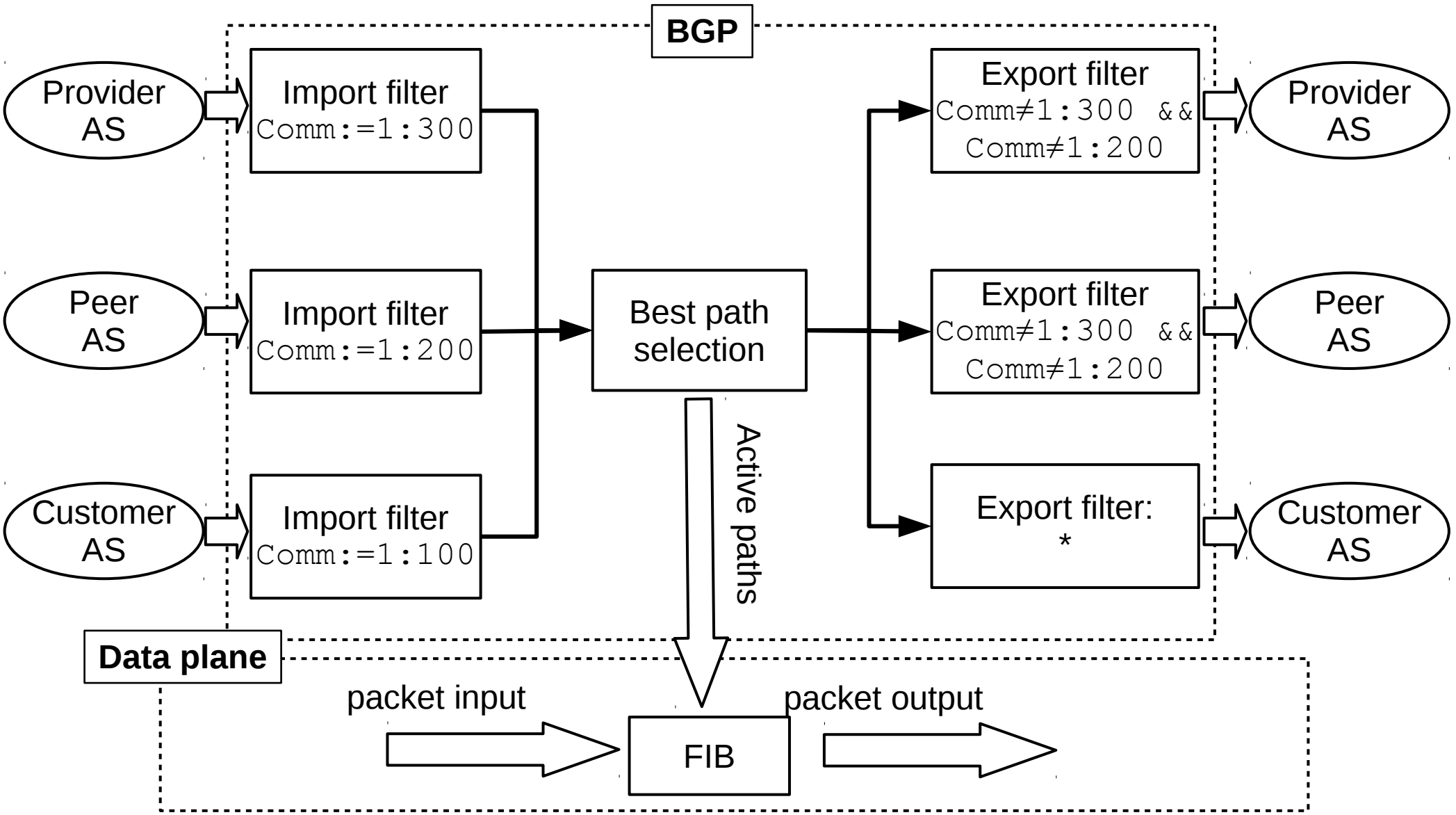
Valley-free routing: BGP conf

- Translate this logic to the BGP configuration language
- **1) Tag BGP announcements with the type of the link it was received on:** set the BGP COMMUNITY attribute with import filters
- So announcements describe how we got them:
  - from a customer: `community = 1:100`
  - from a peer: `community = 1:200`
  - from a provider: `community = 1:300`

Valley-free routing: BGP conf

- **2) Filter on BGP Community at export filters**
- Let all announcements through to customers
- Towards peers and providers
  - permit paths received from customers (`Community=1:100`) and all announcements to the AS's own prefixes
  - drop paths via peers (`Community=1:200`) and providers (`Community=1:300`)

