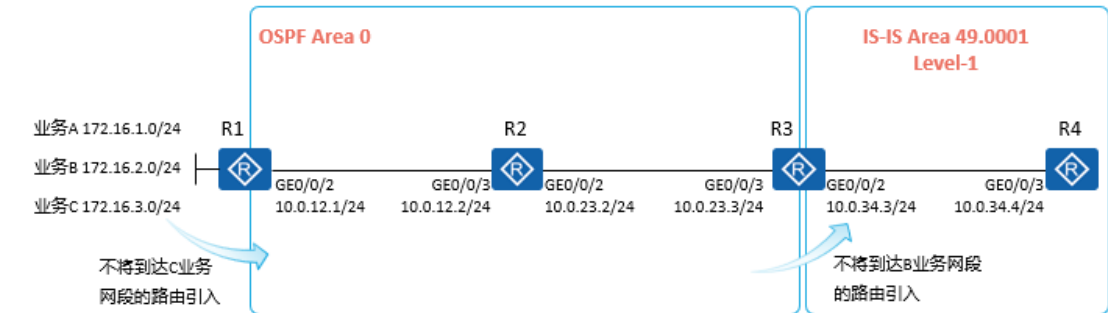


路由策略与路由控制

- 在复杂的数据通信网络中，根据实际组网需求，往往需要实施一些路由策略对路由信息进行过滤、属性设置等操作，通过对路由的控制，可以影响数据流量转发。
- 路由策略并非单一的技术或者协议，而是一个技术专题或方法论，里面包含了多种工具及方法。
- 本课程主要介绍网络中常用的路由选择工具以及路由策略的原理与配置。

技术背景

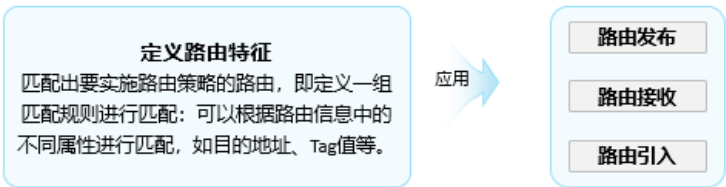


- R1上将业务A、B、C的网段路由引入时希望不引入业务C网段路由，同时R3上将OSPF路由引入到IS-IS中时，同样只希望引入B业务网段路由。
- 此时需要一个工具在引入路由时进行限制。

路由控制概述

路由控制可以通过路由策略（Route-Policy）实现，路由策略应用灵活而广泛，有以下几种常见方式：

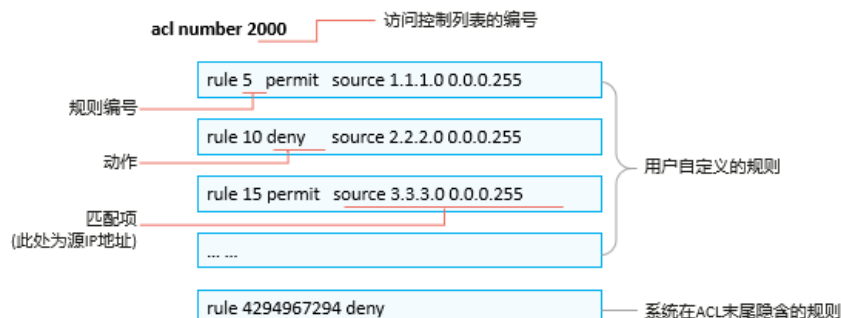
- 控制路由的发布：通过路由策略对发布的路由进行过滤，只发布满足条件的路由。
- 控制路由的接收：通过路由策略对接收的路由进行过滤，只接收满足条件的路由。
- 控制路由的引入：通过路由策略控制从其他路由协议引入的路由条目，只有满足条件的路由才会被引入。





匹配工具1：访问控制列表


- 访问控制列表（Access Control List, ACL）是一个匹配工具，能够对报文及路由进行匹配和区分。
- ACL由若干条permit或deny语句组成。每条语句就是该ACL的一条规则，每条语句中的permit或deny就是与这条规则相对应的处理动作。



- ACL 的组成：
- ACL 编号：在网络上配置 ACL 时，每个 ACL 都需要分配一个编号，称为 ACL 编号，用来标识 ACL。不同分类的 ACL 编号范围不同，这个后面具体讲。
- 规则：前面提到了，一个 ACL 通常由若干条“permit/deny”语句组成，每条语句就是该 ACL 的一条规则。
- 规则编号：每条规则都有一个相应的编号，称为规则编号，用来标识 ACL 规则。可以自定义，也可以系统自动分配。ACL 规则的编号范围是 0 ~ 4294967294，所有规则均按照规则编号从小到大进行排序。
- 动作：每条规则中的 permit 或 deny，就是与这条规则相对应的处理动作。permit 指“允许”，deny 指“拒绝”，但是 ACL 一般是结合其他技术使用，不同的场景，处理动作的含义也有所不同。
- 比如：ACL 如果与流量过滤技术结合使用（即流量过滤中调用 ACL），permit 就是“允许通行”的意思，deny 就是“拒绝通行”的意思。
- 匹配项：ACL 定义了极其丰富的匹配项。例子中体现的源地址，ACL 还支持很多其他规则匹配项。例如，二层以太

网帧头信息（如源 MAC、目的 MAC、以太网协议类型）、三层报文信息（如目的地址、协议类型）以及四层报文信息（如 TCP/UDP 端口号）等。

- 提问：rule 5 permit source 1.1.1.0 0.0.0.255 是什么意思？这个在后续课程中会介绍。



通配符

ACL IP Prefix List


```
acl number 2000
rule 5 deny source 10.1.1.1 0
rule 10 deny source 10.1.1.2 0
rule 15 permit source 10.1.1.0 0.0.255
```

通配符

通配符 (Wildcard)

- 通配符是一个32比特长度的数值，用于指示IP地址中哪些比特位需要严格匹配，哪些比特位无需匹配。
- 通配符通常采用类似网络掩码的点分十进制形式表示，但是含义却与网络掩码完全不同。

- 匹配规则：
“0”表示“匹配”；“1”表示“无需匹配”

 如何匹配192.168.1.0/24网段内的IP地址？

192.168.1.1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0.0.0.255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	严格匹配																无需匹配												

