JUNOS BGP Basics and Policy

JUNOS BGP Basics and Policy

- **This Presentation will show examples of BGP Policies covering:**
  - BGP Basics
  - BGP Show Commands
  - Juniper BGP Policy Basics
  - Import/Export Policies
  - Local Preference
  - MED
  - Communities
  - Next Hop Self
  - AS Path Regular Expressions
  - Route-filters
  - Prefix Lists
  - Route Damping
  - Test and View Policies

Autonomous System

- **Autonomous System (AS)**
  - Group of routers
    - Administered with a common routing policy
    - Running under a single technical administration
    - Viewed externally as a single, coherent interior routing domain
    - Could be running more than one IGP
- AS range
  - 16-bit integer (1-65535)
  - **Private AS Range 64512 – 65535**

BGP Protocol Overview

- **BGP runs over TCP**
  - Port 179

- **Any two routers that have formed a TCP connection to exchange BGP information are called "Peers" or "Neighbors".**

- **Once connection is made, Peers exchange their full BGP routing tables. Updates are then sent as the table changes or new routes added.**

BGP Routes

- **Consists of**
  - Destination, described as an IP address prefix
    - NLRI – Network Layer Reachability Information
  - Information that describes path to the destination
    - AS path
    - Path attributes
- **BGP peers advertise routes to each other in update messages**

BGP Routes

- **BGP stores routes in the JUNOS software routing table**
  - RIB
    - Inet.0 = IPv4 unicast routing table
  - FIB
    - IP forwarding table
- **Routing table stores**
  - Routing information learned from update messages
  - Local routing information selected by applying local policies to routes received in update messages
  - Information selected to advertise to BGP peers

Routing Information Bases (RIB)

- **BGP uses three different storage tables known as Routing Information Bases (RIB)**
  - RIB-IN
    - One for each established BGP peer for routes received from that peer
  - RIB-LOCAL
    - Routes used for traffic forwarding
  - RIB-OUT
    - One for each established BGP peer for routes to be advertised to the peer

Four Types of BGP Messages

- **OPEN**
  - Used to create a BGP connection.
- **UPDATE**
  - Used to exchange Network Reachability Information.
  - Can announce or withdraw routes.
  - Uses “attributes” to keep track of route specific information.
- **KEEPALIVE**
  - Used to determine a link or Peer has failed or not available.
- **NOTIFICATION**
  - Used when an error condition occurs.
  - Terminates BGP Session.

BGP Neighbor States

- ***Also known as the BGP Finite State Machine***
- **Idle**
  - Initial Neighbor State
- **Connect**
  - waiting for a TCP connection to be completed
- **Active**
  - TCP session has been initiated to remote peer
- **Opensent**
  - TCP connection is established. OPEN message is sent to remote Peer. Waiting for returning OPEN message from remote Peer.
- **Openconfirm**
  - OPEN message is received from remote Peer. Waiting for Keep-Alive or Notification message.
- **Established**
  - BGP Peers are fully adjacent. Can start exchanging information.

BGP Attributes

- **BGP Attributes: Influences BGP Routing Decision**
  - Origin
  - AS Path
  - Next Hop
  - Local Preference
  - Multiple Exit Discriminator (MED)
  - Community

BGP Attribute Classes

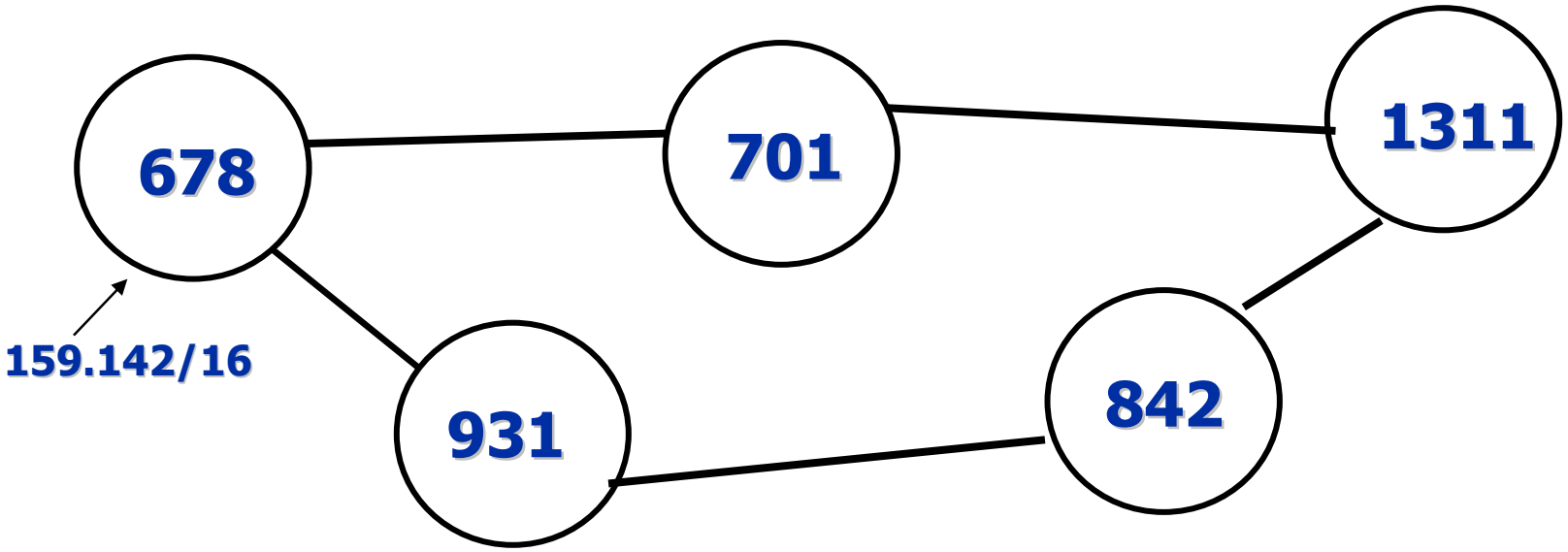
- **Well-known mandatory**
  - Must be supported and included in every update
- **Well-known discretionary**
  - Must be supported but not required
- **Optional transitive**
  - Accepted and passed to peers but not required
- **Optional nontransitive**
  - Not passed to peers and

BGP attributes - Origin

- **Lets a router know how the prefix was originally learned. This has an effect on route selection.**
  - 0 - IGP (I)
    - Network Layer Reachability Information is interior to the originating AS
  - 1 – EGP (E)
    - Network Layer Reachability Information learned via EGP
  - 2 - INCOMPLETE (?)
    - Network Layer Reachability Information learned by some other means
- **All routes are now I or ?**
- **Some providers use it as another metric**

BGP Attributes - AS Path

- Used to prevent looping
- Aids in route selection, shortest AS path



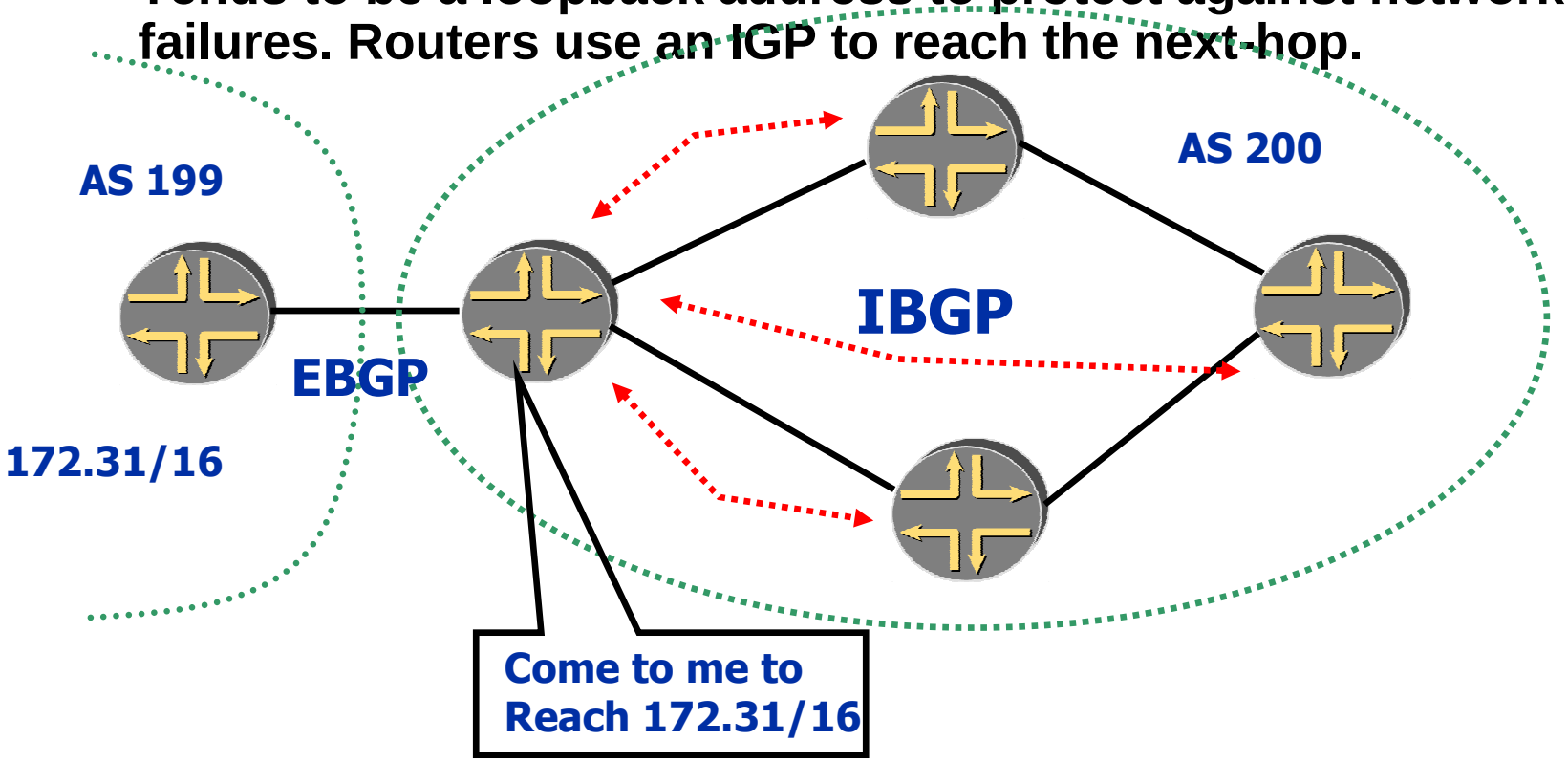
AS 1311 sees two paths to prefix 159.142/16.

159.142/16	701	678

159.142/16	842	931	678

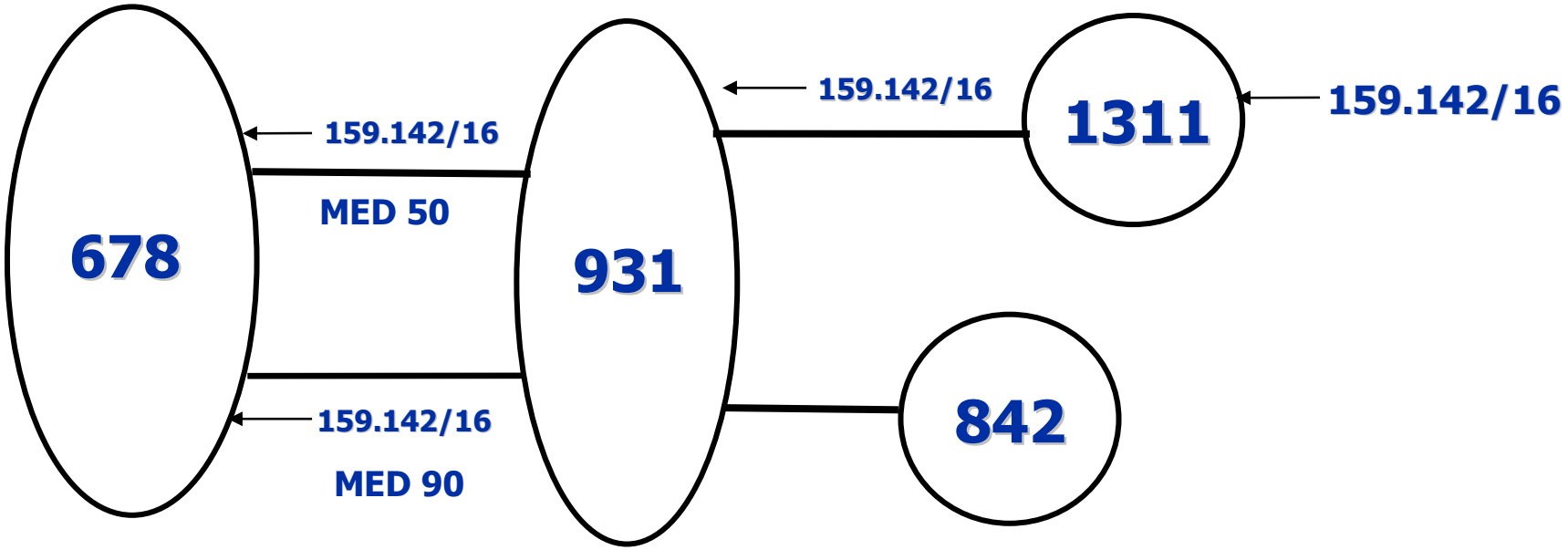
BGP Attributes – Next Hop

- Gives routers a destination to reach prefixes
- Tends to be a loopback address to protect against network failures. Routers use an IGP to reach the next-hop.



