Computer Networks II

Ch.7 Advanced BGP

(bruno.quoitin@umons.ac.be)

Important note : These slides are partly based on a course by Olivier Bonaventure (UCLouvain).

Chapter 5: roadmap

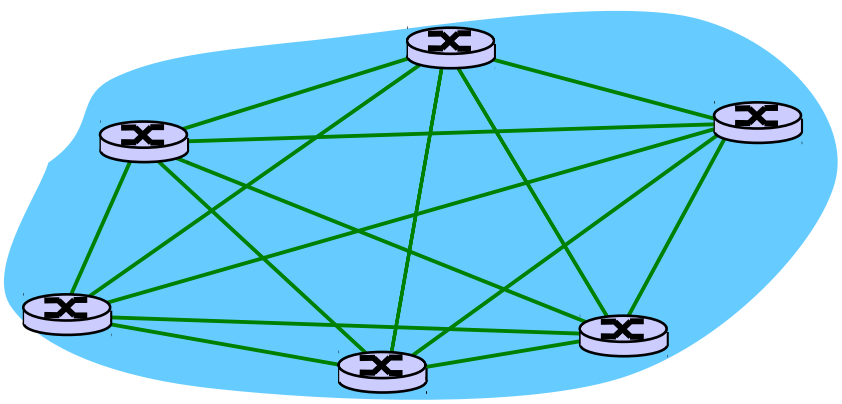
- 5.1 BGP Scalability
  - Confederations
  - Route-Reflection
  - Scalable Filters: Communities
- 5.2 BGP Stability

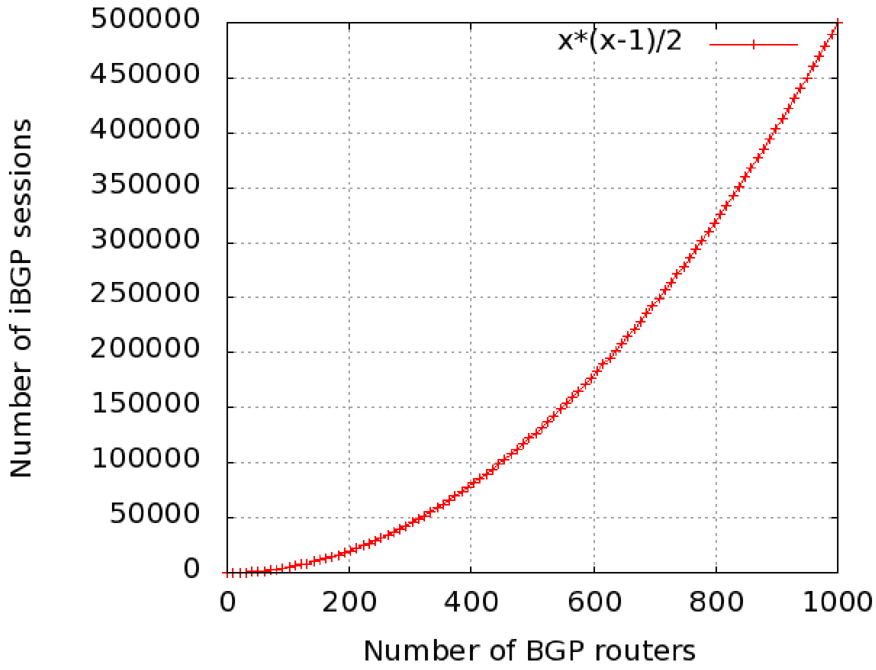
BGP Scalability

- **Objectives**
  - Reduce number of iBGP sessions
    - Management issue : adding a router in a domain requires modification to N-1 other routers in the domain.
    - Memory issue : Each router can store up to N-1 alternative routes received by iBGP
    - Solutions : BGP Confederations and BGP Route-Reflection
  - Allow more scalable routing policies management
    - Solution : BGP Communities

iBGP scalability

- **The iBGP full-mesh**
  - In an AS with N routers, $N*(N-1)/2$ iBGP sessions are needed !





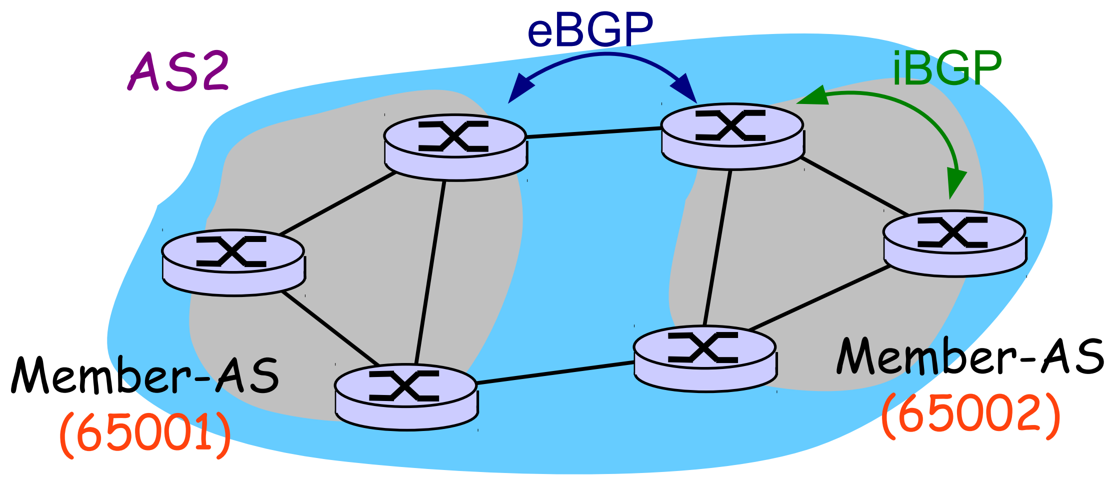
Chapter 5: roadmap

- 5.1 BGP Scalability
  - Confederations
  - Route-Reflection
  - Scalable Filters: Communities
- 5.2 BGP Stability

Confederations - RFC5065[(1)]

❑ Principle

- Divide a large AS in smaller Member-ASes
  - Use iBGP full-mesh inside each Member-AS
  - Use eBGP between Member-ASes
  - Each router as two ASN : confederation ASN and Member-AS ASN

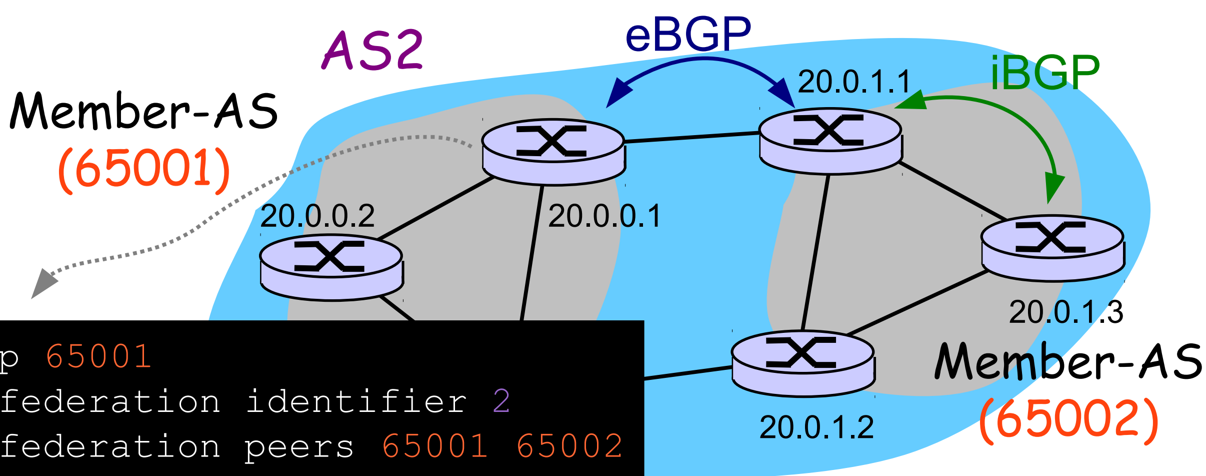


Conferation with 2 members

(1) Obsoletes RFC3065

Confederations

- **Principle**
  - Divide a large AS in smaller Member-ASes
    - Use iBGP full-mesh inside each Member-AS
    - Use eBGP between Member-ASes
    - Each router as two ASN: confederation ASN and Member-AS ASN



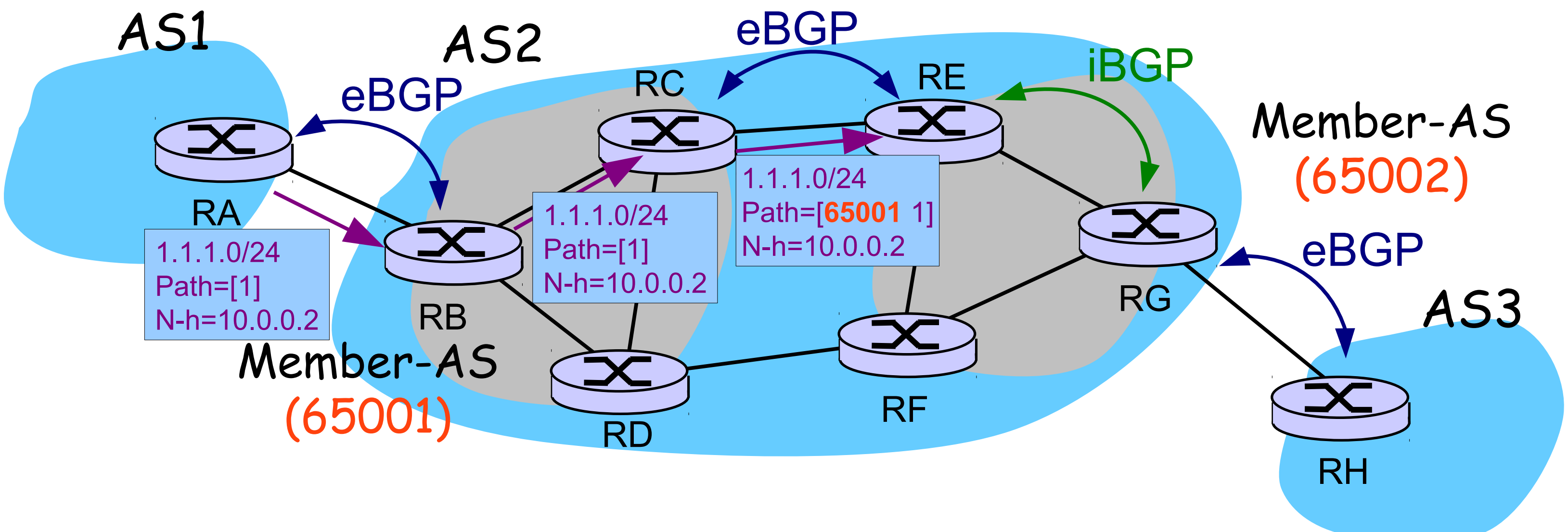

```
# router bgp 65001
#   bgp confederation identifier 2
#   bgp confederation peers 65001 65002
#   neighbor 20.0.0.2 remote-as 65001
#   neighbor 20.0.1.1 remote-as 65002
#   ...
```

A router's configuration specifies the confederation and member ASNs

Confederations

❑ Principle

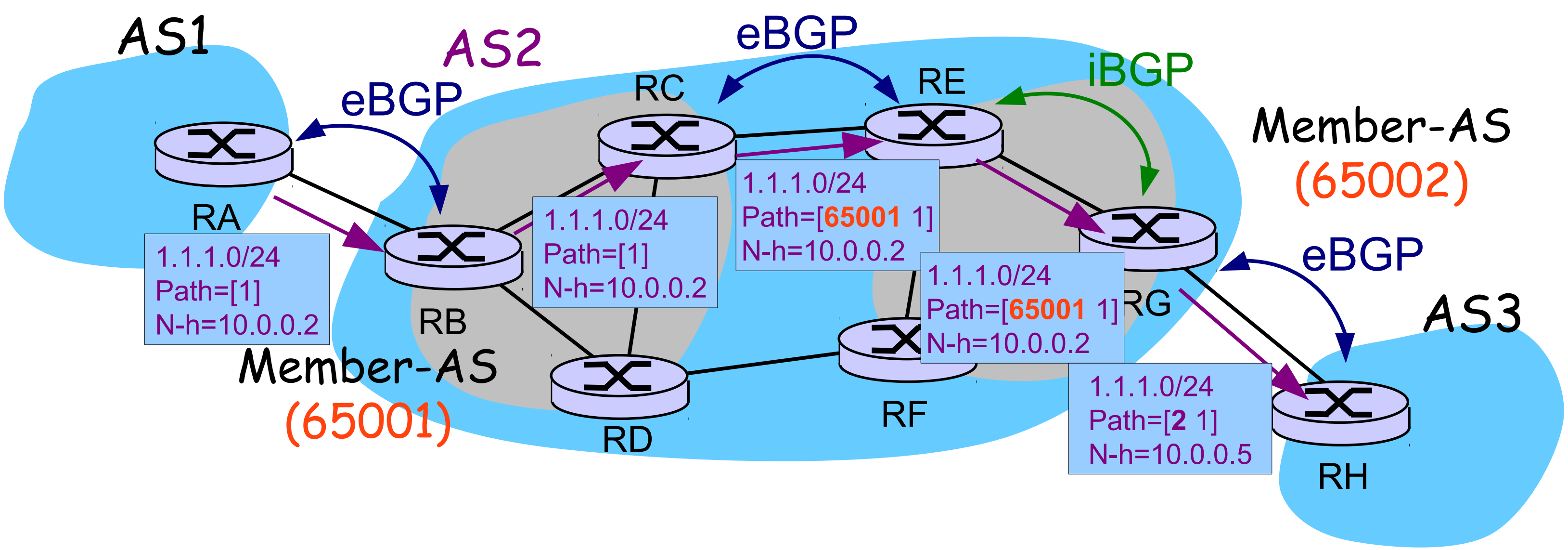
- When a router propagates an UPDATE via eBGP to a router inside the confederation
  - it must insert its Member-AS number in the AS-Path[1].
  - it can advertise unchanged Next-Hop as well as the Local-Pref attribute.



(1) AS-Path can contain new AS_CONFED_SET and AS_CONFED_SEQUENCE segments.

Confederations

- **Principle**
  - When a router propagates an UPDATE via eBGP to a router outside of the confederation it must
    - remove the internal Member-AS ASNs
    - prepend the confederation's ASN.

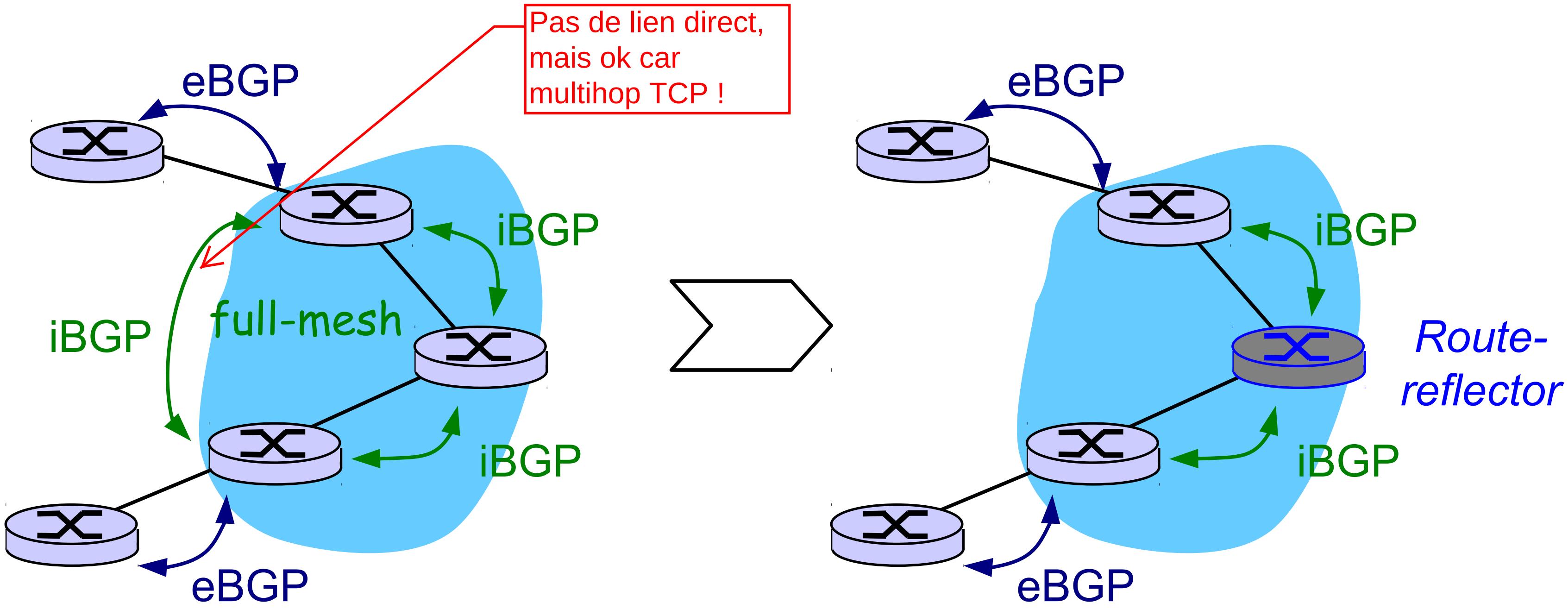


Chapter 5: roadmap

- 5.1 BGP Scalability
  - Confederations
  - Route-Reflection
  - Scalable Filters: Communities
- 5.2 BGP Stability

Route-Reflection (RFC4456)

- **Principle**
  - Special routers named route-reflectors are allowed to propagate over an iBGP session a route received from another iBGP session.

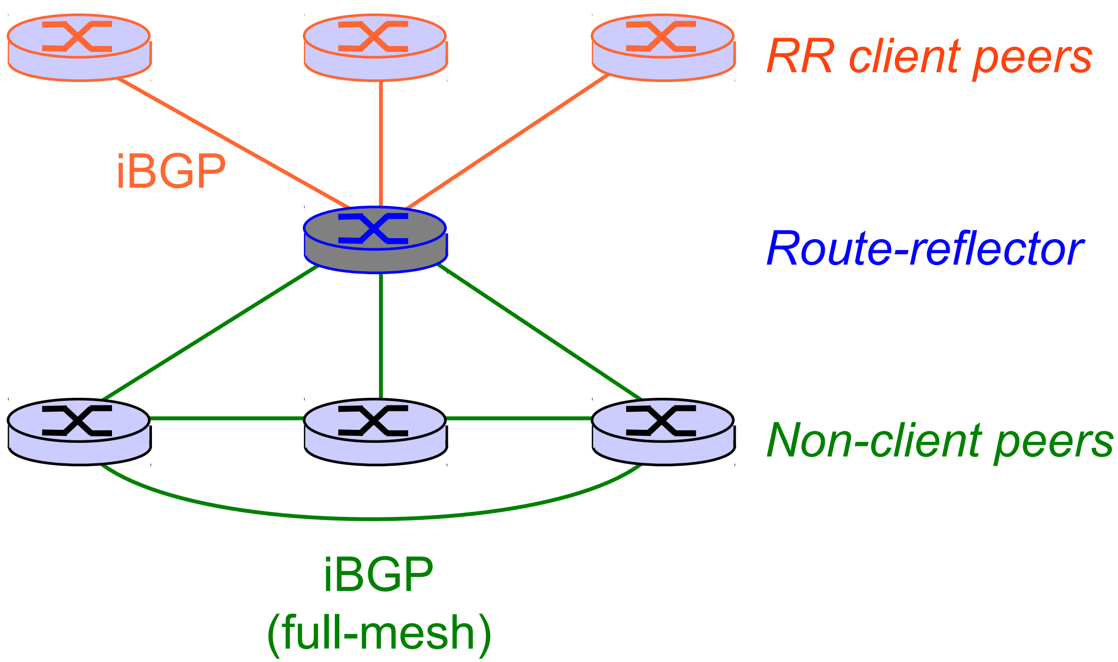


full-mesh iBGP

iBGP with Route-reflectors

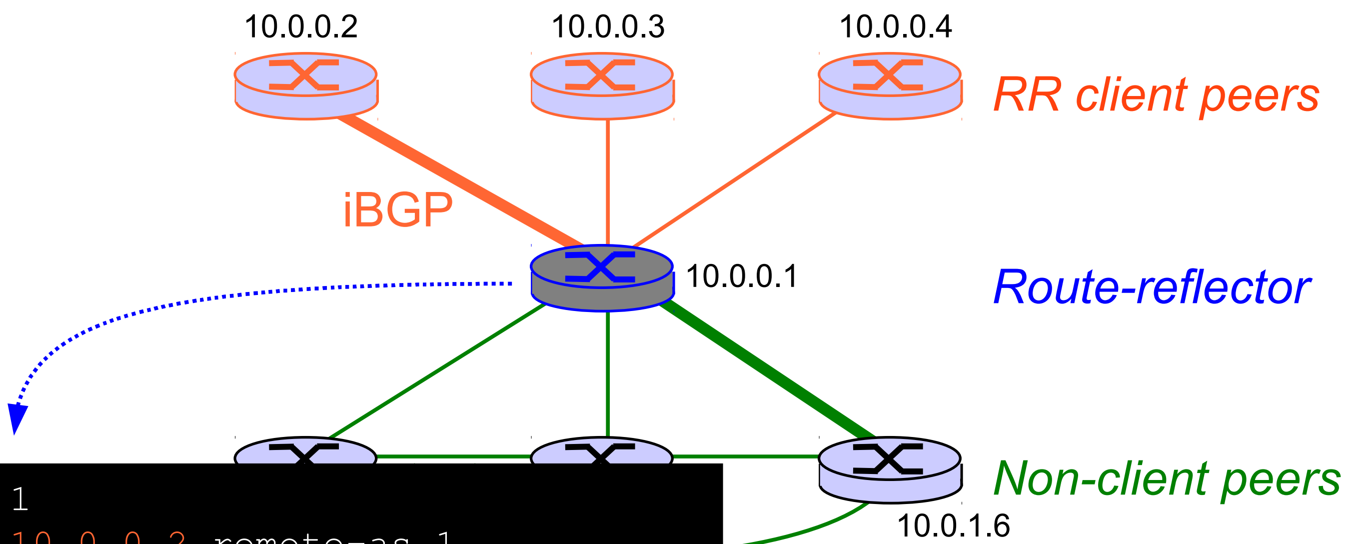
Route-Reflection

- **Clients / non-clients**
  - Route-reflectors have two types of iBGP peers :
    - RR client peers : do not participate in iBGP full-mesh
    - non-client peers : fully iBGP meshed



Route-Reflection

- **Clients / non-clients**
  - Route-reflectors have two types of iBGP peers
    - RR client peers : do not participate in iBGP full-mesh
    - non-client peers : fully iBGP meshed

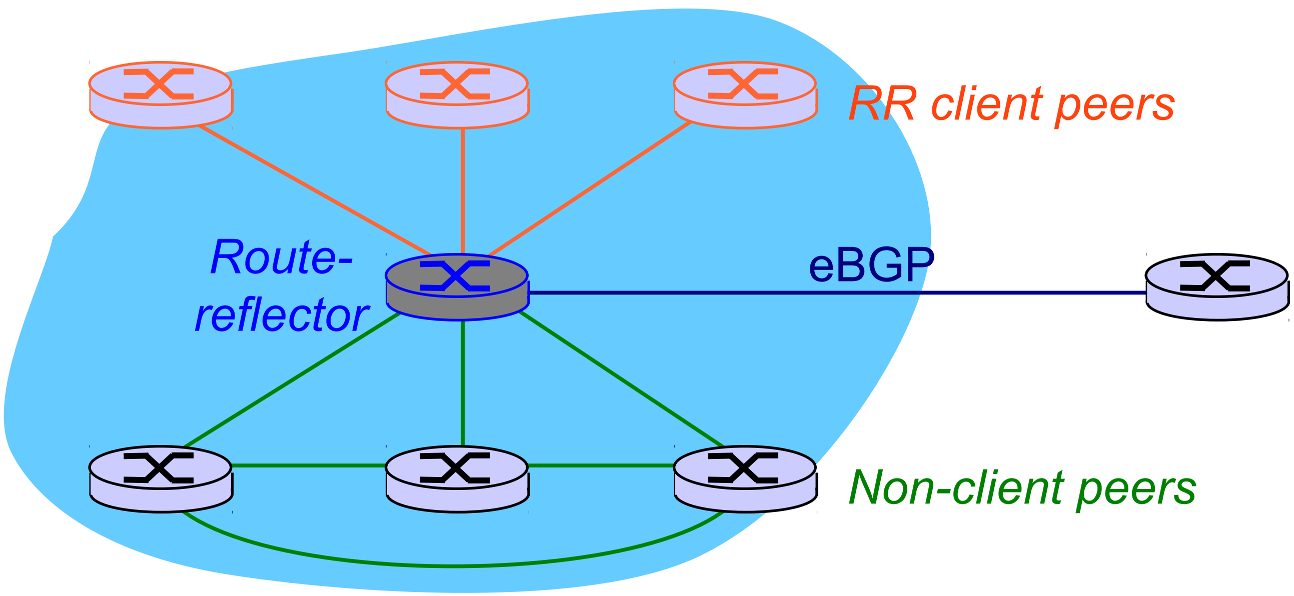



```
# router bgp 1
#   neighbor 10.0.0.2 remote-as 1
#   neighbor 10.0.0.2 route-reflector-client
#   neighbor 10.0.1.6 remote-as 1
#   ...
```

iBGP client peers are identified
in a route-reflector's configuration

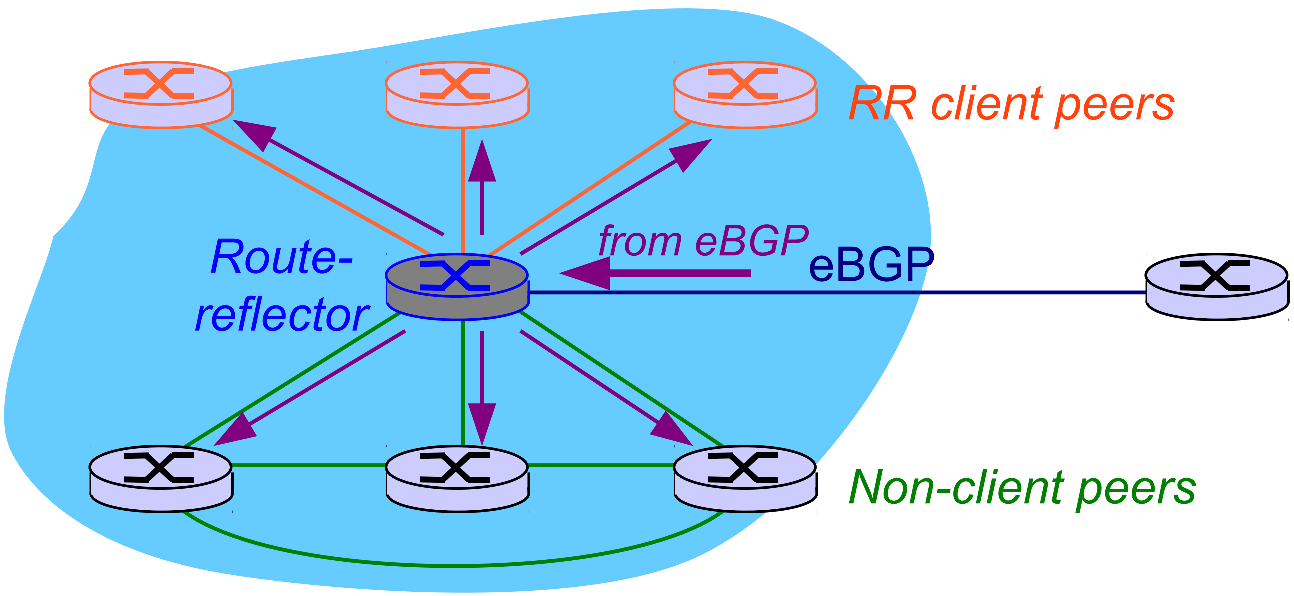
Route-Reflection

- **iBGP redistribution**
  - iBGP route redistribution rules for route-reflectors
    - from eBGP/client → all client peers, all non-client peers
    - from non-client → only client peers



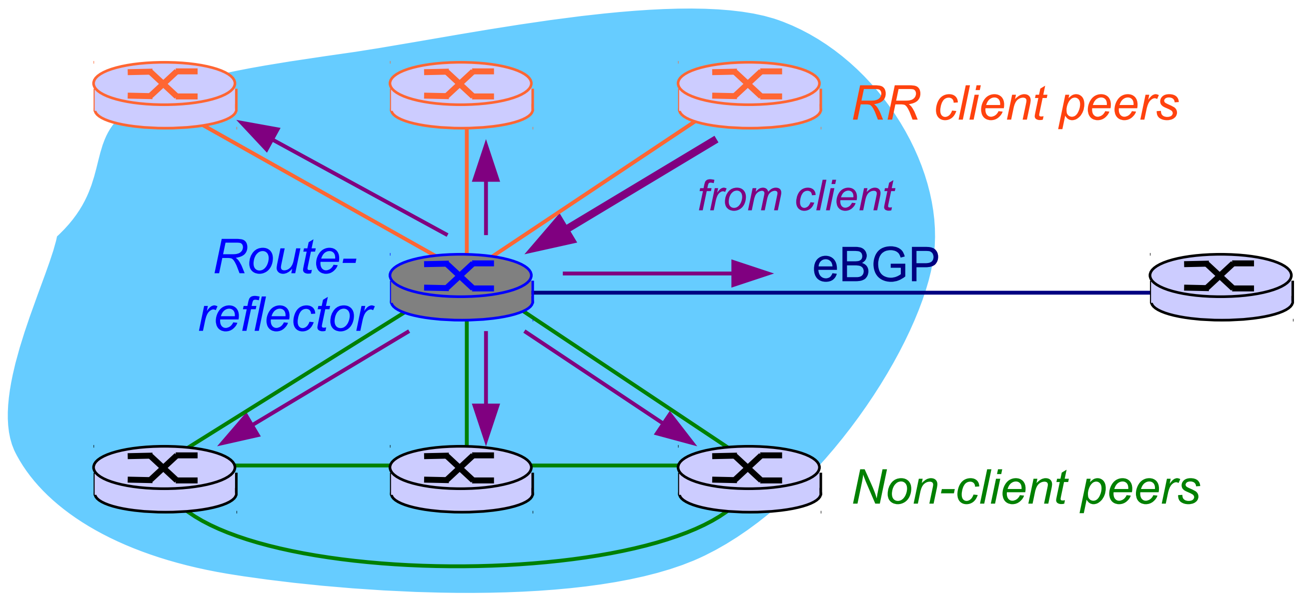
Route-Reflection

- iBGP redistribution
  - iBGP route redistribution rules for route-reflectors
    - from eBGP/client → all client peers, all non-client peers
    - from non-client → only client peers



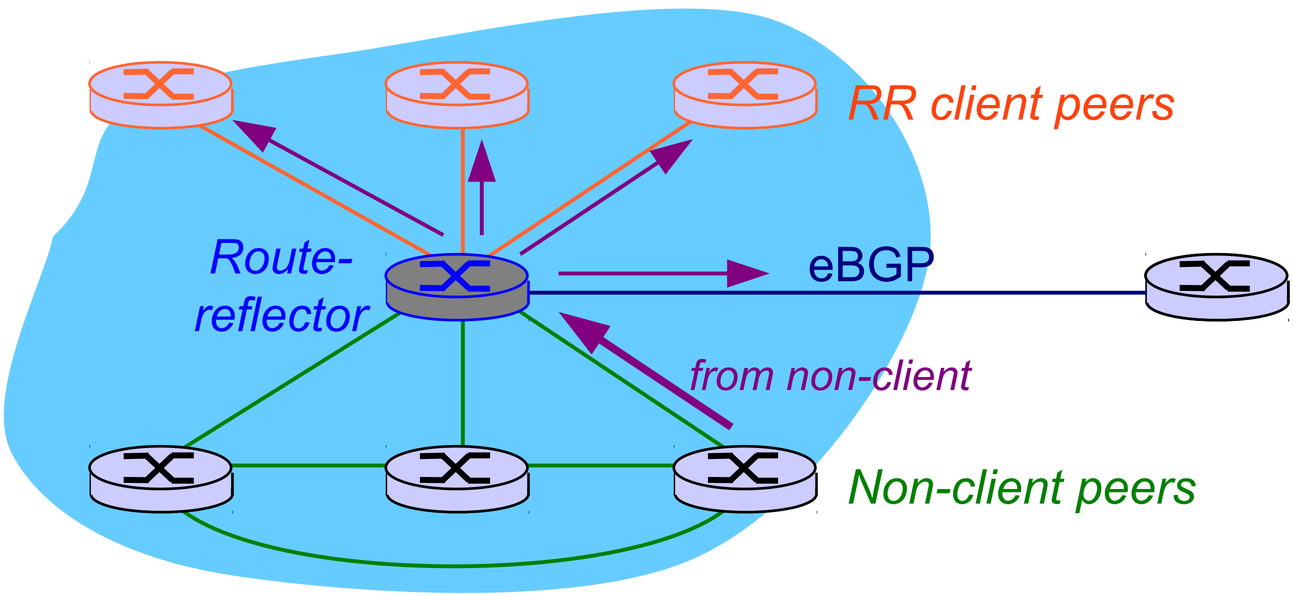
Route-Reflection

- **iBGP redistribution**
  - iBGP route redistribution rules for route-reflectors
    - from eBGP/client → all client peers, all non-client peers
    - from non-client → only client peers