192.168.1.0/24网段

- 当进行 IP 地址匹配的时候，后面会跟着 32 位掩码位，这 32 位称为通配符。
- 通配符，也是点分十进制格式，换算成二进制后，“0”表示“匹配”，“1”表示“不匹配”。通配符中的 1 或者 0 可以不连续。
- 具体看下这 2 条规则：
 - rule 5: 拒绝源 IP 地址为 10.1.1.1 报文通过——因为通配符为全 0，所以每一位都要严格匹配，因此匹配的是主机 IP 地址 10.1.1.1；
 - rule 15: 允许源 IP 地址为 10.1.1.0/24 网段地址的报文通过——因为通配符：0.0.0.11111111，后 8 位为 1，表示不关心，因此 10.1.1.xxxxxxxx 的后 8 位可以为任意值，所以匹配的是 10.1.1.0/24 网段。

- 例子：如果要精确匹配 192.168.1.1/24 这个 IP 地址对应的网段地址，通配符是多少？
- 可以得出：网络位需要严格匹配，主机位无所谓，因此通配符为“0.0.0.255”。
- 两个特殊的通配符：
- 当通配符全为 0 来匹配 IP 地址时，表示精确匹配某个 IP 地址；
- 当通配符全为 1 来匹配 0.0.0.0 地址时，表示匹配了所有 IP 地址。

ACL 的分类与基本 ACL

- 基于 ACL 规则定义方式的划分

分类	编号范围	规则定义描述
基本 ACL	2000~2999	仅使用报文的源 IP 地址、分片信息和生效时间段信息来定义规则。
高级 ACL	3000~3999	可使用 IPv4 报文的源 IP 地址、目的 IP 地址、IP 协议类型、ICMP 类型、TCP 源/目的端口、UDP 源/目的端口号、生效时间段等来定义规则。
二层 ACL	4000~4999	使用报文的以太网帧头信息来定义规则，如根据源 MAC 地址、目的 MAC 地址、二层协议类型等。
用户自定义 ACL	5000~5999	使用报文头、偏移位置、字符串掩码和用户自定义字符串来定义规则。
用户 ACL	6000~6999	既可使用 IPv4 报文的源 IP 地址或源 UCL (User Control List) 组，也可使用目的 IP 地址或目的 UCL 组、IP 协议类型、ICMP 类型、TCP 源端口/目的端口、UDP 源端口/目的端口号等来定义规则。

- 基本 ACL

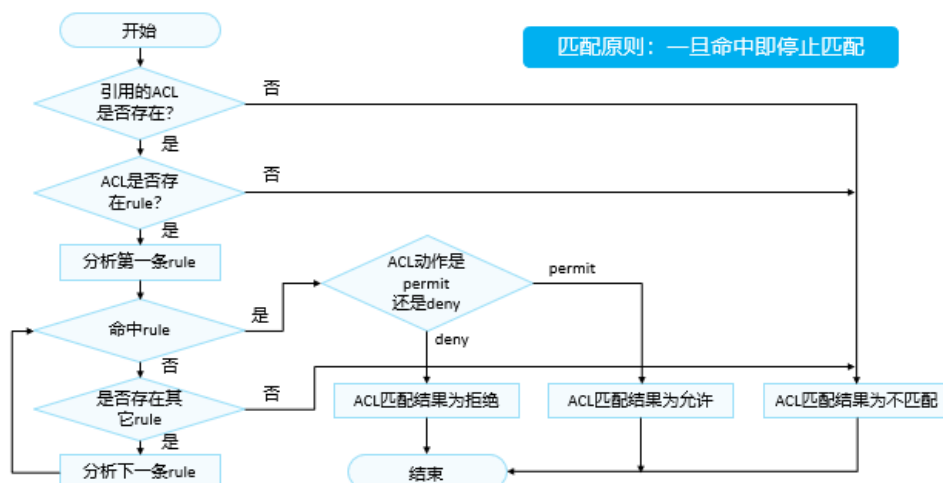
源IP地址			
IP Header		TCP/UDP Header	Data
acl number 2000			
rule	5	deny	source 10.1.1.1 0
rule	10	deny	source 10.1.1.2 0
rule	15	permit	source 10.1.1.0 0.0.0.255

- 对于使用 ACL 进行路由匹配的场景，用户只能使用基本 ACL。



ACL的匹配机制

ACL IP Prefix List



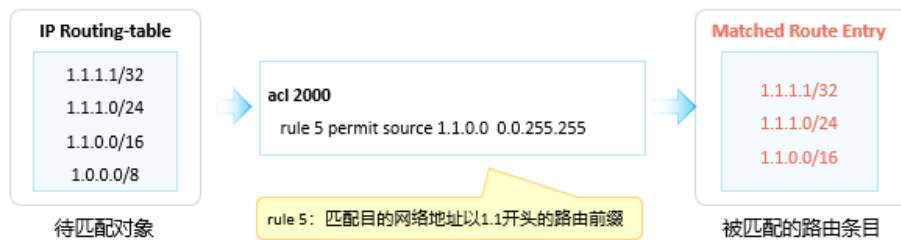
- ACL 的匹配机制概括来说就是：
- 配置 ACL 的设备接收报文后，会将该报文与 ACL 中的规则逐条进行匹配，如果不能匹配上，就会继续尝试去匹配下一条规则。
- 一旦匹配上，则设备会对该报文执行这条规则中定义的处理动作，并且不再继续尝试与后续规则匹配。
- 匹配流程：首先系统会查找设备上是否配置了 ACL。
- 如果 ACL 不存在，则返回 ACL 匹配结果为：不匹配。
- 如果 ACL 存在，则查找设备是否配置了 ACL 规则。
- 如果规则不存在，则返回 ACL 匹配结果为：不匹配。
- 如果规则存在，则系统会从 ACL 中编号最小的规则开始查找。
- 如果匹配上了 permit 规则，则停止查找规则，并返回 ACL 匹配结果为：匹配（允许）。
- 如果匹配上了 deny 规则，则停止查找规则，并返回 ACL 匹配结果为：匹配（拒绝）。
- 如果未匹配上规则，则继续查找下一条规则，以此循环。如果一直查到最后一条规则，报文仍未匹配上，则返回 ACL 匹配结果为：不匹配。