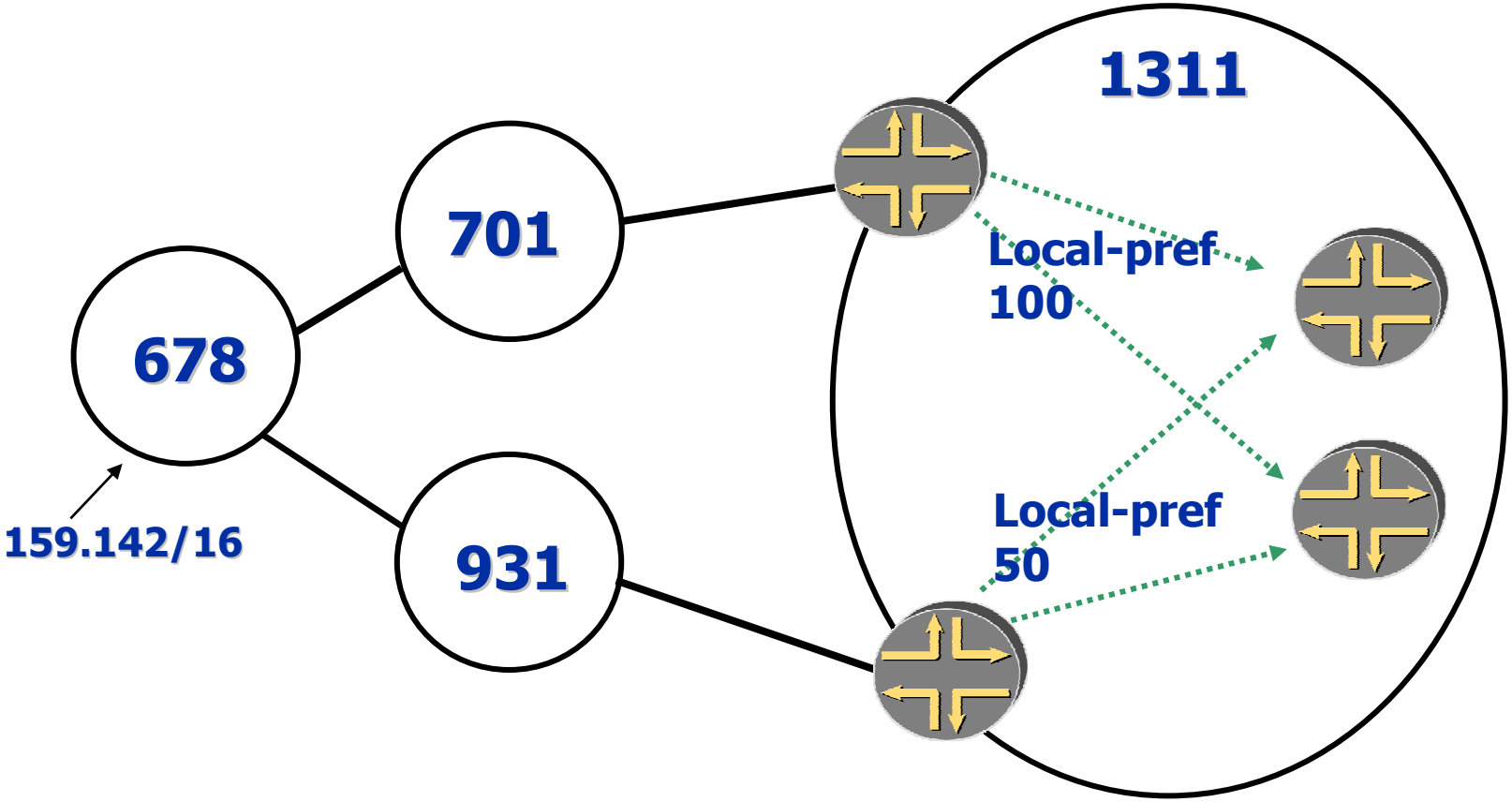
Multi-Exit-Discriminator

Allows an AS to provide information about path selection that may not be obvious to some ASes



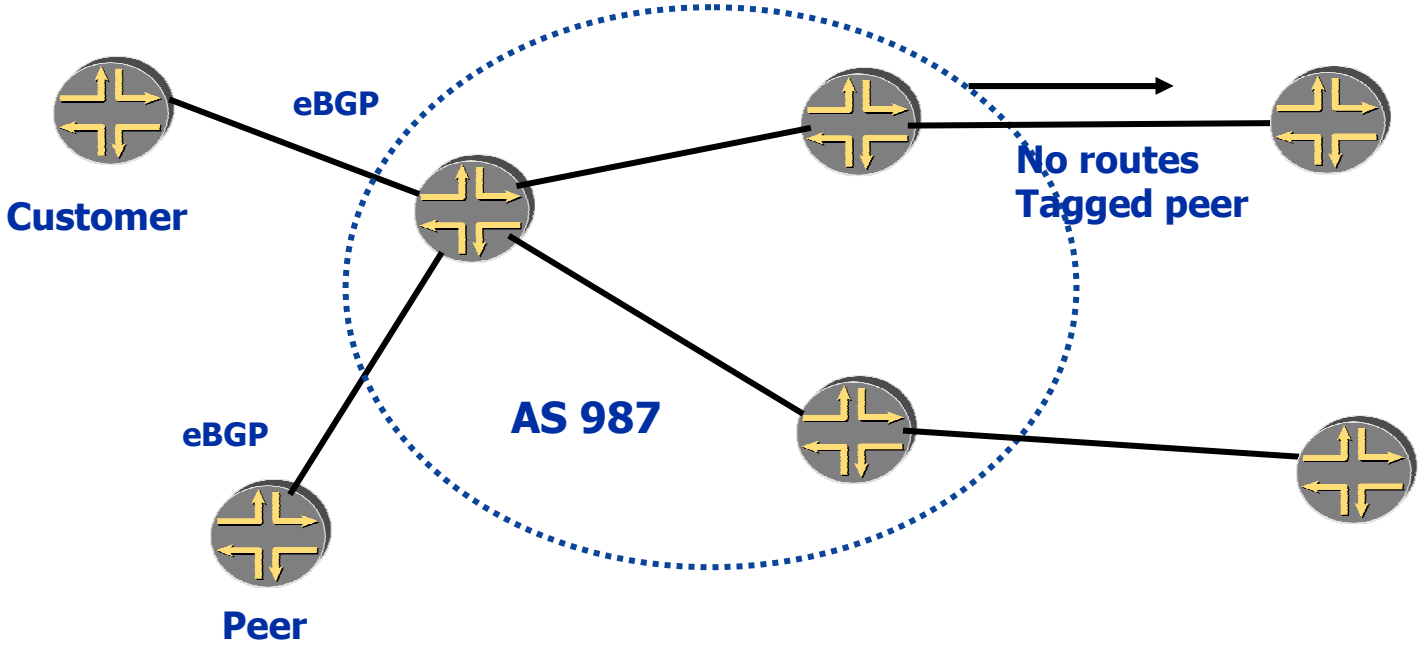
BGP Attributes – Local Preference

Allows an AS to set a value to show internal Routers what exit point is preferred



BGP Attribute - Communities

Used to reduce the complexity of policy maintenance. Tag routes so you can make decisions on them later.



Juniper Default Route Preference Values

Route Source	Default Preference
direct /local	0
Static	5
RSVP	7
LDP	9
OSPF internal route	10
IS-IS Level 1	15
IS-IS Level 2	18
Redirects	30
Generated or aggregate	130
OSPF AS external routes	150
BGP	170

Cisco Default Administrative Distances

Route Source	Default Preference
Connected interface	0
Static route	1
External BGP	20
EIGRP (internal)	90
IGRP	100
OSPF	110
ISIS	115
RIP	120
EGP	140
EIGRP (external)	170
Internal BGP	200
Unknown	255

BGP Route Selection Process (Juniper)

- **Can the BGP next-hop (BNH) be reached?**
  - If Yes, proceed
  - If No, stop processing
- **Prefer the path with the higher local preference.**
- **Prefer the path with the shorter AS path.**
- **Prefer the path with lower Origin code.**
- **Prefer the path with the lower MED.**
- **Prefer paths learned via EBGP over routes via IBGP**
- **Prefer the path with the lower IGP metric to BGP next-hop**
- **Prefer paths where BNH is resolved in inet.3 over inet.0**
- **Prefer paths where BNH has greater number of equal cost paths**
- **Prefer paths with the shortest cluster-List length**
- **Prefer the path with lower router ID (RID)**
- **Prefer the path with the lower peer IP address.**

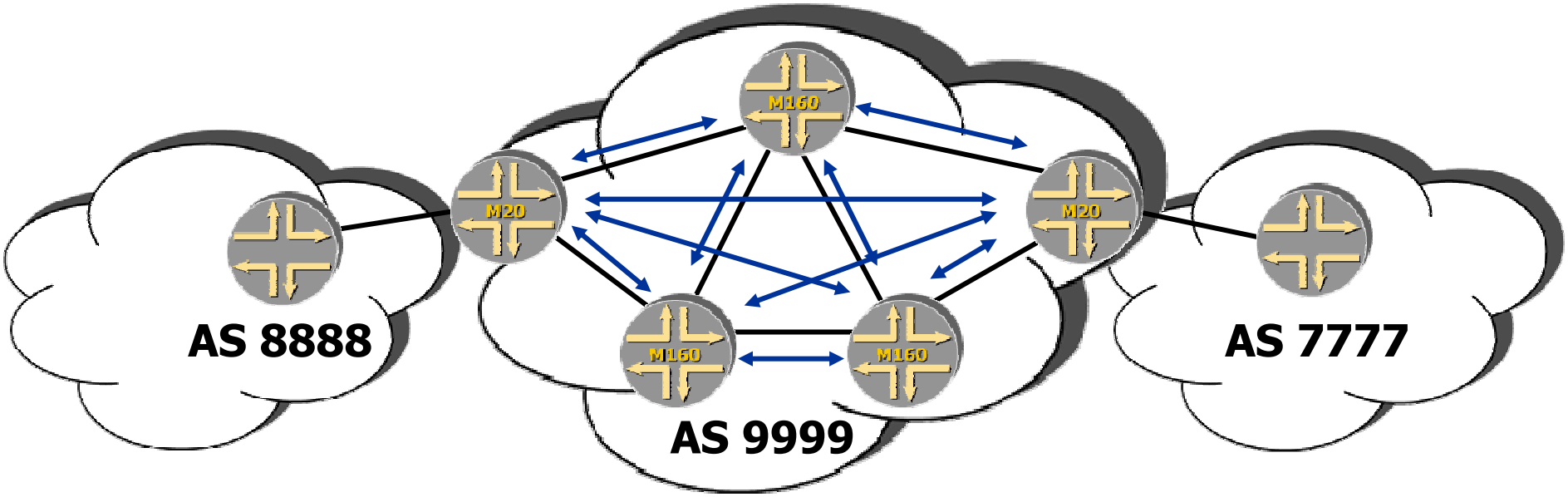
Why Do You Need an IGP?

- **Isn't IBGP enough?**
  - Loopback addresses for iBGP peers
    - Must have a entry in routing table for TCP session to form
  - Internal links
  - NMS

Interior BGP

- **IBGP used inside an AS**
- **Typically implemented as full IBGP mesh**
- **Why do you need a full mesh?**
  - AS-path check not applicable
  - IBGP speaker cannot forward IBGP learned paths to other IBGP speakers
- **BGP next-hop not reset**
- **Local preference used for internal routing policy**
  - Selecting preferred exit point
  - Common practice to set local preference based on community matching
- **Typically peer to loopbacks**

Interior BGP



IBGP Peering

Interior BGP

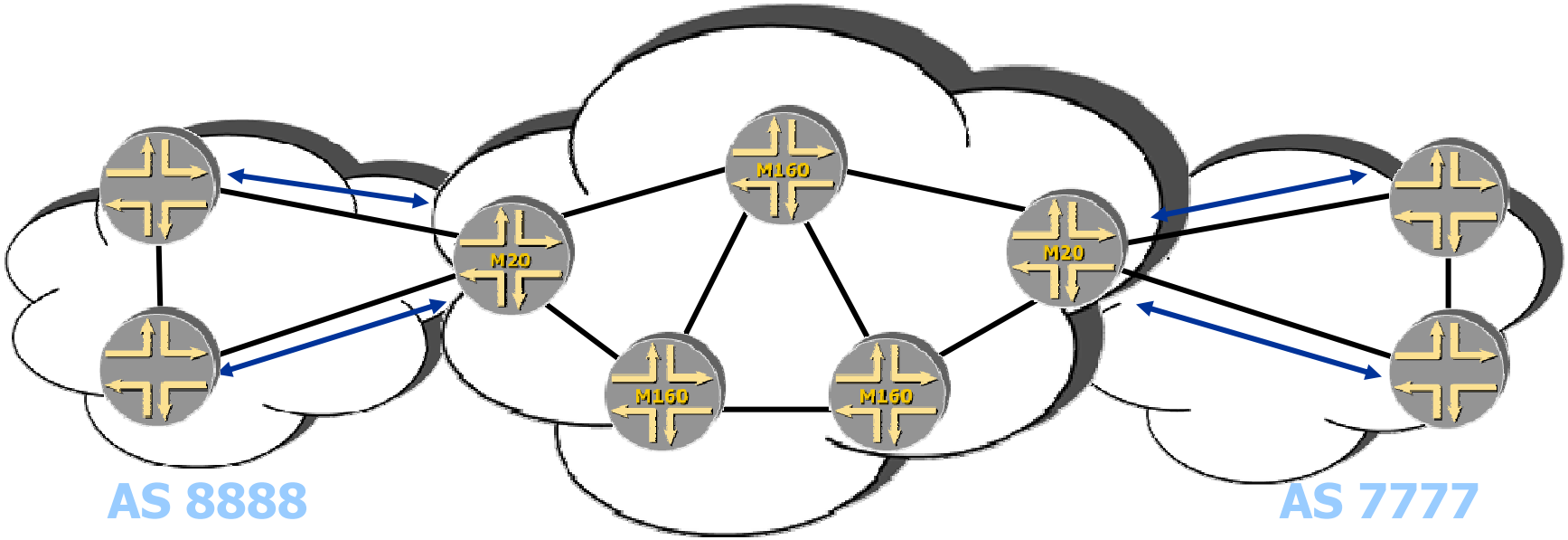
- **What is synchronization?**
  - BGP speakers should not advertise paths unless IGP knows about them
  - Not needed if running full IBGP mesh
- **Sync disabled by default in JUNOS software**
- **Sync on enabled by default on Cisco**