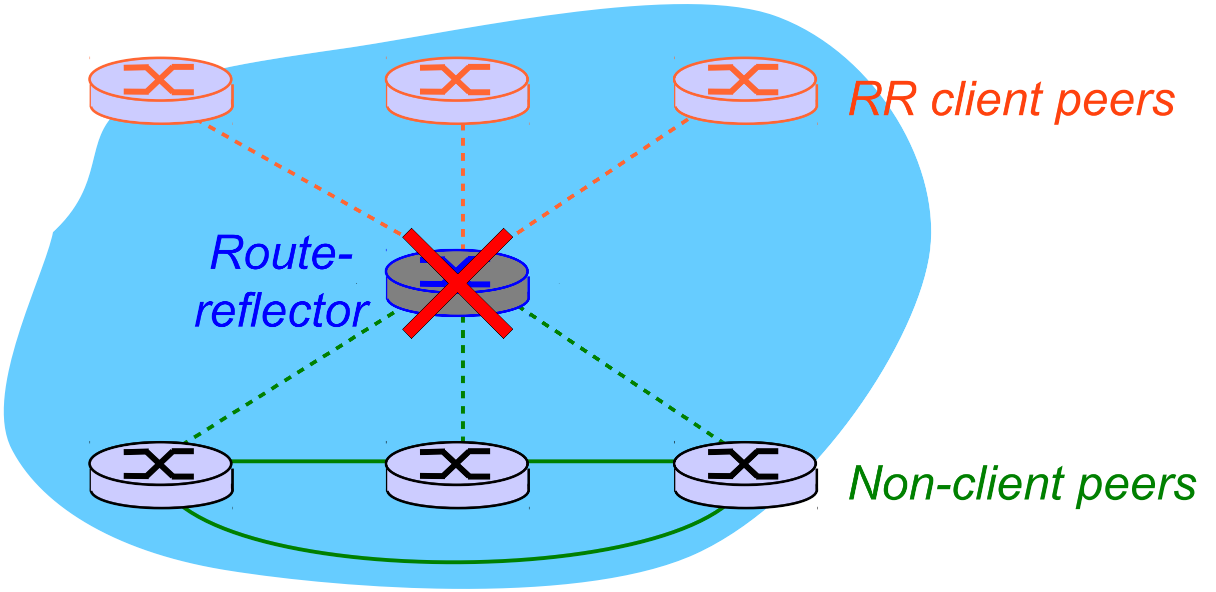
Route-Reflection

- **iBGP redistribution**
  - Route redistribution rules for route-reflectors are as follows:
    - from eBGP/client → all client peers, all non-client peers
    - from non-client → only client peers



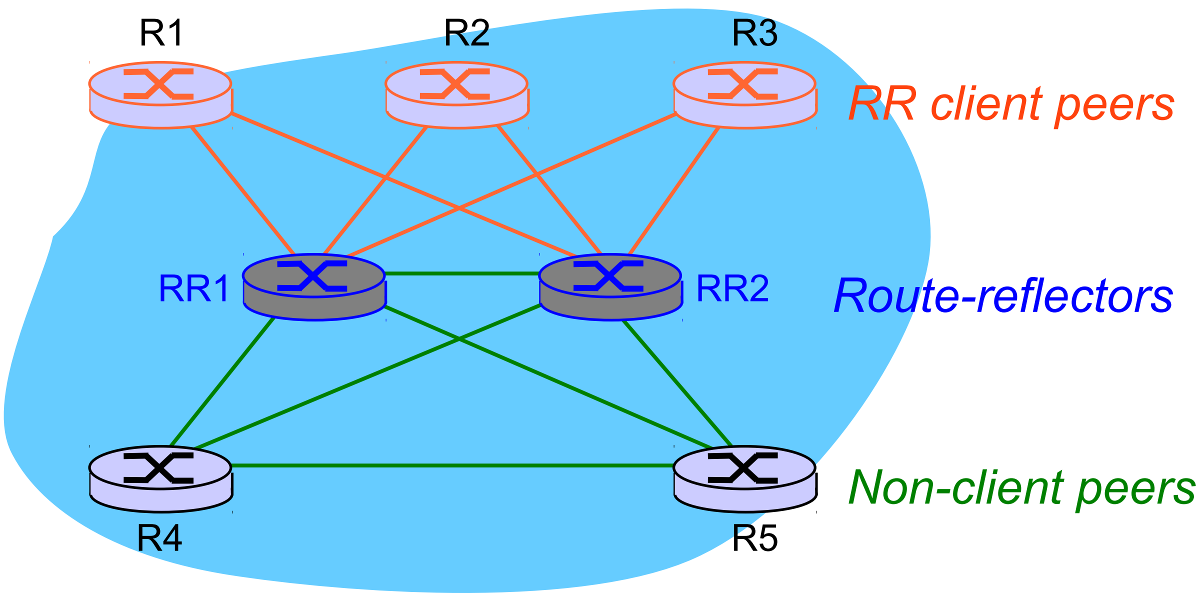
Route-Reflection

- **Fault Tolerance**
  - If the RR fails, all its clients are disconnected from the iBGP topology : single point of failure
  - How to avoid this ?



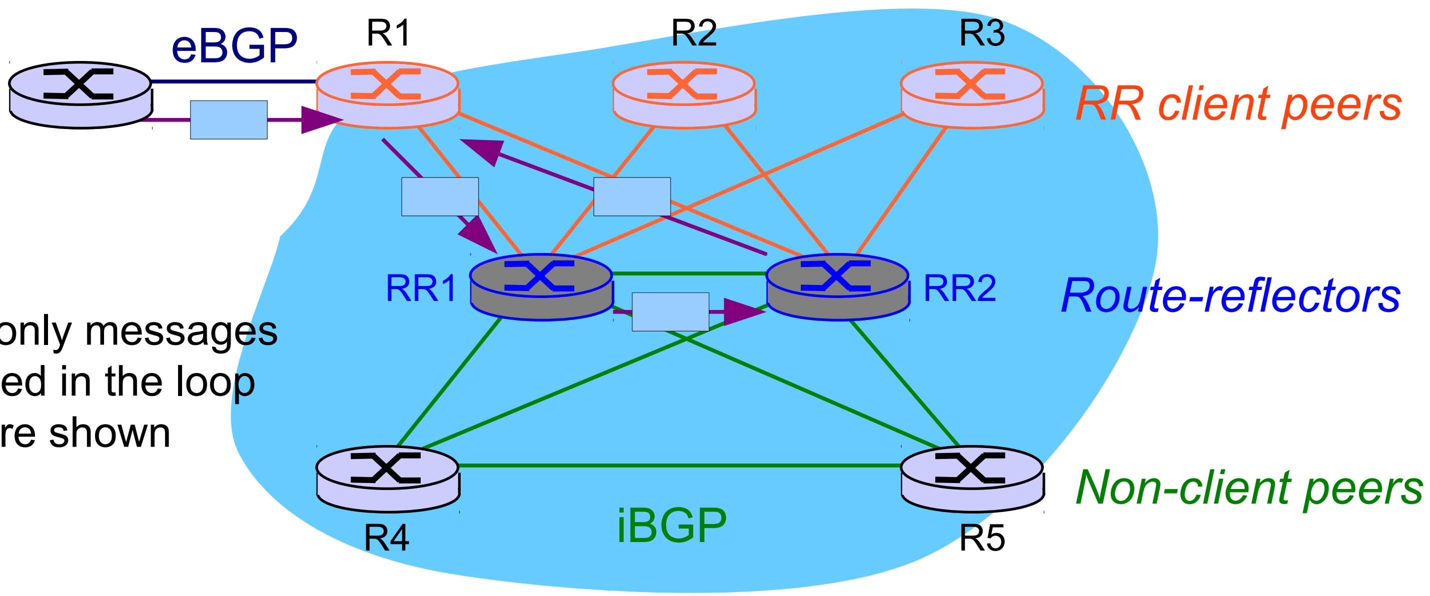
Route-Reflection

- **Fault Tolerance**
  - Use multiple Route-Reflectors.
  - Each client is connected to at least 2 Route-Reflectors.



Route-Reflection

- **Issue: routing plane loop**
  - We need to make sure an update cannot loop back[(1)] to its sender !

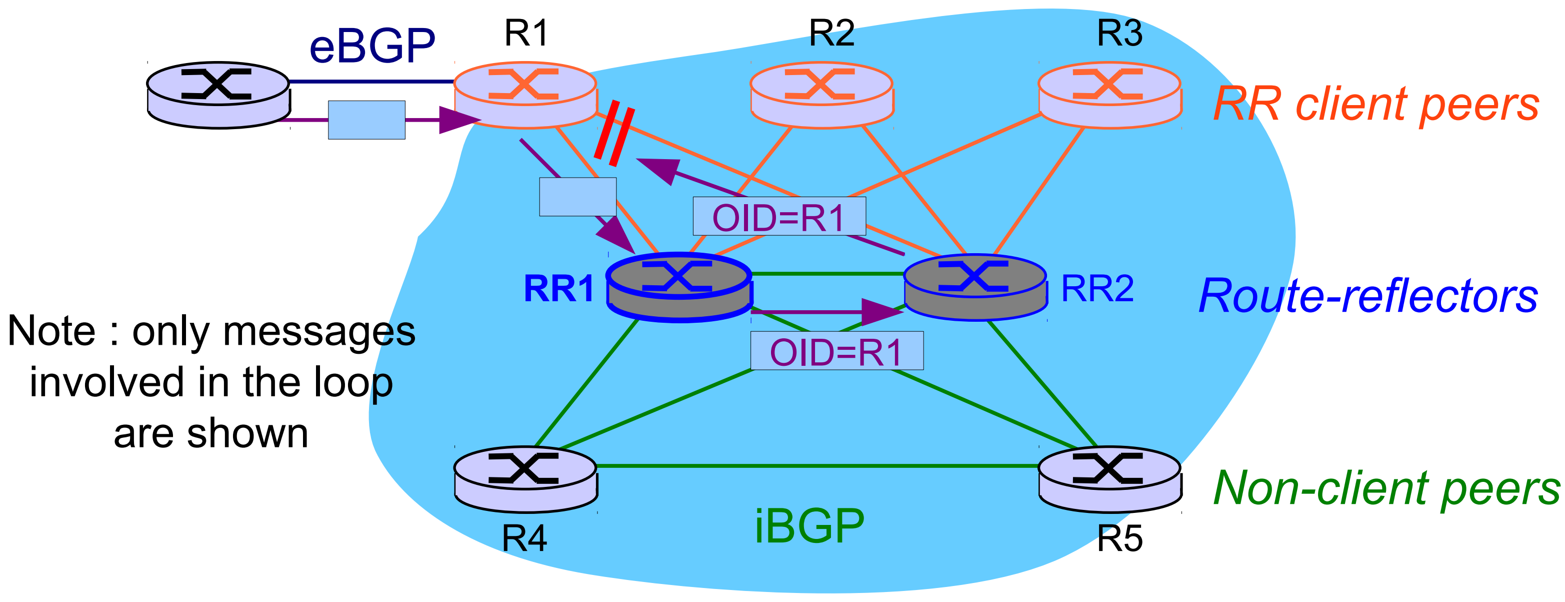


Note : only messages involved in the loop are shown

(1) Recall loops not possible :with eBGP thanks to AS-Path; with iBGP, redistribution not allowed

Route-Reflection

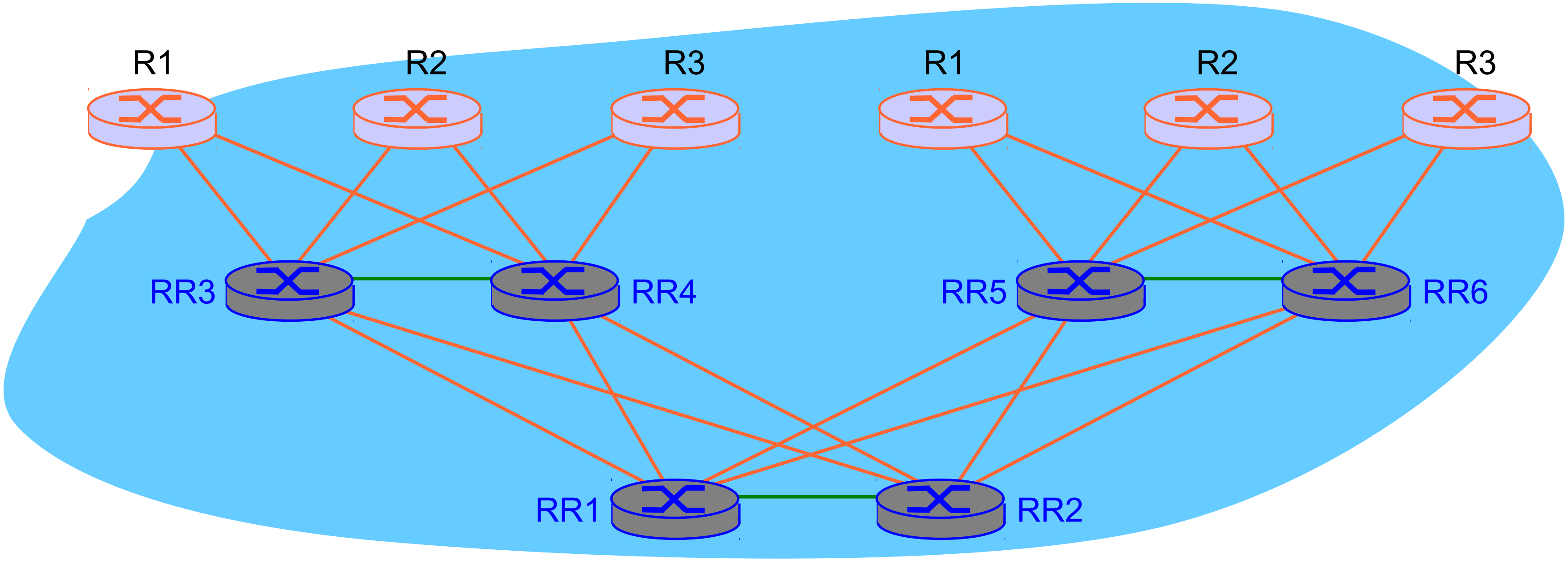
- **Solution: `Originator-Id`**
  - When a RR reflects a route, add `Router-Id` of router which first sent the route in the iBGP (only if this identifier not yet present). Deny route with itself as `Originator-Id`.



R1 refuse le message dupliqué

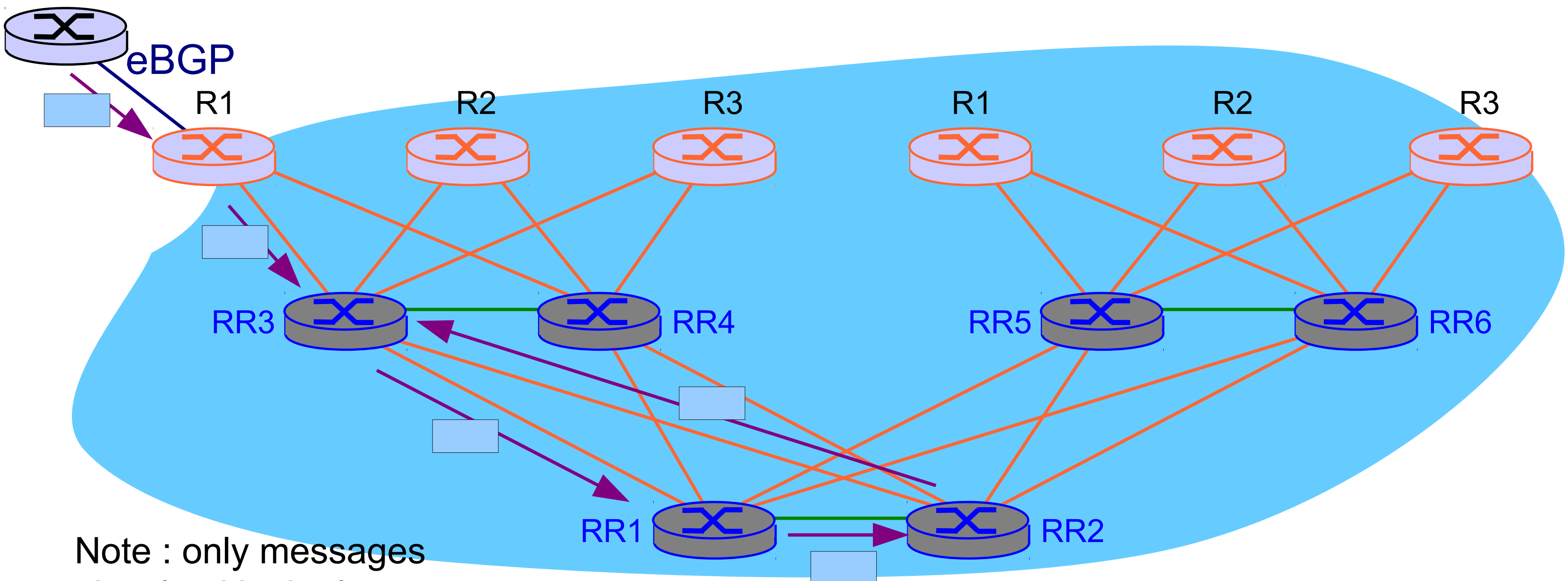
Route-Reflection

- **Hierarchy of Route-Reflectors**
  - A route-reflector can also be a client of another route-reflector. Same redistribution rules apply.



Route-Reflection

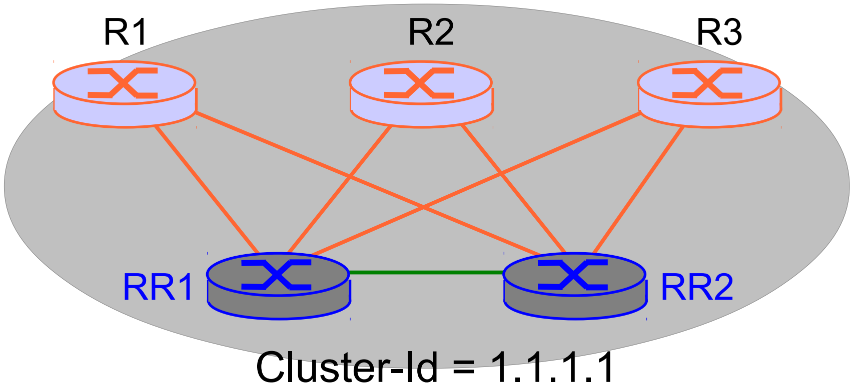
- **Hierarchy of Route-Reflectors**
  - Ensuring there is no routing loop is trickier. `Originator-Id` does not solve every instance...



Note : only messages involved in the loop are shown

Route-Reflection

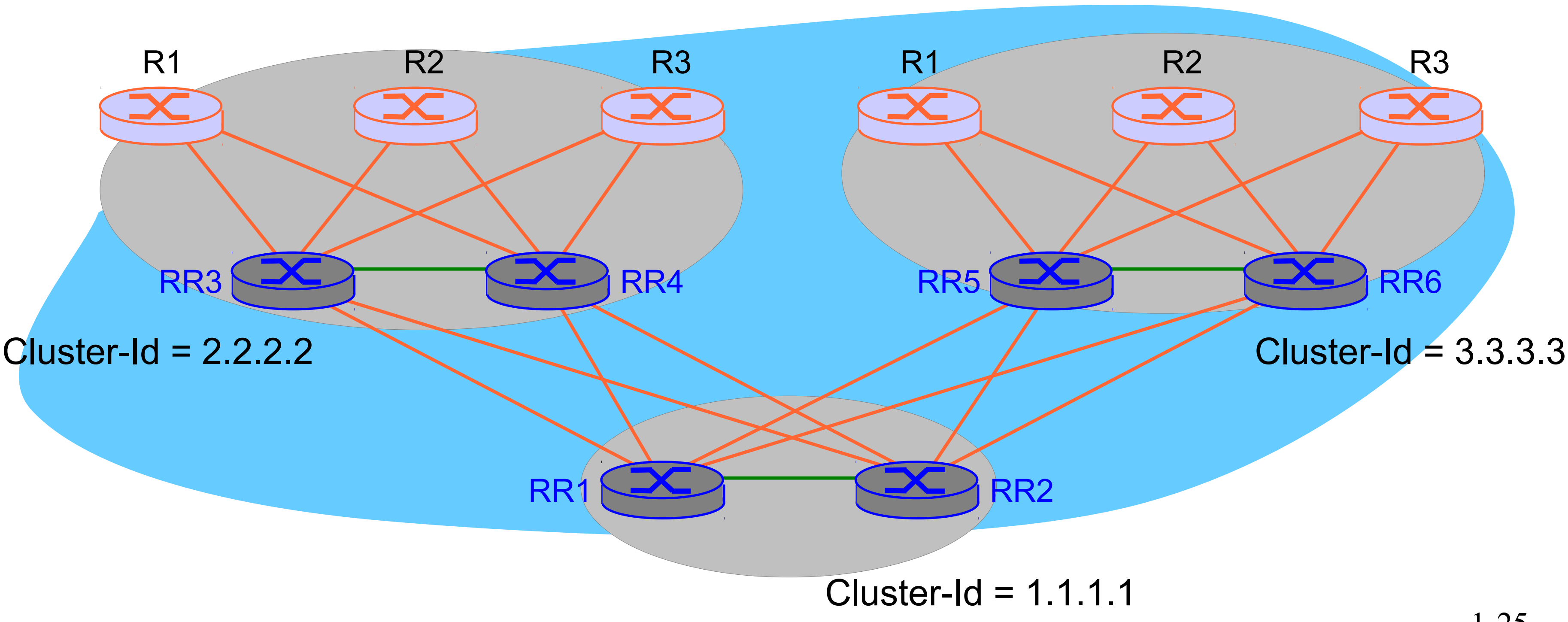
- **Hierarchy of Route-Reflectors**
  - Definition : a cluster is a set of RR-clients. A cluster is identified by the BGP identifier of its Route-reflector. When there are multiple RRs in a cluster, a common `Cluster-Id` is assigned to all these RRs.



A cluster with two RRs.

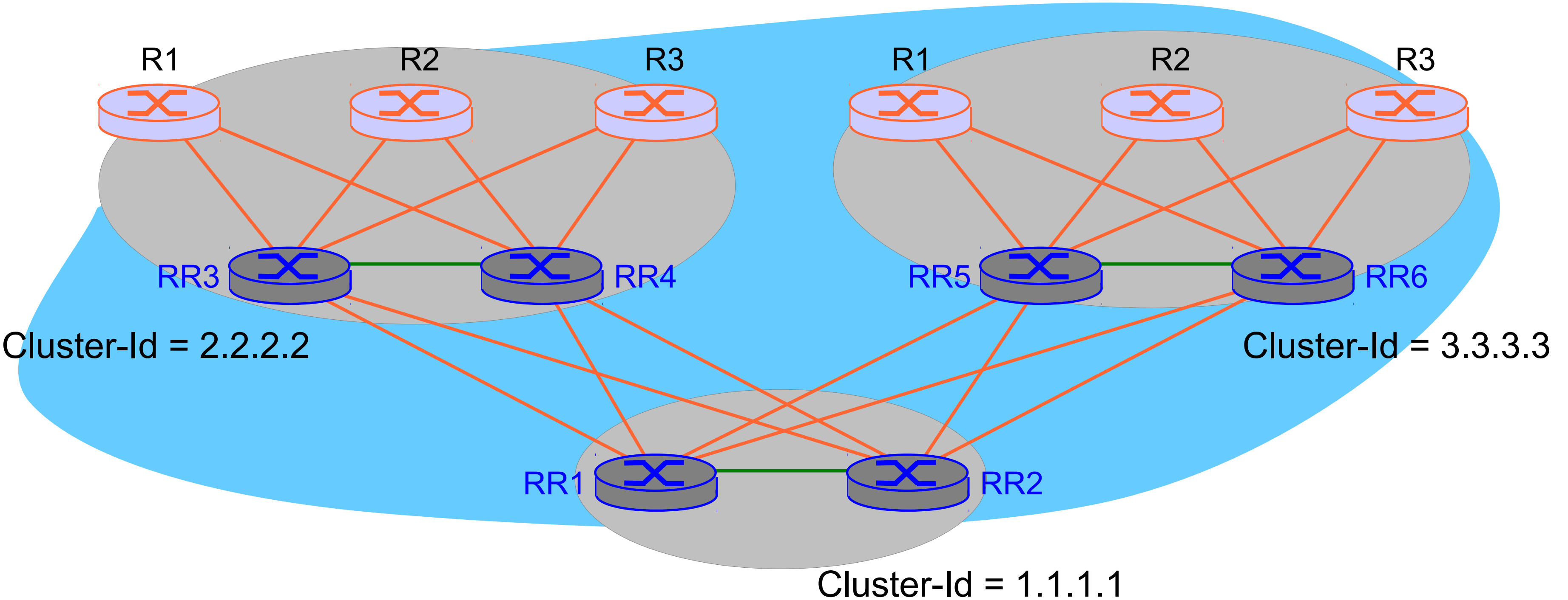
Route-Reflection

- **Hierarchy of Route-Reflectors**
  - Another example...



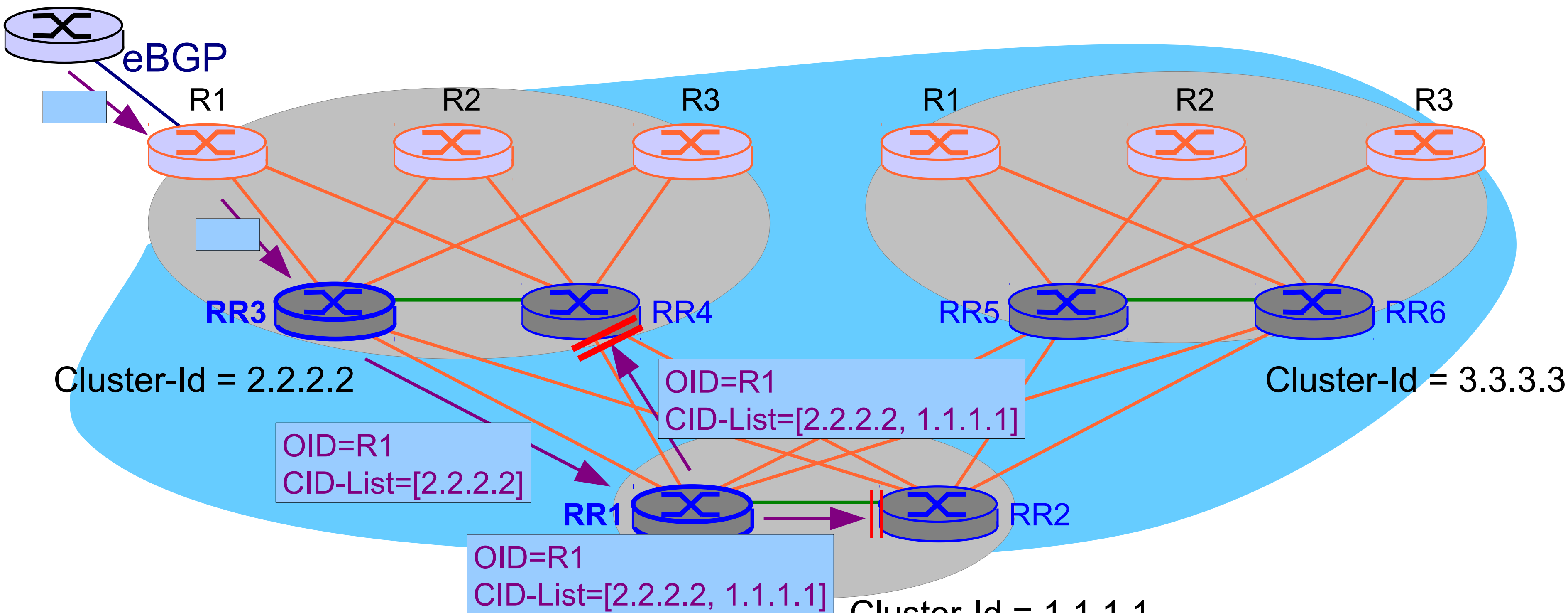
Route-Reflection

- **Hierarchy of Route-Reflectors**
  - New attribute: `Cluster-Id-List`. When a RR reflects a route, it appends its own `Cluster-Id` to the route's `Cluster-Id-List`.



Route-Reflection

- **Hierarchy of Route-Reflectors**
  - To avoid routing loops, prevent a route to reach a cluster if it has already traversed that cluster.



Note: not all messages are shown.

Decision Process (complete)

1. Ignore if next-hop unreachable
2. Prefer locally originated networks
3. Prefer **highest** **LOCAL-PREF**
4. Prefer **shortest** **AS-PATH**
5. Prefer **lowest** **ORIGIN**
6. Prefer **lowest** **MED**
7. Prefer **eBGP over iBGP**
8. Prefer **nearest next-hop**

tie-breaks

9. Prefer **lowest Router-ID /** **ORIGINATOR-ID**
10. Prefer **shortest** **CLUSTER-LIST**
11. Prefer **lowest neighbor address**

The exact decision process may slightly differ. See for example
http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094431.shtm

iBGP scalability

- **Route-reflectors vs Confederations**
  - With Route-reflectors,
    - clients need not be aware of hierarchy (only configure RRs)
    - higher load put on RRs → usually on biggest routers (number of messages to process, memory required)
  - With Confederation,
    - easy to migrate a multi-AS topology to a single confederation(1)
    - all routers need to be aware of confederation identifier and member-AS ASN

(1) e.g. after a merge of companies

Chapter 5: roadmap

- 5.1 BGP Scalability
  - Confederations
  - Route-Reflection
  - Scalable Filters: Communities
- 5.2 BGP Stability

Communities (RFC1997)

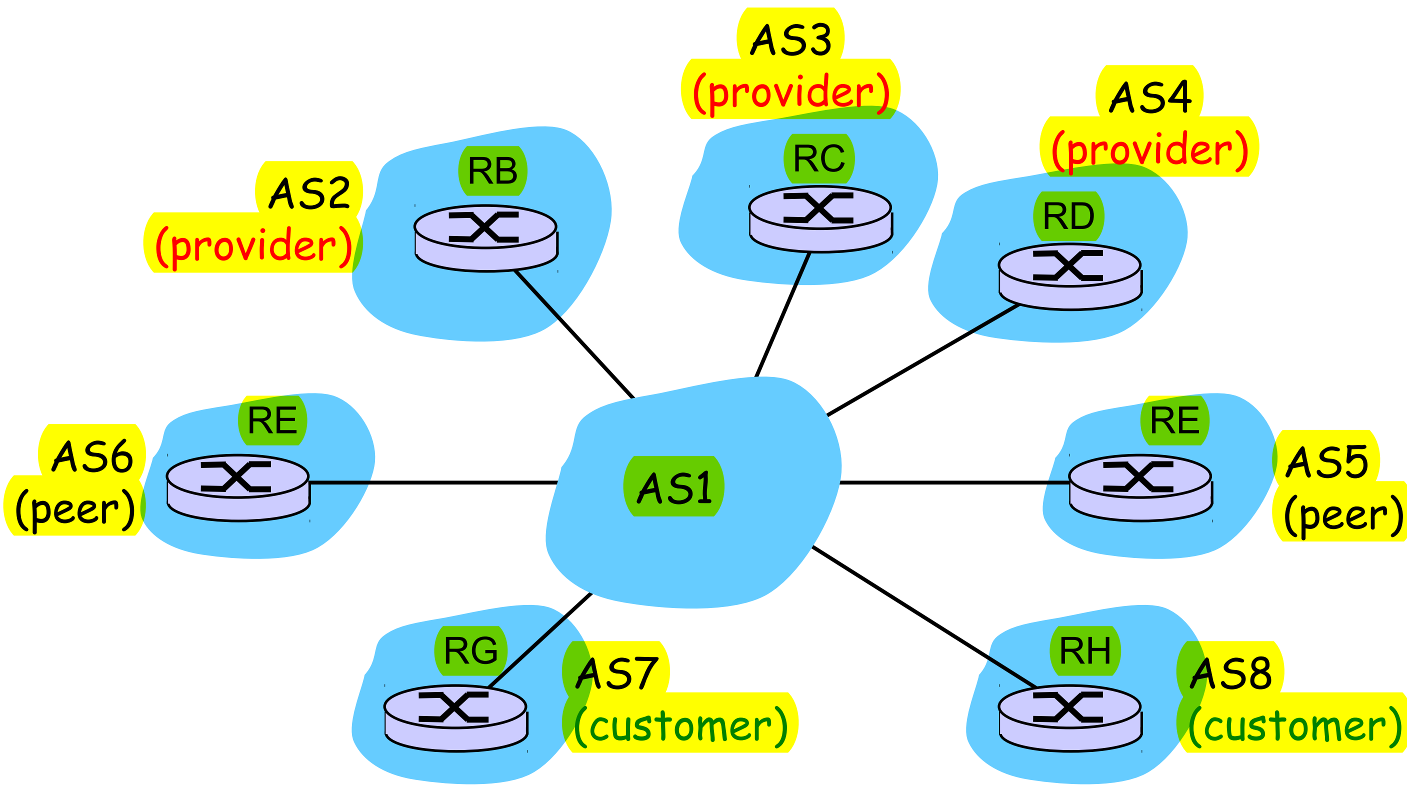
- **Principle**
  - A **Community** is a special integer value that can be attached to a route
    - A Community value is encoded as a 32 bits value
    - Routes with the same community value are usually treated in the same manner
  - *Standardized* community values
    - `NO_EXPORT` (0xFFFFFF01)
    - `NO_ADVERTISE` (0xFFFFFF02)
  - *Delegated* community values
    - Each AS has been delegated 65536 community values in the range ASN:0 to ASN:0xFFFF
    - Semantics of these values is up to the owner AS

Communities

- **Principle**
  - The `Community` attribute contains the set of community values (communities) attached to a route.
  - Attach same community value to all routes that need to receive the same treatment.