- 从整个 ACL 匹配流程可以看出，报文与 ACL 规则匹配后，会产生两种匹配结果：“匹配”和“不匹配”。
- 匹配（命中规则）：指存在 ACL，且在 ACL 中查找到了符合匹配条件的规则。不论匹配的动作是“permit”还是“deny”，都称为“匹配”，而不是只是匹配上 permit 规则才算“匹配”。
- 不匹配（未命中规则）：指不存在 ACL，或 ACL 中无规则，再或者在 ACL 中遍历了所有规则都没有找到符合匹配条件的规则。以上三种情况，都叫做“不匹配”。
- 匹配原则：一旦命中即停止匹配。

ACL 的匹配顺序及匹配结果

配置顺序（config 模式）

- 系统按照 ACL 规则编号从小到大的顺序进行报文匹配，规则编号越小越容易被匹配。



“允许”是指什么？

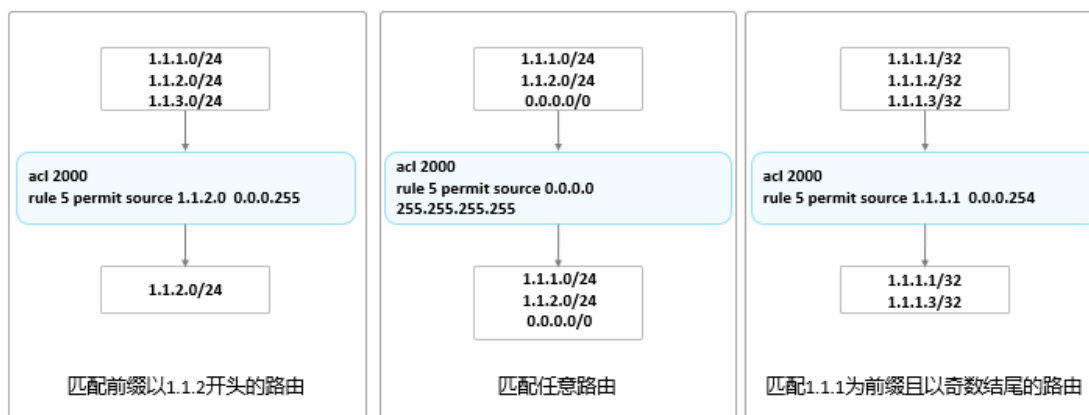
- 一条 ACL 可以由多条“deny 或 permit”语句组成，每一条语句描述一条规则，这些规则可能存在包含关系，也可能有重复或矛盾的地方，因此 ACL 的匹配顺序是十分重要的。
- 华为设备支持两种匹配顺序：自动排序（auto 模式）和配置顺序（config 模式）。缺省的 ACL 匹配顺序是 config 模式。
- 自动排序，是指系统使用“深度优先”的原则，将规则按照精确度从高到低进行排序，并按照精确度从高到低的顺序进行报文匹配。——这个比较复杂，这里就不具体展开了，感兴趣的同学可以课后查看资料。
- 配置顺序，系统按照 ACL 规则编号从小到大的顺序进行

报文匹配，规则编号越小越容易被匹配。——这个就是前面提到的匹配顺序。

- 如果后面又添加了一条规则，则这条规则会被加入到相应的位置，报文仍然会按照从小到大的顺序进行匹配。
- 注意：ACL 技术总是与其他技术结合在一起使用的，因此，所结合的技术不同，“允许 (permit)”及“拒绝 (deny)”的实际作用也会不同。例如，当 ACL 技术与路由过滤技术结合使用时，permit 就是“匹配该条路由条目”的意思，deny 就是“不匹配该条路由条目”的意思。

ACL IP Prefix List

常用匹配举例



ACL只能匹配路由的前缀，无法匹配路由的网络掩码。

ACL IP Prefix List

基本ACL的基础配置命令

1. 创建基本ACL

```
[Huawei] acl [ number ] acl-number [ match-order config ]
```

使用编号（2000 ~ 2999）创建一个数字型的基本ACL，并进入基本ACL视图。

```
[Huawei] acl name acl-name { basic | acl-number } [ match-order config ]
```

使用名称创建一个命名型的基本ACL，并进入基本ACL视图。

2. 配置基本ACL的规则

```
[Huawei-acl-basic-2000] rule [ rule-id ] { deny | permit } [ source { source-address source-wildcard | any } ] [ time-range time-name ]
```

在基本ACL视图下，通过此命令来配置基本ACL的规则。

- 创建基本 ACL
- [Huawei] **acl** [**number**] *acl-number* [**match-order config**]
- *acl-number* : 指定访问控制列表的编号。
- **match-order config** : 指定 ACL 规则的匹配顺序 , config 表示配置顺序。
- [Huawei] **acl name** *acl-name* { **basic** | *acl-number* } [**match-order config**]
- *acl-name* : 指定创建的 ACL 的名称。
- **basic** : 指定 ACL 的类型为基本 ACL。
- 配置基本 ACL 规则
- [Huawei-acl-basic-2000] **rule** [*rule-id*] { **deny** | **permit** } [**source** { *source-address source-wildcard* | **any** } | **time-range** *time-name*]
- *rule-id* : 指定 ACL 的规则 ID。
- **deny** : 指定拒绝符合条件的报文。
- **permit** : 指定允许符合条件的报文。
- **source** { *source-address source-wildcard* | **any** } : 指定 ACL 规则匹配报文的源地址信息。如果不配置 , 表示报文的任何源地址都匹配。其中 :
 - *source-address* : 指定报文的源地址。
 - *source-wildcard* : 指定源地址通配符。
 - **any** : 表示报文的任意源地址。相当于 source-address 为 0.0.0.0 或者 source-wildcard 为 255.255.255.255。
- **time-range** *time-name* : 指定 ACL 规则生效的时间段。其中 , *time-name* 表示 ACL 规则生效时间段名称。如果不指定时间段 , 表示任何时间都生效。



匹配工具2：IP前缀列表

- IP前缀列表（IP-Prefix List）是将路由条目的网络地址、掩码长度作为匹配条件的过滤器，可在各路由协议发布和接收路由时使用。
- 不同于ACL，IP-Prefix List能够同时匹配IP地址前缀长度以及掩码长度，增强了匹配的精确度。