Exterior BGP

- **Used for passing routes between autonomous systems**

- **Differences with IBGP**
  - BGP next-hop is reset
  - AS path is pre-pended
  - MED used for routing policy
    - Selecting preferred entry point
    - Common practice to set MED based on IGP metric
  - Typically peer between physical interface addresses

Exterior BGP



AS 9999

EBGP Peering

EBGP vs IBGP

- **EBGP – between different ASs**
  - Peer to connected interface
- **IBGP – within same AS**
  - Full Mesh Required
  - (exception to full mesh is Route Reflectors and Confederations)
  - Peer to loopback (Local-address command)
  - IGP required for peer TCP reachability

When to Apply Policy

- You do not want to *import* all learned routes into the routing table
- You do not want to *advertise* all learned routes to neighboring routers
- You want one protocol to receive routes from another protocol (redistribution)
- You want to modify information (attributes) associated with a route

JUNOS BGP Route Advertisement

- **JUNOS software default BGP advertisement rules**
  - Active routes only
    - All BGP learned routes (except IBGP rule)
    - Advertise-inactive knob available
  - Export policies needed to
    - Advertise static routes
    - Advertise aggregate routes
    - Advertise default route
    - Redistribute other routes to BGP

BGP Default Policy

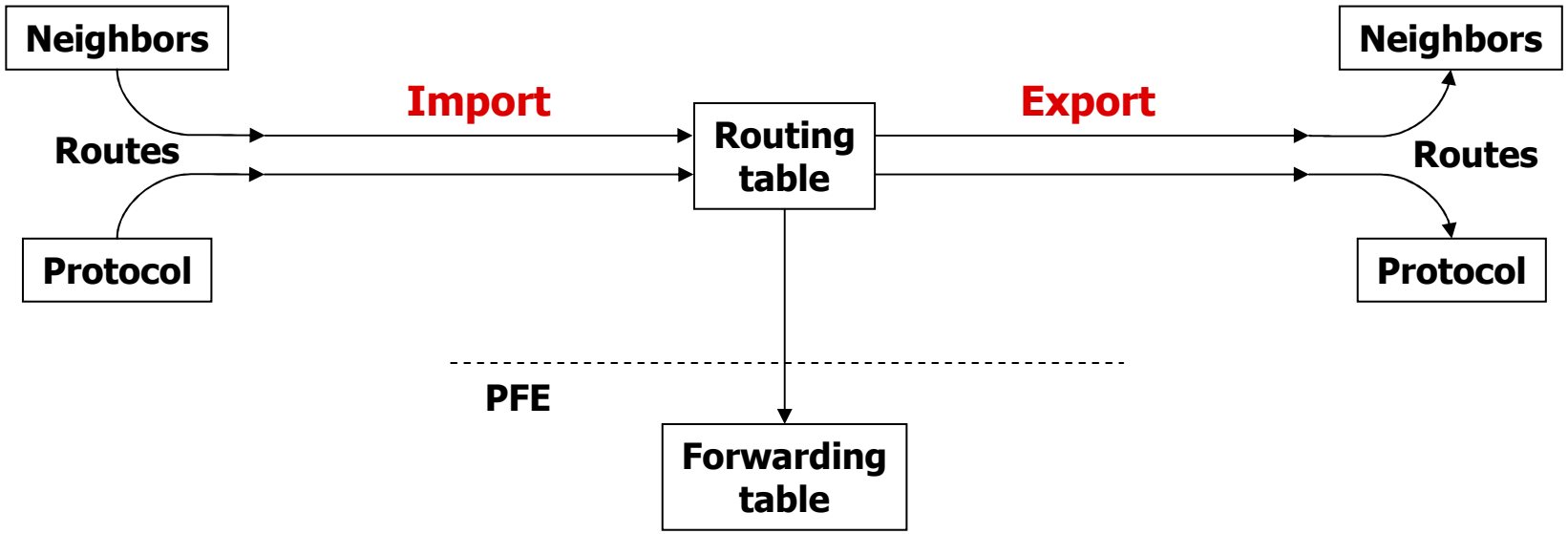
- **BGP**
  - Import
    - All routes learned from BGP neighbors
  - Export
    - Transmit all routes learned from BGP neighbors to all BGP neighbors
    - Only active routes can be exported

Import vs Export

- **Import Policy**
  - Apply an import routing policy to control the routes that the routing protocol process uses to determine active routes.
  - Affect routes that BGP *receives from* a neighbor.

- **Export Policy**
  - Apply an export routing policy to control the routes that a BGP *advertises to* its neighbors.

Import vs Export



Applying Policy

- **BGP global filter syntax**

```
protocols {
   bgp {
      export [ policy-list ];
      import [ policy-list ];
   }
}
```

Import Policy Use

- **Example:**

```
protocols {
    bgp {
        group SomeRegional.ISP {
            type external;
            multihop;
            import customer-routes;
            peer-as 500;
            neighbor 6.6.6.6;
            }
        }
}
```

Export Policy Use

- **Example:**

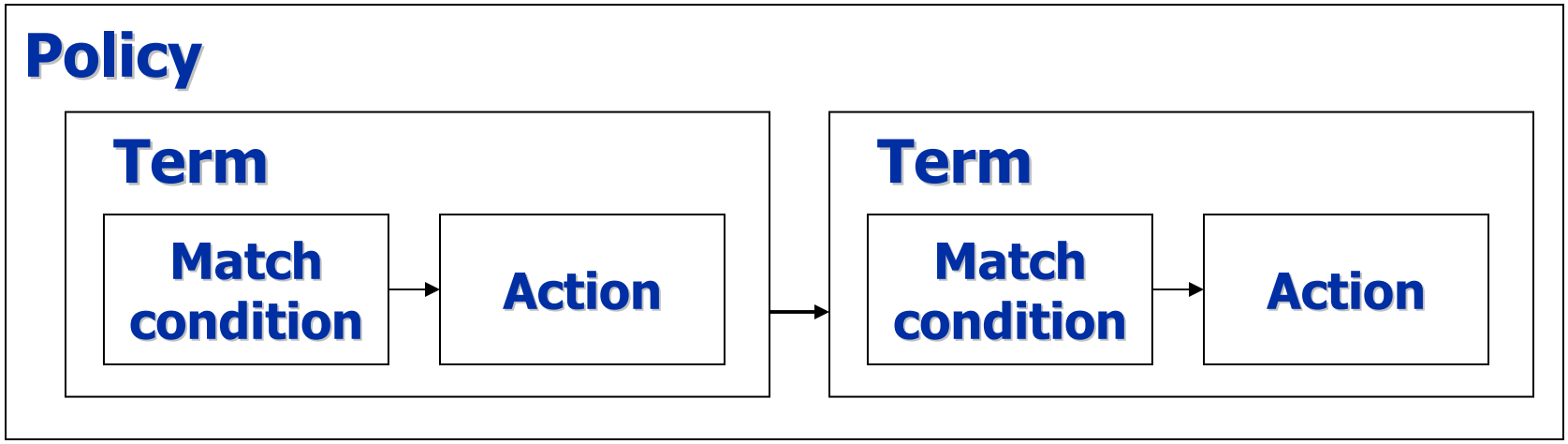
```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Internal-Peers {
            type internal;
            export nexthopself;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
            neighbor 9.9.9.9;
```

You can have both Import/Export

```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Some-Customer {
            import customer-routes;
            export advertise-policy;
            type external;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
            neighbor 9.9.9.9;
            }
        }
}
```

Configuring Policy

- Policies are made up of terms
- Terms are made up of match conditions and actions
- Match conditions can be split into “from” and “to” parts



Basic Policy syntax

```
policy-options {
   policy-statement policy-name {
         term term-name {
               from {
                     match-conditions;
               }
               then {
                     action;
               }
         }
         final-action;
   }
}
```

Match Conditions

- **General**
  - Route metrics
    - Metric
    - Preference
    - Color
  - Interface name
  - Neighbor address
  - Next-hop address
  - Protocol
    - bgp, direct, dvmrp, isis, local, mpls, ospf, pim-dense, pim-sparse, rip, static, aggregate

Match Conditions

- **Some match conditions are protocol specific**

- **OSPF**
  - Area ID
  - Tag and tag2 fields
- **IS-IS**
  - Level number
- **BGP**
  - Autonomous system path (AS path)
  - Community name
  - Local preference
  - Origin

Match Actions

- **Modify**
  - Metric
    - (protocol specific)
  - Preference
    - (global routing preference)
  - Color
  - Next-hop address

Match Actions

- **Modify**
  - OSPF
    - Type 1 or type 2 external link advertisement
    - Tag and tag2 fields
  - BGP
    - Prepend AS path
    - Add, delete, or set community
    - Change route damping parameters
    - Change local preference value
    - Change protocol origin

Policy Match Actions

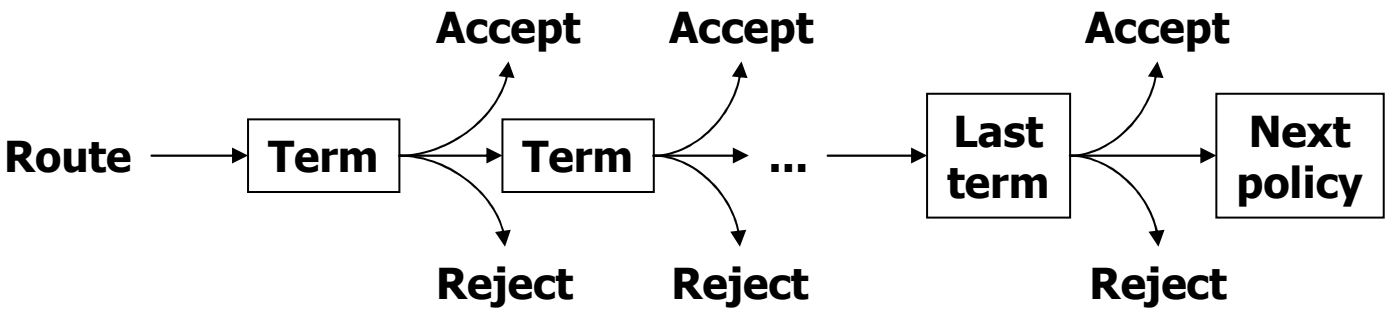
- **Terminate**
  - Accept route
  - Reject (or suppress) route
- **Flow Control**
  - Skip to next policy
  - Skip to next term
- **Trace**
  - Log the match to a trace file, continue processing term

Policy Match Example

```
policy-statement advertise-policy {
    term advertise {
        from community transit;
        then next policy;
        then accept;
    }
    term catch-all {
        then reject;
    }
```

Policy Terms

- Policies contain collections of terms
- Terms contain a condition and an action to apply to each route



Policy Terms

- **Example:**

```
policy-statement advertise-policy {
    term advertise {
        from community transit;
        then accept;
    }
    term do-not-advertise {
        from community Tier-1;
        then reject;
    }
    term catch-all {
        then reject;
    }
```

BGP Filtering points

- BGP has three filtering points
  - Global
  - Groups of neighbors
  - Individual neighbors

  - Neighbor policy overrides group and global policies
  - Group policy overrides global policy

BGP Filtering points

```
protocols {
    bgp {
        export nexthopself1;
        local-address 5.5.5.5;
        group Internal-Peers {
            type internal;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
            neighbor 9.9.9.9;
            }
        }
}
```

BGP Filtering points

```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Internal-Peers {
            export nexthopself2;
            type internal;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
            neighbor 9.9.9.9;
            }
        }
}
```

BGP Filtering points

```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Internal-Peers {
            type internal;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
              export nexthopself3;
            neighbor 9.9.9.9;
            }
        }
}
```

BGP Filtering points

```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Internal-Peers {
            export nexthopself;
            type internal;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
              export localpref;        << localpref over rides nexthopself
            neighbor 9.9.9.9;
              }
        }
}
```

Advertising Networks

- **A network that resides within an AS is said to originate from that network. To inform other ASs about its networks, the AS advertises them. BGP provides three ways for an AS to advertise the networks that it originates:**
  - Redistributing Dynamic Routes
  - Redistributing Static Routes
  - Redistributing Aggregates

Redistributing Dynamic Routes

- This is typically not done.
- Causes instability in the internet

- BAD BAD BAD BAD

Redistributing Dynamic Routes

```
protocols {
    bgp {
        group peer-to-someNET {
            export redistribute-ospf;
            peer-as 9999;
            neighbor 23.43.12.16;
        }
    }
}
policy-options {
    policy-statement redistribute-ospf {
        from {
            protocol ospf;
        }
        then accept;
    }
}
```

BAD BAD BAD BAD

Redistributing Static Routes

- **This is the Juniper equivalent of Cisco's null0 route and network statement**

```
routing-options {
    static {
        route 9.0.0.0/8 discard;
    }
}
```

- **Reject**
  - Drop packets to destination; send ICMP unreachables
- **Discard**
  - Drop packets to destination; send no ICMP unreachables

The whole config

```
routing-options {
    static {
        route 9.0.0.0/8 discard;
    }
}
protocols {
    bgp {
        group peer-to-BIGNET {
            export redistribute-static;
            peer-as 9999;
            neighbor 23.43.12.16;
        }
    }
}
policy-options {
    policy-statement redistribute-static {
        from {
            protocol static;
            route-filter 9.0.0.0/8 exact;
        }
```

Redistributing Aggregate Routes

- Route aggregation allows you to combine groups of routes with common addresses into a single entry in the routing table.
- This decreases the size of the routing table as well as the number of route advertisements sent by the router.

Redistributing Aggregate Routes

- An aggregate route becomes active when it has one or more contributing routes.
- A contributing route is an active route that is a more specific match for the aggregate destination.