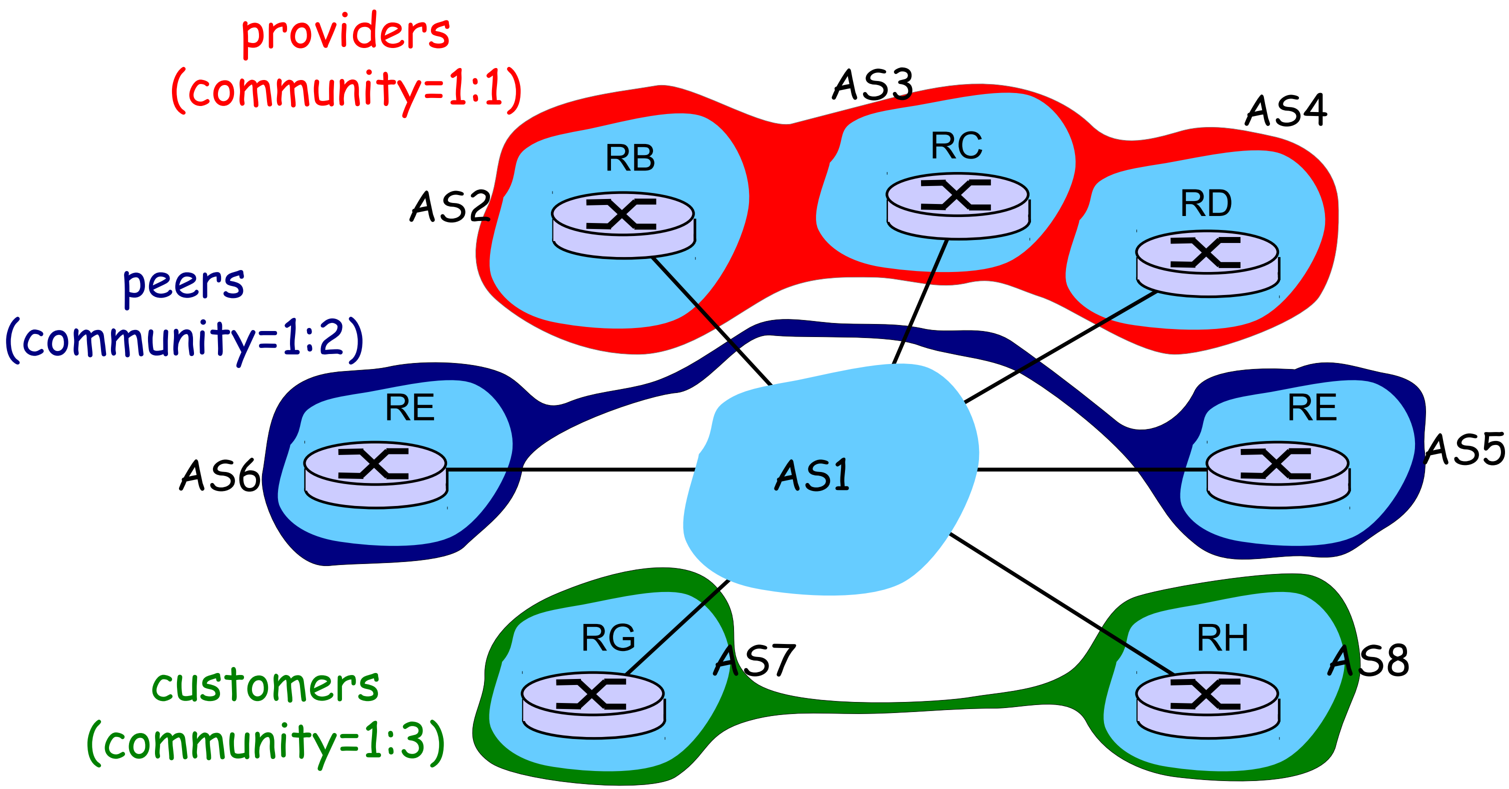
Communities

- **Utilization**
  - How to deal with simple policies that involve a large number of different neighbors ?



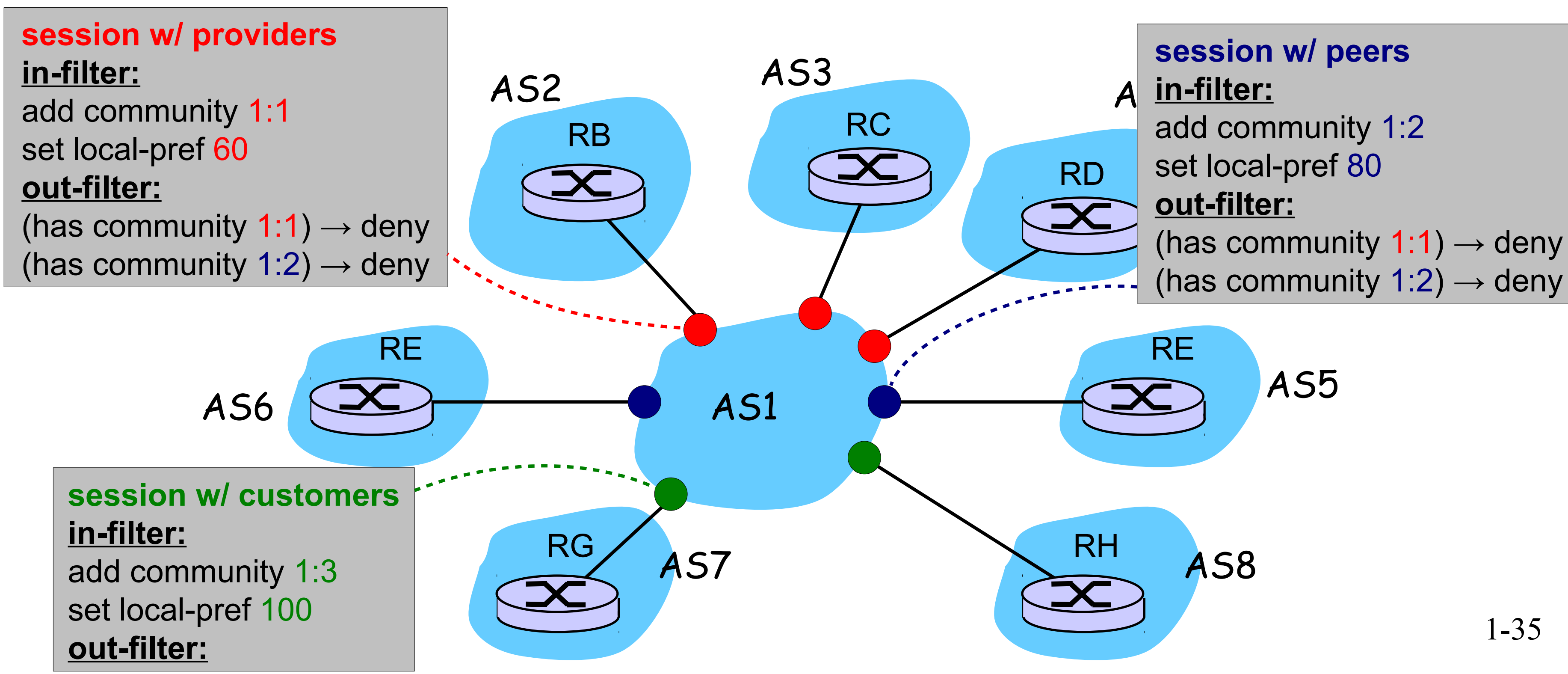
Communities

- **Utilization**
  - Attach same community value to all routes that need to receive the same treatment.



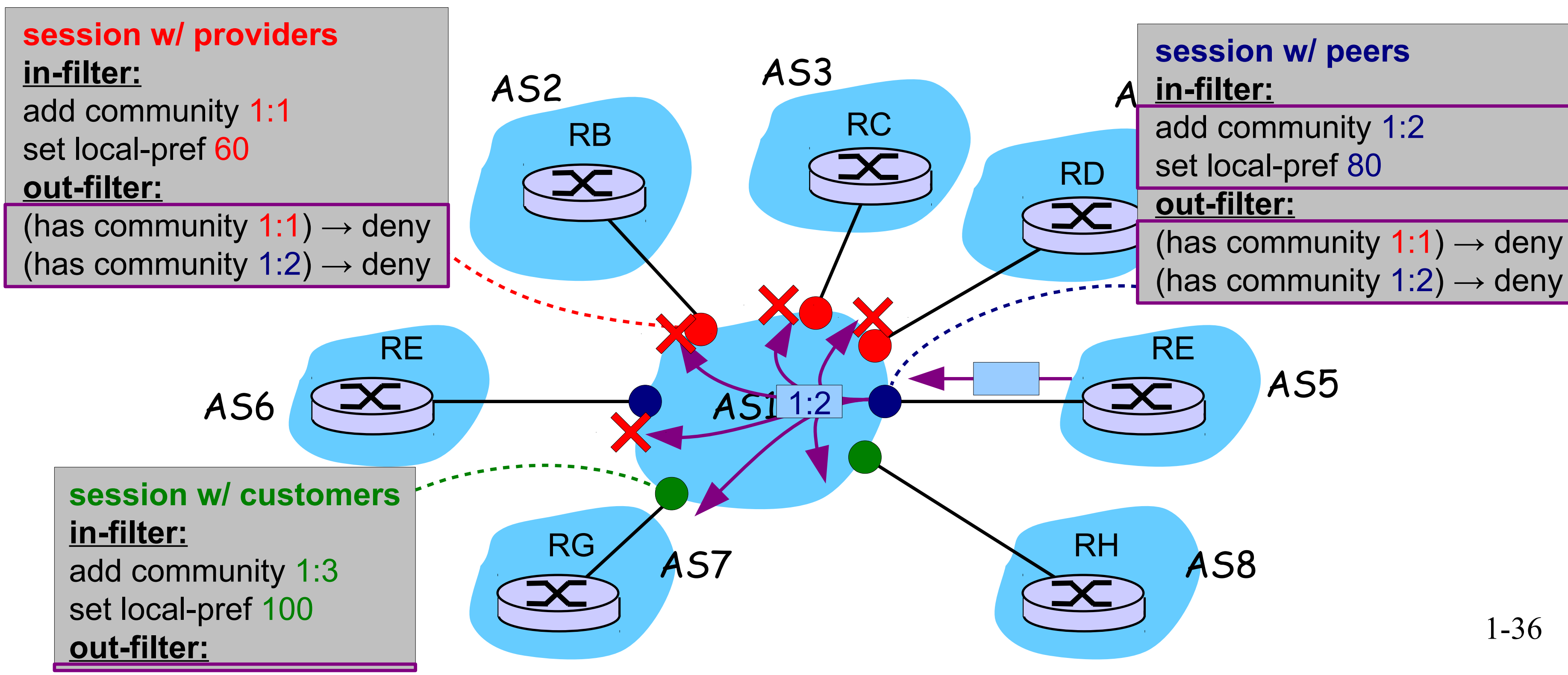
Communities

- **Utilization**
  - Use same set of policies on each session with a neighbor of the same category (prov/peer/cust)



Communities

- **Utilization**
  - Example: a route received from a peer is marked with 1:2, then out-filtered on every session according to category



Communities

- **More complex routing policies**
  - Research ISP providing 2 types of services
    - Access to research networks for universities
    - Access to the commercial Internet for universities and government institutions
  - How ?
    - Tag routes learned from research/commercial destinations.
    - Only announce research networks to universities and universities to research networks.

Communities

- **More complex routing policies**
  - Commercial ISP providing 2 transit services
  - Full transit service
    - Announce all known routes to all customers
    - Announce customer routes to all peers, customers, providers
  - Client routes only
    - Only advertise to those customers the routes learned from customers, but not routes learned from peers and providers
    - Advertise the routes learned from those customers only to customers

Communities

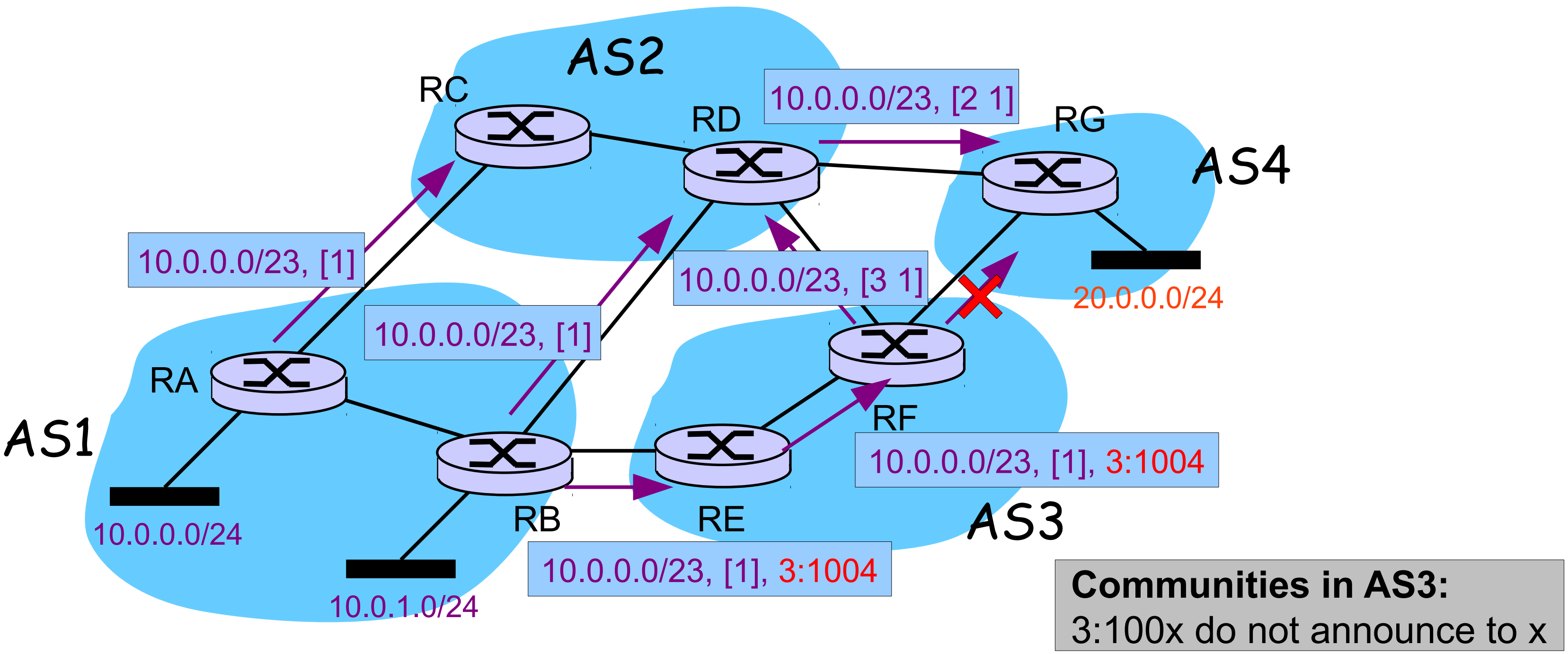
- **Communities used for tagging**
  - to indicate country where route was received
    - Example: Cable&Wireless (AS3561)
    - Community values 3561:SRCC where S is peer/customer, R is regional code, CC:ISO-3166-1 country code
  - to indicate IX where route was learned
    - Example Global Access Telecommunications (AS13129)
    - 13129:2110 routes learned at DE-CIX
    - 13129:2120 routes learned at INXS
    - 13129:2130 routes learned at SFINX

Communities

- **Communities used for TE**
  - Attach a special community value to request downstream router to perform a special action on the route
  - Possible actions
    - set local-pref in downstream AS
    - do not announce the route to ASx
    - prepend AS-Path when announcing to ASx
    - ...

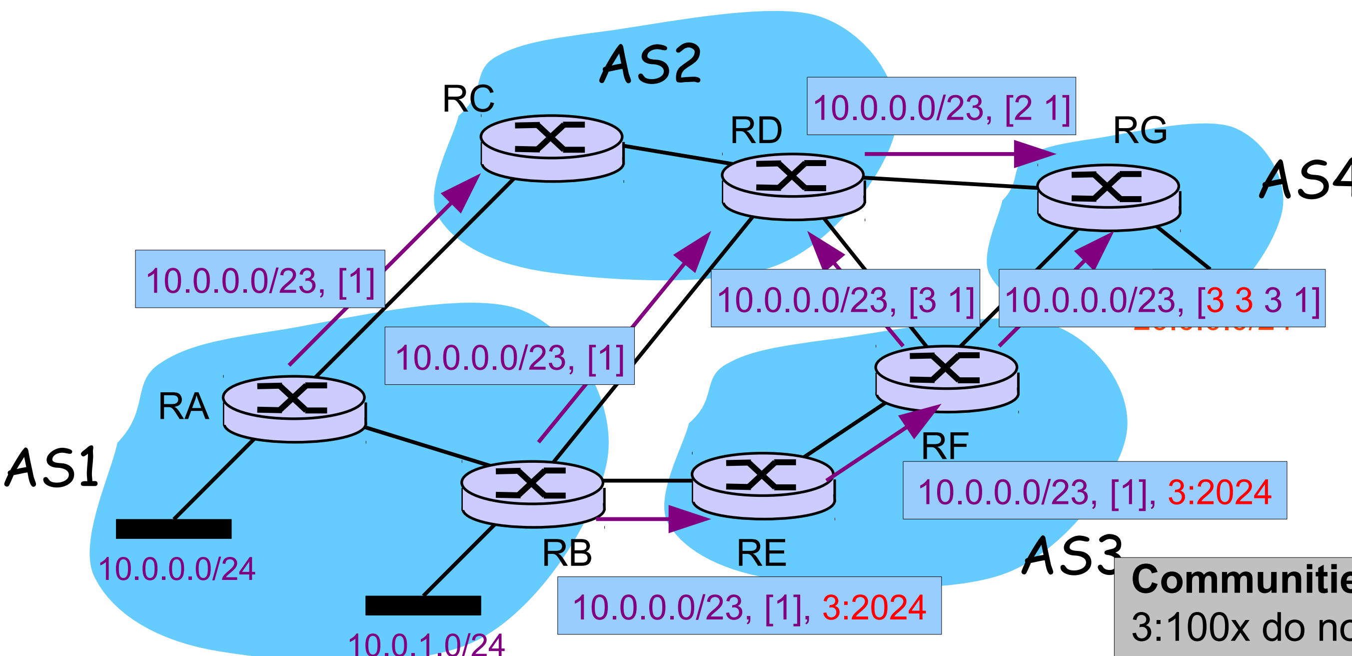
Communities

- Community-based selective announcements



Communities

- Community-based AS-Path prepending



Communities in AS3:
3:100x do not announce to x
3:20nx prepend n times to x

Communities

- **Issues**
  - A router may easily add community values to routes
  - The community attribute is transitive
    - A community value added by one router could be propagated to the global Internet
    - Example observed in 2002 (see NANOG25)

```
TABLE DUMP|1019465346|B|64.200.199.3|7911|57.249.147.0/24| 7911 3561 5511
3215|INCOMPLETE|64.200.199.3|0|0| 3215:101 3215:204 3215:500 3215:589
3215:903 3215:1001 3215:2001 3215:7503 3215:50000 3561:11840 3561:30010
3561:30020 3561:30030 3561:30040 3561:30050 3561:30060 3561:30070
3561:30080 3561:30090 3561:30100 3561:30110 3561:30120 3561:30130
3561:30140 3561:30150 3561:30160 3561:30170 3561:30180 3561:30190
3561:30200 3561:30410 3561:30420 3561:30430 3561:30440 3561:30450
3561:30460 5511:500 5511:502 5511:999 7911:999|NAG||
```

(3215=France Telecom, 3561=C&W, 5511=OpenTransit)

Communities

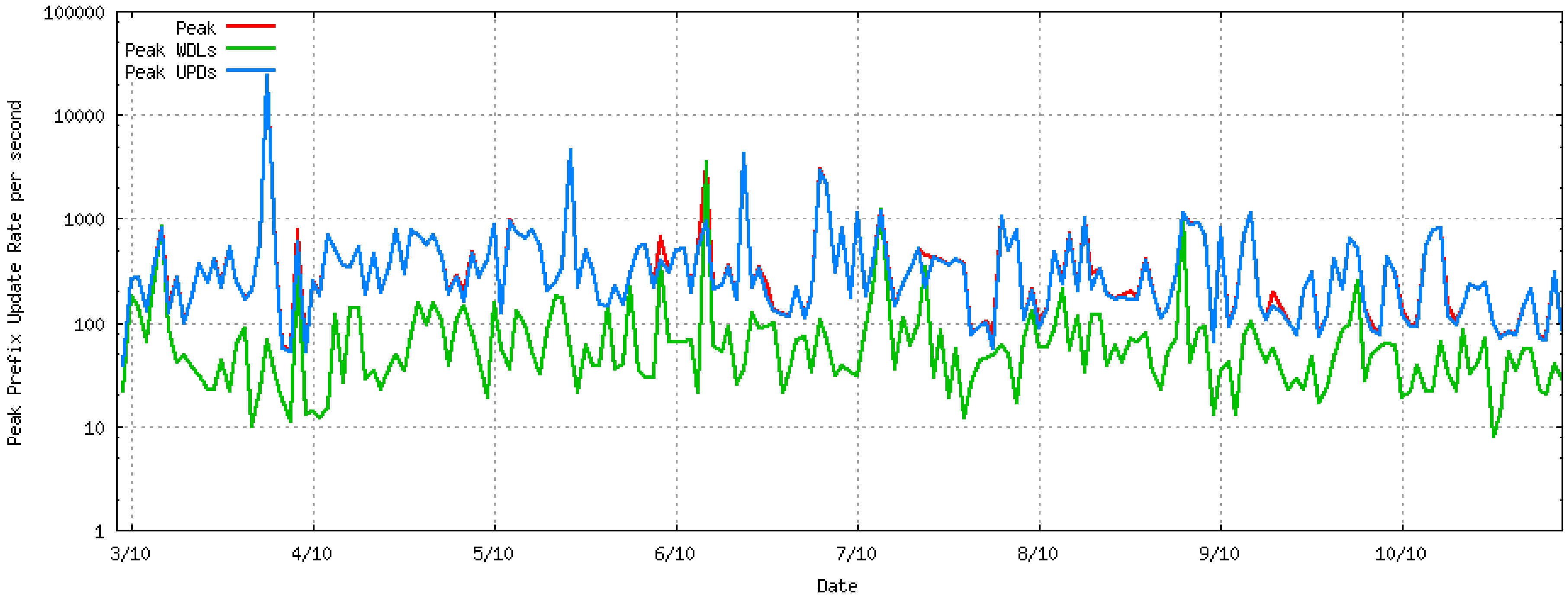
- **Issues**
  - The semantics of Communities is defined locally
    - Some ASes advertise the semantics of their communities by using RPSL in whois databases
    - Most of the community values that a router receives are useless, but they consume memory / CPU cycles and may cause BGP UPDATEs to be widely distributed
  - Best current practice
    - If an operator uses communities, it should make sure that they are not advertised uselessly to the entire Internet !

Chapter 5: roadmap

- 5.1 BGP Scalability
- 5.2 BGP Stability
  - A first look at BGP stability
  - Path Exploration
  - Stable Path Problem
  - Stable eBGP without Global Coordination
  - iBGP stability : MED-EVIL

Some statistics

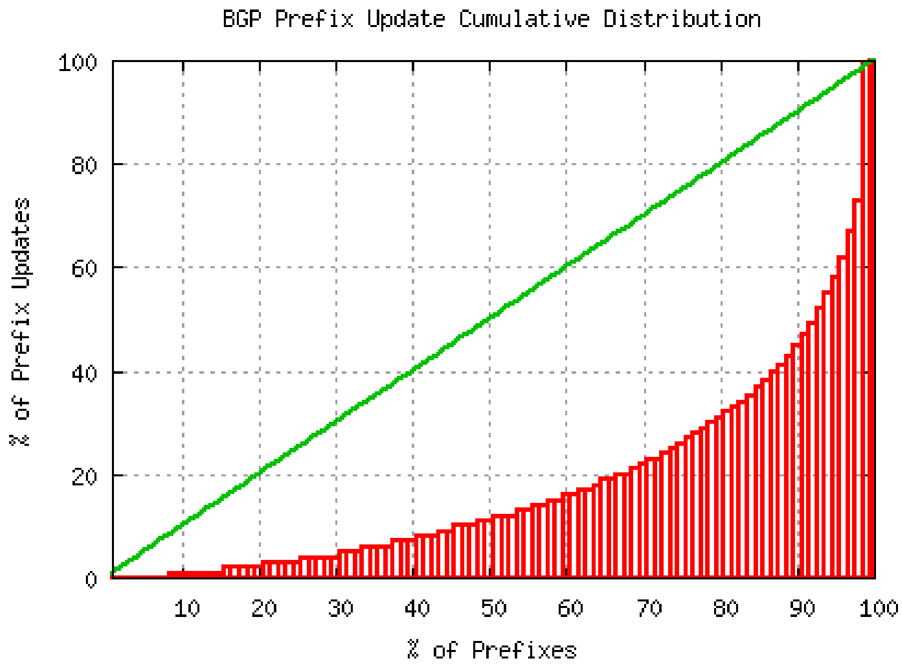
- **Peak number of BGP messages**
  - (on a single session)
  - Looks chatty for an incremental protocol...

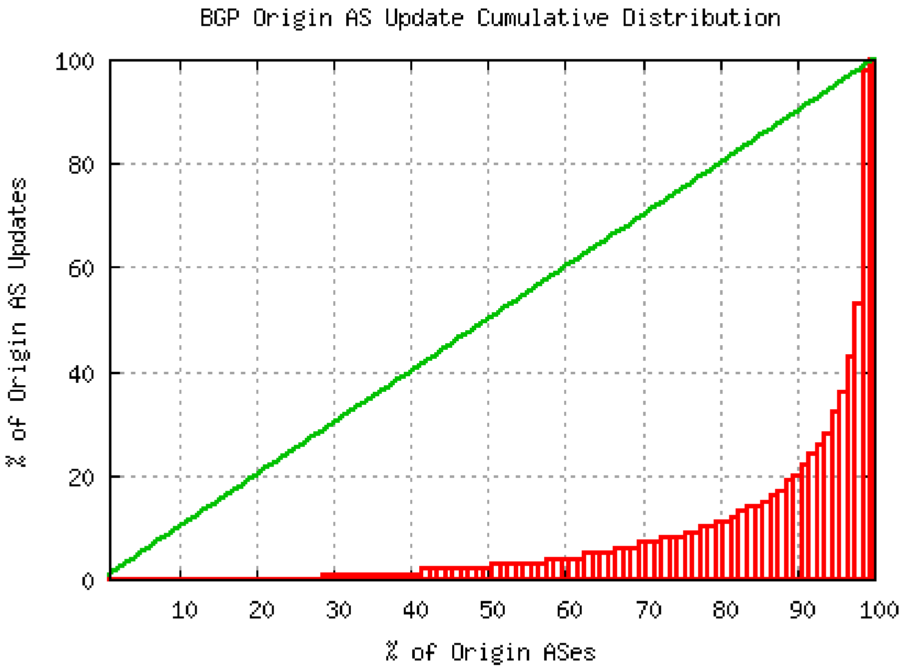


Source: http://www.cidr-report.org (11/08/2010)

Some statistics

- Are all prefixes/ASes equally chatty ?
  - 1% of prefixes → 24% of updates
  - 1% of origin ASes → 44% of updates





Source: http://www.cidr-report.org (11/08/2010)

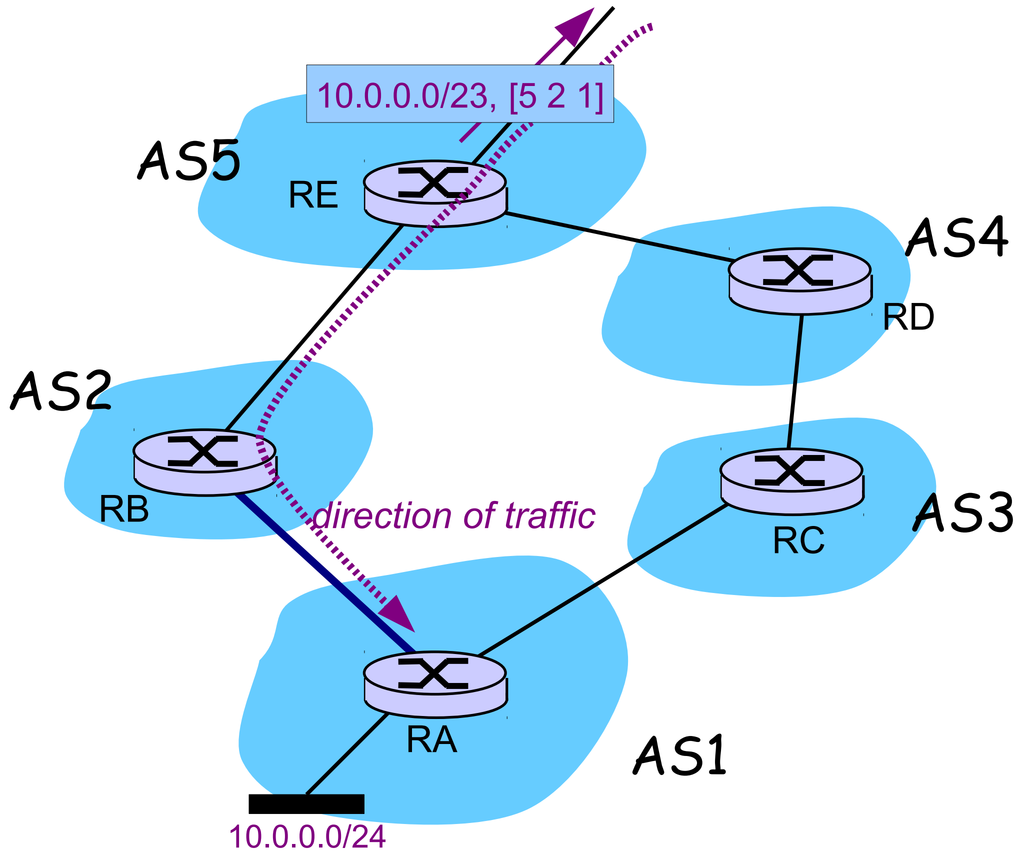
Some statistics

- **Why all these updates ?** [Griffin, ICNP'02]
  - Networks come, networks go
  - There is always a router rebooting somewhere
  - Hardware failure, flaky interface cards, backhoes digging, ...
  - Misconfiguration
  - Path exploration
  - Software bugs
  - IGP instability exported outside AS
  - Secret sauce routing algorithms attempting fancy-dancy tricks
  - Weird policy interactions
  - Gnomes, sprites and fairies. Who knows ?...