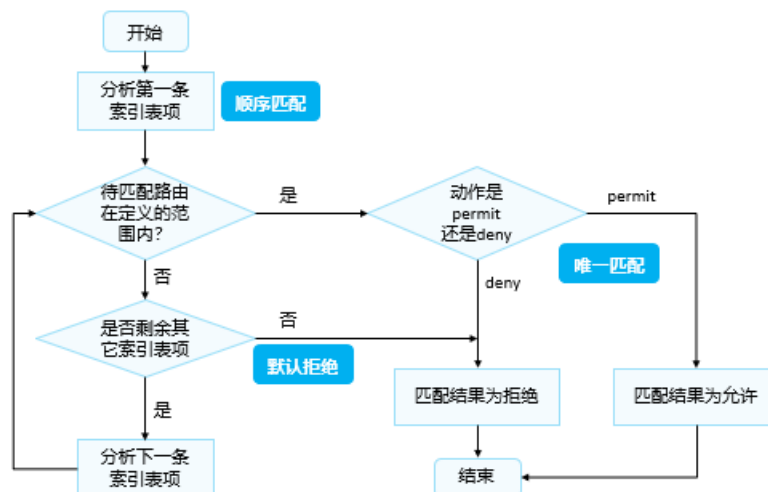
```
[Huawei] ip ip-prefix test index 10 permit 192.168.1.0 22 greater-equal 24 less-equal 26
```

ip-prefix-name 序号 动作 IP网段与掩码 掩码范围

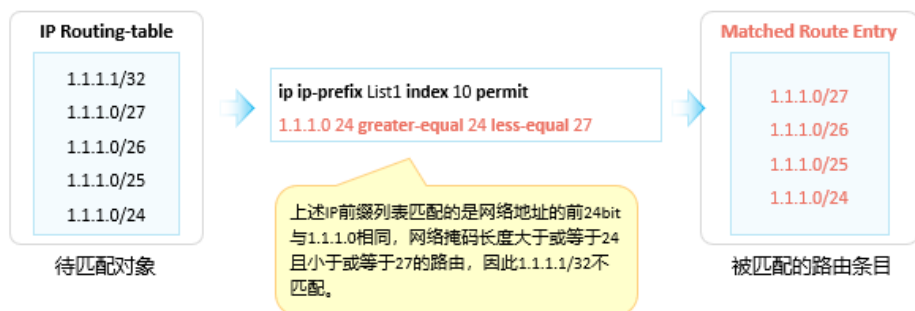
- ip-prefix-name**：地址前缀列表名称
- 序号**：本匹配项在地址前缀列表中的序号，匹配时根据序号从小到大进行顺序匹配
- 动作**：permit/deny，地址前缀列表的匹配模式为允许/拒绝，表示匹配/不匹配
- IP网段与掩码**：匹配路由的网络地址，以及限定网络地址的前多少位需严格匹配
- 掩码范围**：匹配路由前缀长度，掩码长度的匹配范围 $\text{mask-length} \leq \text{greater-equal-value} \leq \text{less-equal-value} \leq 32$



IP-Prefix的匹配机制



IP-Prefix的匹配示例



IP-Prefix的基础配置命令

1. 创建IPv4地址前缀列表

```
[Huawei] ip ip-prefix ip-prefix-name [ index index-number ] { permit | deny } ipv4-address mask-length [ match-network ] [ greater-equal greater-equal-value ] [ less-equal less-equal-value ]
```

创建IPv4地址前缀列表或增加其中一个表项。

- *ip-prefix-name*: 指定地址前缀列表的名称。
- *index index-number*: 指定本匹配项在地址前缀列表中的序号。
- *permit*: 指定地址前缀列表的匹配模式为允许。
- *deny*: 指定地址前缀列表的匹配模式为拒绝。
- *ipv4-address mask-length*: 指定IP地址和指定掩码长度。
- *greater-equal greater-equal-value*: 指定掩码长度匹配范围的下限。
- *less-equal less-equal-value*: 指定掩码长度匹配范围的上限。

- *ip-prefix-name*: 指定地址前缀列表的名称。字符串形式，长度范围是 1 ~ 169，不支持空格，区分大小写。
- *index index-number*: 指定本匹配项在地址前缀列表中的序号。整数形式，取值范围是 1 ~ 4294967295。缺省情况下，该序号值按照配置先后顺序依次递增，每次加 10，第一个序号值为 10。
- *permit*: 指定地址前缀列表的匹配模式为允许。在该模式下，如果过滤的 IP 地址在定义的范围內，则通过过滤，进行相应的设置；否则，必须进行下一节点的测试。
- *deny*: 指定地址前缀列表的匹配模式为拒绝。在该模式

下，如果过滤的 IP 地址在定义的范围內，该 IP 地址不能通过过滤从而不能进入下一节点的测试；否则，将进行下一节点的测试。

- **ipv4-address mask-length**：指定 IP 地址和指定掩码长度。其中掩码长度为整数形式，取值范围是 0 ~ 32。
- **greater-equal greater-equal-value**：指定掩码长度匹配范围的下限。若不配置 **greater-equal greater-equal-value** 和 **less-equal less-equal-value**，则使用 mask-length 作为掩码长度。
- 参数 **greater-equal-value** 的取值限制为： $mask-length \leq greater-equal-value \leq less-equal-value \leq 32$ 。
- 如果只配置了 **greater-equal**，则掩码长度范围在 **greater-equal-value** 和 32 之间。
- **less-equal less-equal-value**：指定掩码长度匹配范围的上限。若不配置 **greater-equal greater-equal-value** 和 **less-equal less-equal-value**，则使用 **mask-length** 作为掩码长度。
- 参数 **less-equal-value** 的取值限制为： $mask-length \leq greater-equal-value \leq less-equal-value \leq 32$ 。
- 如果只配置了 **less-equal**，则掩码长度范围在 **mask-length** 和 **less-equal-value** 之间。

ACL > IP Prefix List

IP-Prefix的配置举例 (1)



单语句匹配

```
ip ip-prefix aa index 10 permit 10.1.1.0 24
```

- Case1: 路由10.1.1.0/24被Permit，其他都被Deny。

```
ip ip-prefix bb index 10 deny 10.1.1.0 24
```