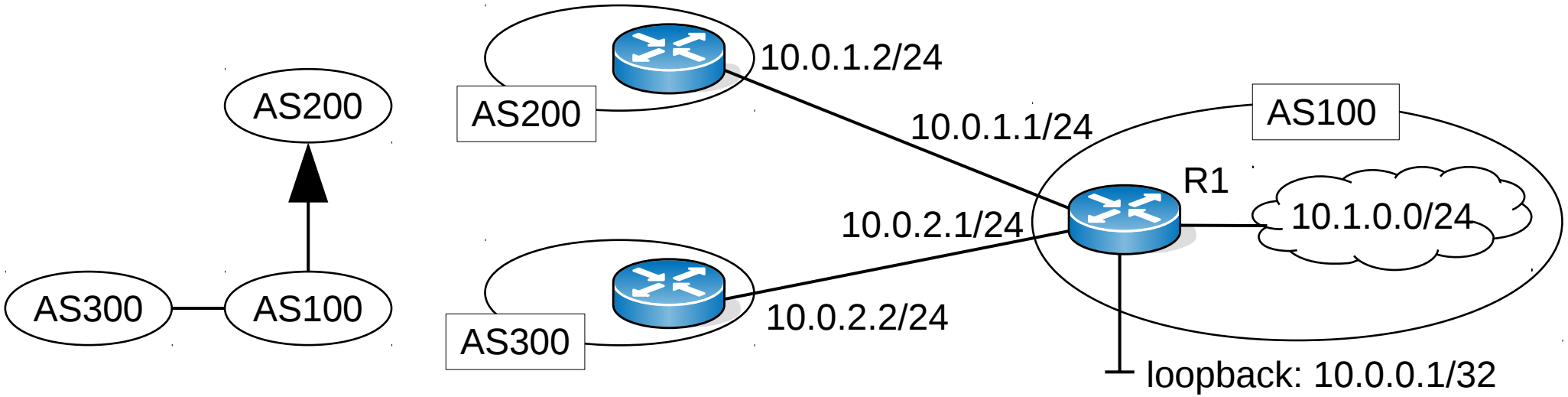
Valley-free routing: BGP conf



Valley-free routing: Import filters

- Suppose `AS200` is a provider and `AS300` is a peer of `AS100`
- Import filter to tag the announcements of `AS300`

```
route-map rm-peer-set-cm permit 20
  set community 1:200
```

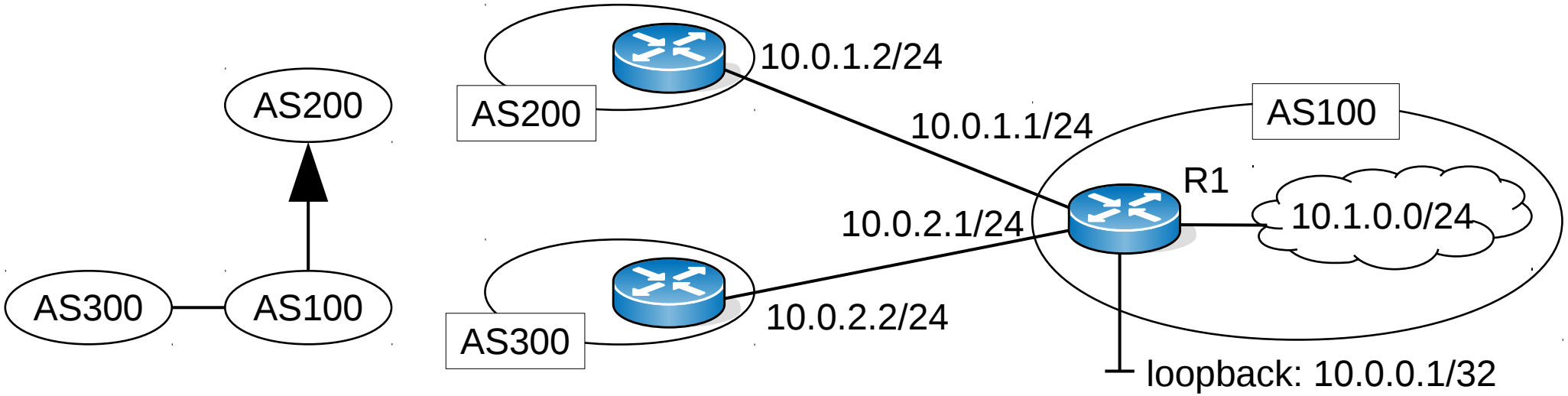


Valley-free routing: Import filters

- Order the filter to neighbor `AS300`

  ```
  neighbor 10.0.2.2 route-map rm-peer-set-cm in
  ```

- Similarly, define an import filter for tagging the announcements from provider `AS200` with the BGP community `1:300`



Valley-free routing: Export filters

- Announcements to peer `AS300` and provider `AS200` must be filtered on export filters
- First, define the set of communities to drop

```
ip community-list standard cm-no-export permit 1:300
ip community-list standard cm-no-export permit 1:200
```

- Match communities (`1:200` OR `1:300`) and `permit` announcements with such communities
- Match on both communities (`1:200` AND `1:300`):

```
ip community-list standard cm-list permit 1:200 1:300
```

Valley-free routing: Export filters

- The first `route-map` below drops announcements as per our community list (`1:200` and `1:300`)

```
route-map rm-no-export deny 10
  match community cm-no-export
```

- `deny`: matching announcements are dropped
- Set an empty, low-priority “catch-all” `route-map` to capture and `permit` remaining announcements

```
route-map rm-no-export permit 20
```

- Otherwise all these announcements would be dropped: the default policy for a `route-map` is to drop everything that did not match

Valley-free routing: Example

```
!!! BGP router configuration
!!! Communities:
!!!    1:100: customer
!!!    1:200: peer
!!!    1:300: provider
router bgp 100
 bgp router-id 10.0.0.1
 network 10.1.0.0/24
 neighbor 10.0.1.2 remote-as 200
 neighbor 10.0.1.2 route-map rm-prov-set-cm in
 neighbor 10.0.1.2 route-map rm-no-export out
 neighbor 10.0.2.2 remote-as 300
 neighbor 10.0.2.2 route-map rm-peer-set-cm in
 neighbor 10.0.2.2 route-map rm-no-export out

!!! cont'd on next page
```

Valley-free routing: Example

```
!!! import filters
route-map rm-prov-set-cm permit 10
 set community 1:300

route-map rm-peer-set-cm permit 10
 set community 1:200

route-map rm-cust-set-cm permit 10
 set community 1:100

!!! export filters
ip community-list standard cm-no-export permit 1:200
ip community-list standard cm-no-export permit 1:300

route-map rm-no-export deny 10
 match community cm-no-export

route-map rm-no-export permit 20
```