This is what dynamic routing has been designed for

Flap flap

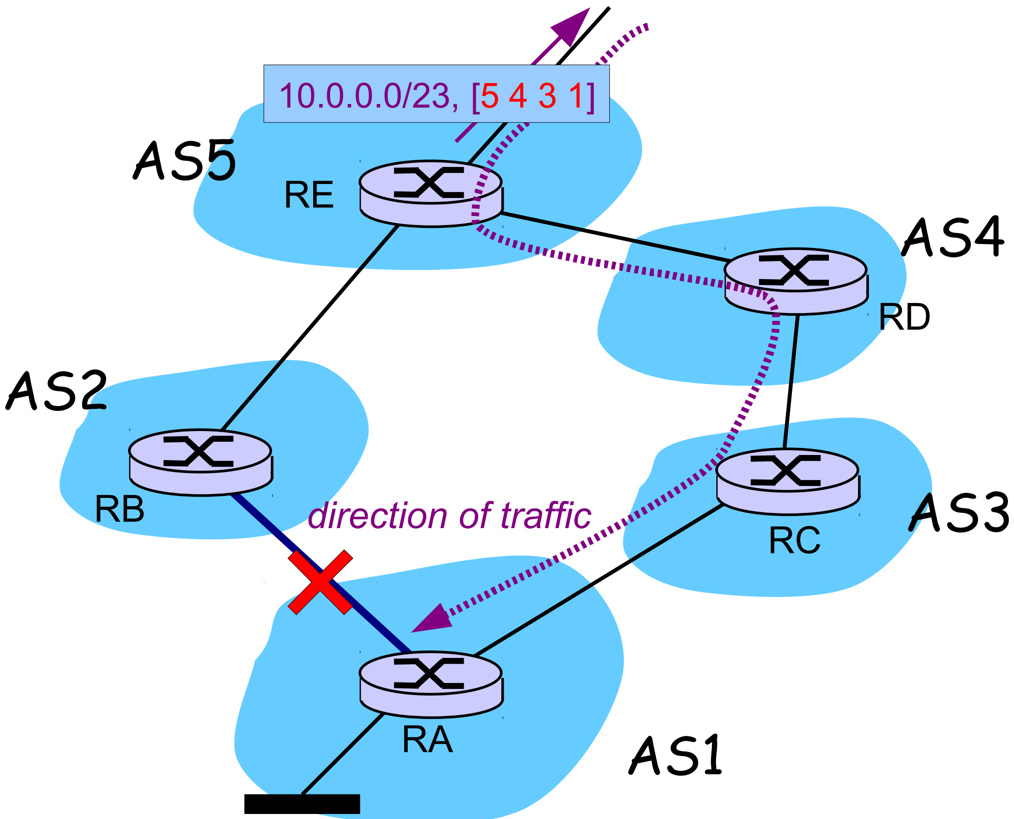
- Flapping interdomain link



Flap flap

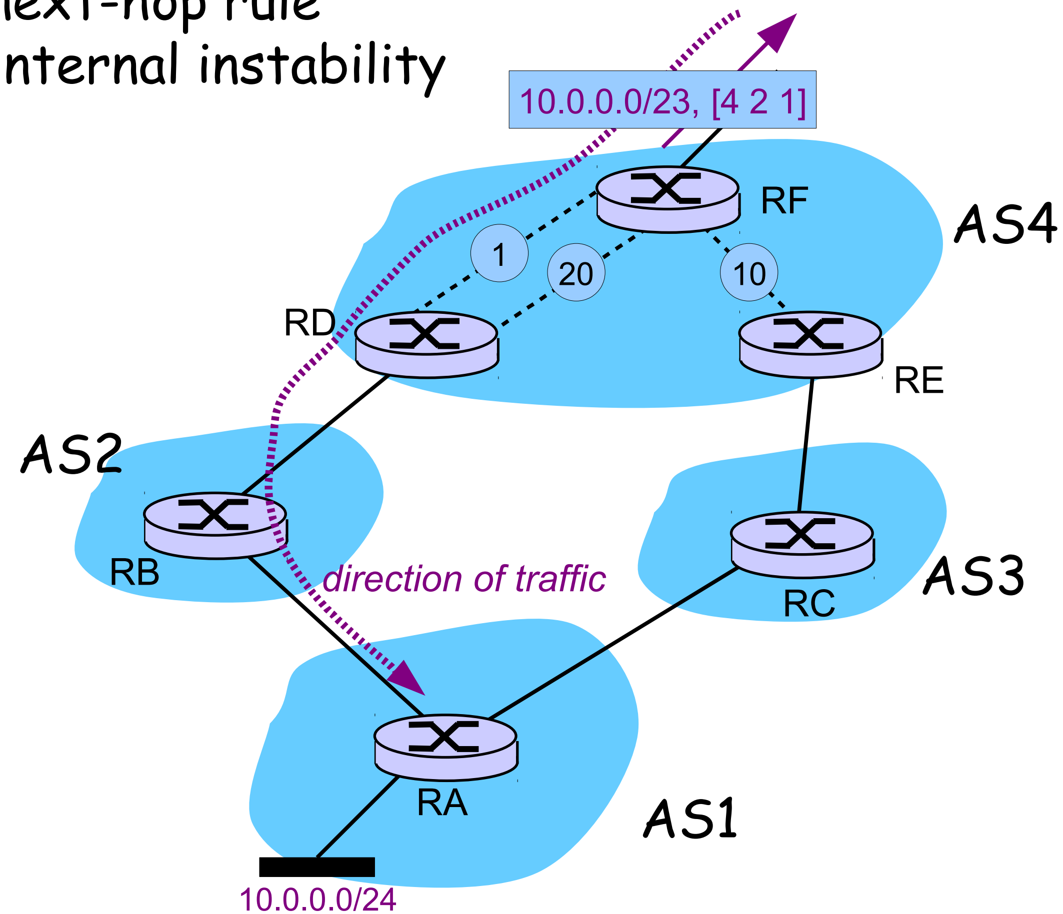
Flap tout le temps et à chaque fois, on advertise tous les AS de l'internet...

- Flapping interdomain link



IGP instability

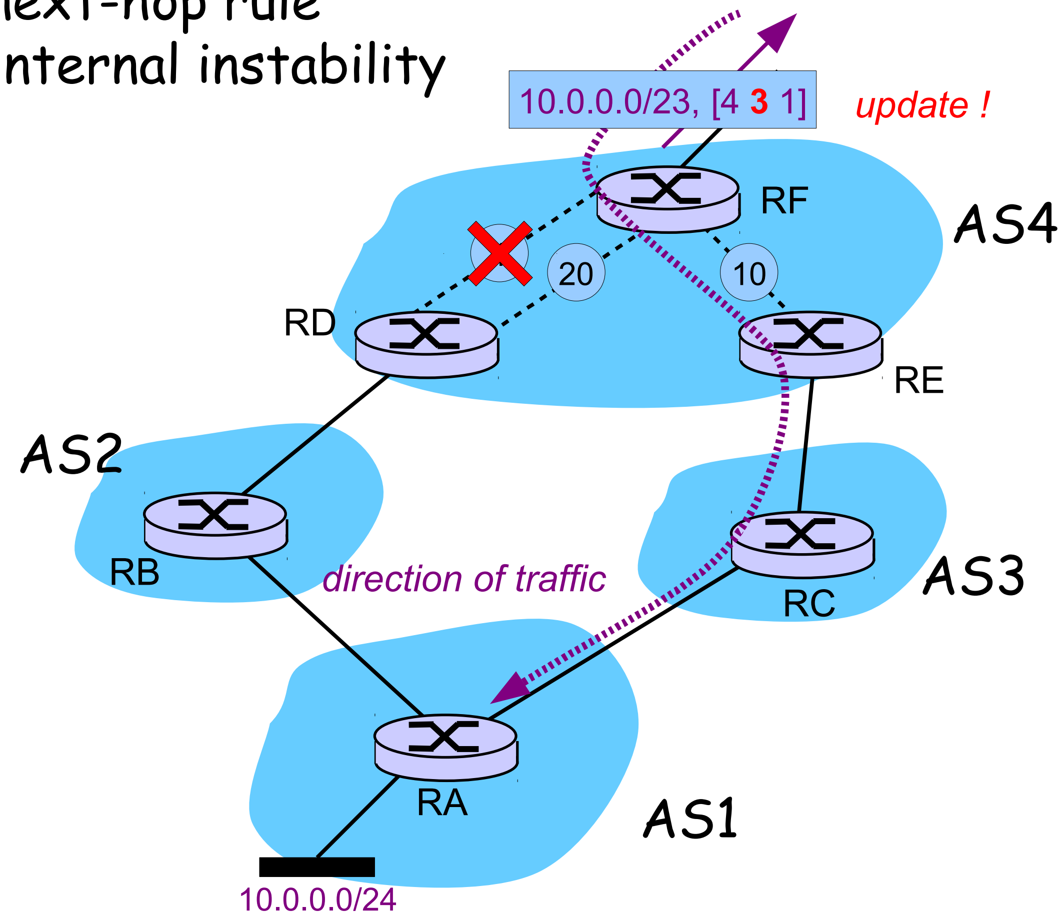
- IGP tie-breaking
  - nearest next-hop rule exports internal instability



IGP instability

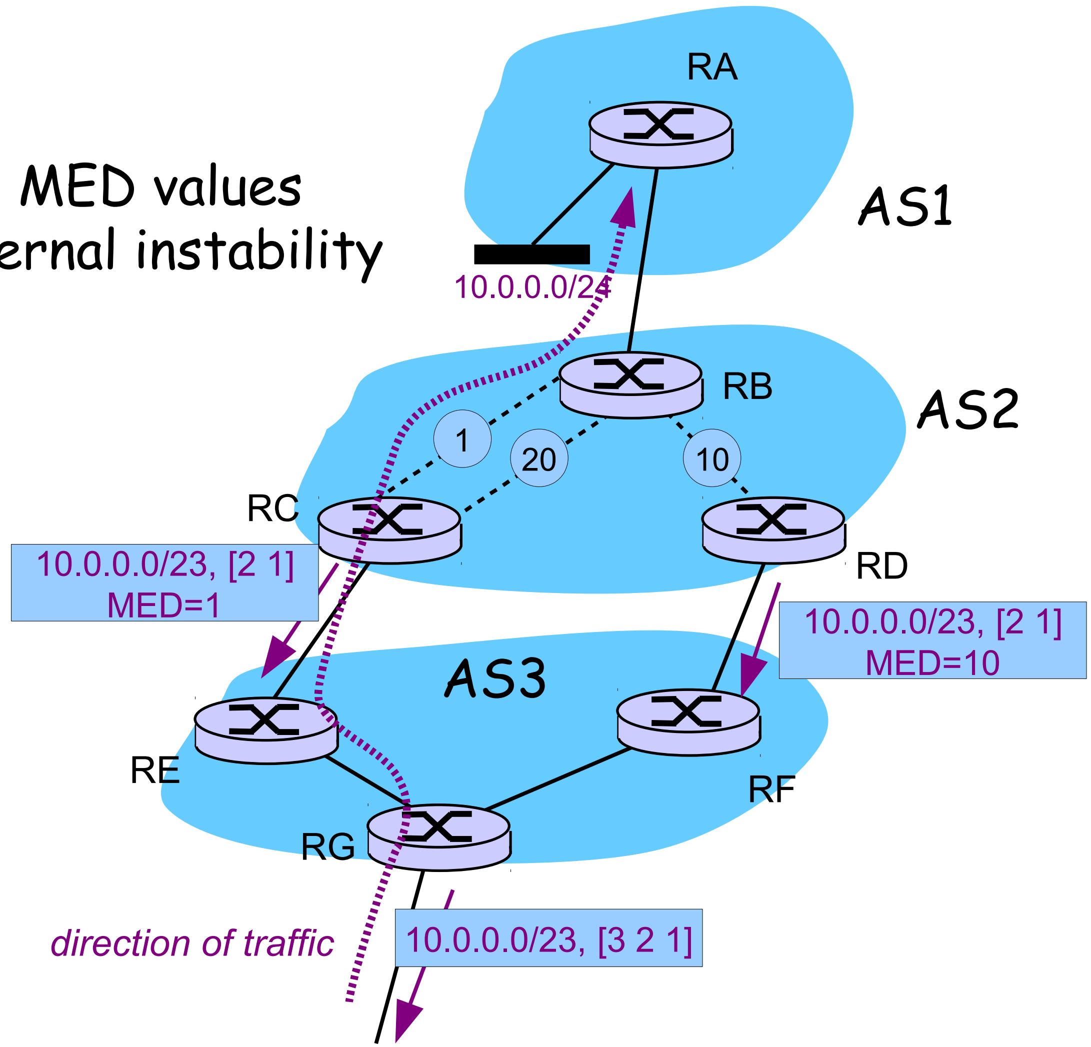
Pareil que pour le flap, on advertise tout le monde à chaque fois...

- **IGP tie-breaking**
  - nearest next-hop rule exports internal instability



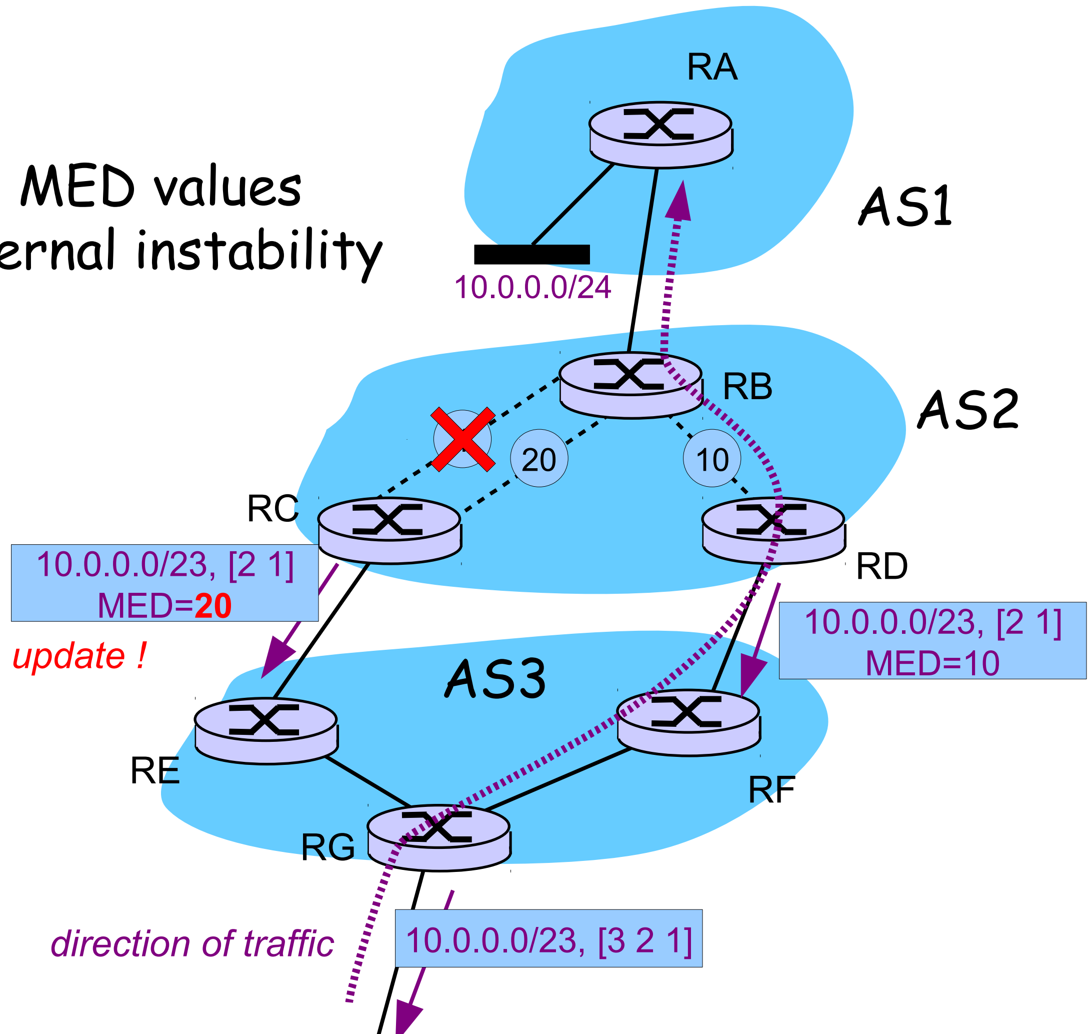
IGP instability

- **MED**
  - IGP-based MED values export internal instability



IGP instability

- **MED**
  - IGP-based MED values export internal instability



Rarement capable de gérer les duplicats

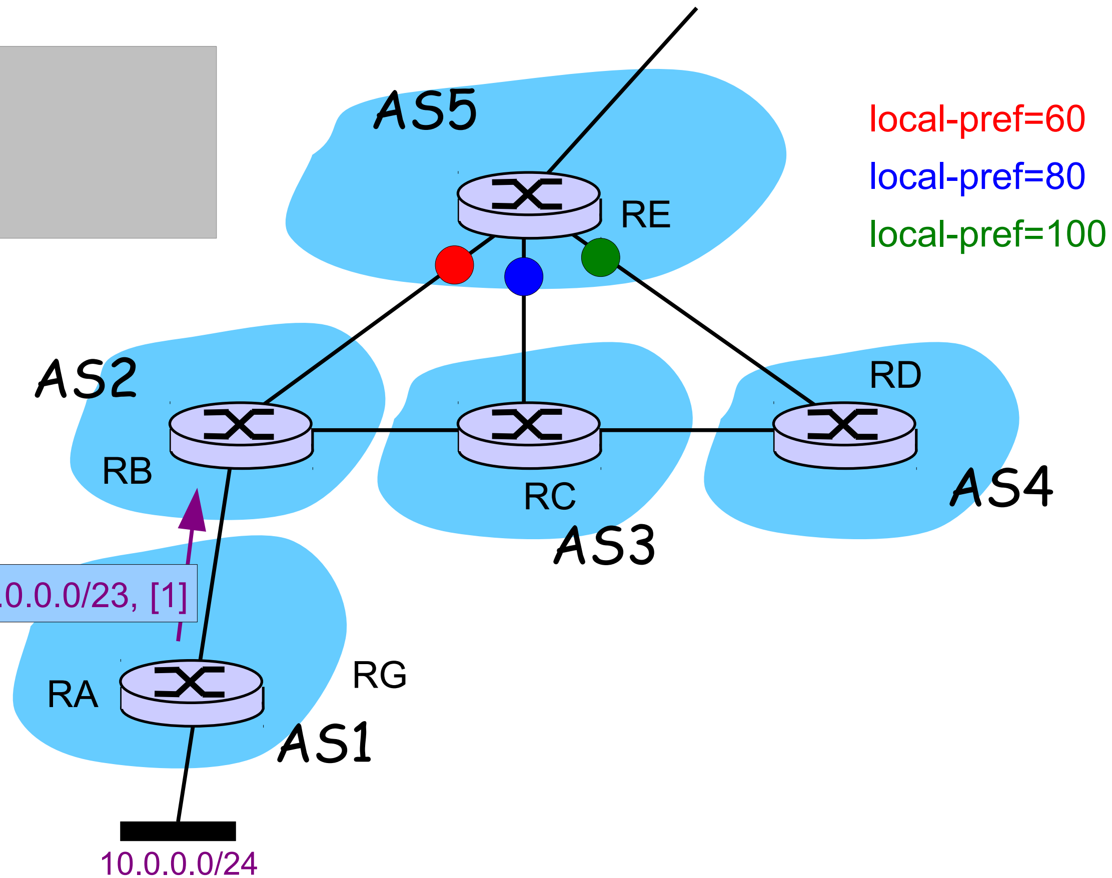
Chapter 5: roadmap

- 5.1 BGP Scalability
- 5.2 BGP Stability
  - A first look at BGP stability
  - Path Exploration
  - Stable Path Problem
  - Stable eBGP without Global Coordination
  - iBGP stability : MED-EVIL

Path Hunting / Exploration

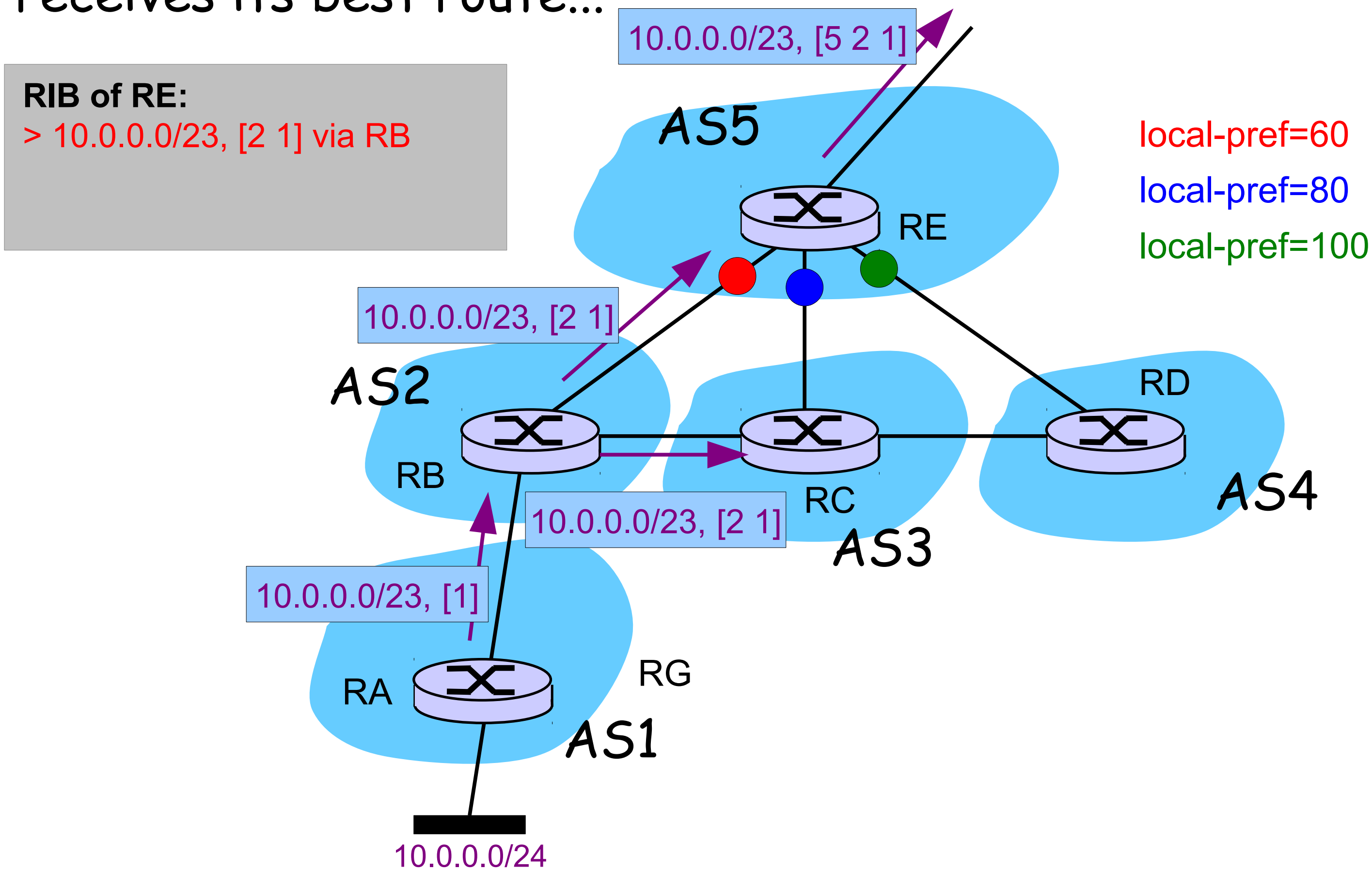
- Due to policies, it can take several seconds before BGP receives its best route...

RIB of RE:
(empty)



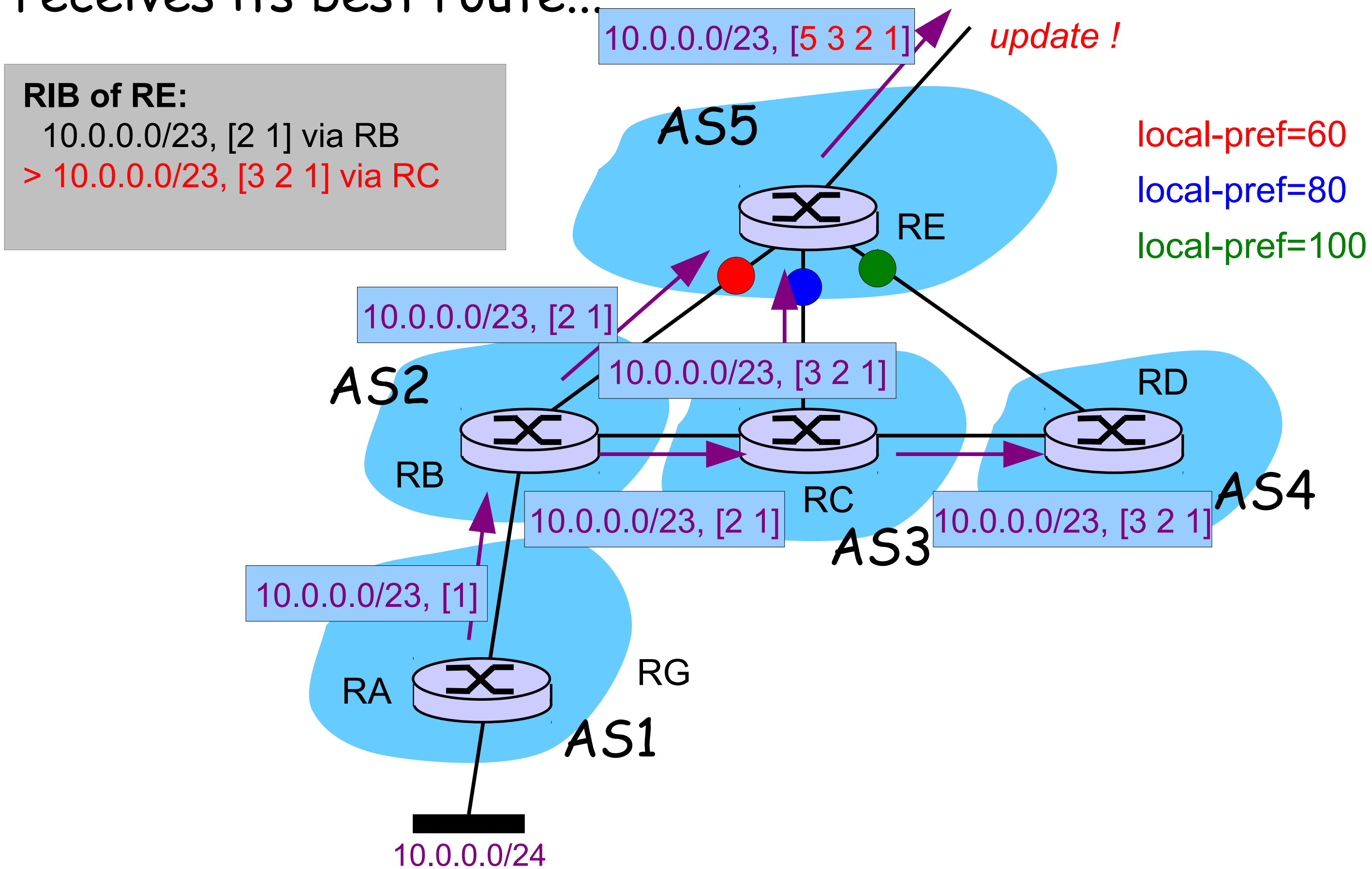
Path Hunting / Exploration

- Due to policies, it can take several seconds before BGP receives its best route...



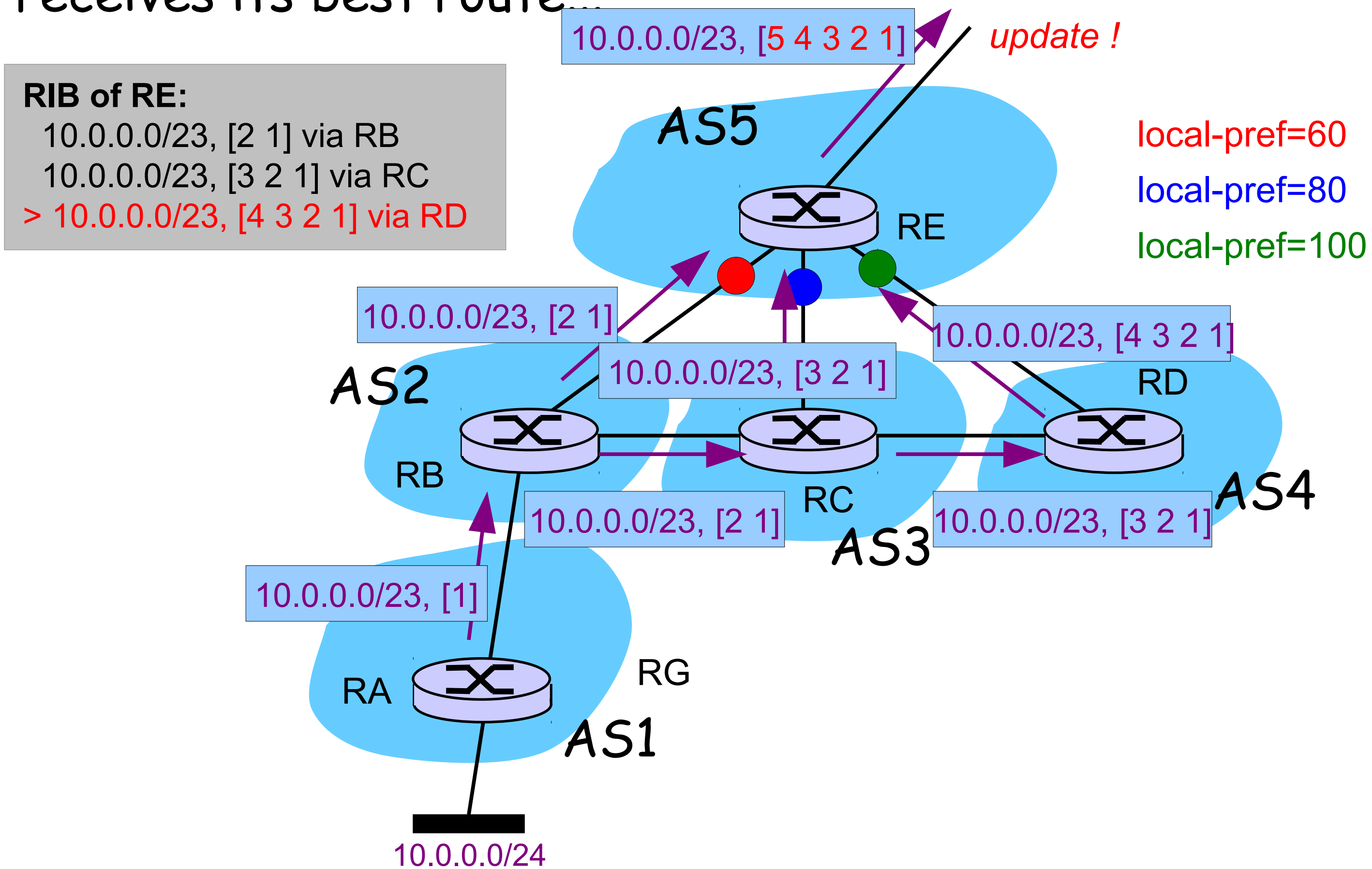
Path Hunting / Exploration

- Due to policies, it can take several seconds before BGP receives its best route...



Path Hunting / Exploration

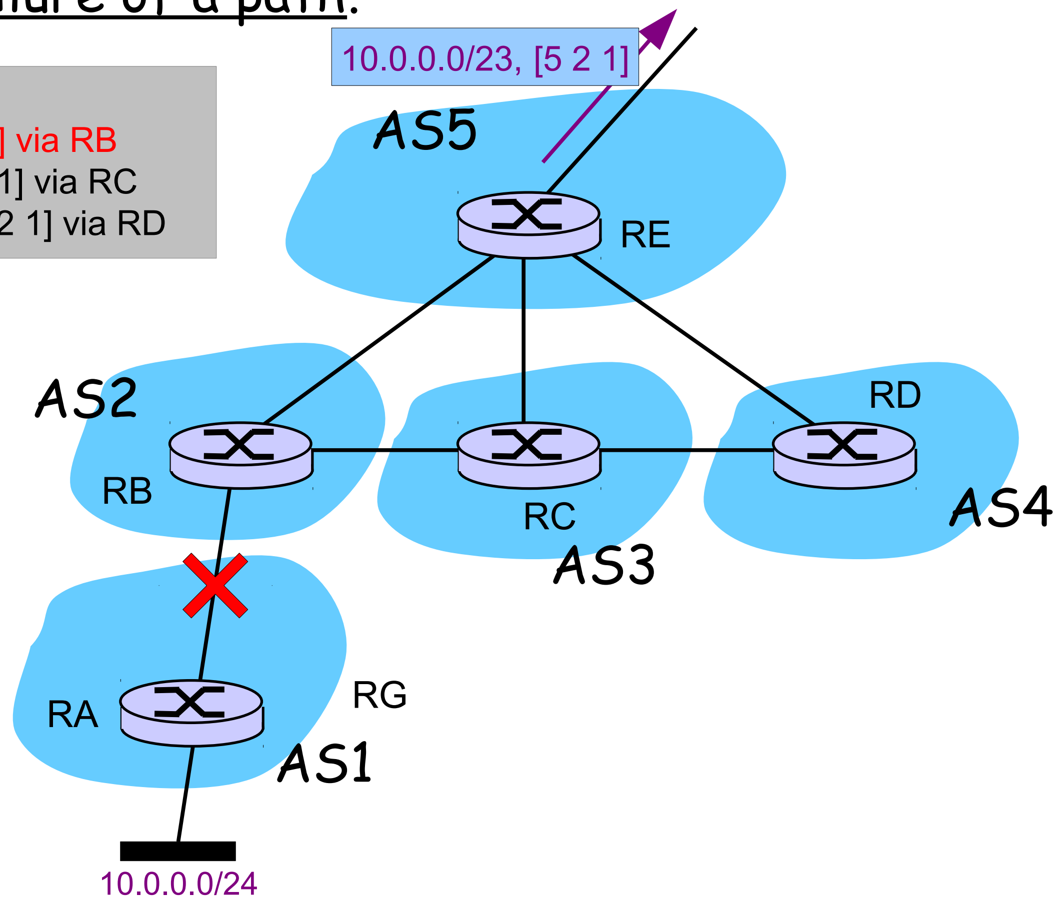
- Due to policies, it can take several seconds before BGP receives its best route...