Redistributing Aggregate Routes

```
routing-options {
    aggregate {
        route 9.0.0.0/8;
    }
}
  protocols {
    bgp {
        group peer-to-BIGNET {
            export redistribute-aggregates;
            peer-as 9999;
            neighbor 23.43.12.16;
        }
    }
}
policy-options {
    policy-statement redistribute-aggregates {
        from {
            protocol aggregate;
            route-filter 9.0.0.0/8 exact;
        }
        then accept;
```

Hands on

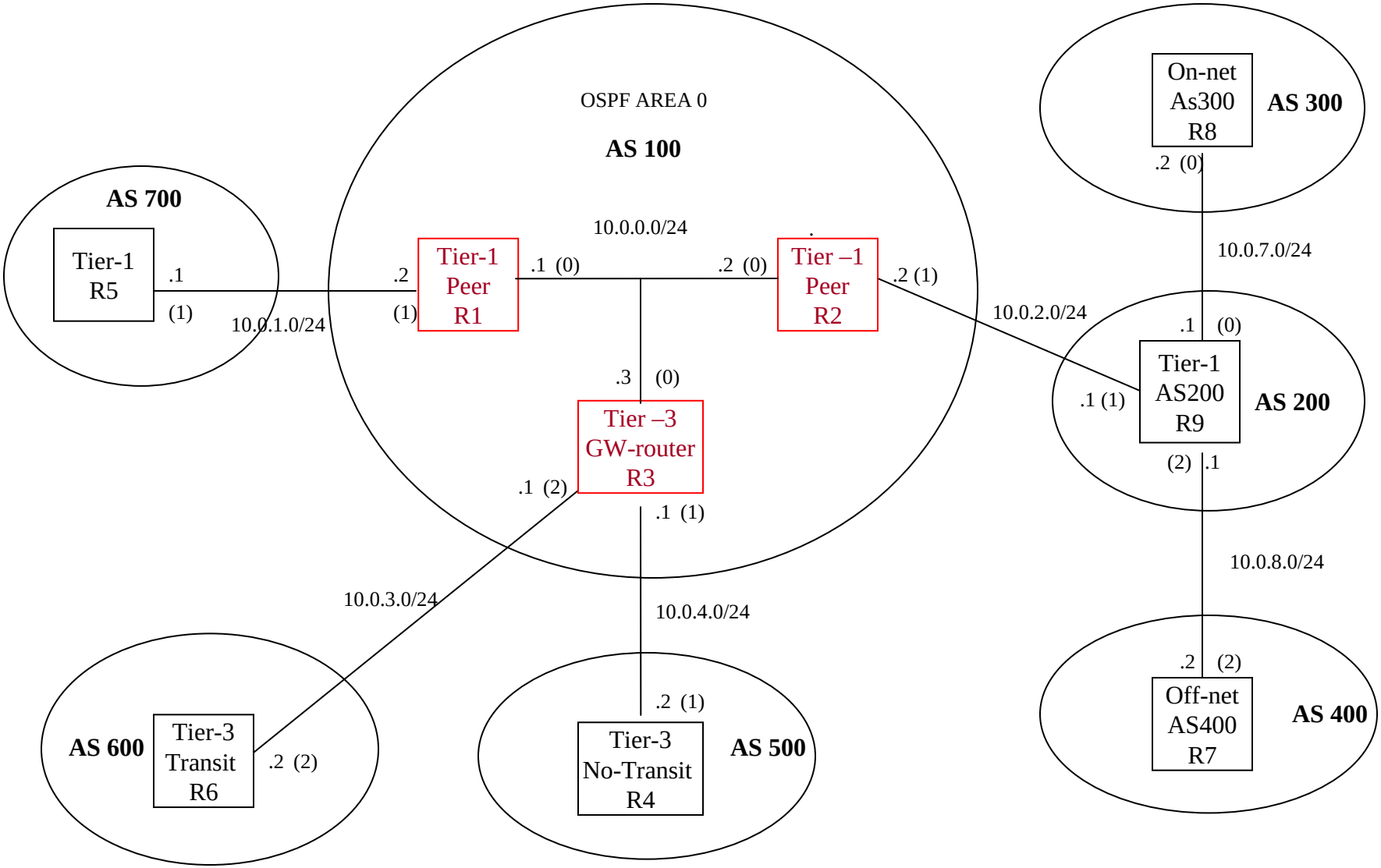
Next-hop Self

- Next-hop is Well-known mandatory attribute
- Set the next hop address
- If you specify address as self, the next-hop address is set to the local IP address used for the BGP adjacency

Resolving BGP Next Hops

- **Two common methods**
  - Next-hop self
    - Resets BGP next-hop address when advertising to internal peers
  - Passive interface
    - Adds external link subnet to IGP database
    - Allows peer router to be pinged from internal network

Next Hop Self Example



Next-hop Self Example

- **Create the policy**

```
policy-options {
    policy-statement nexthopself {
        from protocol bgp;
        then {
            nexthop self;
            next-policy;
```

Next-hop Self Example

- **Apply the policy on border routers**

```
protocols {
    bgp {
        local-address 5.5.5.5;
        group Internal-Peers {
            type internal;
            export nexthopself;
            neighbor 2.2.2.2;
            neighbor 1.1.1.1;
            neighbor 9.9.9.9;
```

Route Filter

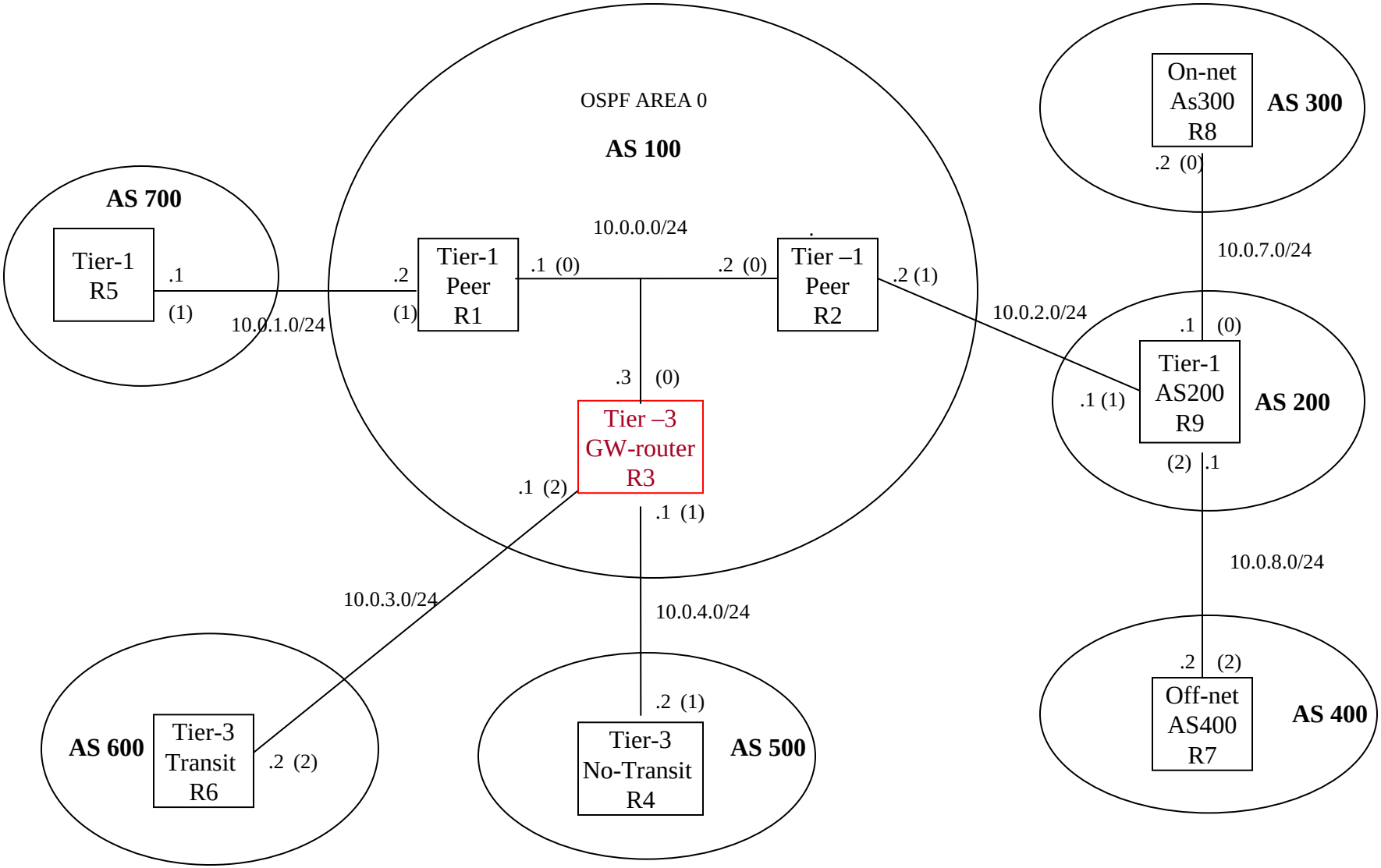
- To specify route prefixes in policies, include one or more route-filter options in the from statement of the policy-statement statement
- Multiple route filters in a single term

Route Filters

```
[…]
    term term-name {
        from {
            route-filter prefix/prefix-length match-type <actions>;
            […]
        }
        <then actions>;
    }
[…]
```

- **Match type can be**
  - Exact, orlonger, longer, upto, through
- **Multiple route filters in a single term**
- **Evaluation of route filters has special rules**

Route Filter



Route Filter Example 1

- **Create the Policy on the gateway router**

```
policy-options {
policy-statement AS600customer-routes {
    term routes-to-accept {
        from {
            protocol BGP;
            route-filter 1.0.0.0/8 exact;
            route-filter 2.0.0.0/8 exact;
            route-filter 3.0.0.0/8 exact;
            route-filter 4.0.0.0/8 exact;
        }
        then accept;
    }
    term reject-routes {
        then reject;
    }
```

Logical OR function

Route Filter Example 1

- **Apply the policy on the gateway router**

```
protocols {
    bgp {
        group As600Regional.ISP {
            type external;
            multihop;
            import AS600customer-routes
            peer-as 500;
            neighbor 6.6.6.6;
```

Match Types—exact

- **Exactly match a single prefix and prefix length**

```
term sample {
     from route-filter 192.168/16 exact;
     then accept;
 }
```

Includes

192.168.0.0/16

Excludes

Everything else

Match Types—orlonger

- Greater than or equal to
- Match a range of routes having the most-significant bits in common as described by the prefix length

```
term sample {
     from route-filter 192.168/16 orlonger;
     then accept;
 }
```

Includes

192.168.0.0/16	192.168.12.4/30
192.168.0.0/17	192.168.12.128/32
192.168.4.0/24	

Excludes

192.0.0.0/8	192.169.1.0/24
192.170.0.0/16	

Match Types—longer

- **Match a range of routes having the most-significant bits in common as described by the prefix length, except the exact match**
- **Greater than**

```
term sample {
     from route-filter 192.168/16 longer;
     then accept;
 }
```

Includes

192.168.0.0/17
192.168.4.0/24
192.168.12.4/30
192.168.12.128/32

Excludes

192.0.0.0/8
192.170.0.0/16
192.169.1.0/24
192.168.0.0/16

Match Types—upto

- **Match a range of routes having the most-significant bits in common as described by the first prefix length, but not exceeding the second prefix length**

```
term sample {
     from route-filter 192.168/16 upto /24;
     then accept;
 }
```

Includes

192.168.0.0/16
192.168.0.0/17
192.168.4.0/24

Excludes

192.0.0.0/8
192.169.1.0/24
192.170.0.0/16
192.168.5.4/30
192.168.12.128/32

Match Types—through

- Match a contiguous set of routes from the first prefix-prefix length pair to the second prefix-prefix length pair
- Not used very often–covers a corner case

```
term sample {
    from route-filter 192.168/16 through 192.168.16/20;
    then accept;
}
```

Includes	
192.168.0.0/16	192.168.0.0/19
192.168.0.0/17	192.168.16.0/20
192.168.0.0/18	

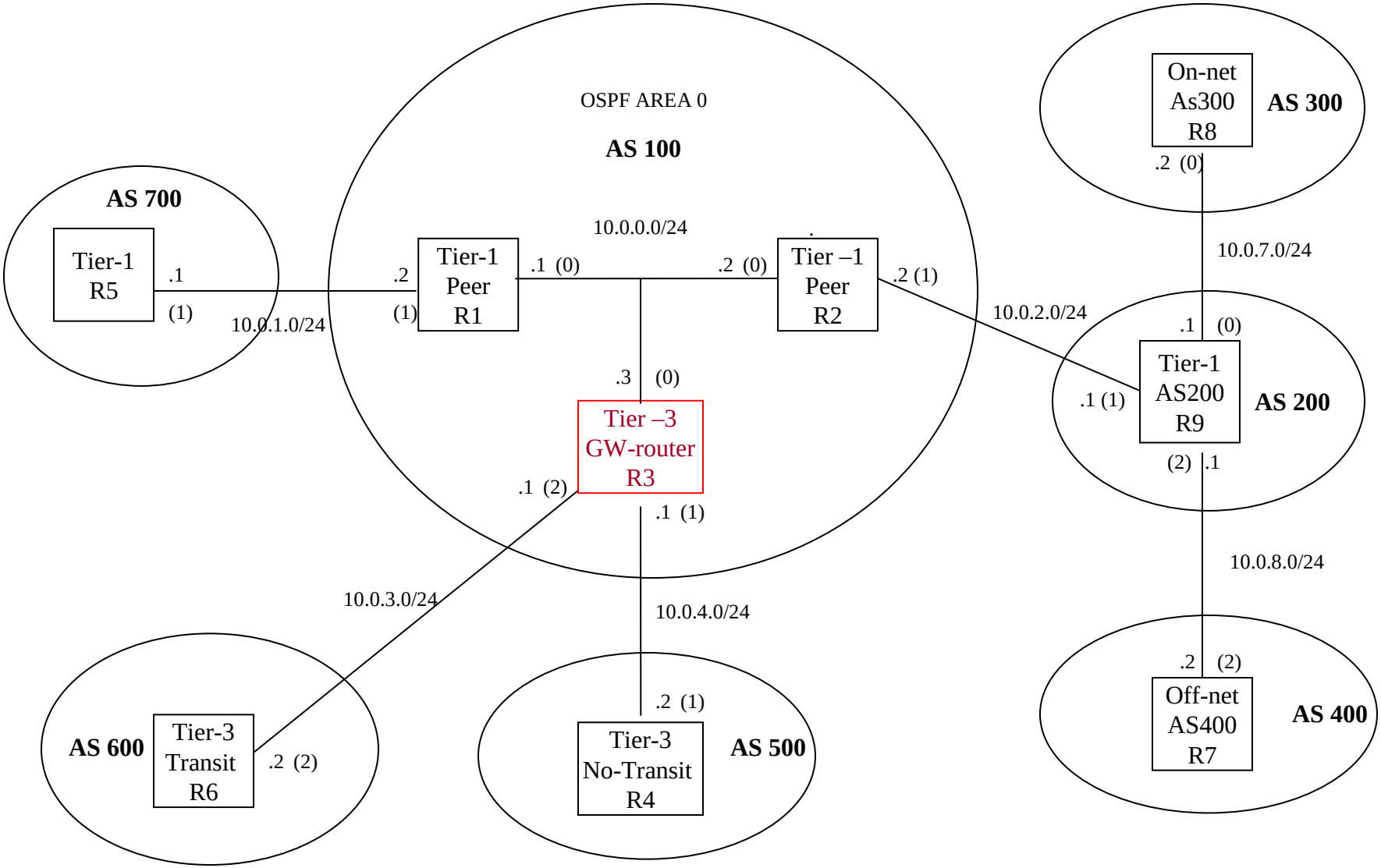
Excludes	
192.168.128.0/17	192.168.0.0/20
192.168.192.0/18	
192.168.224.0/19	

Prefix List

Prefix List

- You can define and name a set of IP address prefixes and use them in the configuration for routing policy statements and firewall filters.
- Note: Per-prefix policy actions cannot be applied to individual prefixes in the list or to the collection of prefixes in the list. It is all or nothing.