- Case2: 路由全部被Deny。

- Case1 说明：这种情况属于单节点的精确匹配，只有目的地址，掩码完全相同的路由才会匹配成功，而且节点的匹配模式为 Permit，所以路由 10.1.1.0/24 被 Permit，属于匹配成功并被 Permit，其他路由由于未匹配成功被 Deny。
- Case2 说明：这种情况属于单节点的精确匹配，但节点的匹配模式为 deny，所以路由 10.1.1.0/24 还是被 Deny，属于匹配成功但被 Deny，其他路由则属于未匹配成功被默认 Deny。

IP-Prefix的配置举例 (2)



多语句匹配

```
ip ip-prefix aa index 10 deny 10.1.1.0 24
ip ip-prefix aa index 20 permit 10.1.1.1 32
```

- Case1：路由10.1.1.0/24被Deny，路由10.1.1.1/32被Permit，其他路由都被Deny。

```
ip ip-prefix bb index 10 permit 10.1.1.0 24 greater-equal 26 less-equal 32
```

- Case2：路由10.1.1.0/26，10.1.1.1/32被Permit，其他路由被Deny。

- Case1 说明：这种情况属于多节点的精确匹配。
- 路由 10.1.1.0/24 在匹配 index 10 时，满足匹配条件，但匹配模式是 deny，属于匹配成功但被 Deny。
- 路由 10.1.1.1/32 在匹配 index 10 时，不满足匹配条件，则继续匹配 index 20，此时匹配成功，且 index 20 的匹配模式是 permit，属于匹配成功并被 Permit。
- 其他路由由于都不符合 index 10 和 20 的条件，属于未匹配成功被默认 Deny。
- Case2 说明：此时 *greater-equal-value*=26，*less-equal-value*=32。配置时，需满足 *mask-length*≤*greater-equal-value*

$greater\text{-}equal\text{-}value \leq less\text{-}equal\text{-}value$ ，否则配置不成功。

ACL IP Prefix List

IP-Prefix的配置举例 (3)



通配地址匹配

```
ip ip-prefix aa index 10 permit 10.0.0.0 8 less-equal 32
```

- Case1: 所有掩码长度在8到32的路由都被Permit。

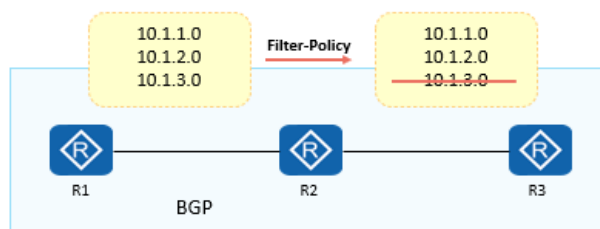
```
ip ip-prefix bb index 10 deny 10.1.1.0 24 less-equal 32  
ip ip-prefix bb index 20 permit 10.1.0.0 16 less-equal 32
```

- Case2: 路由10.1.0.0/16被Permit，其他路由被Deny。

- Case1 说明：此时 $greater\text{-}equal\text{-}value=8$ ， $less\text{-}equal\text{-}value=32$ ，前 8 个 bit 与 10.0.0.0 相同，掩码长度在 8-32 之间的路由全部被匹配。
- Case2 说明：
 - 对于 index 10， $greater\text{-}equal\text{-}value=24$ ， $less\text{-}equal\text{-}value=32$ ，前 24 个 bit 与 10.1.1.0 相同，所有掩码长度在 24 到 32 的路由全部被 Deny；
 - 对于 index 20， $greater\text{-}equal\text{-}value=0$ ， $less\text{-}equal\text{-}value=32$ ，前 16 个 bit 与 10.1.0.0 相同，掩码长度在 16 到 32 的路由被匹配，此时只有 10.1.0.0/16 被匹配，剩下的被默认 deny。

策略工具1：Filter-Policy

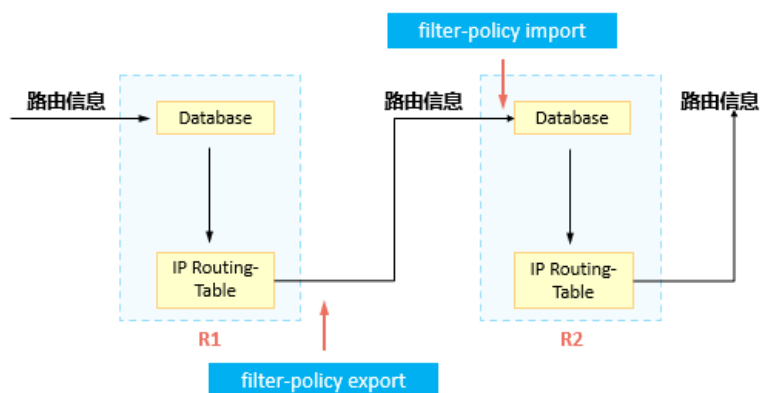
- Filter-Policy（过滤-策略）是一个很常用的路由信息过滤工具，能够对接收、发布、引入的路由进行过滤，可应用于IS-IS、OSPF、BGP等协议。



- 如图所示，R1、R2、R3之间运行BGP路由协议，路由在各个设备之间传递，当需要根据实际需求过滤某些路由信息的时候可以使用Filter-Policy实现。

Filter-Policy在距离矢量路由协议中的应用

在距离矢量路由协议中，设备之间传递的是路由信息，如果需要对这种路由信息进行某种过滤，可以使用Filter-Policy实现，出方向和入方向的生效位置如图所示。

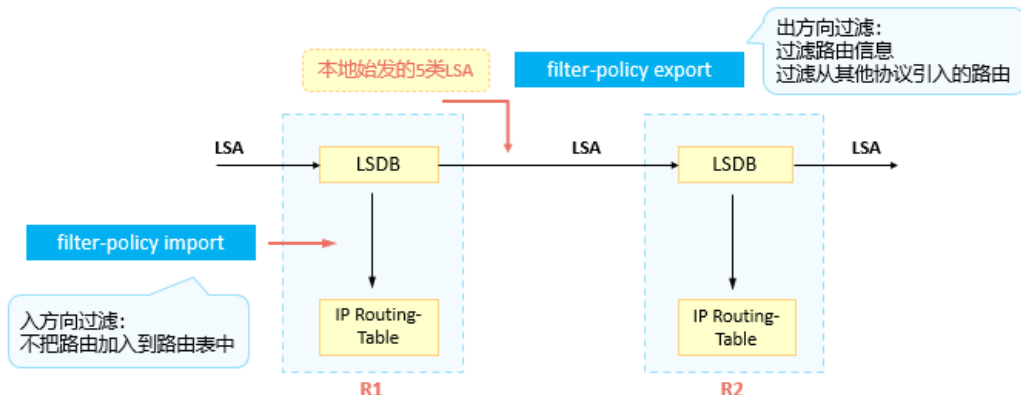


- 距离矢量协议是基于路由表生成路由的，因此过滤器会影响从邻居接收的路由和向邻居发布的路由。
- 如果要过滤掉上游设备到下游设备的路由，只需要在上游设备配置 filter-policy export 或者在下游设备上配置 filter-policy import。