Path Hunting / Exploration

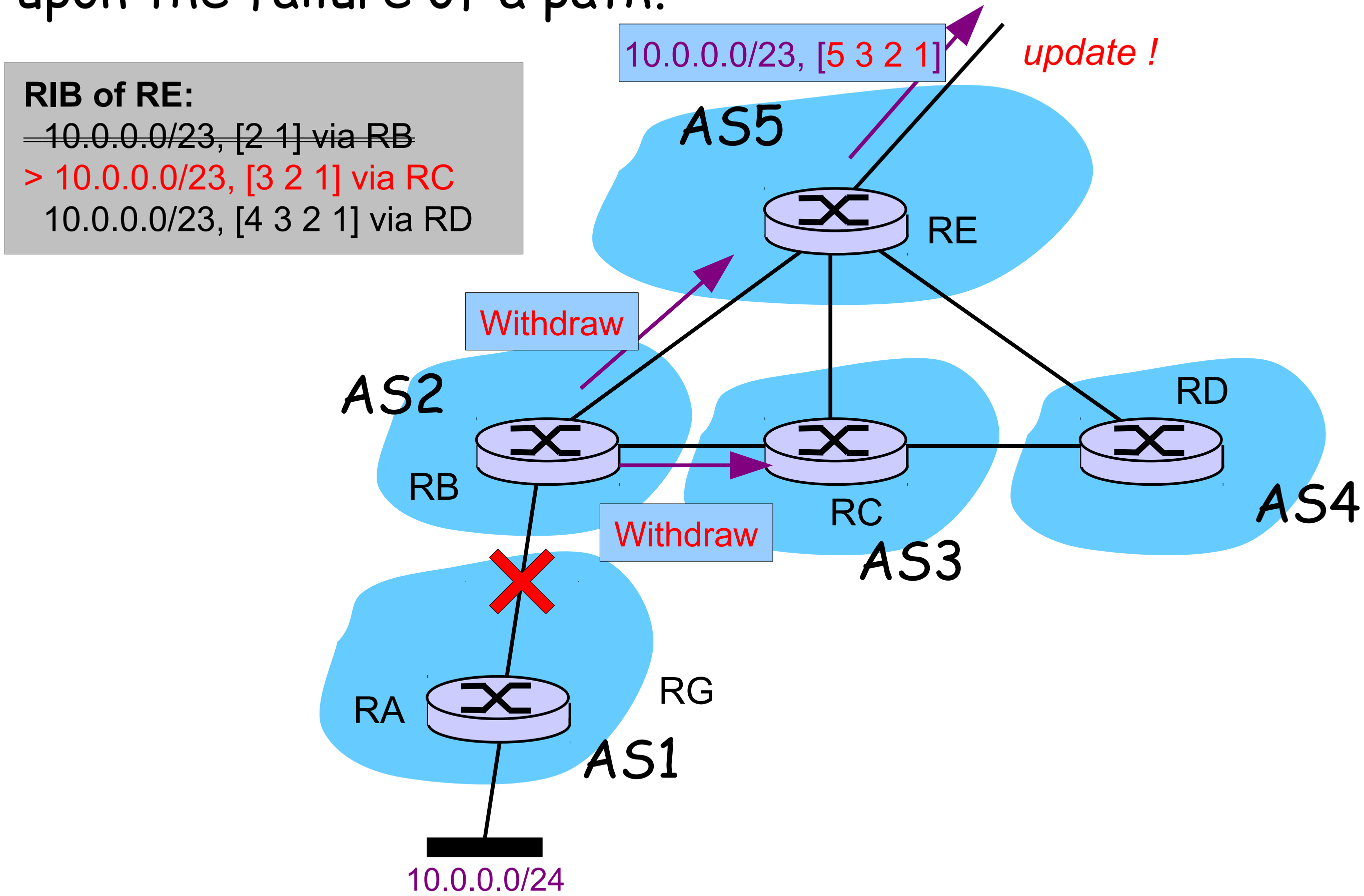
- In the absence of policies path exploration can also occur upon the failure of a path.

RIB of RE:
> 10.0.0.0/23, [2 1] via RB
10.0.0.0/23, [3 2 1] via RC
10.0.0.0/23, [4 3 2 1] via RD



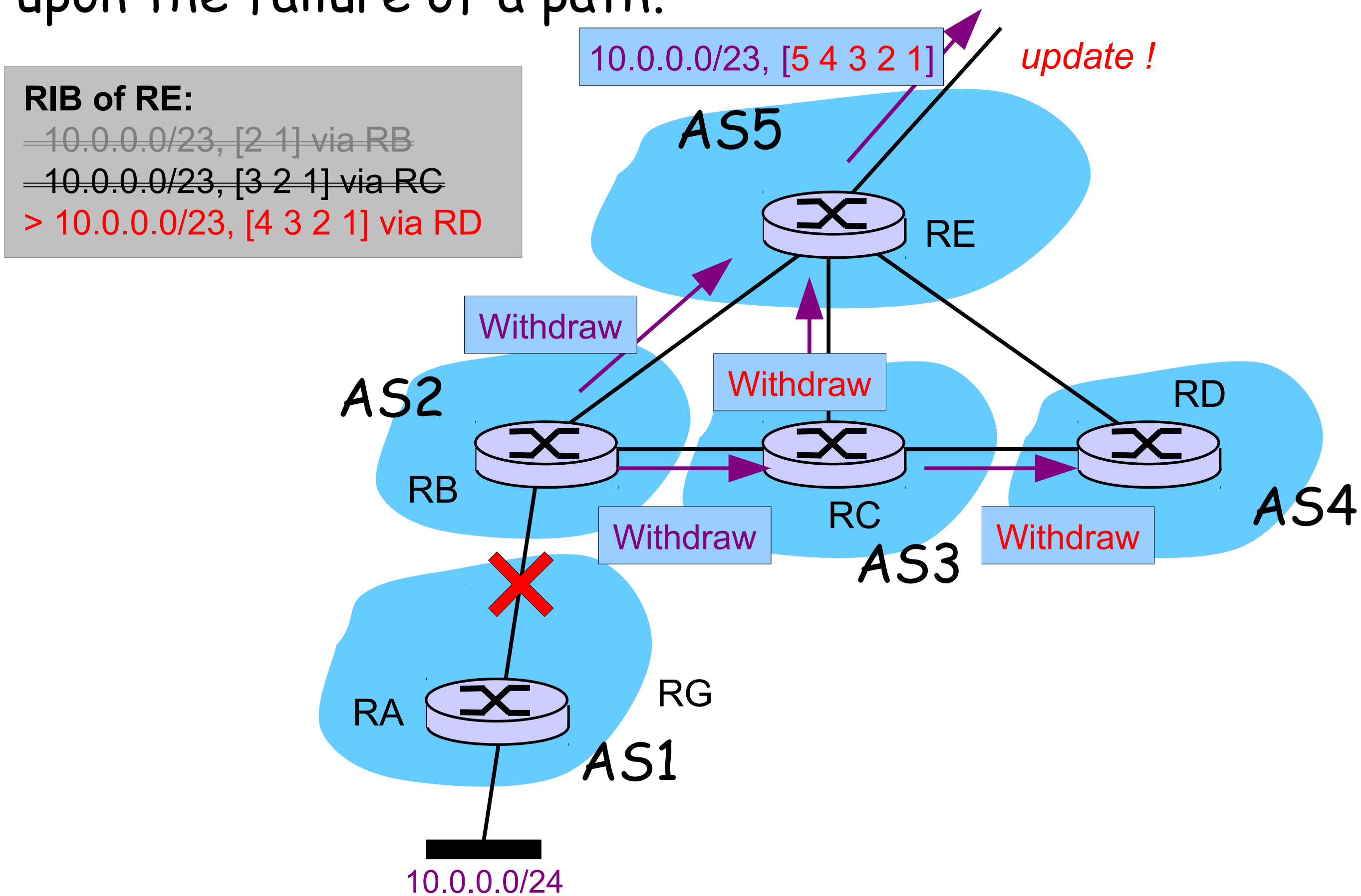
Path Hunting / Exploration

- In the absence of policies path exploration can also occur upon the failure of a path.



Path Hunting / Exploration

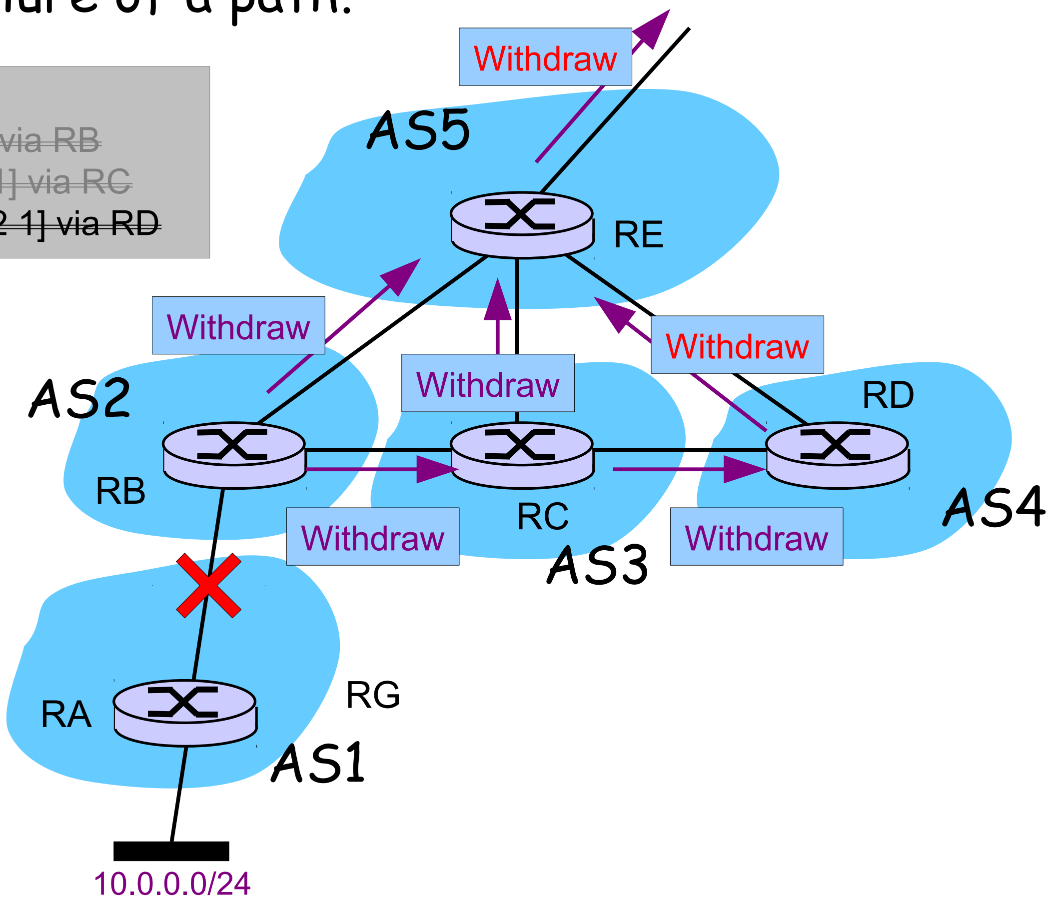
- In the absence of policies path exploration can also occur upon the failure of a path.



Path Hunting / Exploration

- In the absence of policies path exploration can also occur upon the failure of a path.

RIB of RE:
~~10.0.0.0/23, [2 1] via RB~~
~~10.0.0.0/23, [3 2 1] via RC~~
~~10.0.0.0/23, [4 3 2 1] via RD~~



Profile of BGP updates

Type of update		# updates
Announcement of an already announced prefix	w/ longer AS-Path (AA+)	607,093
	w/ shorter AS-Path (AA-)	555,609
	w/ equal-length AS-Path (AA0)	594,029
	w/ same AS-Path, != attributes (AA*)	782,404
	w/ same AS-Path, same attributes (AA)	195,707
Announcement of a withdrawn prefix	w/ longer AS-Path (WA+)	238,141
	w/ shorter AS-Path (WA-)	190,328
	w/ equal-length AS-Path (WA0)	51,780
	w/ same AS-Path, != attributes (WA*)	30,797
	w/ same AS-Path, same attributes (WA)	77,440
Withdrawal of an announced prefix (AW)		627,538
Withdrawal of a withdrawn prefix (WW)		0

Source : The ISP Column, June 2007, Geoff Huston, http://www.potaroo.net/ispcol/2007-06/dampbgp.html

Solutions (?)

- **How to reduce number of updates ?**
  - Avoid sending messages too frequently
    - Two update messages sent by the same router for the same prefix should be spaced by at least the *Min. Route Advertisement Interval* (MRAI)
    - Default value=30 seconds
  - Advantage
    - reduces the number of BGP messages
  - Drawback
    - may delay the propagation of BGP messages and increase the total convergence time

BGP Dampening (RFC2439)

- **Observation**
  - Most routes do not change frequently
  - A small fraction of routes are responsible for most BGP messages
    - Question : can we penalize those unstable routes ?
- **Principle**
  - Associate a penalty counter to each route
    - Increase penalty counter each time route changes
    - Use exponential decay to slowly decrease penalty counter with time
  - Routes with a too large penalty are suppressed

BGP Dampening

- **Parameters**
  - Penalty per message
    - can be specified separately for withdraw, updates, ...
  - Cutoff threshold
    - Route is suppressed when penalty value is above this threshold
  - Reuse threshold
    - Route is re-used when penalty value has fallen under this threshold. Strictly lower than cutoff → hysteresis
  - Halftime
    - used for the exponential decay
  - Maximum suppress time
    - A route cannot be suppressed longer than this time

BGP Dampening

- **Exponential decay**
  - Given a value at time 0 and a decay factor of K

$$V_0$$

  - After t units of time, value is obtained by formula

$$V_t = \frac{V_0}{K^t}$$

  - Here, the decay factor is specified as 2 per "half-time". Therefore, after n half-times, the value is decreased 2^n, i.e. obtained by

$$V_{n.HT} = \frac{V_0}{2^n}$$

BGP Dampening

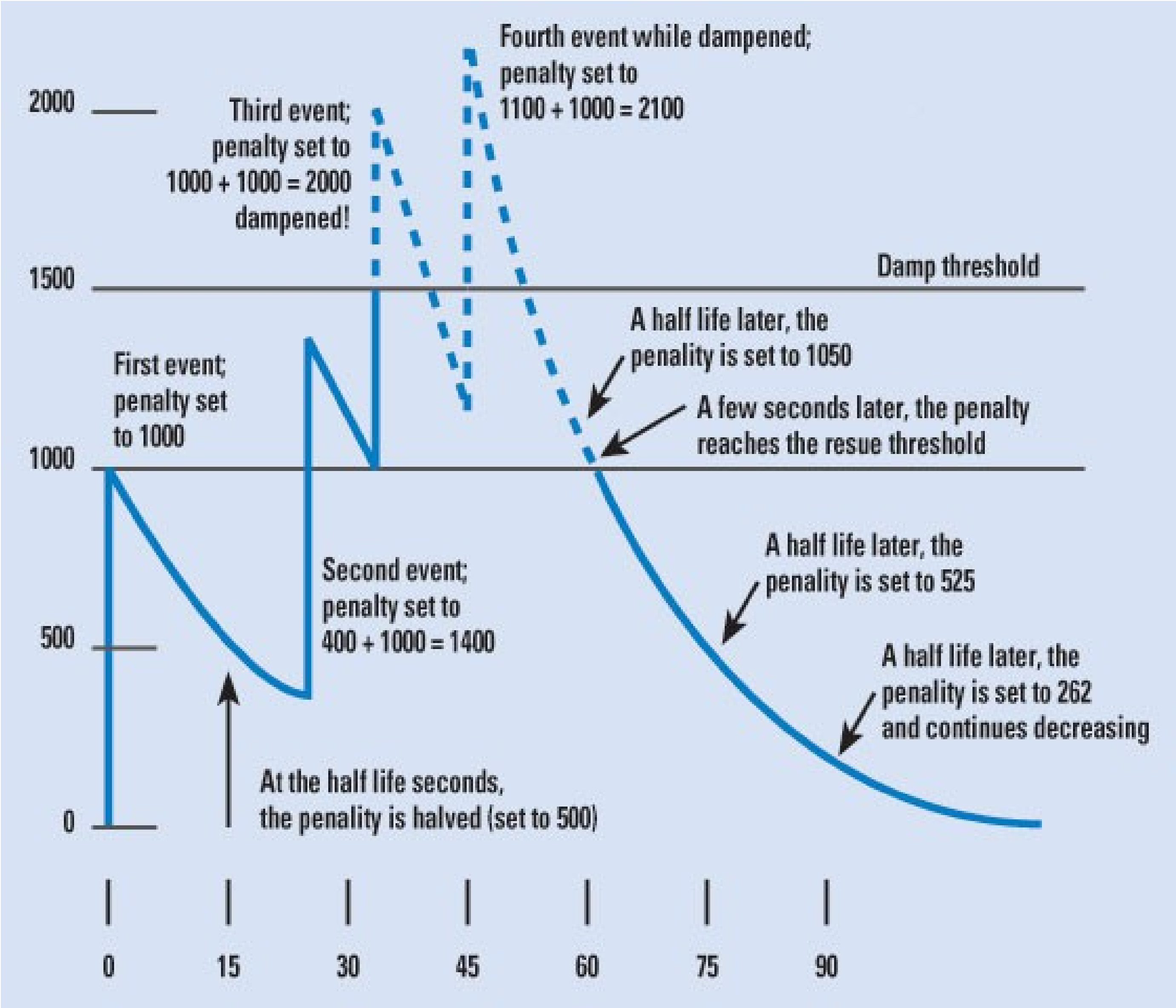
- **Algorithms**

Pas vu au cours

```
on_message_received(m):
    penalty[m.prefix]+= PENALTY_PER_MESSAGE
    if (penalty[m.prefix] > CUTOFF_THRESHOLD):
        dampen(m.prefix)
```

```
on_timer_expire():    # every HALF_TIME
    for prefix in dampened prefixes:
        penalty[prefix]/= 2
        if (penalty[prefix] < REUSE_THRESHOLD):
            reuse(prefix)
```

BGP Dampening



Source : **High Availability in Routing**, Russ White,
The Internet Protocol Journal,
Volume 7, Number 1, March 2004

BGP Dampening

- **Issues**
  - What are the best configuration values to use ?
    - No definite scientific answer today
  - Today, the use of BGP dampening is discouraged
    - It has adverse effects : can slow down convergence even for valid prefixes for which there are frequent updates due, for example, to path exploration

Chapter 5: roadmap

- 5.1 BGP Scalability
- 5.2 BGP Stability
  - A first look at BGP stability
  - Path Exploration
  - Stable Path Problem
  - Stable eBGP without Global Coordination
  - iBGP stability : MED-EVIL

Stable Path Problem (SPP)

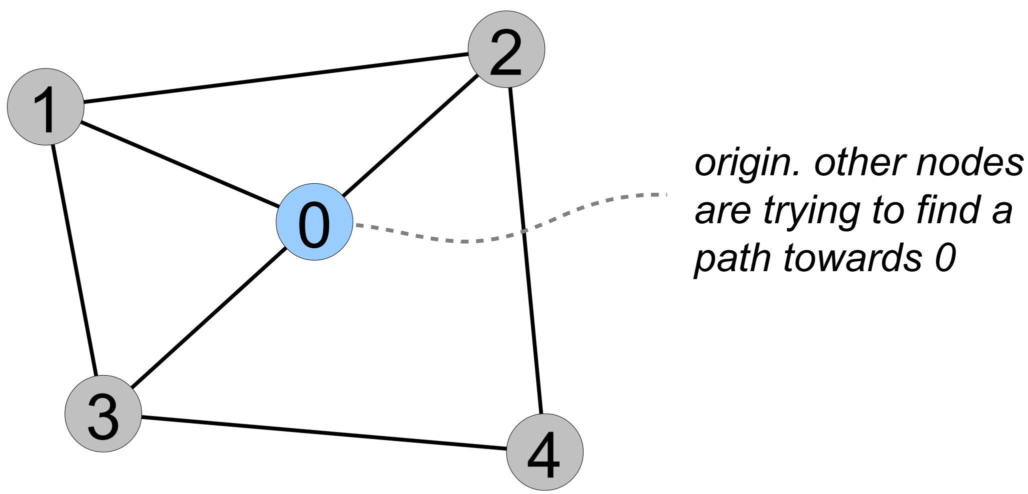
- **Introduction**
  - Reasoning about the correctness of a BGP routing system is still an open issue today
  - Interesting questions are
    - Does a BGP system admit a solution ?
    - How long will last the convergence ?
    - ...
  - The **Stable Path Problem** is a first attempt at formalizing how BGP works.
    - [Stable Paths Problem and Interdomain Routing, T. Griffin et al, IEEE/ACM Transactions on Networking, Vol.10, No.2, April 2002]

Stable Path Problem

- **Definitions**
  - Let $G=(V, E)$ be a simple, undirected graph. $V=\{0, 1, ..., n\}$ is the set of nodes and E is the set of edges.
  - For any node u, $peers(u)=\{w \mid \{u,w\} \in E\}$ is the set of peers for u.
  - Convention : node 0 is the origin
    - (destination to which all other nodes attempt to establish a path)

Stable Path Problem

- **Example**
  - $V=\{0, 1, 2, 3, 4\}$
  - $E=\{(0,1), (0,2), (0,3), (1,2), (1,3), (2,4), (3,4)\}$



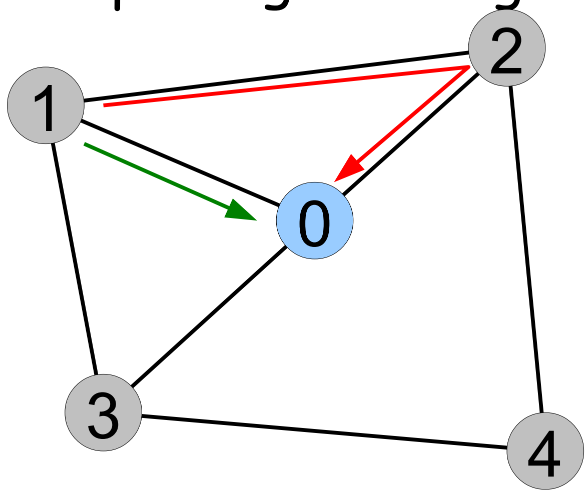
  - $peers(1)=\{0, 2, 3\}$

Stable Path Problem

- **Definitions: Paths**
  - A path P in G is a sequence of nodes $(v_k\ v_{k-1} \dots v_1\ v_0)$, $k \geq 0$ such that for each i *such that* $\mathrm{k} \geq i > 0$, $\{v_i, v_{i-1}\} \in E$. The empty path is denoted by ε
  - Each non empty path has a direction from its first node v_k to its last node v_0
  - For each node v, P^v denotes the set of permitted paths from v to 0.
  - If $P=(v\ v_k \dots v_0) \in P^v$, then the node v_k is called the next-hop of path P.
  - Let P^* be the union of all sets P^v

Stable Path Problem

- **Example**
  - P_1=(1 2 0) and P_2=(1 0) are paths from 1 to 0
  - P^1 could be limited to $\{P_1, P_2\}$, meaning that node 1 does not accept to go through node 3.



  - Note: (1 4 0) is not a path in G.

Stable Path Problem

- **Definition: ranking**
  - A path ranking for a node v is a function $\lambda^v : P^v \rightarrow R^+$ which represents how node v ranks its permitted paths.
  - If $P_1, P_2 \in P^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be preferred over P_1.
  - Let $\Lambda=\{\lambda^v \mid v \text{ in } V-\{0\}\}$

Stable Path Problem

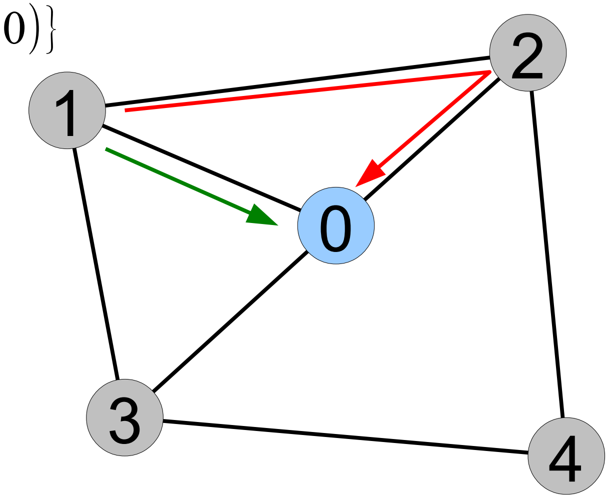
- **Example**
  - $\lambda^1(P_1) > \lambda^1(P_2)$, or node 1 prefers path P_1 over path P_2

$P^1 = \{(1\,2\,0), (1\,0)\}$

$\mathbf{P_1 = (1\ 2\ 0)}$

$P_2 = (1\ 0)$

convention in lecture:
permitted paths appear next to each node.
+ highest ranked paths appear on top



- Note : λ^1 is not defined for $(1\ 3\ 0)$. Why ?

Stable Path Problem

Lambda est une fonction qui va ordonner les chemins. (Ranking)

- **Definition : SPP instance**
  - An instance of the SPP problem, $S=(G, P^*, \Lambda)$, is a graph together with the permitted paths and the ranking functions at each node.
  - It is assumed that $P^0=\{(0)\}$
  - Assumptions for all $v \in V-\{0\}$,
    - $\varepsilon \in P^v$ *(empty is permitted)*
    - $\lambda^v(\varepsilon)=0$ and $\lambda^v(P)>0$ for all $P \neq \varepsilon$ *(empty ranked lowest)*
    - if $P_1, P_2 \in P^v$ and $\lambda^v(P_1)=\lambda^v(P_2)$, then $\exists\, u \in V$: $P_1=(v\ u)P_1'$ and $P_2=(v\ u)P_2'$ *(strictness [1])*
    - if $P \in P^v$, then P has no repeated nodes *(simplicity)*

(1) *Strictness i.e. same rank implies same next-hop*

Stable Path Problem

- **Definition : Path Assignment**
  - A path assignment is a function π that maps each node $v \in V$ to a path $\pi(v) \in P^v$.
  - We interpret $\pi(v)=\varepsilon$ as v is not assigned a path to the origin.
  - Given W a subset of V that contains node 0, a partial path assignment π is a path assignment such that for every v in W, $\pi(v)$ only contains nodes in W.

Stable Path Problem

- **Definition : choices of a node**
  - The set of possible paths for a node given a path assignment π is the set $choices(\pi, v)$ defined as follows

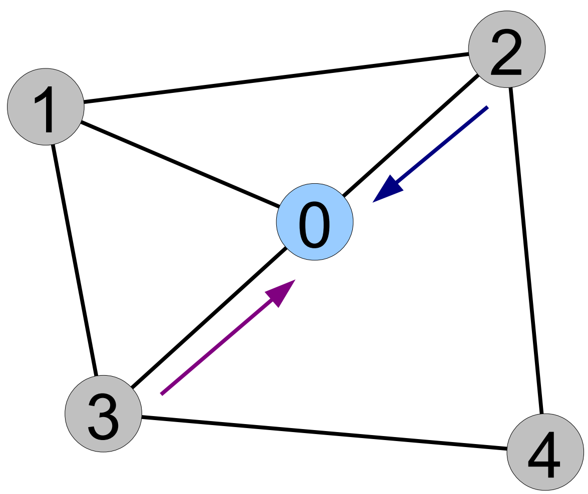
$$choices(\pi, 0) = \{(0)\}$$

$$choices(\pi, v) = \{(v\,u)\pi(u) : (v, u) \in E\} \cap P^v$$

Stable Path Problem

- **Example**
  - $\pi(1)=\varepsilon$; $\pi(2)=(2\ 0)$; $\pi(3)=(3\ 0)$; $\pi(4)=\varepsilon$
    $\rightarrow$ $choices(\pi, 1)=\{(1\ 2\ 0), (1\ 3\ 0)\}$

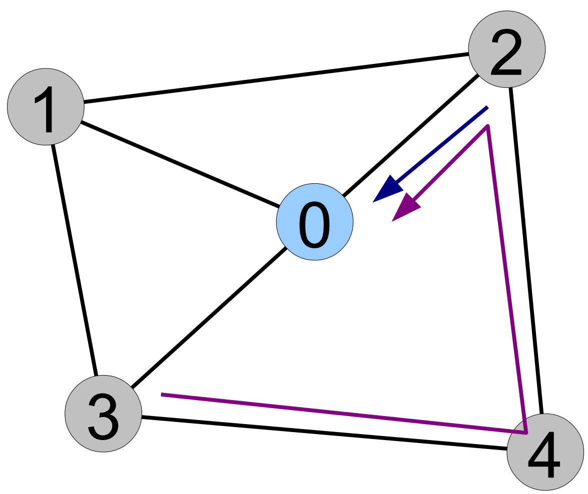
$P^1=\{(1\,2\,0),(1\,3\,0)\}$



Stable Path Problem

- **Example**
  - $\pi(1)=\varepsilon$; $\pi(2)=(2\ 0)$; $\pi(3)=(3\ 4\ 2\ 0)$; $\pi(4)=(4\ 2\ 0)$
    $\rightarrow$ $choices(\pi, 1)=\{(1\ 2\ 0)\}$

$P^1=\{(1\ 2\ 0), (1\ 3\ 0)\}$



  - ... indeed, $(1\ 3\ 4\ 2\ 0)$ is not in P^1

Stable Path Problem

- **Best path**
  - Let W be a subset of P^v
    - (for example the possible *choices* based on a path assignment π).
  - The best path in W is defined to be

$$best(\emptyset, u) = \epsilon$$

$$best(W, u) = P \in W \quad such\ that \quad \lambda^u(P)\ is\ maximal$$

Processus de décision BGP

Stable Path Problem

- **Stability**
  - A path assignment π is stable at node u if

    $$\pi(u) = best(choices(\pi, u), u)$$

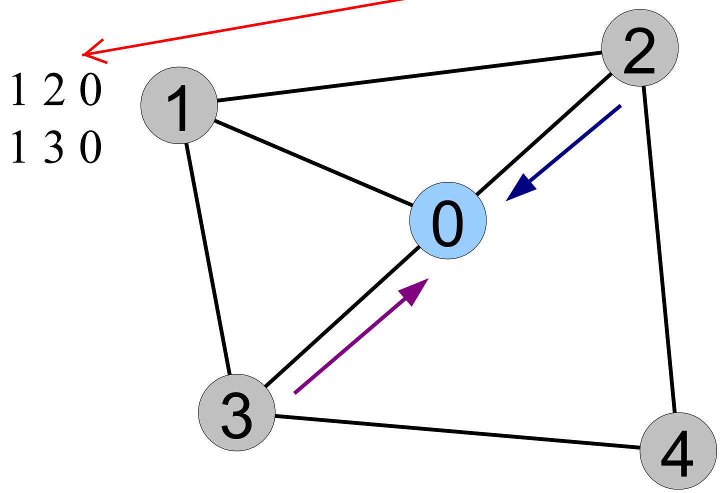
    which means u is assigned the best path among its possible choices.

  - A path assignment π is stable if it is stable at each node u in V

Stable Path Problem

- Example
  - $\pi(1)=(1\ 2\ 0)\ ;\ \pi(2)=(2\ 0)\ ;\ \pi(3)=(3\ 0)\ ;\ \pi(4)=\varepsilon$ is a stable path assignment at node 1

meilleur ranking = premier path indiqué = 1 2 0



car le chemin pour 1 n'est pas le chemin avec le meilleur ranking

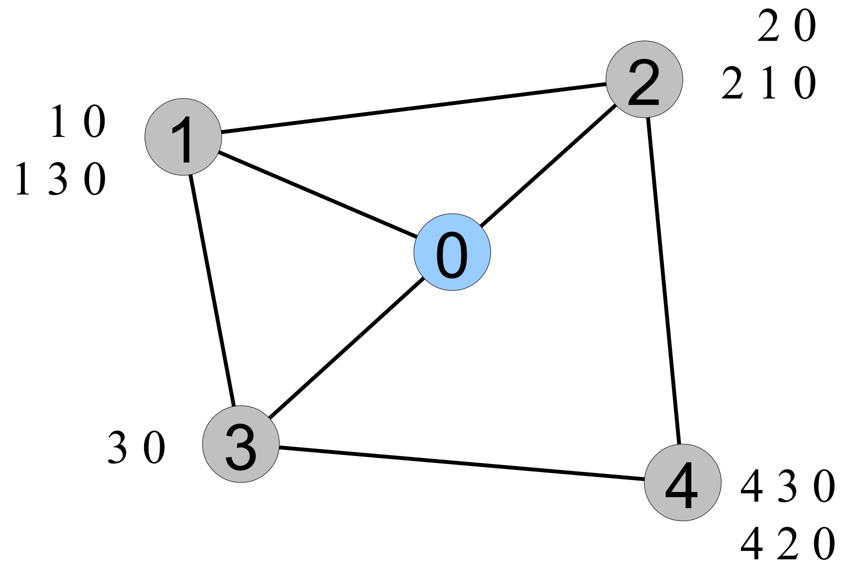
  - $\pi(1)=(1\ 3\ 0)\ ;\ \pi(2)=(2\ 0)\ ;\ \pi(3)=(3\ 0)\ ;\ \pi(4)=\varepsilon$ is not !

Stable Path Problem

- **Solvability**
  - An SPP instance $S=(G, P^*, \Lambda)$ is solvable if there is a stable path assignment for S.
  - Such a stable path assignment is also called a solution for S.
  - If S admits no stable path assignment, S is unsolvable.