Prefix List



Prefix List Example

- **Create the named prefix list**

```
[edit policy-options]
prefix-list ISP1-acceptable-routes {
    1.0.0.0/8;
    2.0.0.0/8;
    3.0.0.0/8;
}
```

Prefix List Example

- **Use the named prefix list in a policy**

```
policy-statement customer-routes {
    term routes-to-accept {
        from {
            prefix-list ISP1-acceptable-routes;
        }
        then accept;
    }
    term reject-routes {
        then reject;
    }
    then next policy;
```

Local Preference

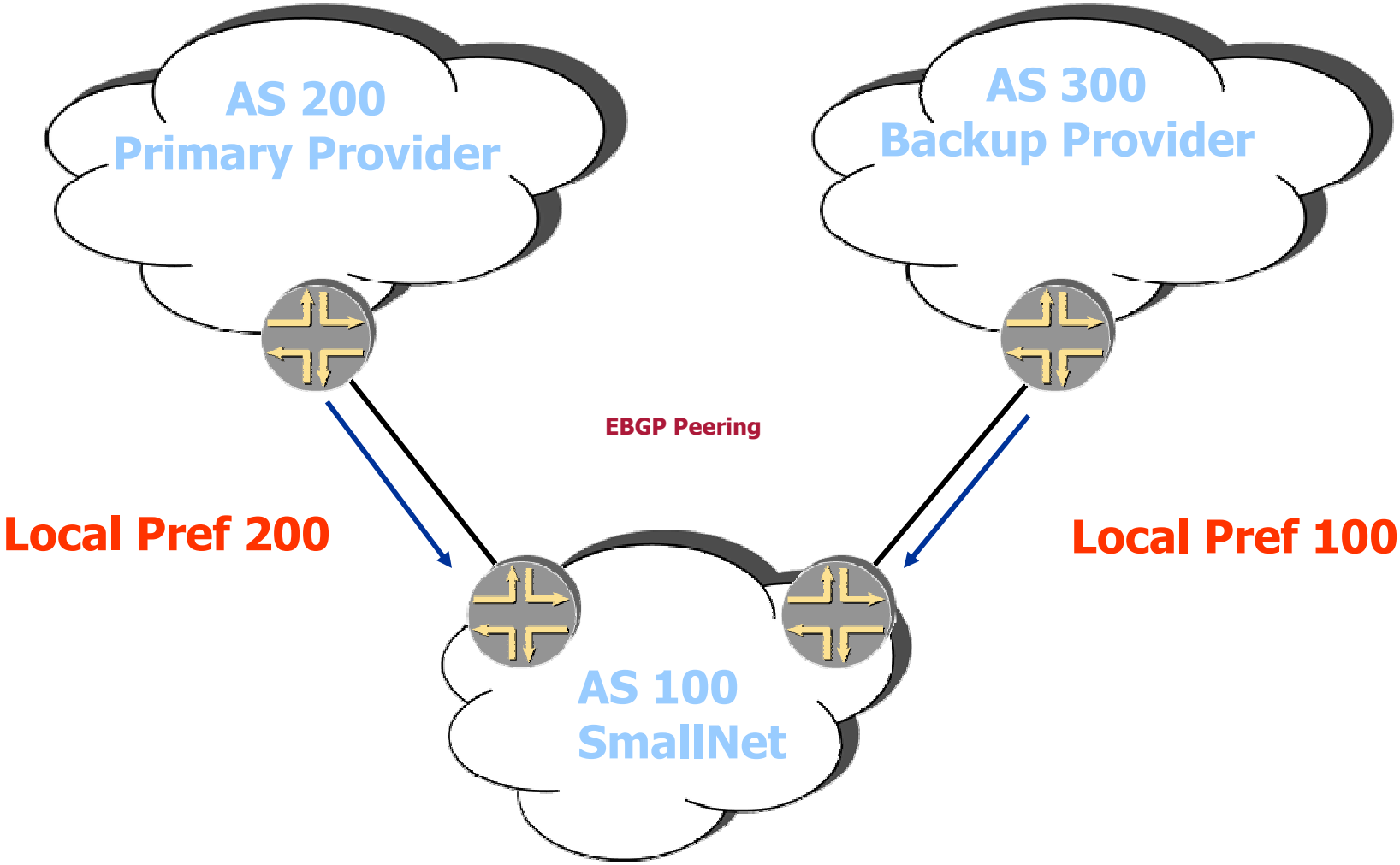
Local Preference

- **Well-known, discretionary**
- **Used with Internal BGP**
  - Carried in internal BGP update packets in the path attribute LOCAL_PREF.
- **This metric indicates the degree of preference for an external route.**
- **The route with the *highest* local preference value is preferred.**
- **Default local preference is 100.**
  - (Highest is 2^32)

Local Preference

- **Two ways to set local preference**
  - On the peer
  - In a policy

Local Preference Example

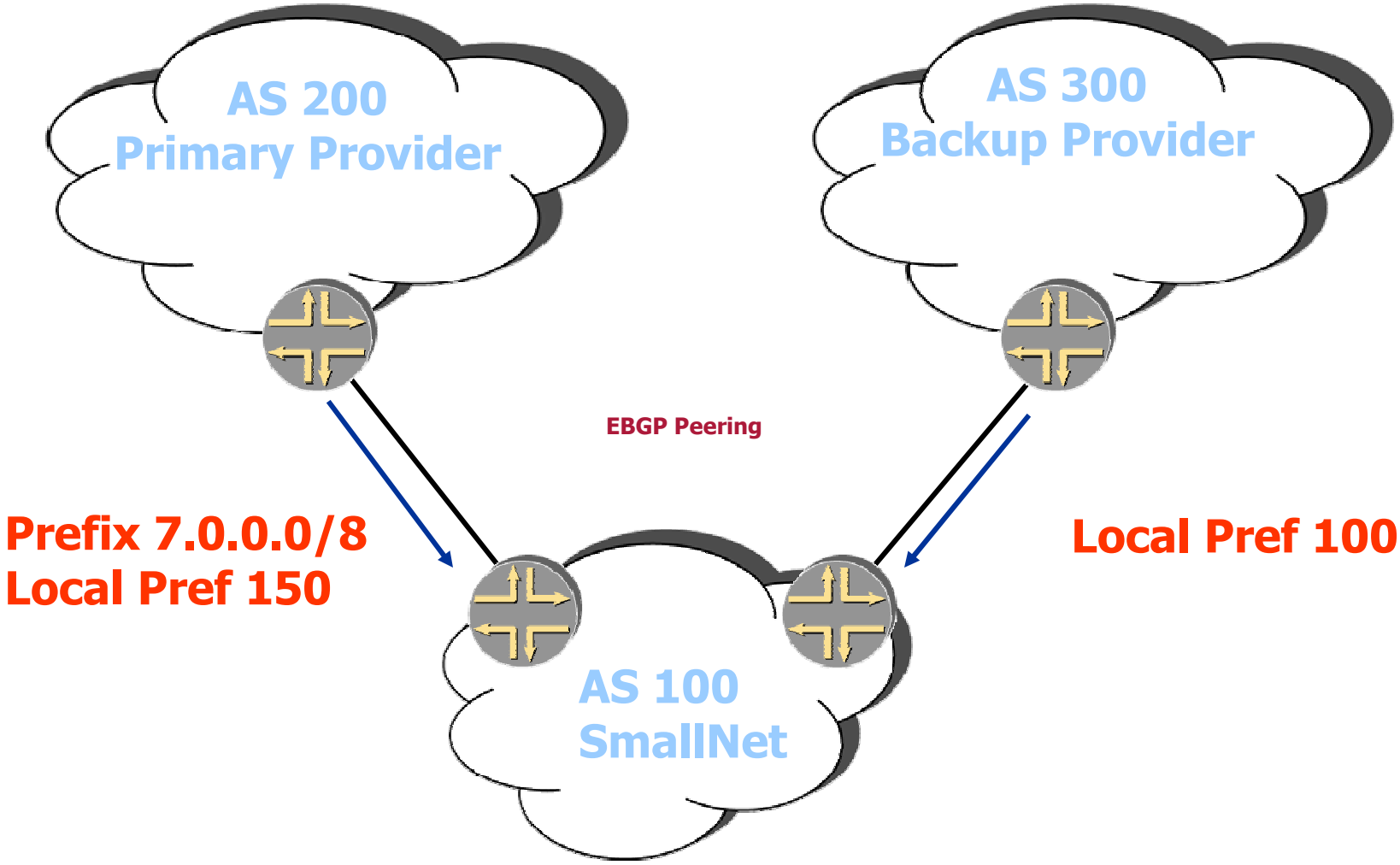


Local Preference on the peer

```
protocols {
    bgp {
        group Primary-provider-AS200 {
        type external;
        multihop ttl 2;
        peer-as 200;
        local-preference 200;
         local-address 2.2.2.2;
        neighbor 3.3.3.3;
```

All routes learned from this peer will be have local preference of 200

Local Preference Example



Local Preference in a policy

- **Create the Policy**

```
policy-statement routes-from-Primary-provider {
        term path1 {
            from {
                route-filter 7.0.0.0/8 exact;
            }
            then {
                local-preference 150;
                accept;
            }
        }
```

Local Preference in a policy

- **Apply the policy**

```
protocols {
    bgp {
        group Primary-provider-AS200 {
            type external;
            import  routes-from-Primary-provider;
            peer-as 200;
            neighbor 10.0.1.2;
```

Multiple Exit Discriminator (MED)

Multiple Exit Discriminator (MED) Attribute

- **Optional non-transitive attribute**
- **Helps describe which of multiple exit points is the optimal or more preferred link.**
- **Commonly used if one AS connects to another AS in more than one place (that is, there are multiple entry points)**
- **MED represents the external metric of a route.**
- **If the route needs to be readvertised to another AS, the MED value must be reset to zero, unless the associated export policy sets an outgoing MED value.**
- ***The route with the lower MED is more preferred*.**

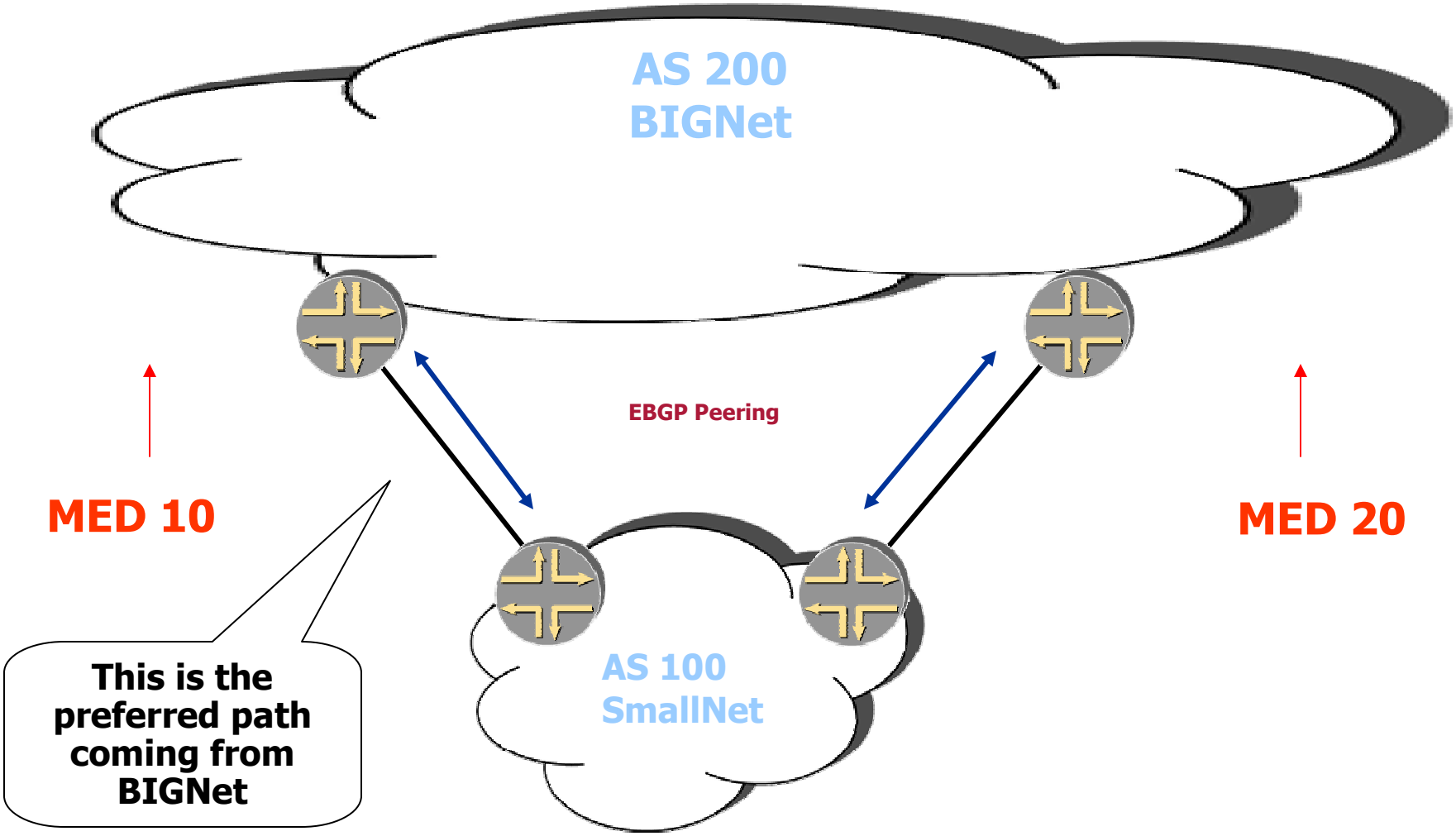
Multiple Exit Discriminator (MED) in a Policy