Filter-Policy在链路状态路由协议中的应用

在链路状态路由协议中，各路由设备之间传递的是LSA信息，然后设备根据LSA汇总成的LSDB信息计算出路由表。但是Filter-Policy只能过滤路由信息，无法过滤LSA。



- OSPF 把网络中所泛洪的 LSA 存储到自己的 LSDB 中，并且运行 SPF 算法，计算出一颗以自己为根，无环的最短路径树，Filter-Policy 对 OSPF 计算出来的路由（加载到路由表之前）进行过滤，而不会对 LSA 进行过滤。
- 上面的实例以 OSPF 为例，展示了 Filter-Policy 在链路状态中的应用。



Filter-Policy的基础配置命令 (1)

1. 在OSPF中的应用

```
[Huawei-ospf-100] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name | route-policy route-policy-name [ secondary ] } import
```

按照过滤策略，设置OSPF对接收的路由进行过滤。

```
[Huawei-ospf-100] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name | route-policy route-policy-name } export [ protocol [ process-id ] ]
```

按照过滤策略，设置对引入的路由在向外发布时进行过滤。

- 命令：`[Huawei-ospf-100] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name | route-policy route-policy-name [secondary] } import`
- *acl-number*：指定基本访问控制列表号。整数形式，取值范围是 2000 ~ 2999。

- **acl-name** *acl-name* : 指定访问控制列表名称。字符串形式，不支持空格，区分大小写，长度范围是 1 ~ 32，以英文字母 a ~ z 或 A ~ Z 开始。
- **ip-prefix** *ip-prefix-name* : 指定地址前缀列表名称。字符串形式，长度范围是 1 ~ 169，不支持空格，区分大小写。当输入的字符串两端使用双引号时，可在字符串中输入空格。
- **route-policy** *route-policy-name* : 指定路由策略名称。字符串形式，区分大小写，不支持空格，长度范围是 1 ~ 40。当输入的字符串两端使用双引号时，可在字符串中输入空格。
- **secondary** : 设置优选次优路由。
- 命令 : [Huawei-ospf-100] **filter-policy** { *acl-number* | **acl-name** *acl-name* | **ip-prefix** *ip-prefix-name* | **route-policy** *route-policy-name* } **export** [*protocol* [*process-id*]]
- *protocol process-id* : 指定需要对引入的特定的路由协议进行过滤。目前的协议包括 direct、isis、bgp、ospf、unr 和 static。当指定路由协议为 RIP、IS-IS、OSPF 时，还可以指定进程号。整数类型，取值范围是 1 ~ 65535，缺省值是 1。



Filter-Policy的基础配置命令 (2)

Filter-Policy

Route-Policy

2. 在IS-IS中的应用

```
[Huawei-isis-1] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name | route-policy route-policy-name } import
```

配置IS-IS路由加入IP路由表时的过滤策略。

```
[Huawei-isis-1] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name | route-policy route-policy-name } export [ protocol [ process-id ] ]
```

配置IS-IS对已引入的路由在向外发布时进行过滤的过滤策略。



Filter-Policy的基础配置命令 (3)

3. 在BGP中的应用

```
[Huawei-bgp-af-ipv4] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name } import
```

配置对接收的路由信息进行过滤。

```
[Huawei-bgp-af-ipv4] filter-policy { acl-number | acl-name acl-name | ip-prefix ip-prefix-name } export [ protocol [ process-id ] ]
```

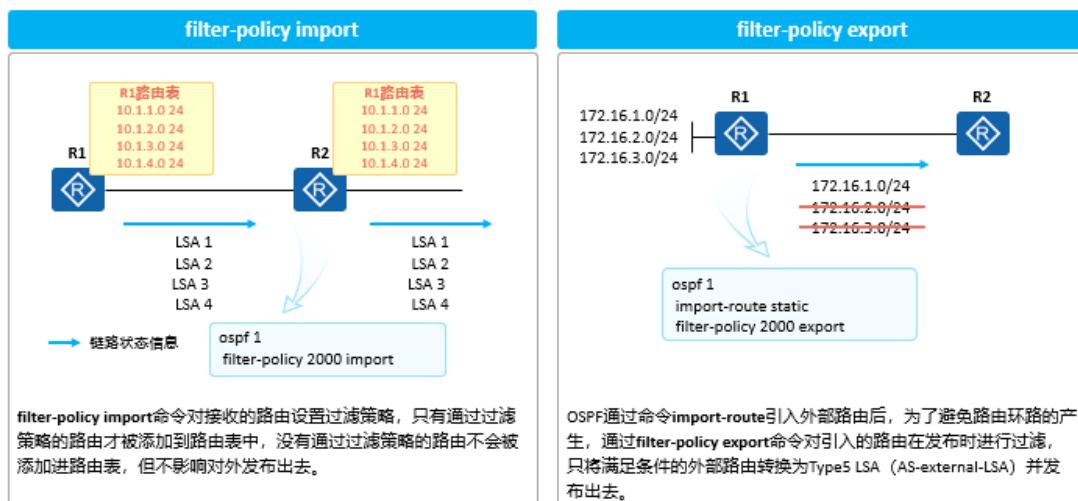
配置对发布的路由进行过滤，只有通过过滤的路由才被BGP发布。

```
[Huawei-bgp-af-ipv4] peer { group-name | ipv4-address } filter-policy { acl-number | acl-name acl-name } { import | export }
```