Stable Path Problem

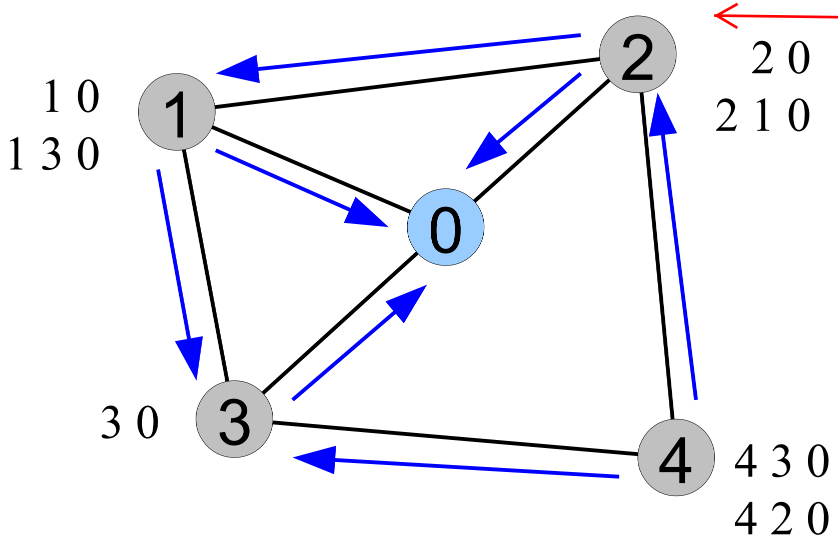
- **Example: SHORTEST 1**
  - How many complete path assignments are **possible** ?



Stable Path Problem

- **Example: SHORTEST 1**
  - Let's have a look at the directed *dependency graph* $G_D=(V, E_D)$ of nodes. A node x depends on another node y if P^x permits a path where y is the next-hop.

Le choix du noeud 2 dépend du choix du noeud 1 => flèche

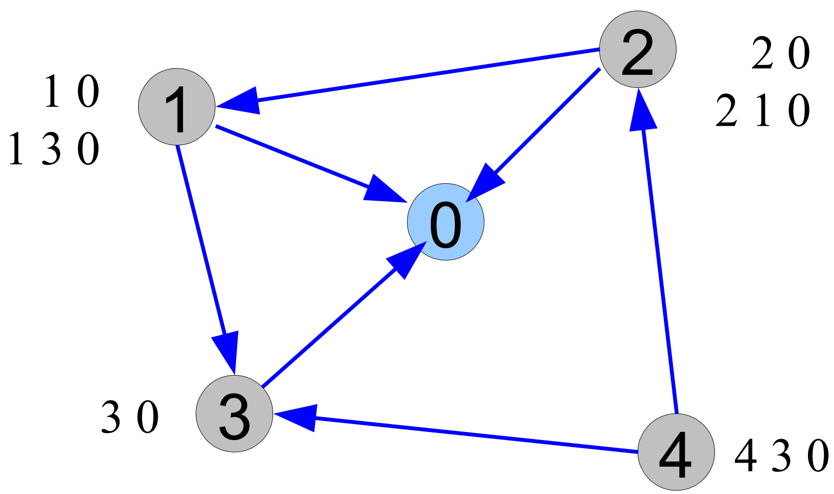


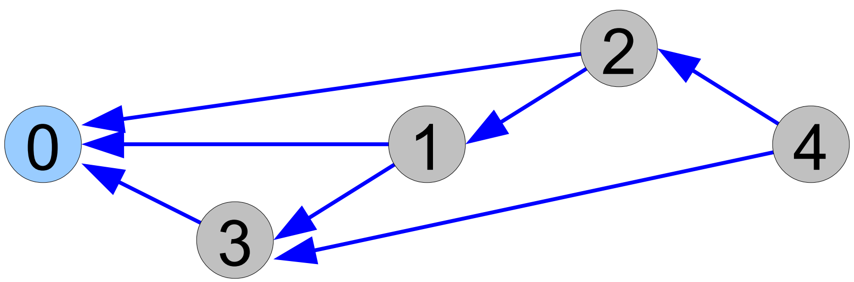
$$P^1=\{(1\,0),(1\,3\,0)\}\Rightarrow E_D\supset\{(1,0),(1,3)\}$$
$$P^2=\{(2\,0),(2\,1\,0)\}\Rightarrow E_D\supset\{(2,0),(2,1)\}$$
$$\ldots$$

Si le chemin 1 0 est choisi pour 1, alors le chemin 2 1 0 de 2 sera faisable, tandis que si le chemin 1 3 0 est choisi dans 1, 2 1 0 n'est pas faisable dans 2. => dépendance.

Stable Path Problem

- **Example: SHORTEST 1**
  - G_D for SHORTEST 1 is a DAG. We can thus compute a *topological sort* of G_D.

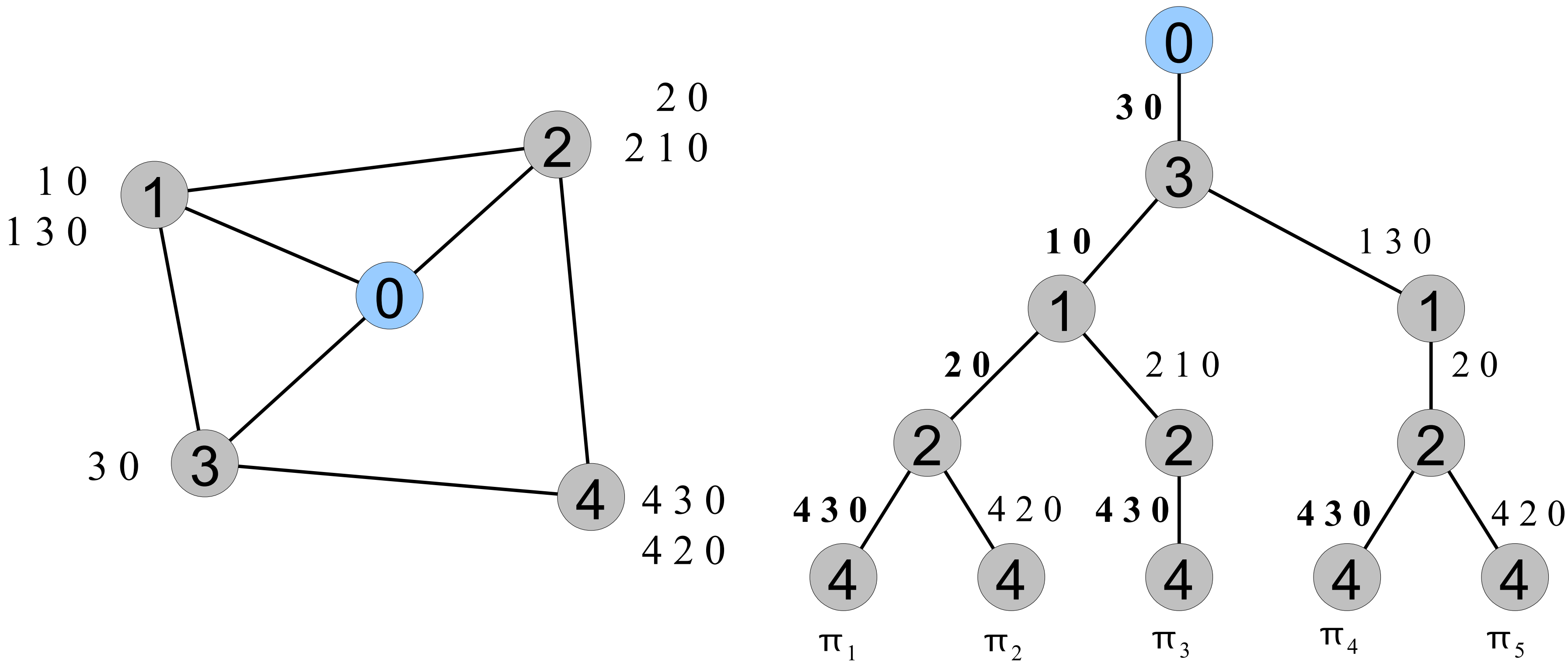




The following linear ordering of nodes in V is a topological sort of G_D: 0, 3, 1, 2, 4

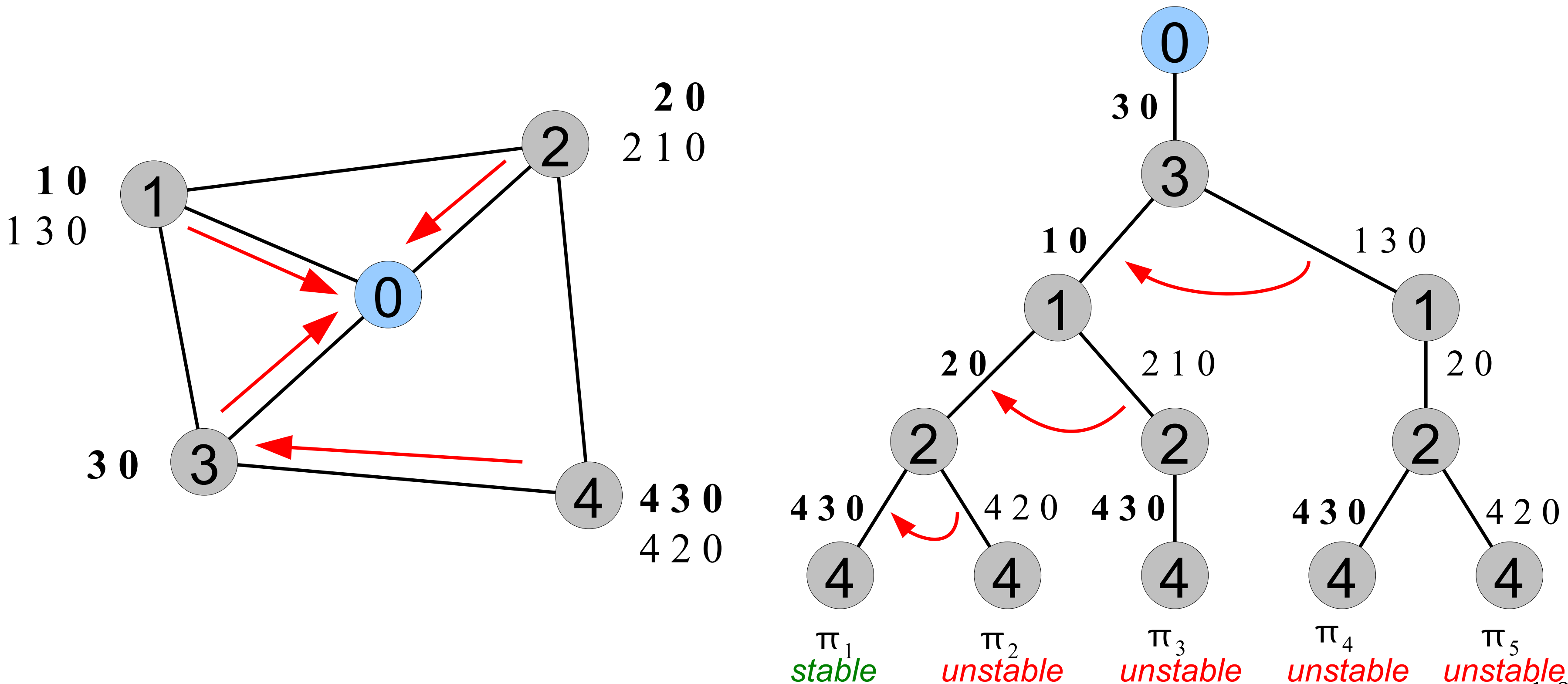
Stable Path Problem

- **Example: SHORTEST 1**
  - How many complete path assignments are **possible** ?



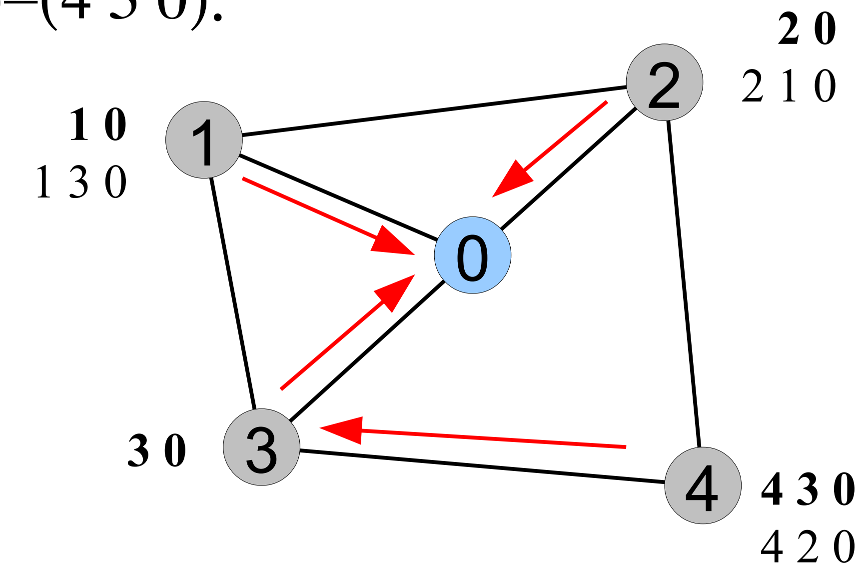
Stable Path Problem

- **Example: SHORTEST 1**
  - How many complete path assignments are **stable** ?



Stable Path Problem

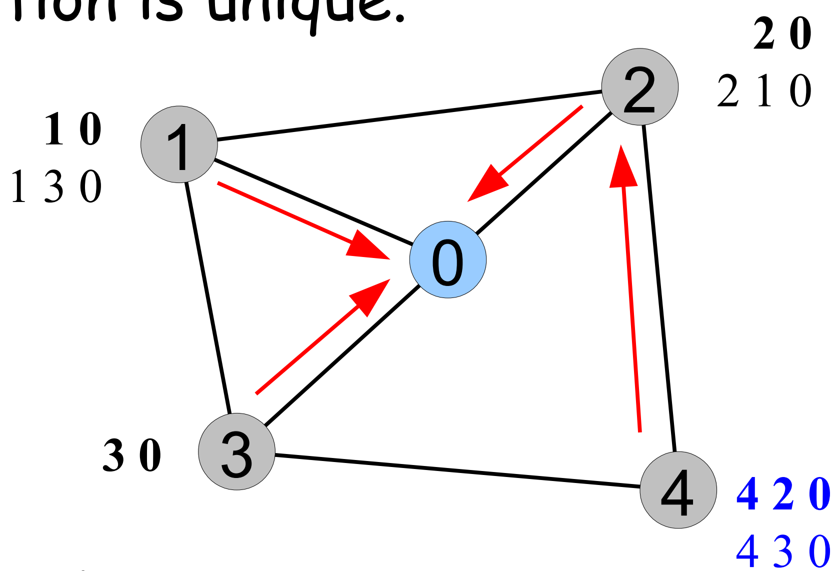
- **Example: SHORTEST 1**
  - This instance is solvable.
  - The solution is $\pi(1)=(1\ 0)$; $\pi(2)=(2\ 0)$; $\pi(3)=(3\ 0)$; $\pi(4)=(4\ 3\ 0)$.



  - The solution is **unique**.

Stable Path Problem

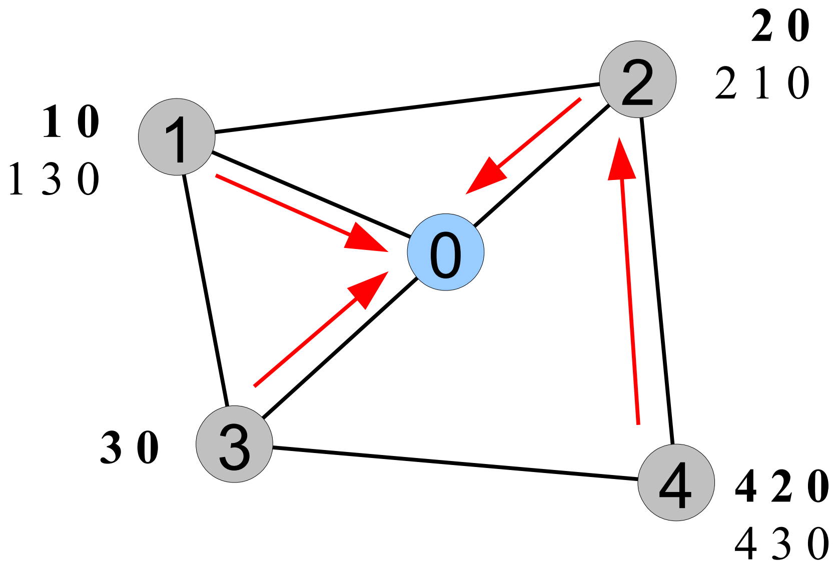
- **Example: SHORTEST 2**
  - Same graph as SHORTEST 1 ; different ranking for node 4.
  - Solution is unique.

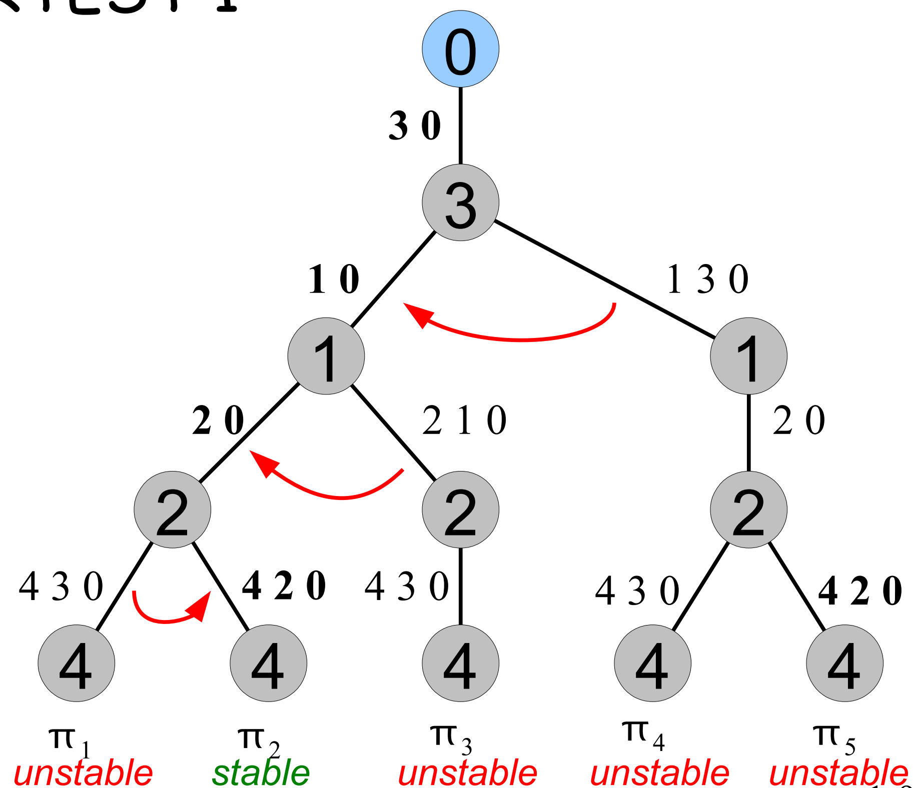


  - The solution is $\pi(1)=(1\ 0)$; $\pi(2)=(2\ 0)$; $\pi(3)=(3\ 0)$; $\pi(4)=(4\ 2\ 0)$

Stable Path Problem

- **Example: SHORTEST 2**
  - Stable path assignments... Note that this is the same tree as in SHORTEST 1



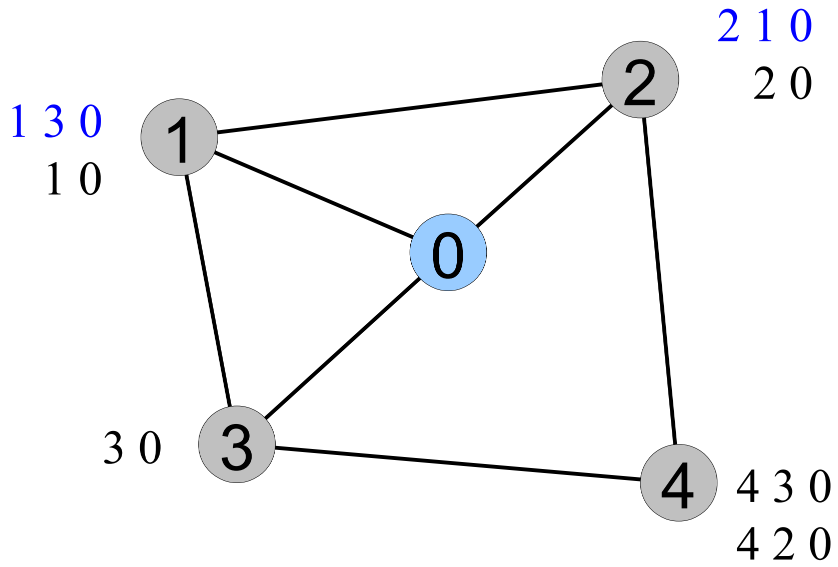


Stable Path Problem

- **Observations**
  - The solutions in SHORTEST 1 and SHORTEST 2 are trees rooted at the origin 0.
  - Both solutions are also shortest paths trees.

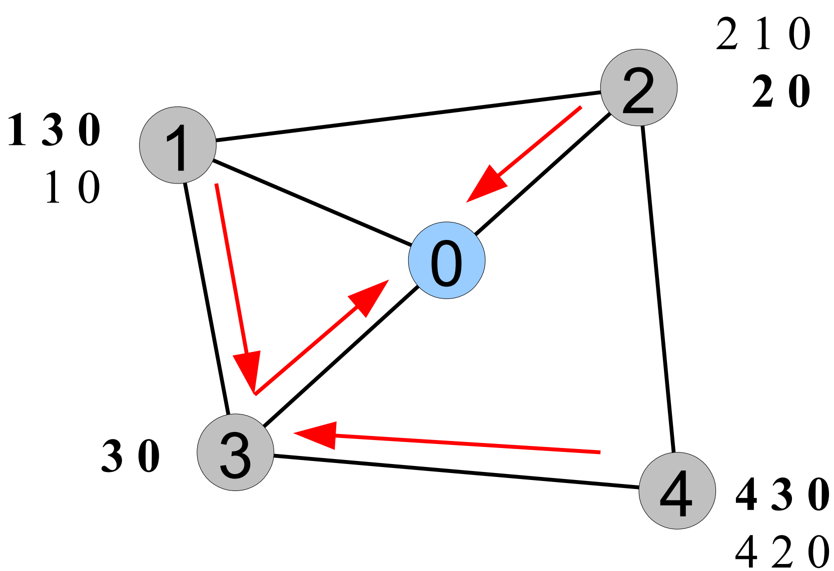
Stable Path Problem

- **Example: GOOD GADGET**
  - An example where nodes 1 and 2 better rank their longest permitted path

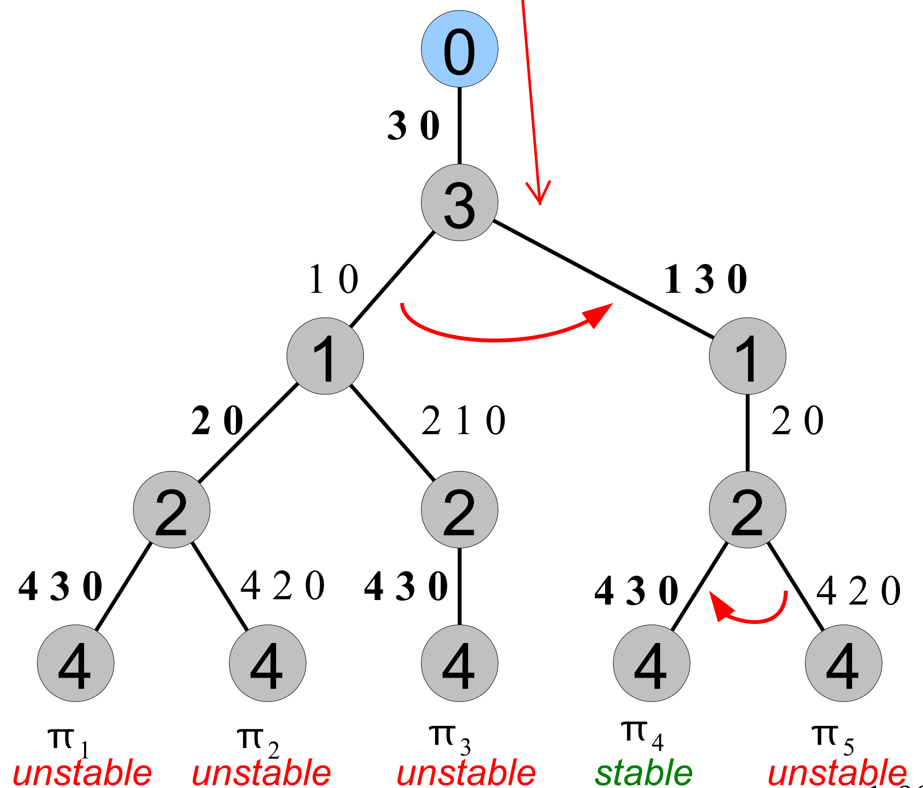


Stable Path Problem

- Example: GOOD GADGET
  - Stable path assignments ?



Importance du graphe de dépendance. En effet, Au branchement entre 3 et 1, il va préférer le meilleur chemin pour 1, c'est pourquoi le chemin pi1 est unstable.



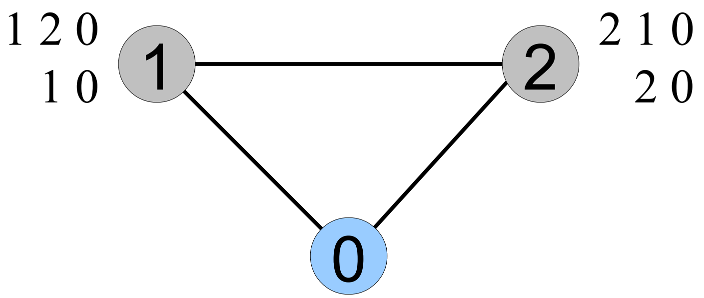
Stable Path Problem

- **Example: GOOD GADGET**
  - The path assignment tree in GOOD-GADGET is identical to SHORTEST 1 and SHORTEST 2.
  - However, the routing tree is not a shortest-paths tree
  - The method we have used to find the possible complete path assignments (topological sort) was possible as the dependency graph was a DAG. **Is this always the case ?**

Le graphe de dépendance n'est pas toujours acyclique.
que se passe-t-il si c'est le cas ?

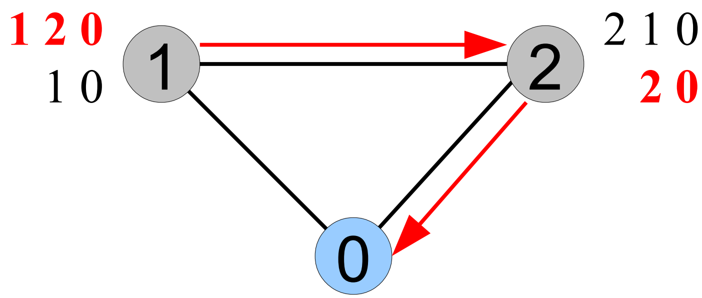
Stable Path Problem

- **Example: DISAGREE**
  - Is this instance solvable ?



Stable Path Problem

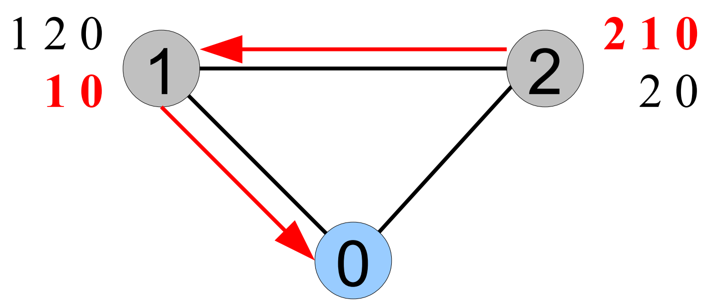
- **Example: DISAGREE**
  - Clearly, this path assignment is stable: $\pi(1)=(1\ 2\ 0)$; $\pi(2)=(2\ 0)$



  - Observe that $\pi'(2)=(2\ 0)$ is a partial path assignment, then $choices(\pi', 1)=\{\{1(2\ 0)\} \cap P^1\} = \{(1\ 2\ 0)\}$ and $best(choices(\pi', 1))=(1\ 2\ 0)$

Stable Path Problem

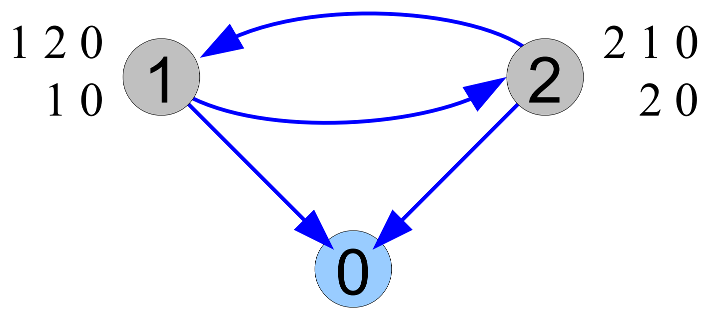
- **Example: DISAGREE**
  - This other path assignment is also stable: $\pi(1)=(1\ 0)$; $\pi(2)=(2\ 1\ 0)$



  - Observe that $\pi'(1)=(1\ 0)$ is a partial path assignment, then $choices(\pi', 2)=\{\{2(1\ 0)\} \cap P^2\} = \{(2\ 1\ 0)\}$ and $best(choices(\pi', 2))=(2\ 1\ 0)$

Stable Path Problem

- **Example: DISAGREE**
  - What does the dependency graph G_D look like ?

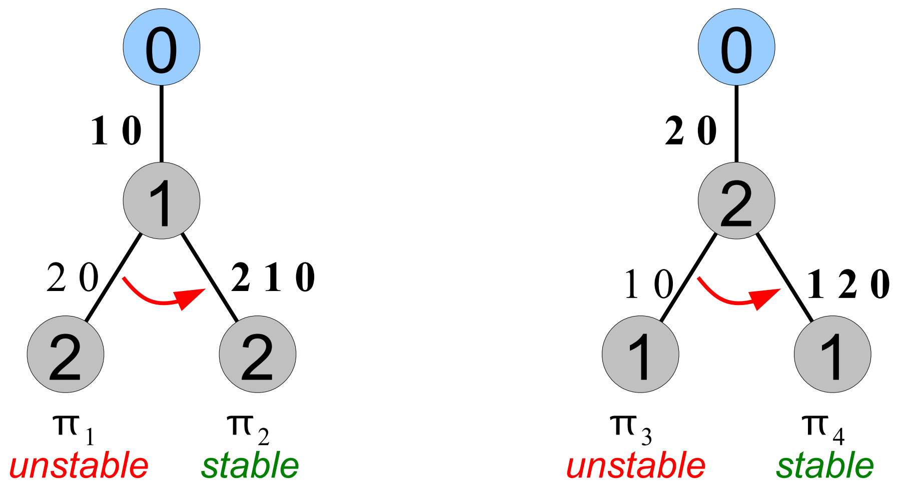


  - G_D is not a DAG → no linear ordering of nodes is possible !

Stable Path Problem

- Example: DISAGREE
  - Constructing the path assignment tree depends on the order the nodes are considered

node 1 learns direct path first



node 2 learns direct path first

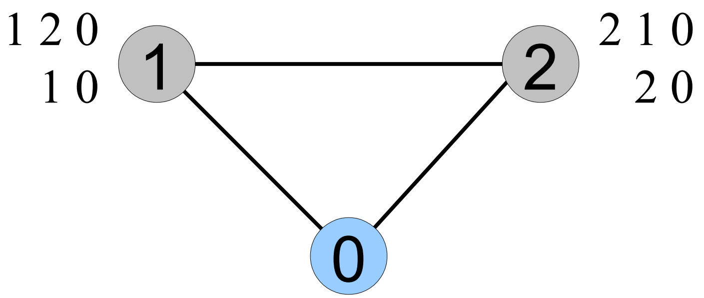
Stable Path Problem

- **Activation Sequence**
  - The route computation is performed in a distributed manner, each node having a partial view of the state of other nodes through the exchange of messages.
  - We will consider a sequence in which nodes learn about the path assignment of one of their neighbors, compute their own *choices* and select their *best* path. This is called an activation sequence.
  - At each step, one node u processes a message of $w \in peers(u)$. If during this step the best route of u has changed, messages informing the change are sent to its neighbors.