- **Apply the policy**

```
protocols {
    bgp {
        group Peer-to-BIGNet {
            type external;
            export setMED
            peer-as 100;
            neighbor 10.0.1.2;
```

- **Route 209.1.1/24 will be advertised with a Metric (MED) of 5**

MED Example



Multiple Exit Discriminator (MED) on the peer

- Outgoing metric (metric-out) value on the peer.

```
protocols {
    bgp {
        group Peer-to-BIGNet {
            type external;
            peer-as 200;
            metric-out 20;
            neighbor 200.1.1.2;
        }
```

All routes advertised from this peer will be have a metric (MED) of 20

Multiple Exit Discriminator (MED) in a Policy

- **In a policy to set an outgoing metric (metric-out) value.**

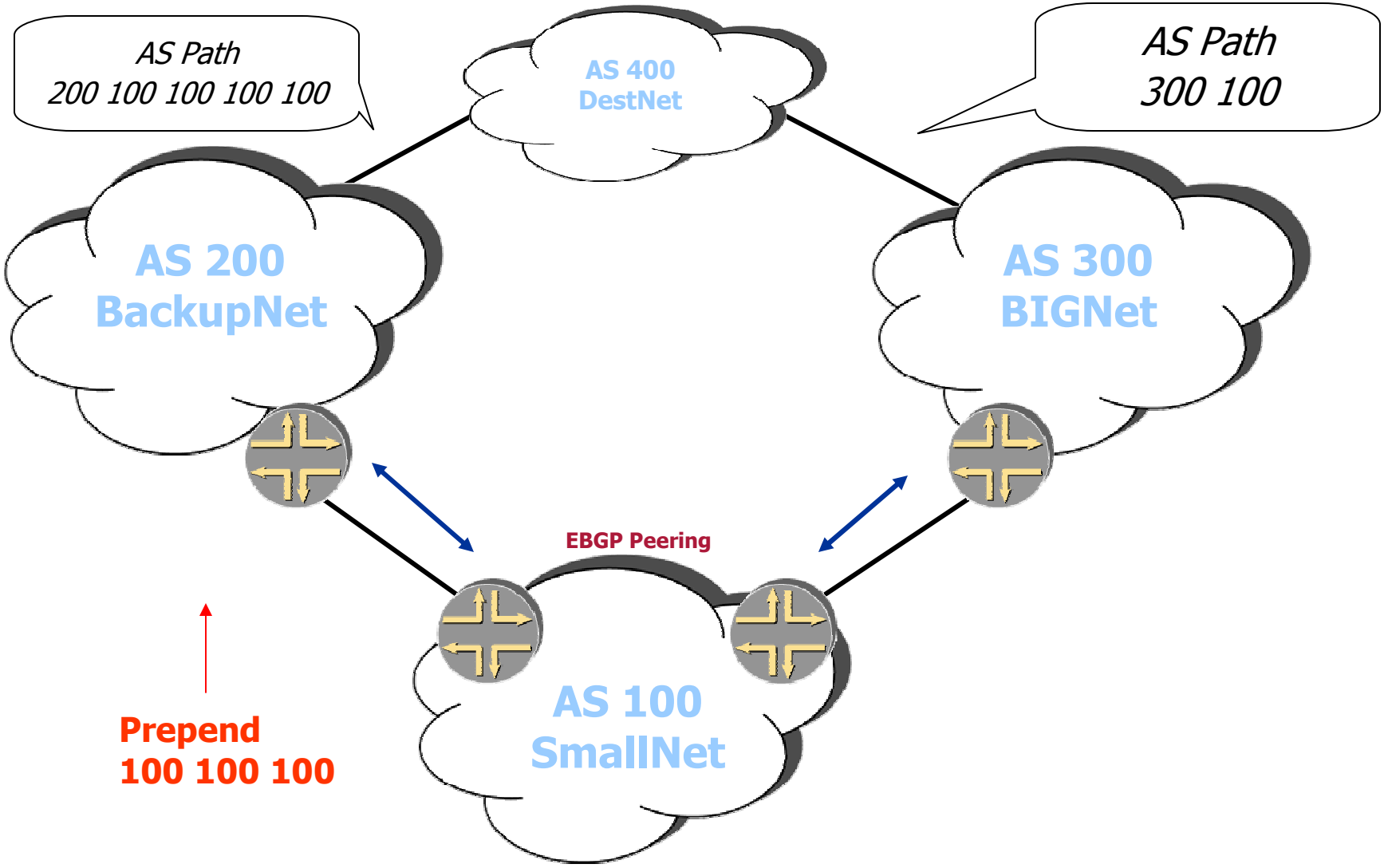
```
policy-options {
    policy-statement setMED {
        from {
            route-filter 209.1.1/24 exact;
        }
        then {
            metric 5;
            accept;
```

AS Path Prepend

AS Path Prepend

- **AS PATH - Well-known mandatory attribute**
- **The AS PATH attribute lists the sequence of AS numbers that the route has passed**
- ***BGP prefers the route with the shortest AS PATH***
- **With AS path prepending, dummy AS numbers are prepended to the AS path to increase the AS path length.**

AS Path Prepend



AS Path Prepend example

- **Create the Policy**

```
policy-options {
    policy-statement prepend {
        from protocol bgp;
        then {
            as-path-prepend "100 100 100";
            accept;
        }
    }
}
```

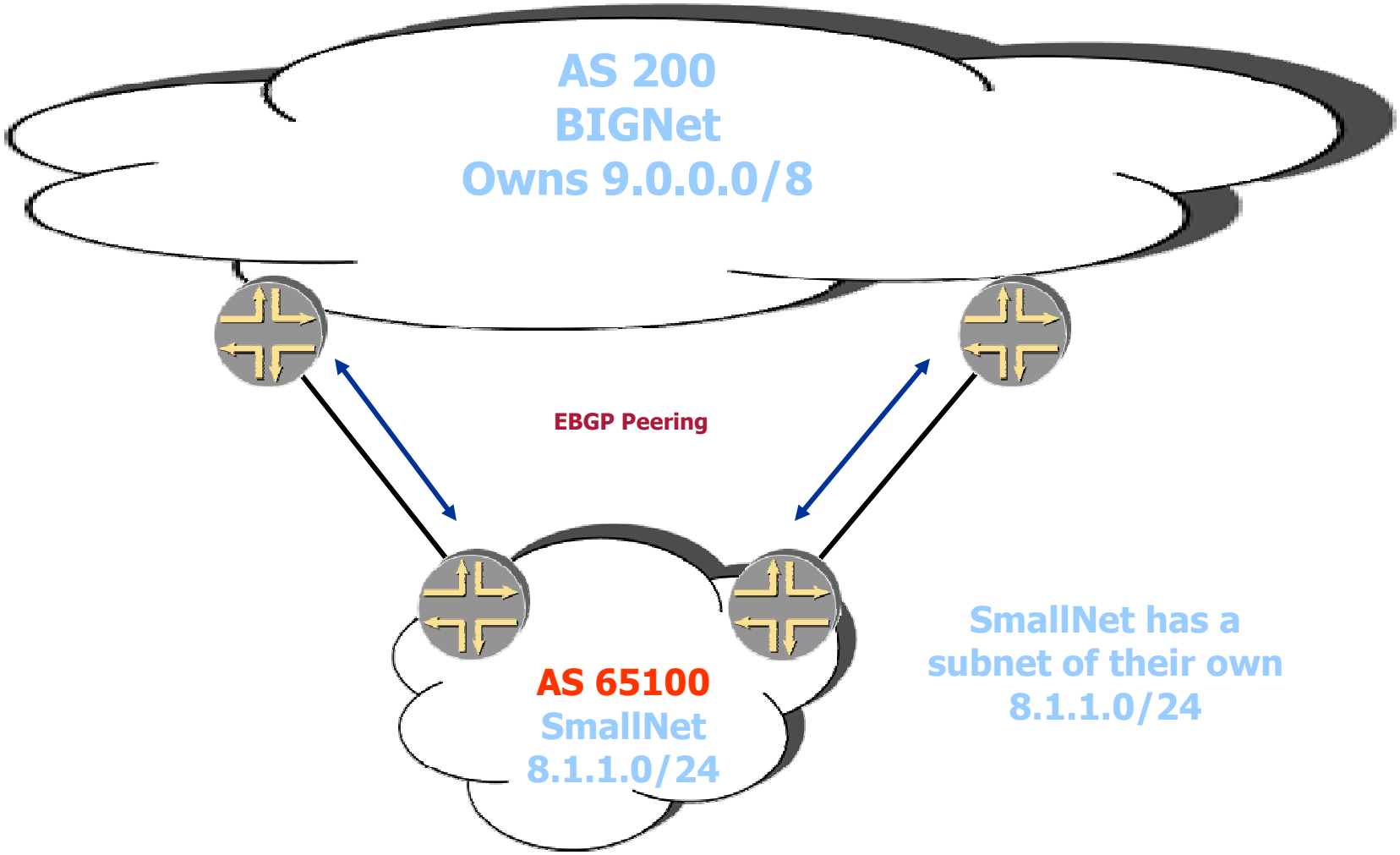
AS Path Prepend example

- **Apply the Policy**

```
protocols {
    bgp {
        local-address 1.1.1.1;
        group BackupNET {
            type external;
            import Tier1-community;
            export prepend;
            peer-as 300;
            neighbor 4.4.4.4;
```

Remove Private ASs

Remove Private AS



Remove Private ASs

- By default, BGP includes all AS numbers when advertising routes to its peers.
- The JUNOS BGP implementation allows the removal of private AS numbers from the AS path list.
- Private AS numbers are those in the range of 64512 through 65535.

Remove Private ASs Example

```
protocols {
   bgp {
      group Some-other-peer {
         type external;
         peer-as 64560;
         neighbor 172.17.4.100;
      }
      group SmallNet-AS300 {
         type external;
         peer-as 65100;
         remove-private;
         neighbor 200.1.1.2;
      }
```

All private ASs will be remove from the AS Path in advertisements sent from this peer

Communities

Communities

- **Optional transitive attribute**
- **RFCs 1997 and 1998**
- **Group destinations that share a common property**
- **Perform action on entire group**
- **Community associations sent in BGP update message**
- **Multiple communities can be associated with a single destination**
- **With routing policy, you can**
  - Create a community
  - Match communities using regular expressions
  - Set, add, or delete communities

Communities

- **Named Community**
  - *Private communities* are user-specific community attributes that are defined for special purposes
  - AS number is usually encoded in the first 2 bytes and the remaining 2 bytes are AS-specific
    - *as-number*:*community-value*
- **JUNOS supports the well known community names defined in RFC 1997**
  - no-export—Don't advertise outside of AS or confederation
  - no-advertise—Don't advertise to other BGP peers
  - no-export-subconfed—Don't advertise to external BGP peers, including peers in other members' ASs inside a BGP confederation

Communities

- **Possible values are 0-65535:0-65535**
- **0:0 – 0:65535 reserved**
  - (so don’t use zero in the AS part)
- **65535:0 – 65535:65535 reserved**
  - (so don’t use 65535 in the AS part)

Creating Communities

- **Define at policy-options level**

```
[edit policy-options]
set community name members [community-ids]
```

- **Where**
  - `name` identifies community
  - `community-ids` specifies one or more members

    as-number:*community-value*

- **Example**

```
policy-options {
    community foobar members 64512:666
    community no-internal-BGP members [64512:111
    64512:222]
}
```