配置向对等体（组）发布或从对等体（组）接收路由时的过滤策略。



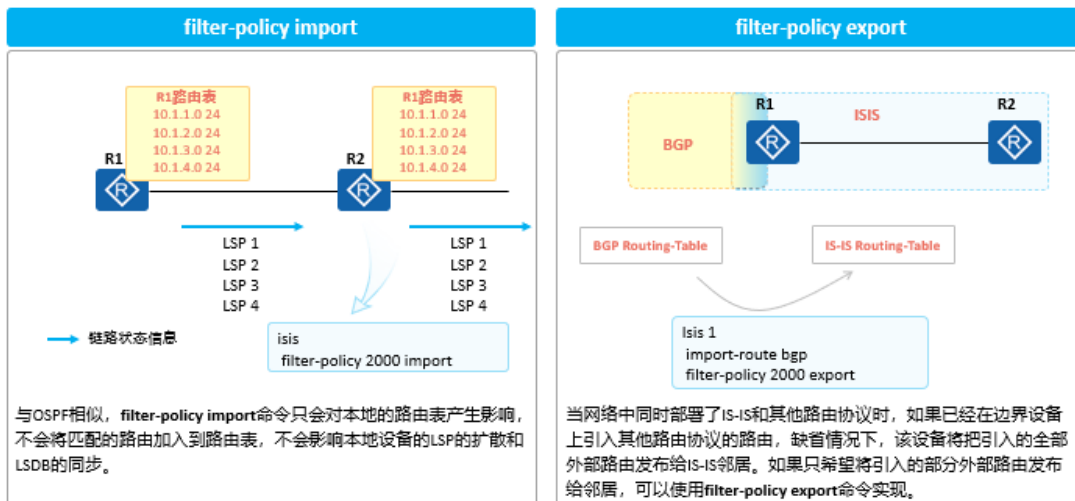
OSPF中使用Filter-Policy



- OSPF 的路由信息记录在 LSDB 中，**filter-policy import** 命令实际上是对 OSPF 计算出来的路由进行过滤，不是对发布和接收的 LSA 进行过滤。
- **filter-policy export** 命令通过指定 *protocol* 或 *process-id* 对特定的某一种协议或某一进程的路由进行过滤。如果没有指定 *protocol* 和 *process-id*，则 OSPF 将对所有引入的路由信息进行过滤。



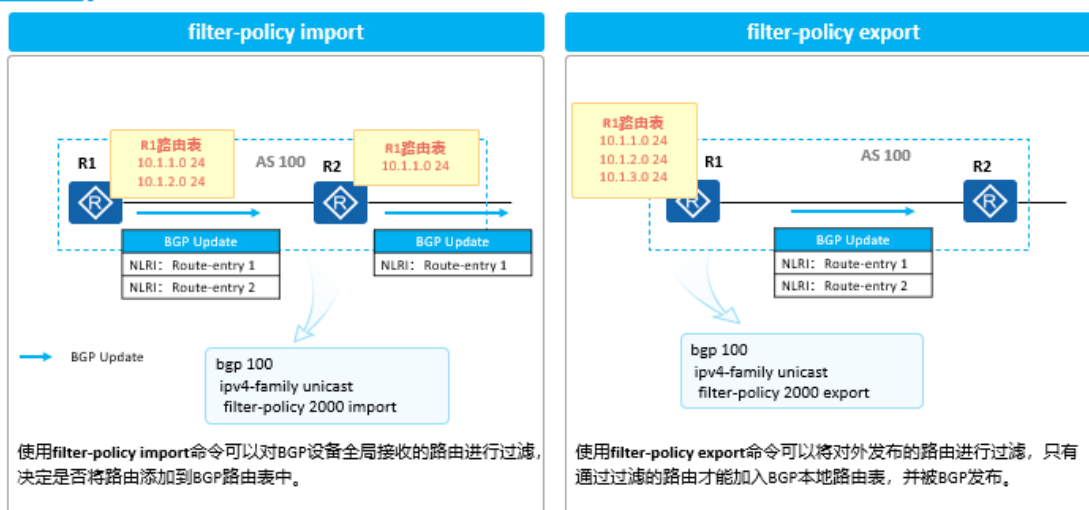
IS-IS中使用Filter-Policy



- IS-IS 的路由表项需要被成功下发到 IP 路由表中，才能用来指导 IP 报文转发。如果 IS-IS 路由表中有到达某个目的网段的路由，但是并不希望将该路由下发到 IP 路由表中，可以使用 filter-policy import 命令结合基本 ACL、IP-Prefix、路由策略等方式，只将部分 IS-IS 路由下发到 IP 路由表中。



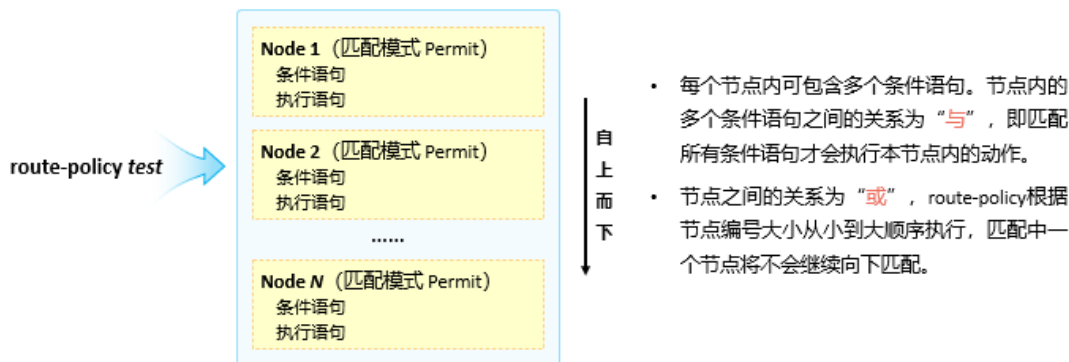
BGP中使用Filter-Policy





策略工具2: Route-Policy

- Route-Policy是一个策略工具，用于过滤路由信息，以及为过滤后的路由信息设置路由属性。
- 一个Route-Policy由一个或多个节点（Node）构成，每个节点都可以是一系列条件语句（匹配条件）以及执行语句（执行动作）的集合，这些集合按照编号从小到大的顺序排列。