Stable Path Problem

- **Example: DISAGREE**
  - Start from a state where only node 0 knows path (0)

$\pi(0)=\epsilon$
$\pi(2)=\epsilon$
$choices(1)=\emptyset$
$best(1)=\epsilon$



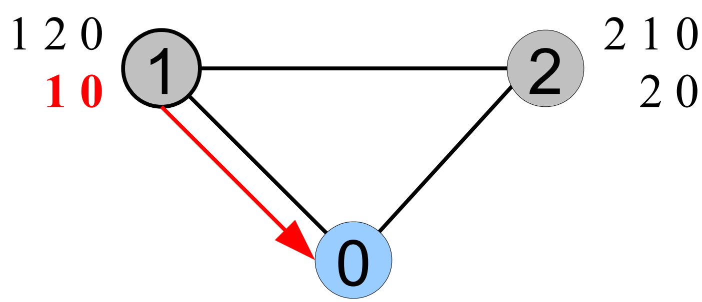
$best(0)=(0)$

$\pi(0)=\epsilon$
$\pi(1)=\epsilon$
$choices(2)=\emptyset$
$best(2)=\epsilon$

Stable Path Problem

- **Example: DISAGREE**
  - Assume that node 1 learns first that $\pi(0)=(0)$

$\pi(0)=(0)$
$\pi(2)=\epsilon$
$choices(1)=\{(1\,0)\}$
$best(1)=(1\,0)$



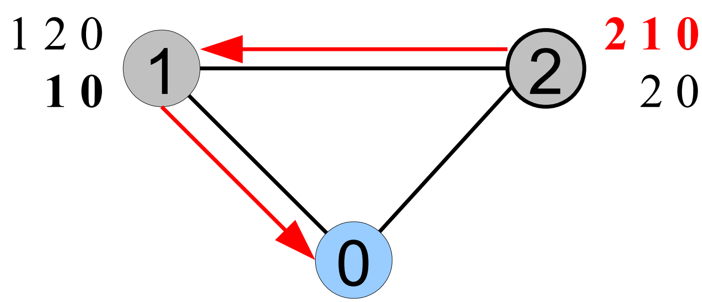
$best(0)=(0)$

$\pi(0)=\epsilon$
$\pi(1)=\epsilon$
$choices(2)=\emptyset$
$best(2)=\epsilon$

Stable Path Problem

- **Example: DISAGREE**
  - Then node 2 learns $\pi(1)=(10)$

$\pi(0)=(0)$
$\pi(2)=\epsilon$
$choices(1)=\{(1\,0)\}$
$best(1)=(1\,0)$



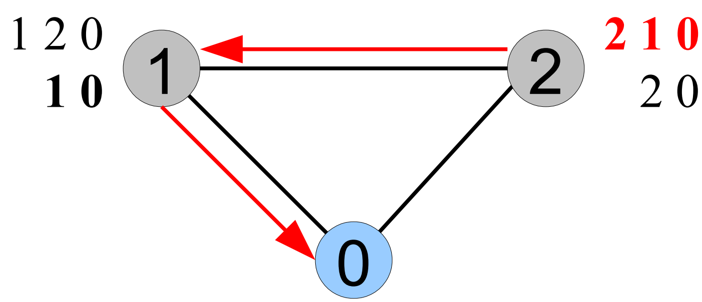
$\pi(0)=\epsilon$
$\pi(1)=(1\,0)$
$choices(2)=\{(2\,1\,0)\}$
$best(2)=(2\,1\,0)$

$best(0)=(0)$

Stable Path Problem

- **Example: DISAGREE**
  - Then node 2 learns $\pi(0)=(0)$
  - This has no impact on its *best* path selection as $\lambda^2(2\ 0) < \lambda^2(2\ 1\ 0)$

$\pi(0)=(0)$
$\pi(2)=\epsilon$
$choices(1)=\{(1\,0)\}$
$best(1)=(1\,0)$



$best(0)=(0)$

$\pi(0)=(0)$
$\pi(1)=(1\,0)$
$choices(2)=\{(2\,0),(2\,1\,0)\}$
$best(2)=(2\,1\,0)$

Stable Path Problem

- **Example: DISAGREE**
  - The previous activation sequence can be summarized as follows:

Step	Event	Node 1	Node 2
0	(0, 1):$\pi(0)=(0)$	*choices*={(1 0)} ***best*=(1 0)**	
1	(1, 2):$\pi(1)=(1\ 0)$		*choices*={(2 1 0)} ***best*=(2 1 0)**
2	(0, 2):$\pi(0)=(0)$		*choices*={(2 1 0), (2 0)} *best*=(2 1 0)
3	(2, 1):$\pi(2)=(2\ 1\ 0)$	*choices*={(1 0)} *best*=(1 0)	

  - Note : the activation of a node is triggered by the reception of a path change from another node. This is called an *edge-triggered* activation sequence.

Stable Path Problem

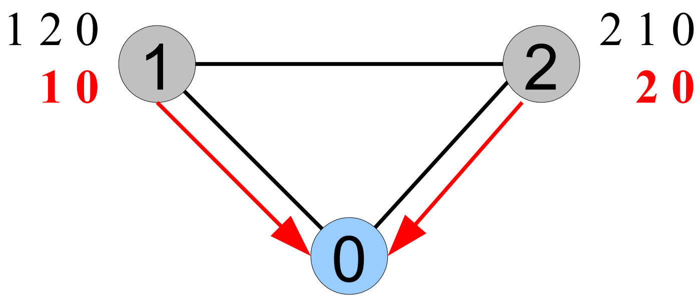
- **Example: DISAGREE**
  - Another possible activation sequence is as follows

Step	Event	Node 1	Node 2
0	(0, 1):π(0)=(0)	*choices*={(1 0)} ***best*=(1 0)**	
1	(0, 2):π(0)=(0)		*choices*={(2 0)} ***best*=(2 0)**
2	(1, 2):π(1)=(1 0)		*choices*={(2 0), (2 1 0)} ***best*=(2 1 0)**
3	(2, 1):π(2)=(2 0)	*choices*={(1 2 0), (1 0)} ***best*=(1 2 0)**	
4	(2, 1):π(2)=(2 1 0)	*choices*={(1 0)} ***best*=(1 0)**	
5	(1, 2):π(1)=(1 2 0)		*choices*={(2 0)} ***best*=(2 0)**
6	(1, 2):π(1)=(1 0)		*choices*={(2 0), (2 1 0)} ***best*=(2 1 0)**
...			

Stable Path Problem

- **Example: DISAGREE**
  - Assume that nodes 1 and 2 learn first together that $\pi(0)=(0)$. They first use direct paths to node 0.

$\pi(0)=(0)$
$\pi(2)=\epsilon$
$choices(1)=\{(1\,0)\}$
$best(1)=(1\,0)$



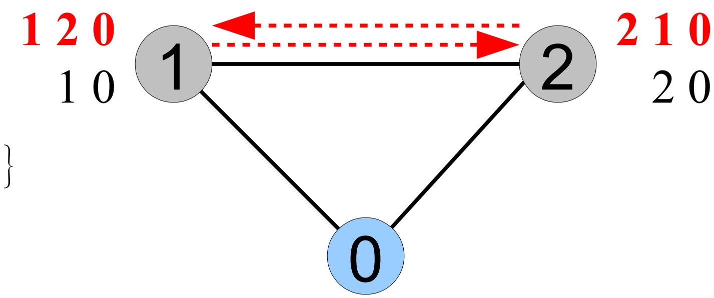
$best(0)=(0)$

$\pi(0)=(0)$
$\pi(1)=\epsilon$
$choices(2)=\{(2\,0)\}$
$best(2)=(2\,0)$

Stable Path Problem

- **Example: DISAGREE**
  - If nodes “update” their best route synchronously, they both select the other's route

$\cancel{choices(1)=\{(1\,0)\}}$
$\pi(0)=(0)$
$\pi(2)=(2\,0)$
$choices(1)=\{(1\,0),(1\,2\,0)\}$
$best(1)=(1\,2\,0)$



$\cancel{choices(2)=\{(2\,0)\}}$
$\pi(0)=(0)$
$\pi(1)=(1\,0)$
$choices(2)=\{(2\,0),(2\,1\,0)\}$
$best(2)=(2\,1\,0)$

$best(0)=(0)$

- Note : the paths used by nodes 1 and 2 at this stage are not coherent with the global routing state. For example, the path (1 2 0) chosen by node 1 is not coherent with path (2 1 0) chosen by node 2. But node 1 doesn't yet know node 2 has changed its mind. For this reason, they are shown with dashed lines.

Stable Path Problem

- **Example: DISAGREE**
  - We are back at initial state with direct paths !

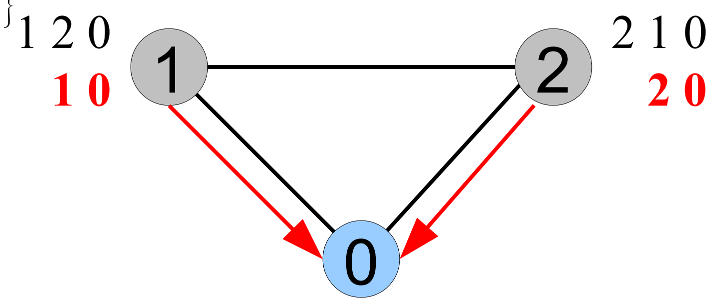
$\cancel{choices(1) = \{(1\,0), (1\,2\,0)\}}$

$\pi(0) = (0)$

$\pi(2) = \epsilon$

$choices(1) = \{(1\,0)\}$

$best(1) = (1\,0)$



$best(0) = (0)$

$\cancel{choices(2) = \{(2\,0), (1\,2\,0)\}}$

$\pi(0) = (0)$

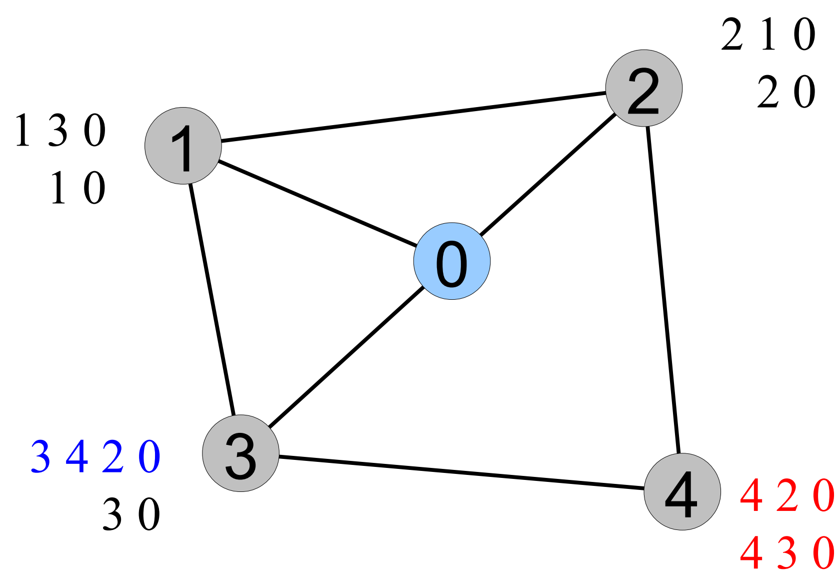
$\pi(1) = \epsilon$

$choices(2) = \{(2\,0)\}$

$best(2) = (2\,0)$

Stable Path Problem

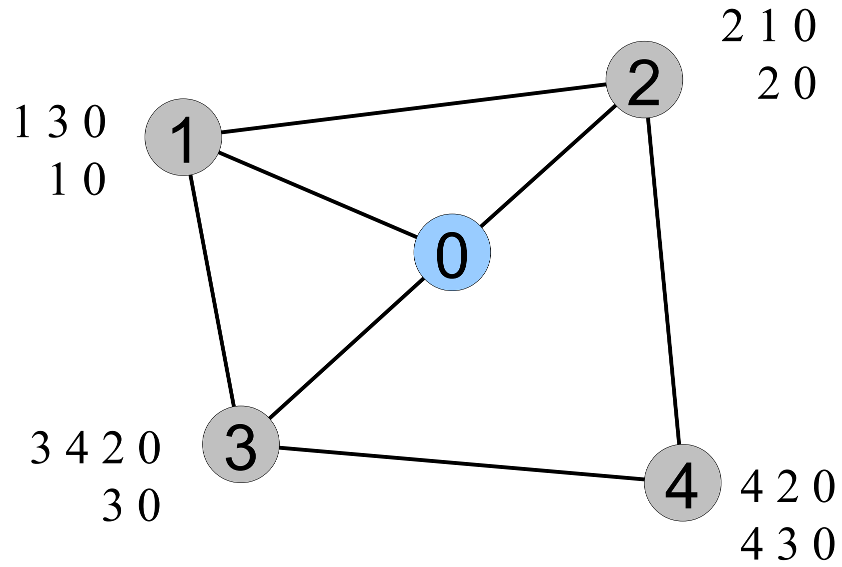
- **Example: BAD GADGET**
  - Same as GOOD GADGET, with an additional path permitted at node 3, and a different ranking at node 4



Stable Path Problem

savoir que le processus de décision boucle mais ne pas connaître le développement qui amène la boucle (slides suivants)

- **Example: BAD GADGET**
  - This instance is not solvable !
    - (it does not admit a stable path assignment)

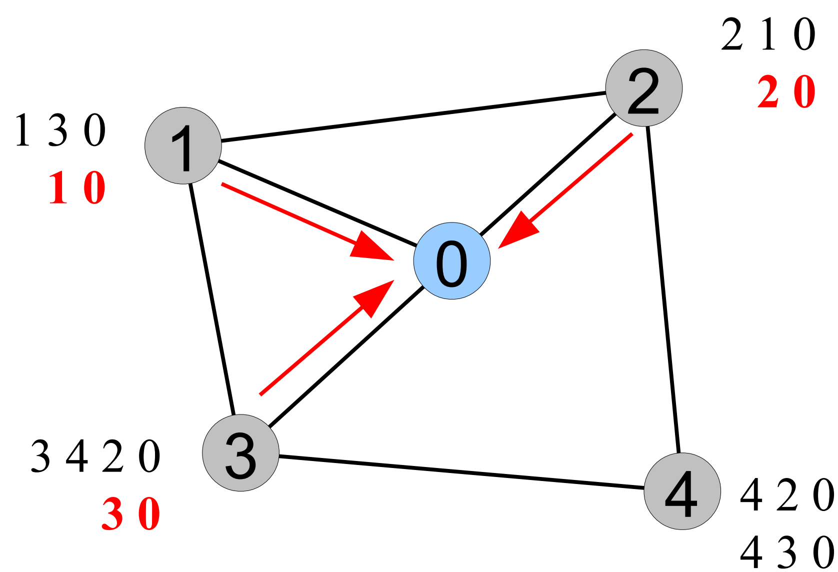


Stable Path Problem

- **Example: BAD GADGET**
  - STEP 0

$choices(1) = \{(1\,0)\}$

$choices(2) = \{(2\,0)\}$



$choices(4) = \emptyset$

$choices(3) = \{(3\,0)\}$

Stable Path Problem

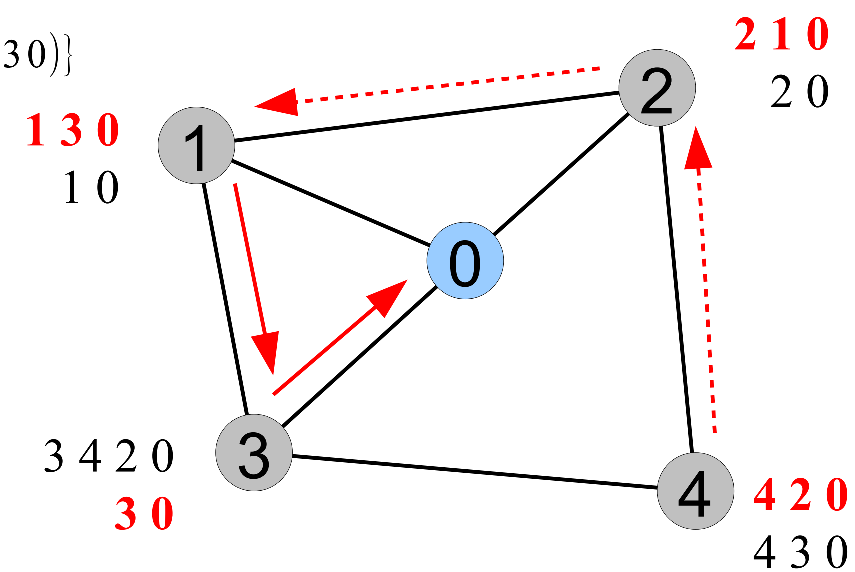
- **Example: BAD GADGET**
  - STEP 1

$choices(1)=\{(1\,0)\}$

$choices(1)=\{(1\,0),(1\,3\,0)\}$

$choices(2)=\{(2\,0)\}$

$choices(2)=\{(2\,0),(2\,1\,0)\}$



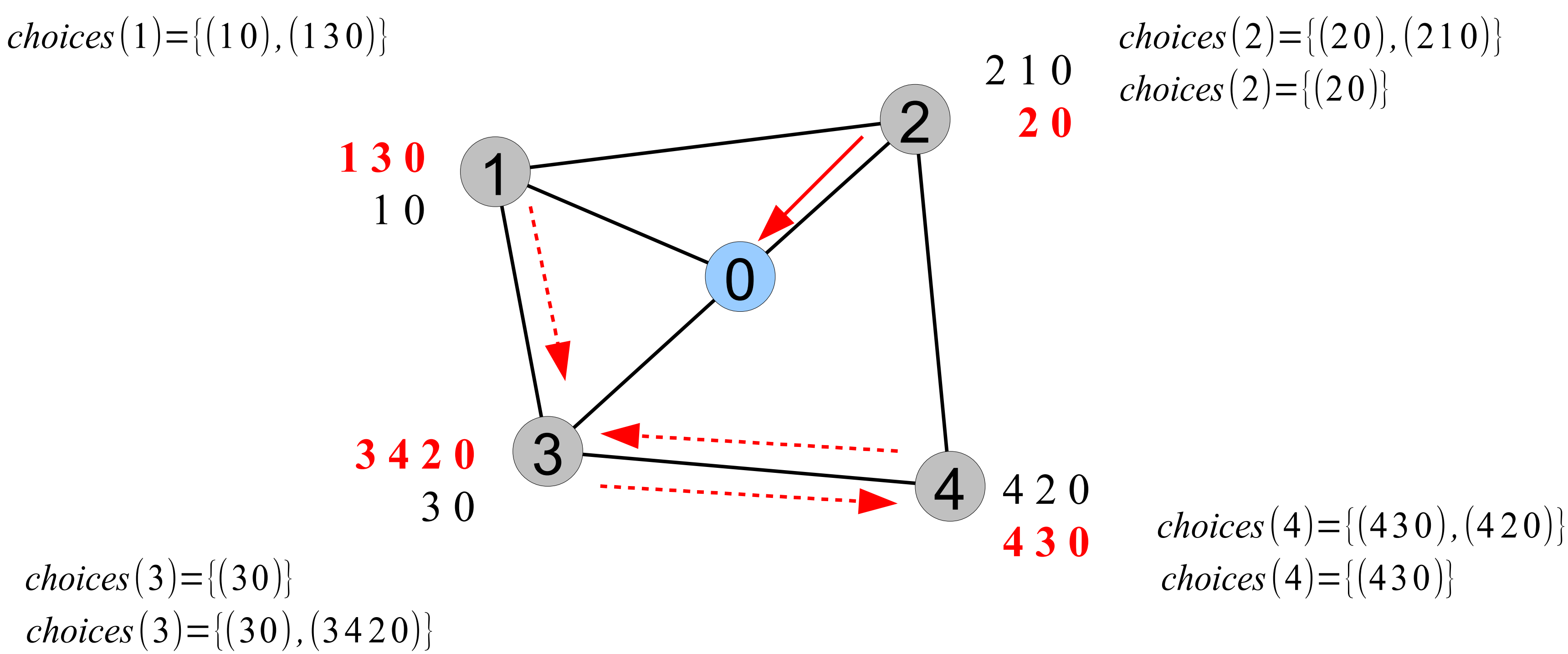
$choices(3)=\{(3\,0)\}$

$choices(4)=\emptyset$

$choices(4)=\{(4\,3\,0),(4\,2\,0)\}$

Stable Path Problem

- **Example: BAD GADGET**
  - STEP 2



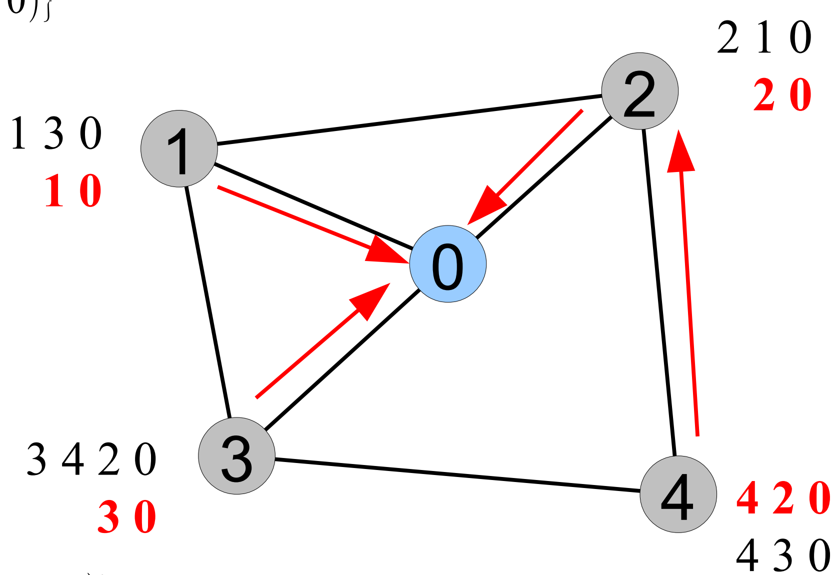
Stable Path Problem

- **Example: BAD GADGET**
  - STEP 3

$choices(1)=\{(1\,0),(1\,3\,0)\}$

$choices(1)=\{(1\,0)\}$

$choices(2)=\{(2\,0)\}$



$choices(4)=\{(4\,3\,0)\}$

$choices(4)=\{(4\,3\,0),(4\,2\,0)\}$

$choices(3)=\{(3\,0),(3\,4\,2\,0)\}$

$choices(3)=\{(3\,0)\}$

Stable Path Problem

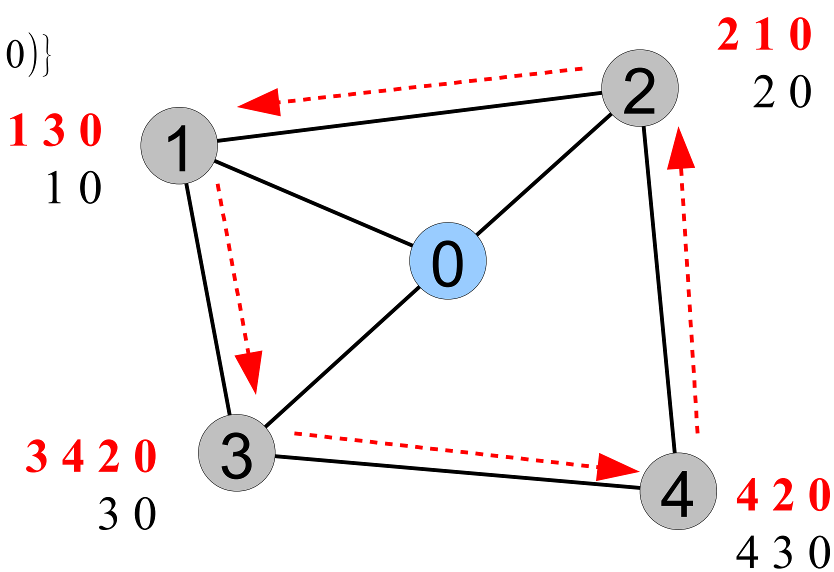
- **Example: BAD GADGET**
  - STEP 4

$choices(1)=\{(1\,0)\}$

$choices(1)=\{(1\,0),(1\,3\,0)\}$

$choices(2)=\{(2\,0)\}$

$choices(2)=\{(2\,0),(2\,1\,0)\}$



$choices(4)=\{(4\,3\,0),(4\,2\,0)\}$

$choices(3)=\{(3\,0)\}$

$choices(3)=\{(3\,0),(3\,4\,2\,0)\}$

Stable Path Problem

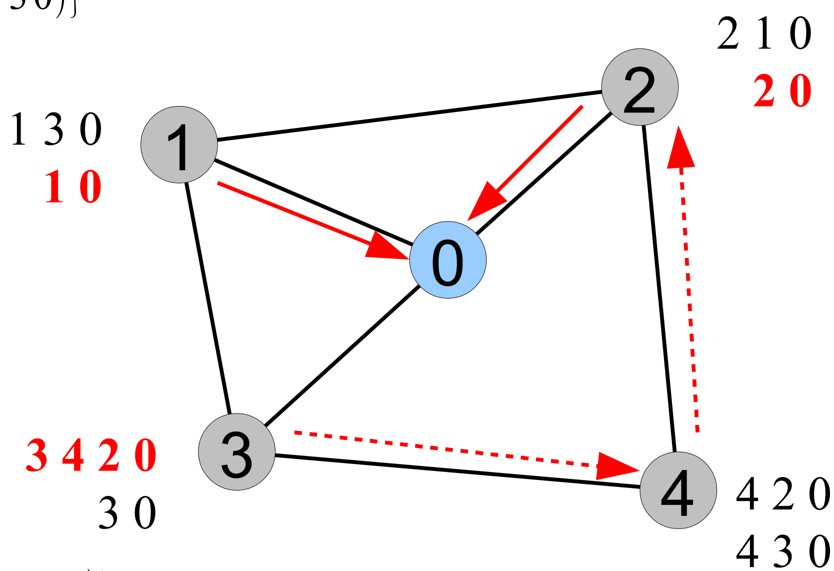
- **Example: BAD GADGET**
  - STEP 5

$choices(1)=\{(1\,0),(1\,3\,0)\}$

$choices(1)=\{(1\,0)\}$

$choices(2)=\{(2\,0),(2\,1\,0)\}$

$choices(2)=\{(2\,0)\}$



$choices(3)=\{(3\,0),(3\,4\,2\,0)\}$

$choices(4)=\{(4\,3\,0),(4\,2\,0)\}$

$choices\,(4)=\emptyset$

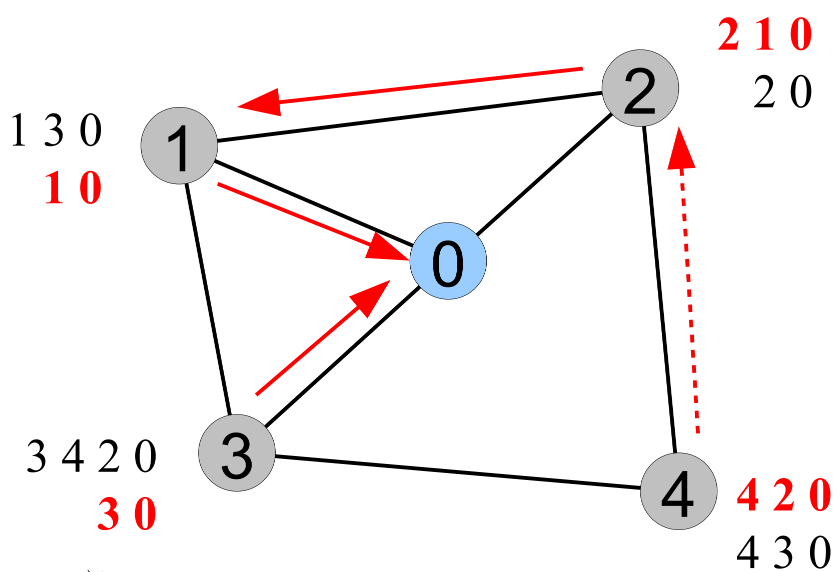
Stable Path Problem

- **Example: BAD GADGET**
  - STEP 6

$choices(1)=\{(1\,0)\}$

$choices(2)=\{(2\,0)\}$

$choices(2)=\{(2\,0),(2\,1\,0)\}$



$choices(4)=\emptyset$

$choices(4)=\{(4\,2\,0)\}$

$choices(3)=\{(3\,0),(3\,4\,2\,0)\}$

$choices(3)=\{(3\,0)\}$

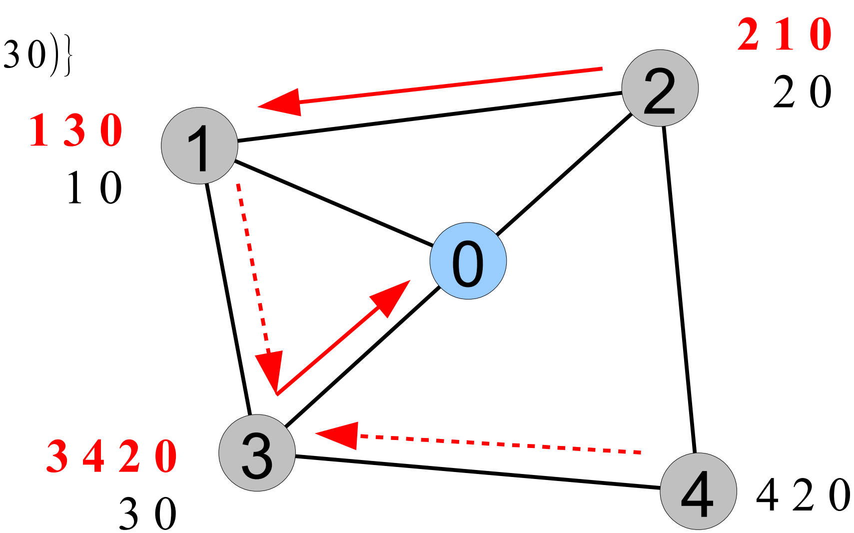
Stable Path Problem

- **Example: BAD GADGET**
  - STEP 7

$choices(1)=\{(1\,0)\}$

$choices(1)=\{(1\,0),(1\,3\,0)\}$

$choices(2)=\{(2\,0),(2\,1\,0)\}$



$choices(4)=\{(4\,2\,0)\}$

$choices(4)=\{(4\,3\,0)\}$

$choices(3)=\{(3\,0)\}$

$choices(3)=\{(3\,0),(3\,4\,2\,0)\}$

Stable Path Problem

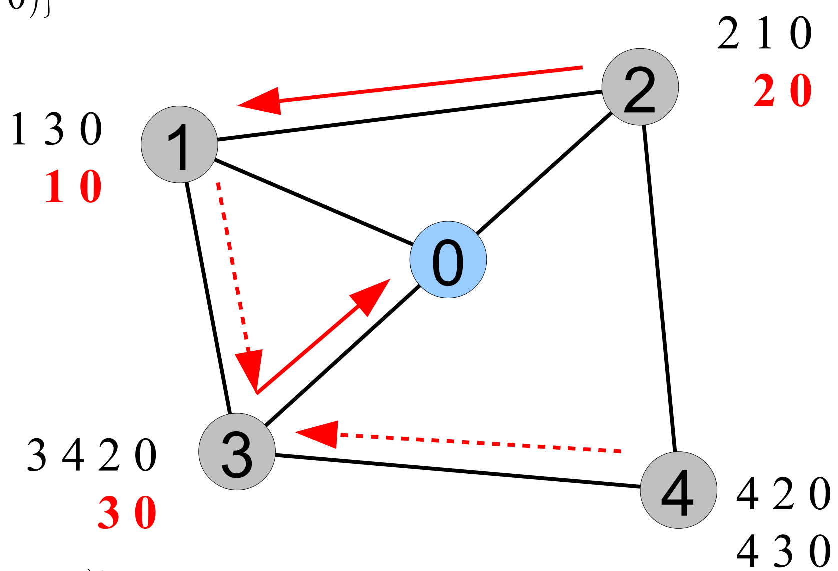
- **Example: BAD GADGET**
  - STEP 8 - same as STEP 0 !

$choices(1) = \{(1\,0), (1\,3\,0)\}$

$choices(1) = \{(1\,0)\}$

$choices(2) = \{(2\,0), (2\,1\,0)\}$

$choices(2) = \{(2\,0)\}$



$choices(4) = \{(4\,3\,0)\}$

$choices(4) = \emptyset$

$choices(3) = \{(3\,0), (3\,4\,2\,0)\}$

$choices(3) = \{(3\,0)\}$

Stable Path Problem

Example: BAD GADGET

Step	Node 1	Node 2	Node 3	Node 4
0	*choices*={(1 0)} ***best*=(1 0)**	*choices*={(2 0)} ***best*=(2 0)**	*choices*={(3 0)} ***best*=(3 0)**	
1	*choices*={(1 0), (1 3 0)} ***best*=(1 3 0)**	*choices*={(2 0), (2 1 0)} ***best*=(2 1 0)**		*choices*={(4 2 0), (4 3 0)} ***best*=(4 2 0)**
2		*choices*={(2 0)} ***best*=(2 0)**	*choices*={(3 0), (3 4 2 0)} ***best*=(3 4 2 0)**	*choices*={(4 3 0)} ***best*=(4 3 0)**
3	*choices*={(1 0)} ***best*=(1 0)**		*choices*={(3 0)} ***best*=(3 0)**	*choices*={(4 2 0)} ***best*=(4 2 0)**
4	*choices*={(1 0), (1 3 0)} ***best*=(1 3 0)**	*choices*={(2 0), (2 1 0)} ***best*=(2 1 0)**	*choices*={(3 0), (3 4 2 0)} ***best*=(3 4 2 0)**	*choices*={(4 2 0), (4 3 0)} *best*=(4 2 0)
5	*choices*={(1 0)} ***best*=(1 0)**	*choices*={(2 0)} ***best*=(2 0)**		*choices*=∅ ***best*=ε**
6		*choices*={(2 0), (2 1 0)} ***best*=(2 1 0)**	*choices*={(3 0)} ***best*=(3 0)**	*choices*={(4 2 0)} ***best*=(4 2 0)**
7	*choices*={(1 0), (1 3 0)} ***best*=(1 3 0)**		*choices*={(3 0), (3 4 2 0)} ***best*=(3 4 2 0)**	*choices*={(4 3 0)} ***best*=(4 3 0)**
8	*choices*={(1 0)} ***best*=(1 0)**	*choices*={(2 0)} ***best*=(2 0)**	*choices*={(3 0)} ***best*=(3 0)**	*choices*=∅ ***best*=ε**

Stable Path Problem

- **Bad News...**

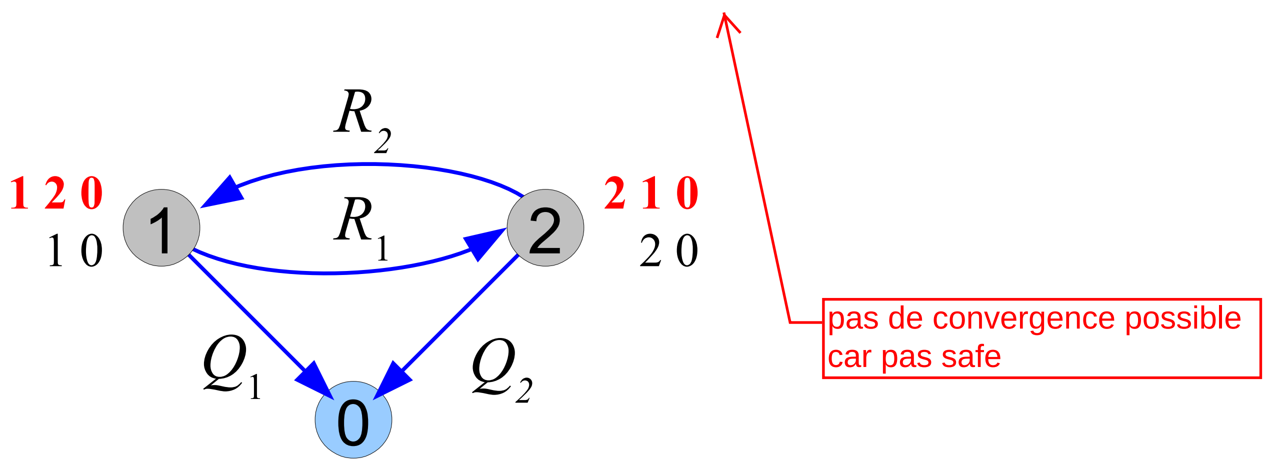
- **Theorem: Solvability is NP-complete**
  - The problem of determining whether an instance of the stable paths problem is solvable is NP-complete

Stable Path Problem

- **Safety**
  - A system can be solvable and yet fail to converge
    - The DISAGREE system is an example: it admits two stable solutions but some activations sequences lead to an oscillation.
  - An SPP instance is **safe** if it is solvable under any *fair* activation sequence.
    - A fair activation sequence is an activation sequence without information loss.