Logical AND function

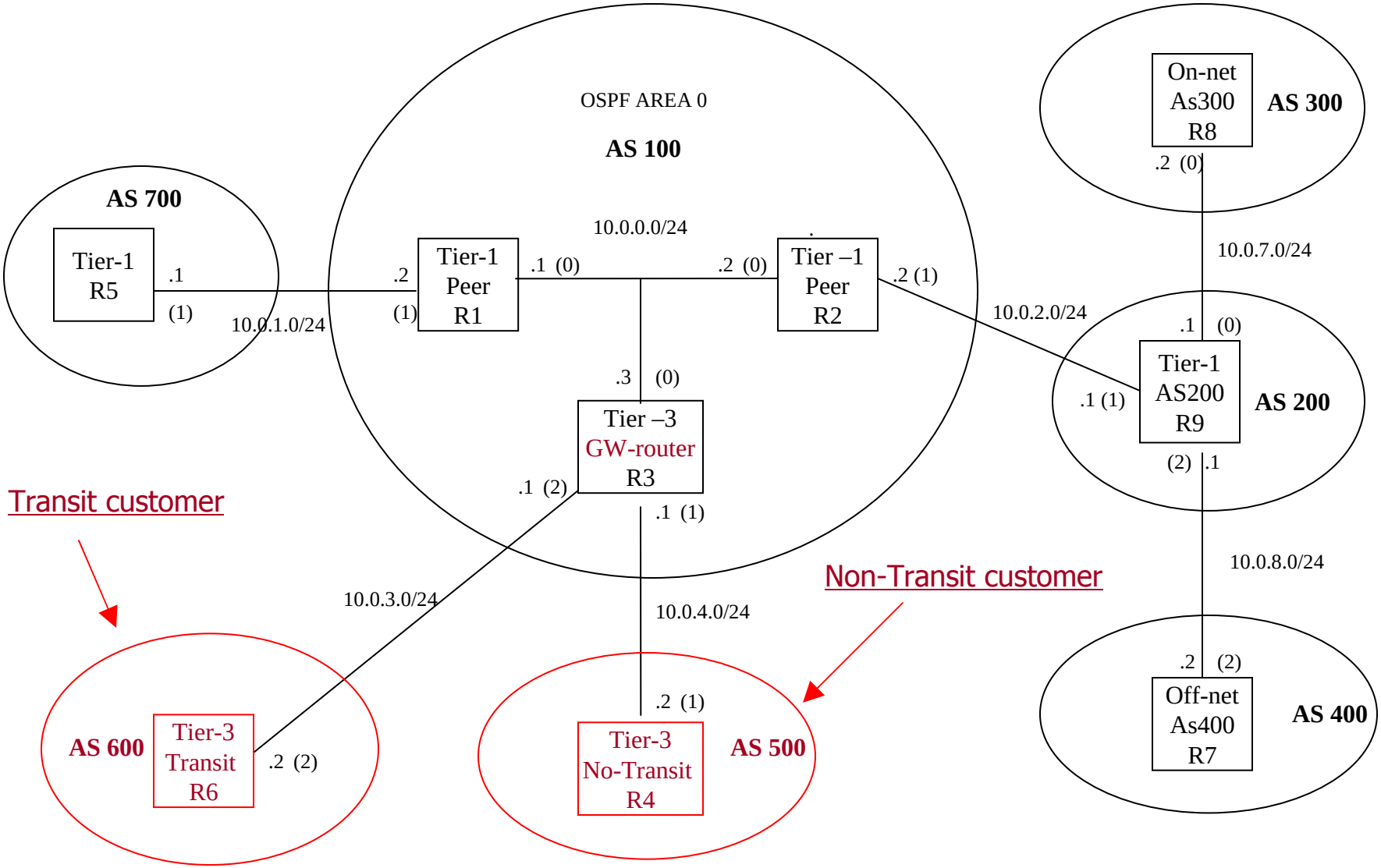
Community Actions

- **Three possible actions**
  - `add`—Add the community to current set of communities
  - `delete`—Delete the community from current set of communities
  - `set`—Set the the community, replacing any existing ones

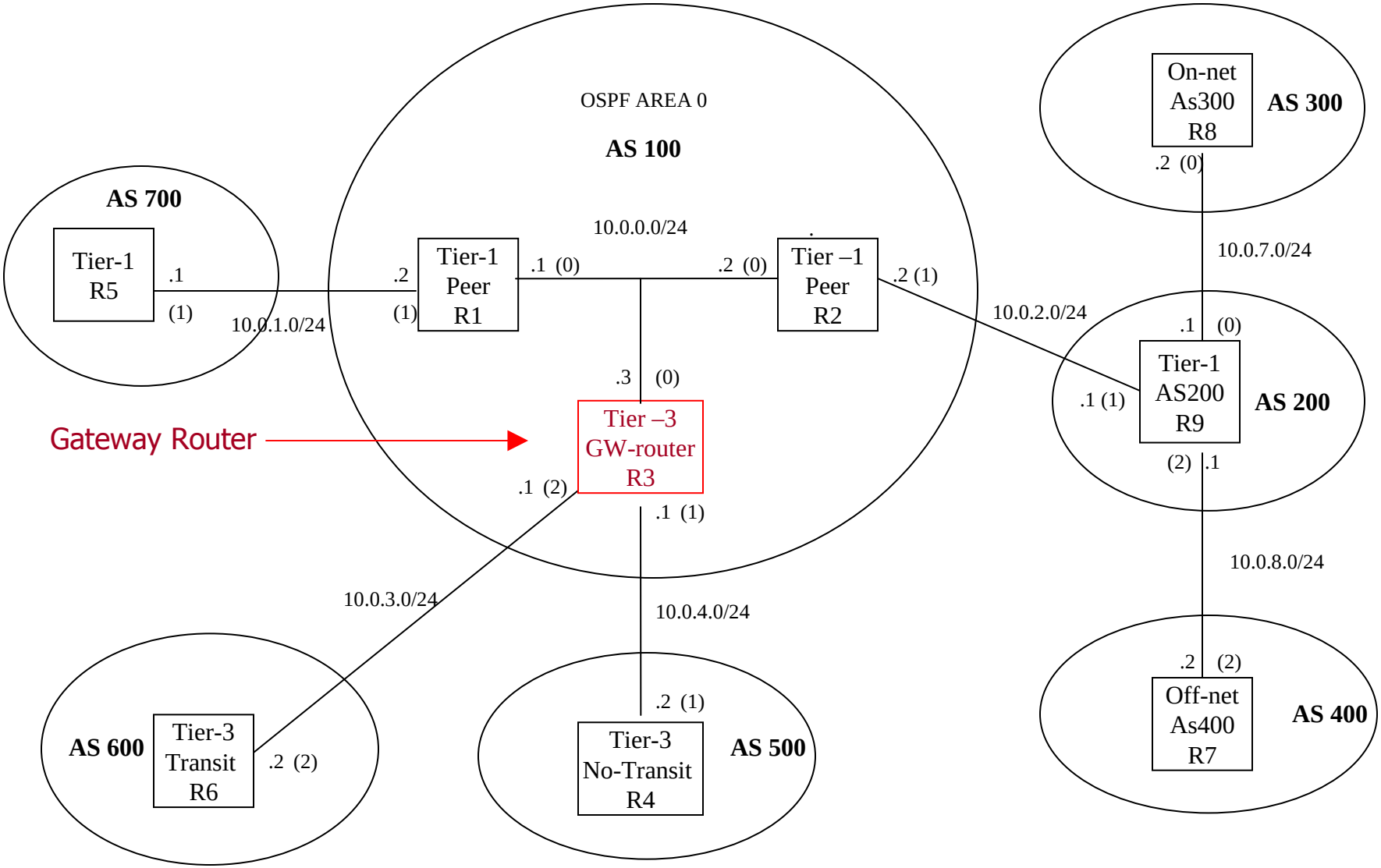
- **Example**

```
policy-options {
    policy-statement accept-but-do-not-export {
        from route-filter 192.11/16 orlonger;
        then {
            community add do-not-export;
            accept;
        }
    }
    community do-not-export members no-export;
}
```

Communities Example



Communities Example



Communities Example

- Create Named Community on gateway router

```
policy-options {
    community no-transit members "no-export;";
    community transit members "100:888;";
}
```

Communities Example

- **Create a policy that marks routes with the community values. On the gateway router:**

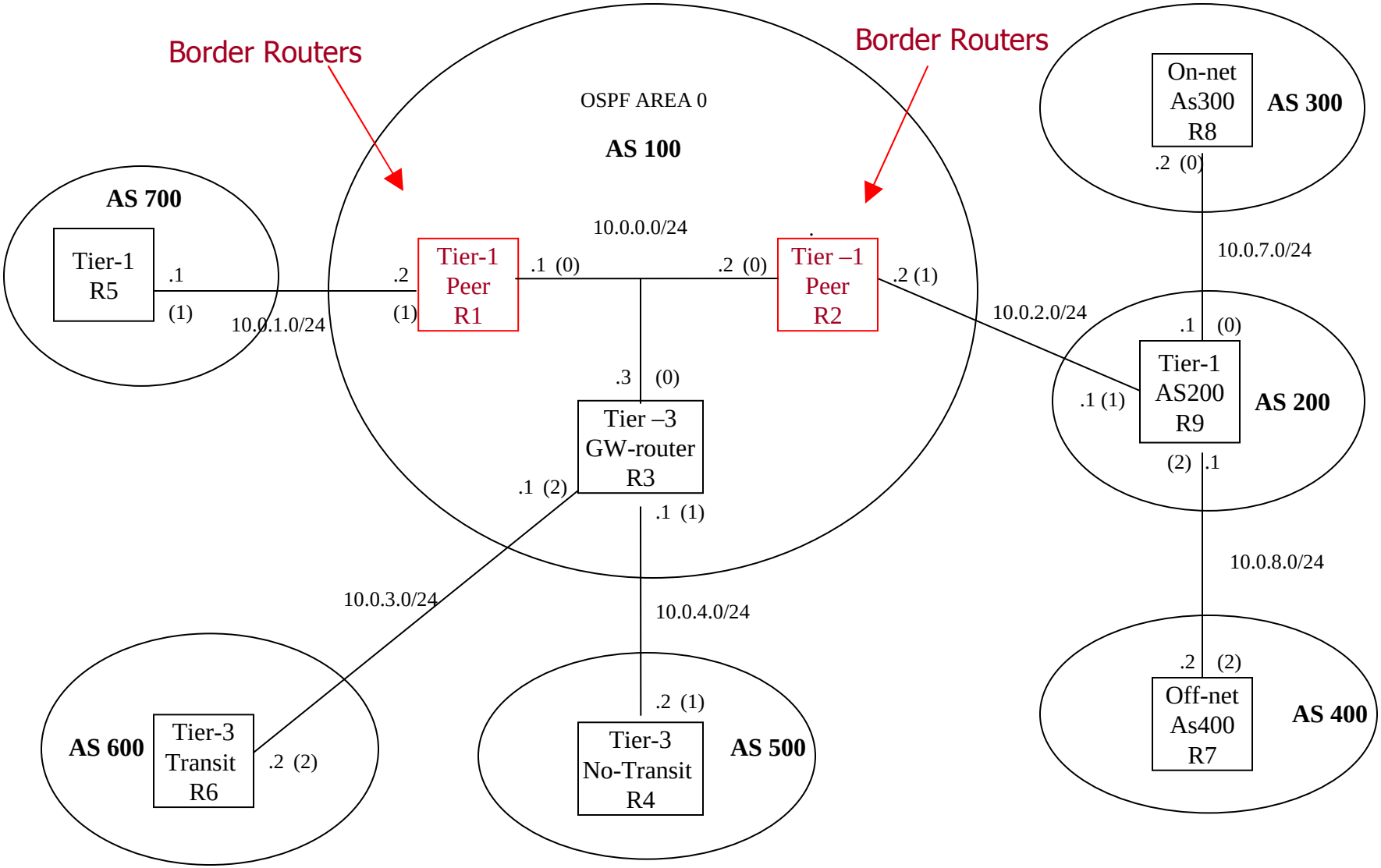
```
policy-options {
    policy-statement transit-service {
        from protocol bgp;
        then {
            community add transit;
            next policy;
            accept;
        }
    }
    policy-statement no-transit-service {
        from protocol bgp;
        then {
            community add no-transit;
            next policy;
            accept;
        }
    }
}
```

Communities Example

- **Apply the policy on the gateway router:**

```
protocols {
    bgp {
      group Regional.ISP1 {
          type external;
          multihop;
          import no-transit-service;
          peer-as 500;
          neighbor 6.6.6.6;
       }
       group Regional.ISP2 {
          type external;
          multihop;
          import transit-service;
          peer-as 600;
          neighbor 8.8.8.8;
```

Communities Example



Communities Example

- **Create a policy to enforce service on BGP border routers:**

```
policy-options {
    policy-statement advertise-policy {
        term advertise {
            from community transit-service;
            then accept;
            then next policy;
        }
        term do-not-advertise {
            from community Tier-1;
            then reject;
        }
        term catch-all {
            then reject;
        }
```

Communities Example

- Apply the policy on BGP Border routers:

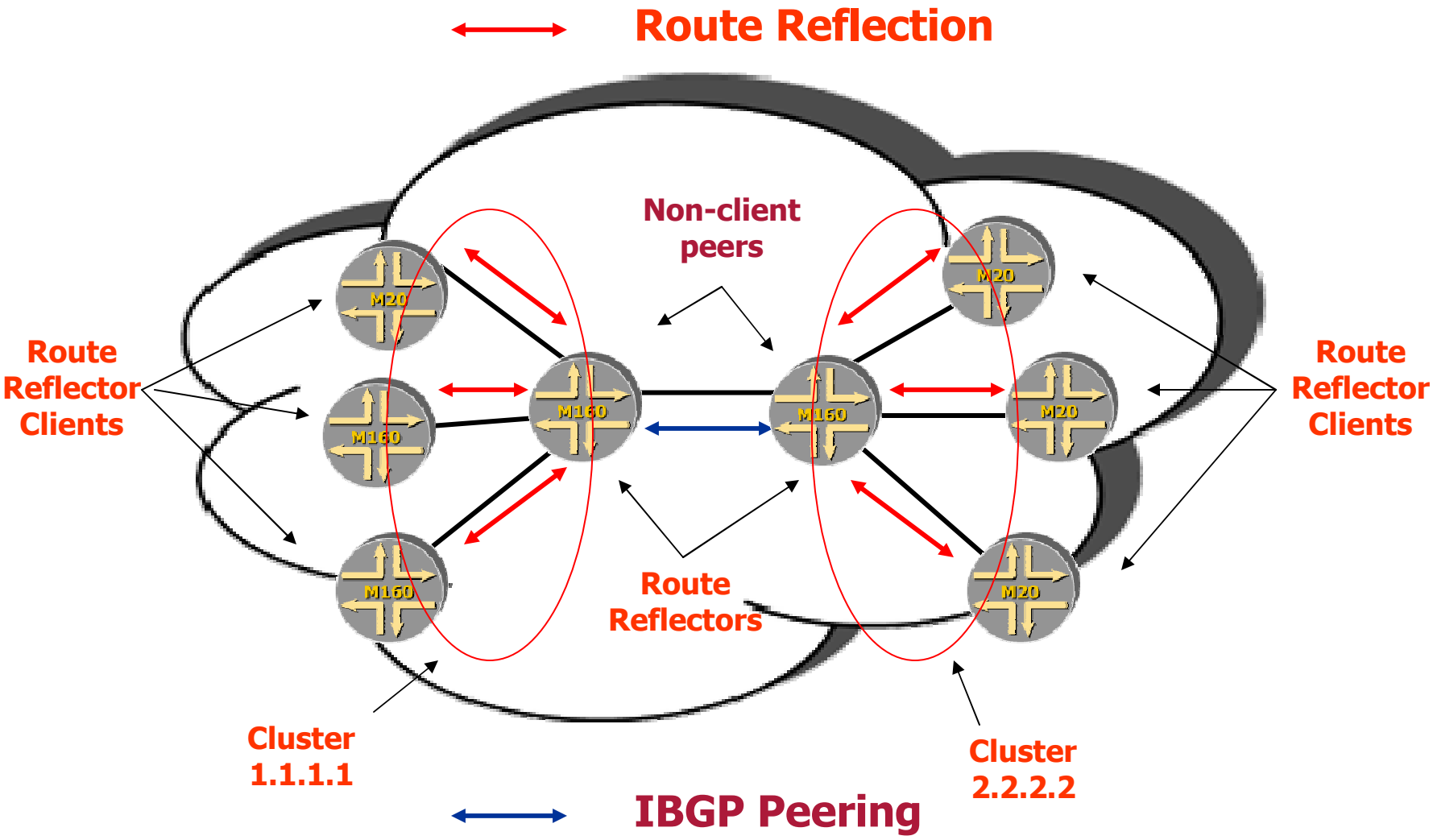
```
protocols {
    bgp {
        local-address 1.1.1.1;
        group Some-Tier1-Provider {
            type external;
            multihop ttl 2;
            import Tier1-community;
            export advertise-policy;
            peer-as 300;
            neighbor 4.4.4.4;
```

Route Reflectors

IBGP Full-mesh Scaling

- **N-squared problem**
  - Add one new router
    - you must peer to all IBGP speakers
    - also add new peer to all IBGP speakers
- **Adds TCP processing overhead**
- **Two methods to scale**
  - Route Reflection (RFC 2796)
  - Confederations (RFC 1965)

BGP Route Reflection (example)



BGP Route Reflection

- **BGP Route Reflection is one method to work around the need to have a full-mesh Internal BGP (IBGP) network.**
- **Some terms:**
  - Route reflector (RR)
    - The RR is the router responsible for the re-advertisement or the reflection of routes.
  - Router reflector client (RR client)
    - The RR client is the router that relies on the RR to advertise and to learn routes to and from the rest of the network. RR clients of the same RR might or might not be fully meshed.
  - Nonclient peer
    - Nonclient peers are routers that have a BGP session with the RR, but are not considered clients. Unlike RR clients, nonclient peers in the same autonomous system (AS) as the RR must be fully meshed.
  - Cluster ID

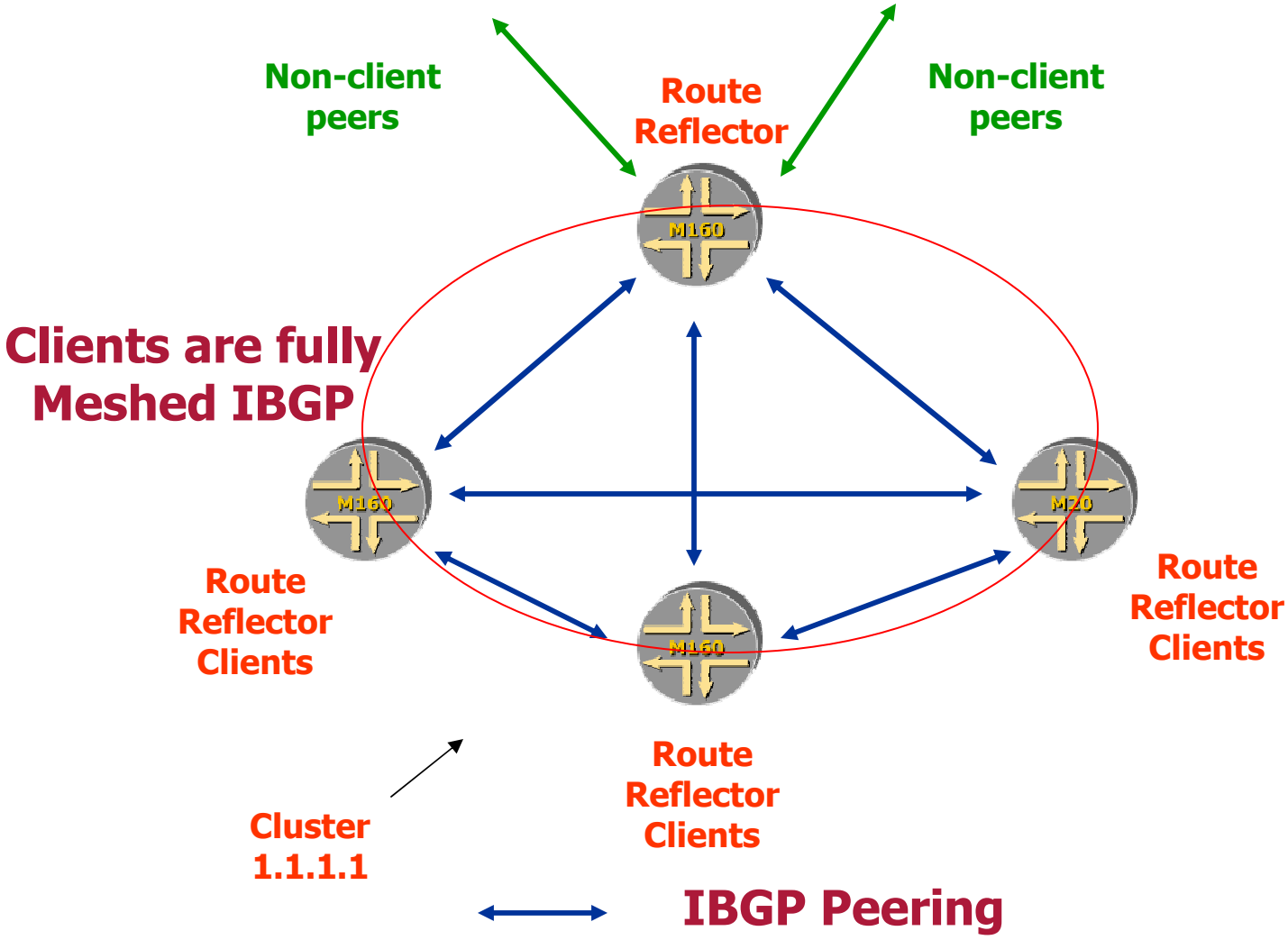
Route Reflection

- **Doesn't this create possible routing loops?**
- **New attributes**
  - Cluster-id
    - Identifies the route reflection cluster
    - Added to the route by the RR
  - Cluster-list
    - Sequence of cluster-ids that an update has traversed
    - Similar to AS-path list
  - Originator-id
    - Identifies the router that originated the route into the AS
    - Added to the route by the RR

BGP Route Reflection (example)

```
protocols {
    bgp {
        export static-to-bgp;
        group ibgp {
            type internal;
            local-address 2.2.2.2;
            peer-as 100;
            neighbor 5.5.5.5;
        }
        group rrcluster {
            type internal;
            local-address 2.2.2.2;
            cluster 192.128.1.1;      <<< indicates that local router is a
  reflector                                   for this cluster
            peer-as 100;
            neighbor 1.1.1.1;                 <<< define list of client neighbors
            neighbor 3.3.3.3;                 <<< in cluster 5.5.5.5
        }
    }
```

no-client-reflect



no-client-reflect

```
protocols {
    bgp {
        export static-to-bgp;
        group ibgp {
            type internal;
            local-address 2.2.2.2;
            peer-as 100;
            neighbor 5.5.5.5;
        }
        group rrcluster {
            type internal;
            local-address 2.2.2.2;
            no-client-reflect;
            cluster 192.128.1.1;
            peer-as 100;
            neighbor 1.1.1.1;
            neighbor 3.3.3.3;
        }
    }
```

BGP Route Reflection: re-advertisement rules

- **When a RR receives a route, it re-advertises the route using the following rules:**
  - If from nonclient peer, reflect the route to all clients.
  - If from a client peer, it reflects the route to all nonclients and all client peers except the client who originated the route.
    - (Note: If you configure the parameter `no-client-reflect`, the RR does not reflect routes to other clients. Include this statement only when all the clients are fully meshed.)
  - If from an EBGP peer, it reflects the route to all client and nonclient peers.

BGP Route Reflection: Additional 2 Attributes

- **In addition to these re-advertisement rules, the RR sets two BGP attributes:**
  - Originator ID indicates the router ID of the router that originated the route into BGP.
  - Cluster list records the sequence of clusters that the route has traversed.

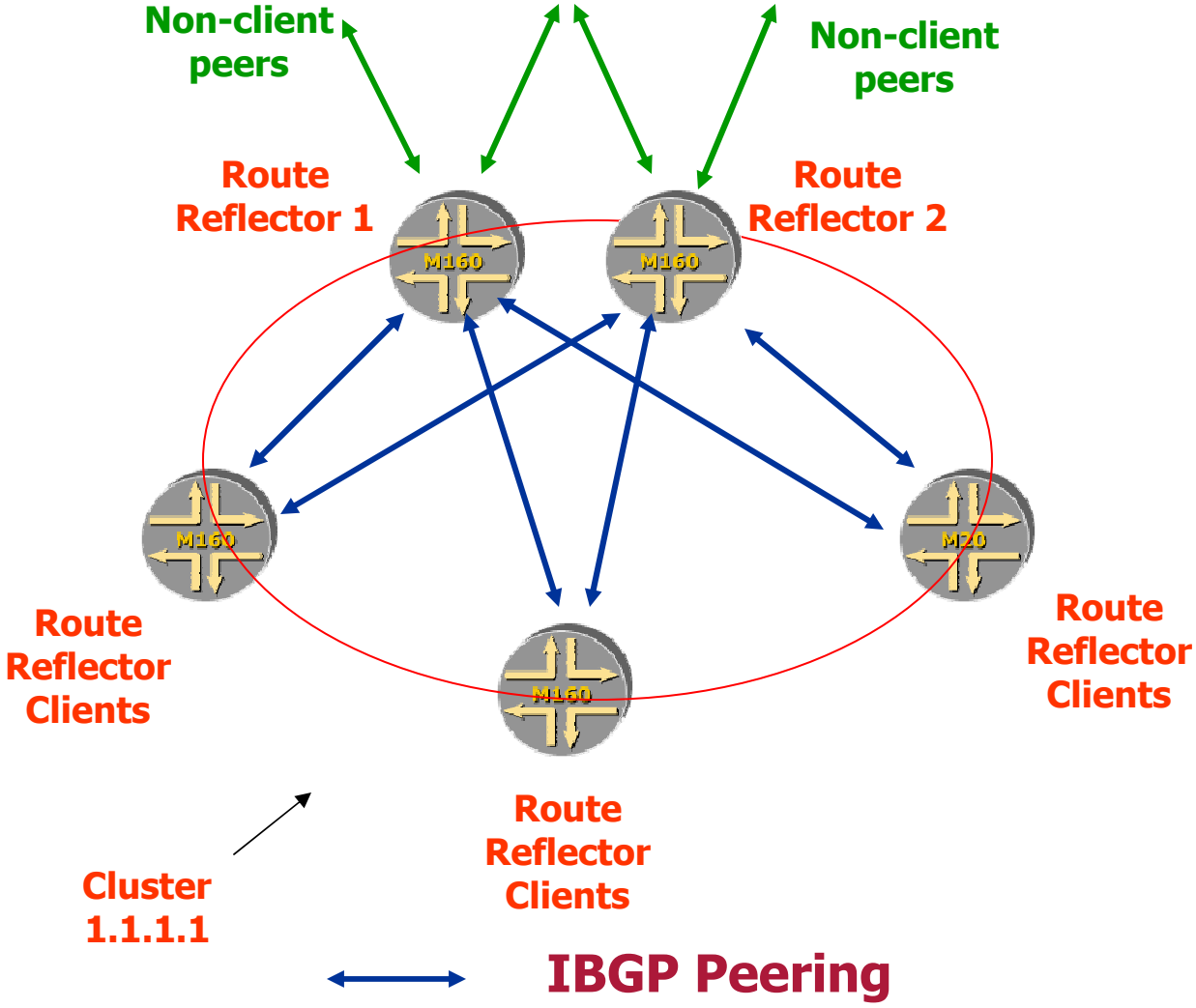
The RR uses the following rules when setting these attributes:

- A RR never adds the cluster ID, cluster list, and originator ID to a route that it received from an EBGP peer or to a route learned from another protocol other than IBGP (such as static route) while reflecting.
- If the reflection is client to client or client to nonclient, the cluster ID, cluster list, and originator ID are added to the route. An originator ID is created only if one is not set. The current cluster ID is added to the existing cluster list or a new cluster list is

BGP Route Reflection: JUNOS Operation

- **Originator ID and cluster list attributes are checked to prevent routing loops.**
- **The following routing loop checks are implemented in the JUNOS software:**
  - If a route is received from a client in a cluster that has our own cluster ID (for the same cluster), the route is discarded and not installed in the routing table.
  - If a route is received from a nonclient that has our own cluster ID, the route is accepted because it might have to be reflected to another cluster if the router is supporting multiple clusters.
  - Do not reflect a path back to the originator of the route.
  - Do not reflect a path to a client if the cluster ID of the cluster to which the client belongs is included in the cluster list.
  - Discard any route received if the originator ID is the same as our router ID.

Redundant Route Reflector



Thank you!