Stable Path Problem

- **Dispute Wheel: DISAGREE**
  - The DISAGREE system has a *dispute wheel*.



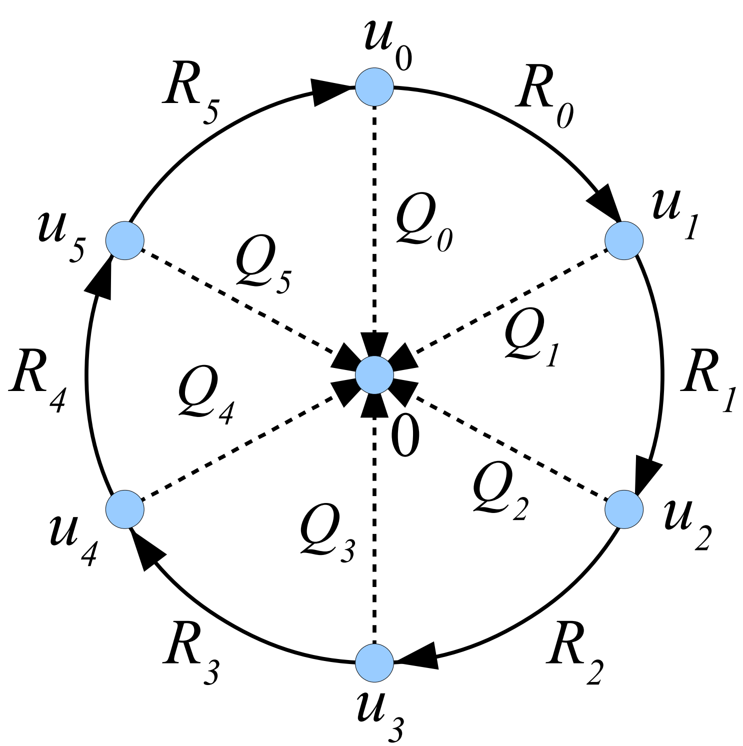
$$\lambda^1(R_1Q_2) > \lambda^1(Q_1)$$

and

$$\lambda^2(R_2Q_1) > \lambda^2(Q_2)$$

Stable Path Problem

- **Dispute Wheel**



Q_i and R_i are non-empty paths

$\forall\, 0 \le i < k$, R_i *path from* u_i *to* $u_{(i+1) \bmod k}$

$$Q_i \in P^{u_i}$$

$$R_i Q_{(i+1) \bmod k} \in P^{u_i}$$

$$\lambda^{u_i}(R_i Q_{(i+1) \bmod k}) \ge \lambda^{u_i}(Q_i)$$

Q_i are *spoke* paths

R_i are *rim* paths

Stable Path Problem

- **Theorems**
  - If an SPP instance has no dispute wheel
    - it is solvable (sufficient condition)
    - it has a unique solution (sufficient condition)
    - it is safe (sufficient condition)

Stable Path Problem

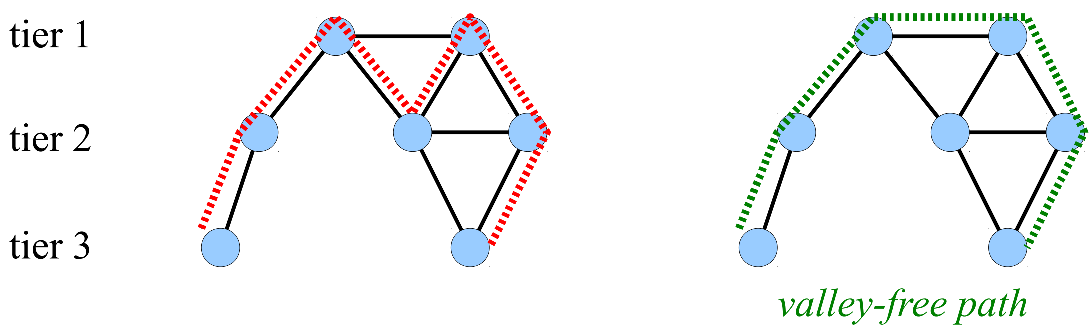
- **Discussion and Limitations**
  - The SPP formalism is difficult to apply directly to BGP systems as the ranking functions are not explicit in BGP routing filters.
    - Usually, an operator will not specify the exact paths that are allowed but rather define classes of paths based on one or several predicates.
  - The no-dispute-wheel condition is too strong in certain cases
    - (might not be required for solvability).
  - Policy Routing is still not well understood and subject of heavy research...

Chapter 5: roadmap

- 5.1 BGP Scalability
- 5.2 BGP Stability
  - A first look at BGP stability
  - Path Exploration
  - Stable Path Problem
  - Stable eBGP without Global Coordination
  - iBGP stability : MED-EVIL

Stable Path Problem

- **Stable Routing without Global Coordination**
  - *[Gao and Rexford, 2000]* have proven formally that a system where every AS uses the following guidelines is safe
    - Use a ranking of routes such that routes from customers are preferred to those of peers, themselves preferred to those of providers
    - Filter routes so that paths between providers and/or peers are not allowed (*valley-free* property)



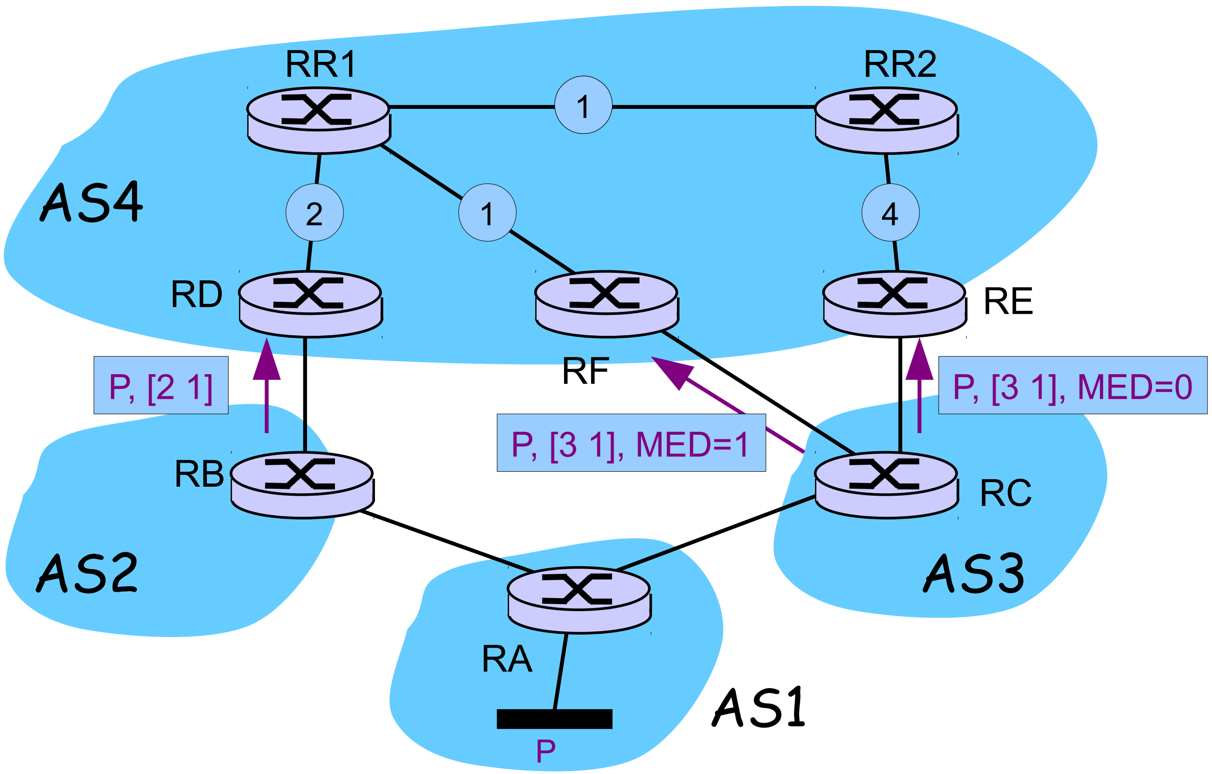
Chapter 5: roadmap

- 5.1 BGP Scalability
- 5.2 BGP Stability
  - A first look at BGP stability
  - Path Exploration
  - Stable Path Problem
  - Stable eBGP without Global Coordination
  - iBGP stability : MED-EVIL

unique slide de ce chapitre qui change par rapport à la version de cette année ds slides

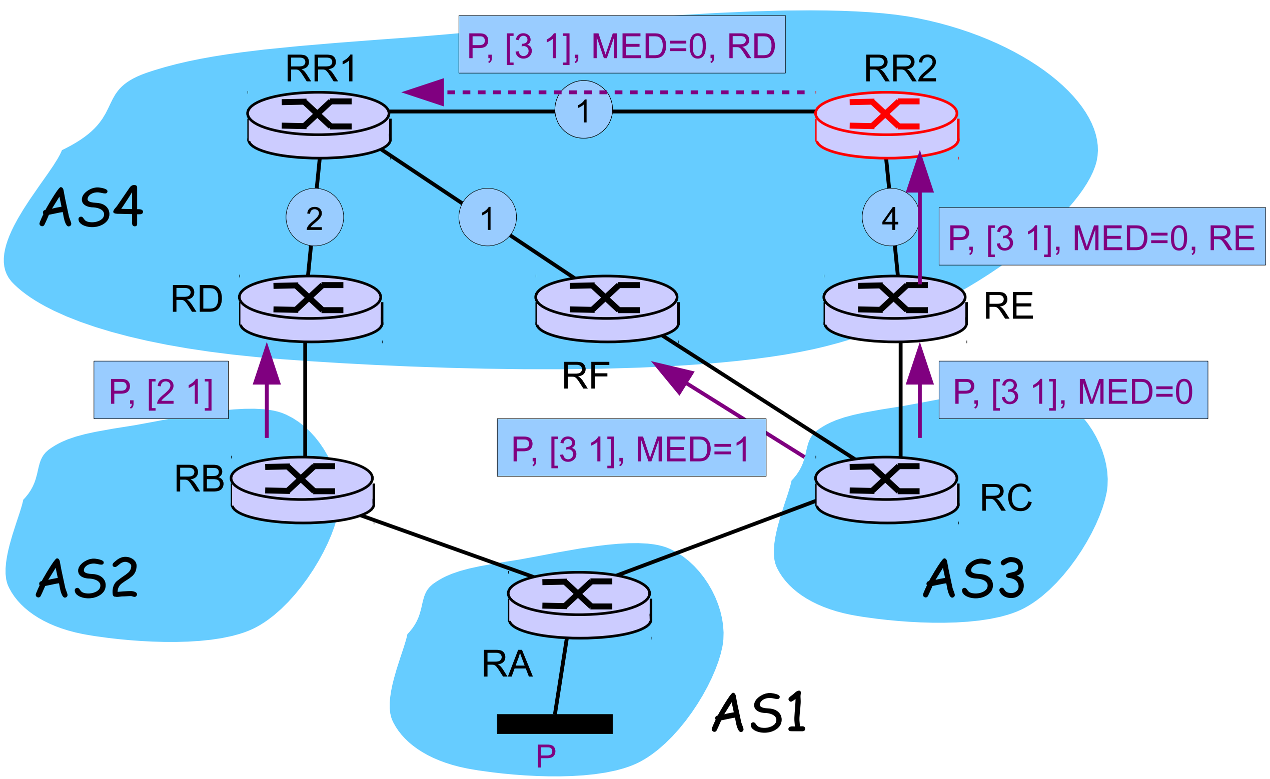
iBGP issues

- **Route Oscillations with MED**
  - RD, RE, RF always prefer their direct eBGP path
  - RR1 and RR2 only know a subset of the 3 possible paths



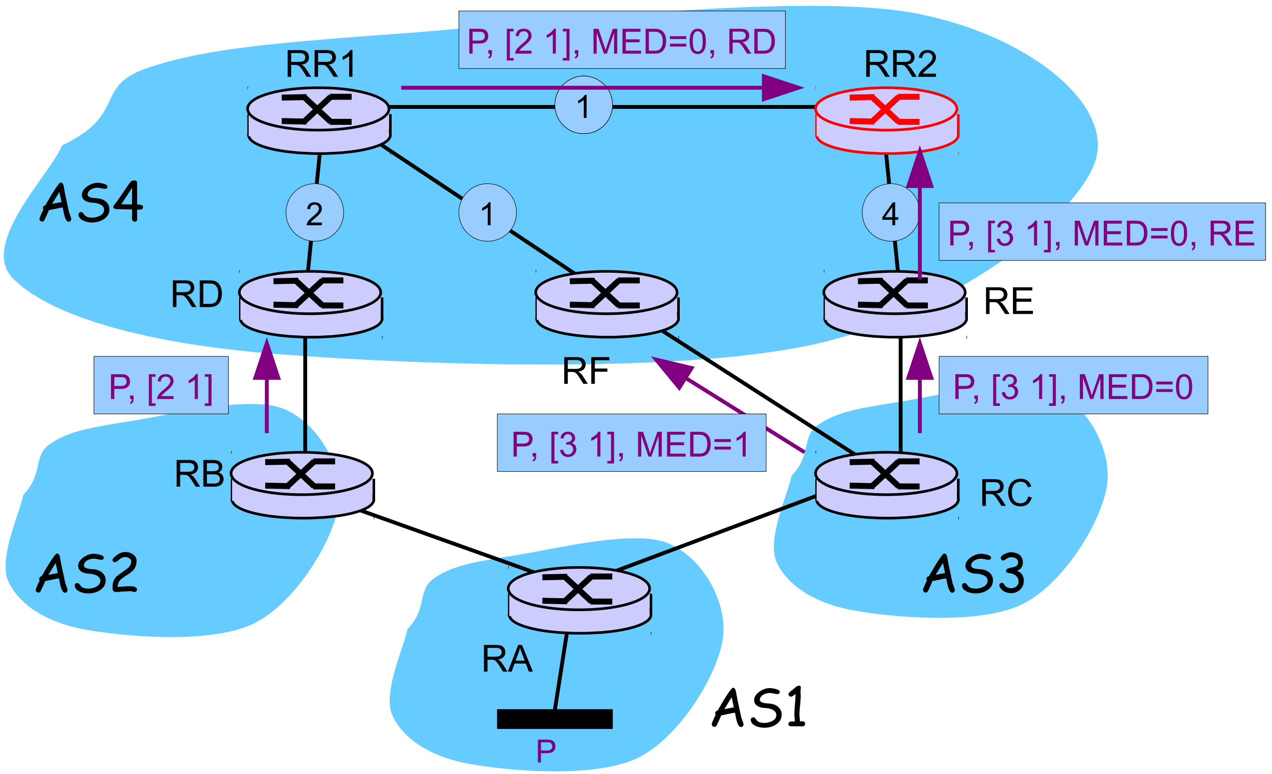
iBGP issues

- **Route Oscillations with MED (RR2 PoV)**
  - if RR2 only knows the path through RE, it advertises it to RR1



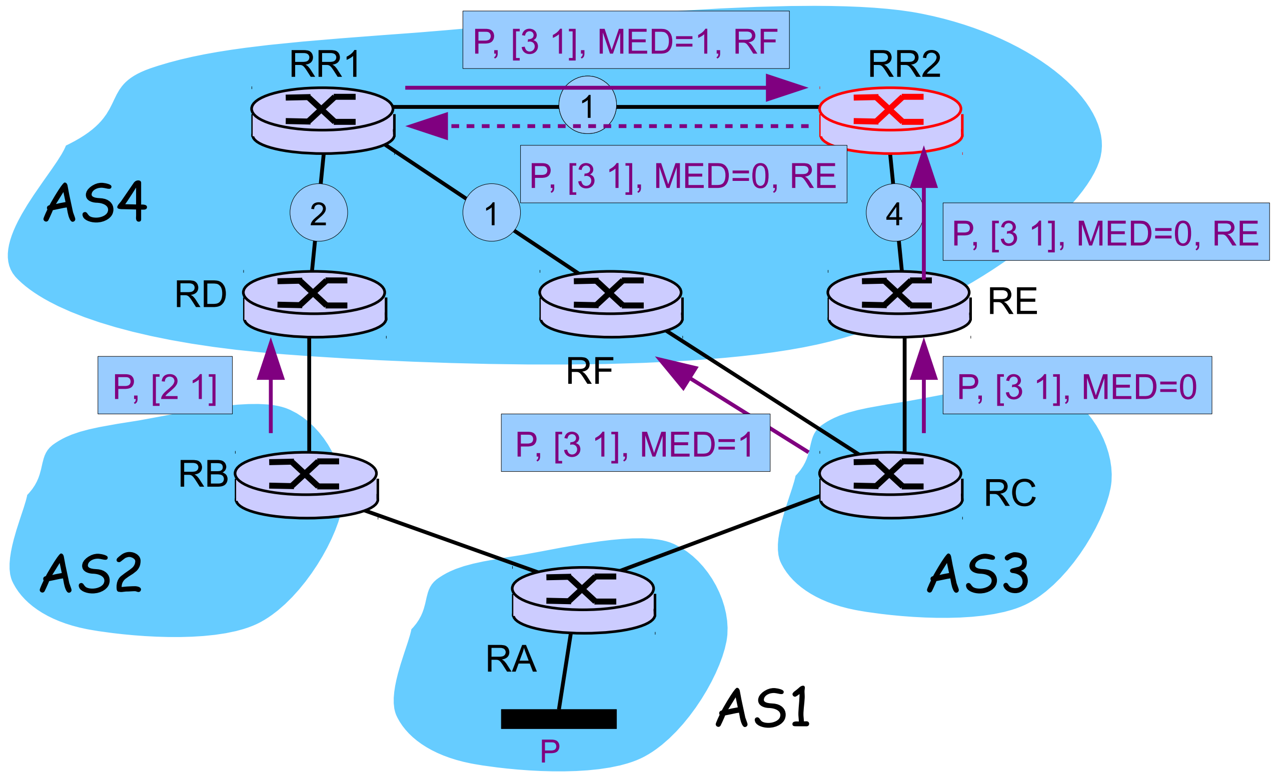
iBGP issues

- **Route Oscillations with MED (RR2 PoV)**
  - if RR2 knows paths through RD and through RE, it prefers RD (nearest next-hop) and doesn't advertise to RR1 path through RE



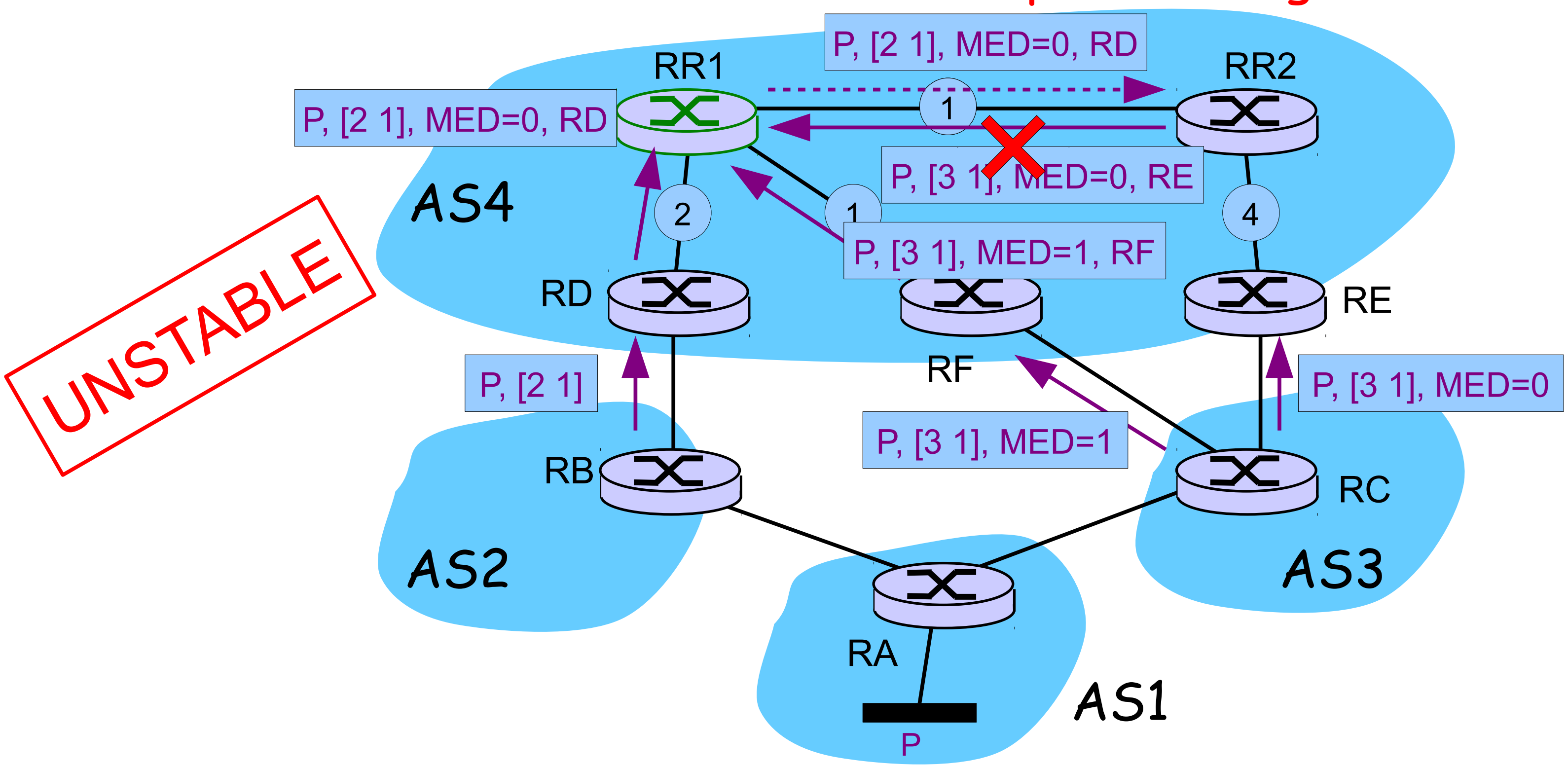
iBGP issues

- **Route Oscillations with MED** (RR2 PoV)
  - if RR2 knows paths through RF and through RE, it prefers RE (lowest MED) and advertises path through RE to RR1



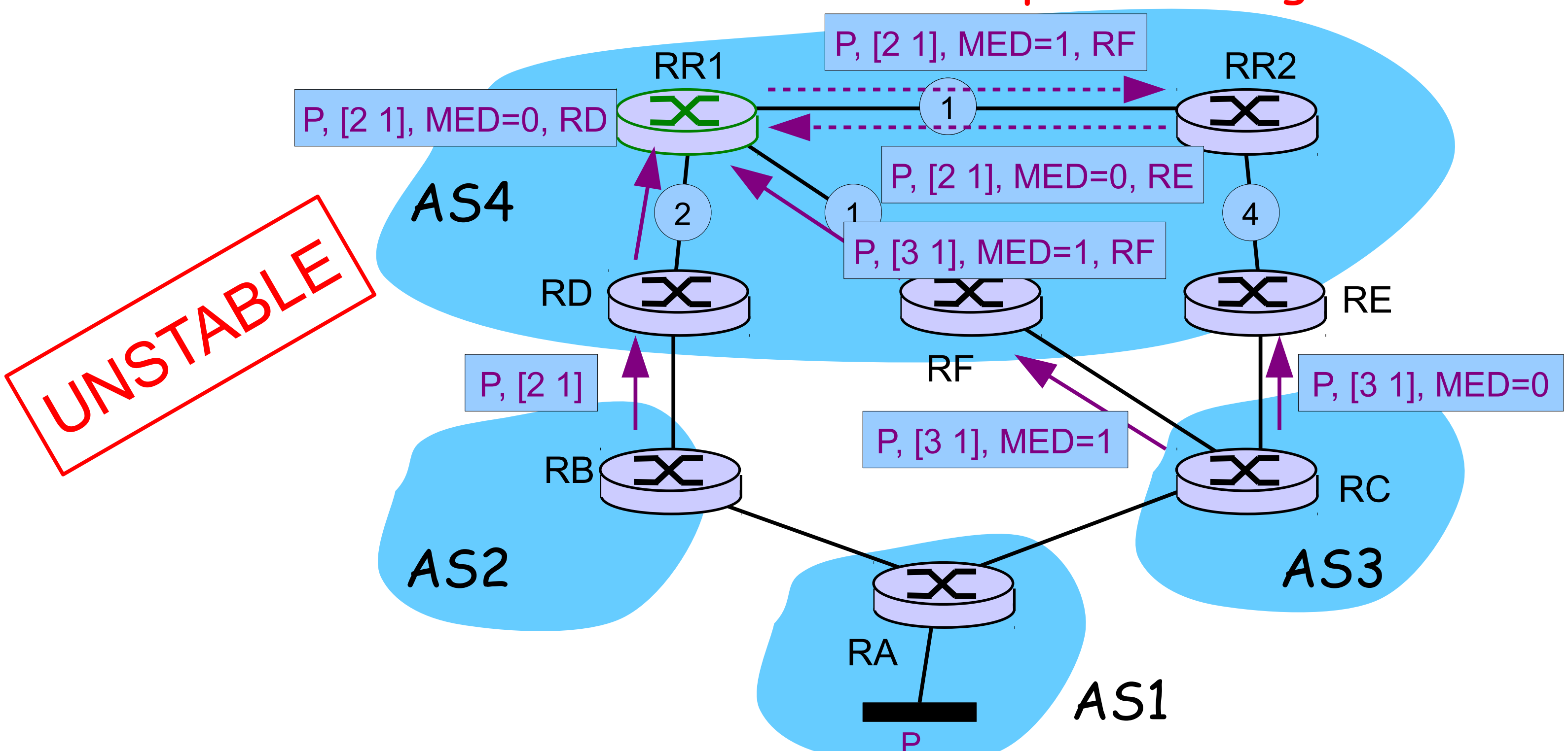
iBGP issues

- **Route Oscillations with MED** (RR1 PoV)
  - if RR1 knows paths through RD, RE and RF, it prefers RD (lowest MED, then nearest next-hop) and advertises it to RR2.
  - But in this case, RR2 withdraws path through RE !



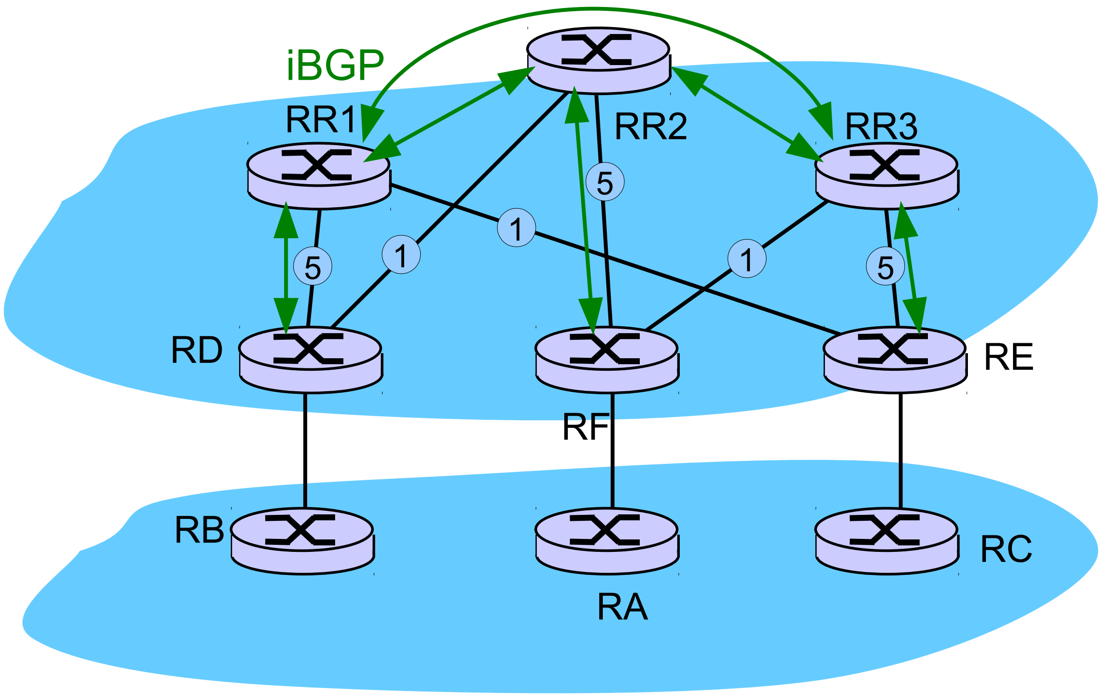
iBGP issues

- **Route Oscillations with MED (RR1 PoV)**
  - if RR1 knows paths through RD and RF, it prefers RF (nearest next-hop) and advertises it to RR2.
  - But in this case, RR2 announces path through RE !



iBGP issues

- BAD GADGET in iBGP
  - RD, RE and RF prefer their eBGP path
  - RR_i prefers path received from its $RR_{(i-1 \bmod 3)}$ due to nearest next-hop rule → never converges !



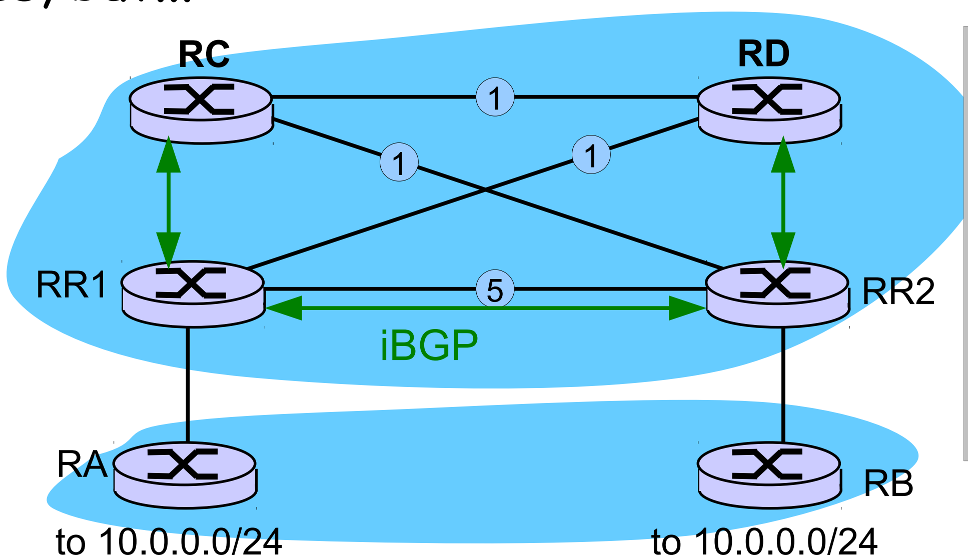
routage converge mais boucle de forwarding

iBGP issues

- **Forwarding deflection and loops**
  - RR1 (resp. RR2) prefers path through RA (resp. RB)
  - RC (resp. RD) only receives path through RA (resp. RB)
  - converges, but...

Router RC
BGP routes
10.0.0.0/24, next-hop=RA

IGP routes
RA via RD (cost=2)
RD direct (cost=1)
RB via RR2 (cost=1)
RR1 via RD (cost=2)
RR2 direct (cost=1)



Router RD
BGP routes
10.0.0.0/24, next-hop=RB

IGP routes
RA via RR1 (cost=1)
RB via RC (cost=2)
RC direct (cost=1)
RR1 direct (cost=1)
RR2 via RC (cost=2)

- Packets from RC go first to RD (to reach next-hop RA) BUT packets from RD go first to RC (to reach next-hop RB)...