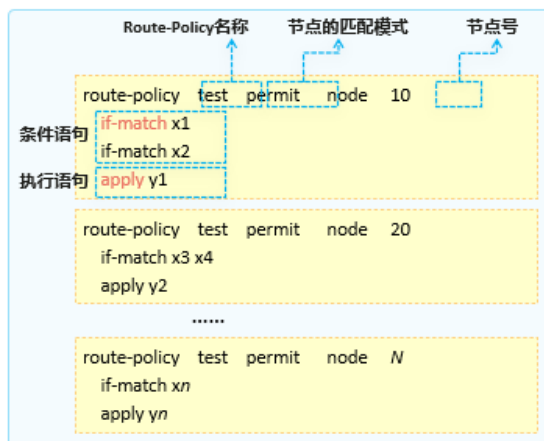


- 当使用 Route-Policy 时，Node 的值小的节点先进行匹配。一个节点匹配成功后，路由将不再匹配其他节点。全部节点匹配失败后，路由将被过滤。



Route-Policy的组成

一个Route-Policy由一个或多个节点构成，每个节点包括多个if-match和apply子句。

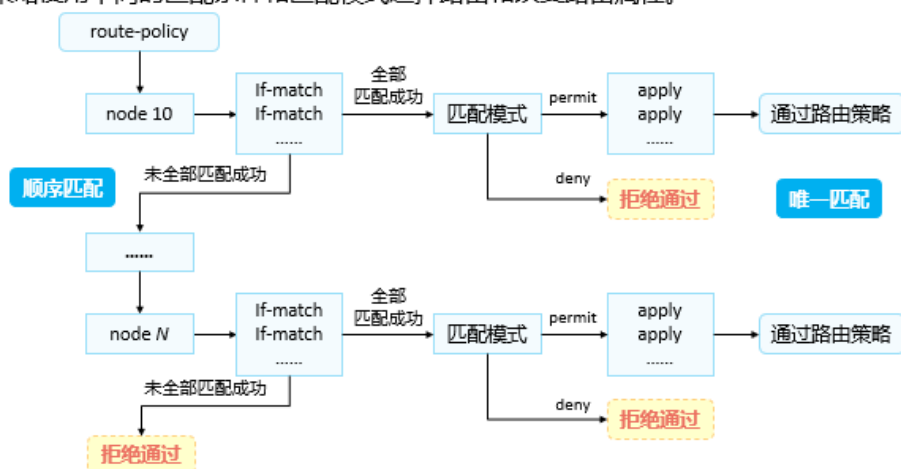


- permit或deny：指定Route-Policy节点的匹配模式为允许或拒绝。
- node：指定Route-Policy的节点号。整数形式，取值范围是0 ~ 65535。
- if-match子句：定义该节点的匹配条件。
- apply子句：定义针对被匹配路由执行的操作。



Route-Policy的匹配顺序

路由策略使用不同的匹配条件和匹配模式选择路由和改变路由属性。



- 一个路由策略中包含 N ($N \geq 1$) 个节点 (Node)。路由进入路由策略后，按节点序号从小到大依次检查各个节点是否匹配。匹配条件由 If-match 子句定义。
- 当路由与该节点的所有 If-match 子句都匹配成功后，进入匹配模式选择，不再匹配其他节点。匹配模式分 permit 和 deny 两种：
 - permit：路由将被允许通过，并且执行该节点的 apply 子句对路由信息的一些属性进行设置。
 - deny：路由将被拒绝通过。
- 当路由与该节点的任意一个 If-match 子句匹配失败后，进入下一节点。如果和所有节点都匹配失败，路由信息将被拒绝通过。



Route-Policy的基础配置命令 (1)

Filter-Policy

Route-Policy

1. 创建Route-Policy

```
[Huawei] route-policy route-policy-name { permit | deny } node node
```

创建路由策略并进入Route-Policy视图。

2. (可选) 配置if-match子句

```
[Huawei-route-policy] if-match ?  
acl          匹配基本ACL  
cost         匹配路由信息的cost  
interface    匹配路由信息的出接口  
ip-prefix    匹配前缀列表  
.....
```

- 命令：**route-policy** *route-policy-name* { **permit** | **deny** } **node** *node*
- **permit**：指定 Route-Policy 节点的匹配模式为允许。如果路由与节点所有的 if-match 子句匹配成功，则执行此节点 apply 子句；否则，进行下一节点。
- **deny**：指定 Route-Policy 节点的匹配模式为拒绝。如果路由与节点所有的 if-match 子句匹配成功，则该路由将被拒绝通过；否则进行下一节点。
- **node** *node*：指定 Route-Policy 的节点号。当使用 Route-Policy 时，node 的值小的节点先进行匹配。一个节点匹配成功后，路由将不再匹配其他节点。全部节点匹配失败后，路由将被过滤。整数形式，取值范围是 0 ~ 65535。



Route-Policy的基础配置命令 (2)

Filter-Policy

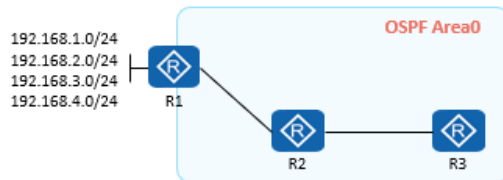
Route-Policy

3. (可选) 配置apply子句

```
[Huawei-route-policy] apply ?  
cost          设置路由的cost  
cost-type {type-1 | type-2} 设置OSPF的开销类型  
ip-address next-hop 设置IPv4路由信息的下一跳地址  
preference    设置路由协议的优先级  
tag           设置路由信息的标记域  
.....
```

- apply 子句用来为路由策略指定动作，用来设置匹配成功的路由的属性。在一个节点中，如果没有配置 apply 子句，则该节点仅起过滤路由的作用。如果配置一个或多个 apply 子句，则通过节点匹配的路由将执行所有 apply 子句。

对接收的路由进行过滤



R2的配置如下:

```
[R2] ip ip-prefix in index 10 permit 192.168.2.0 24
[R2] ip ip-prefix in index 10 permit 192.168.3.0 24
[R2] ip ip-prefix in index 10 permit 192.168.4.0 24
```

```
[R2] ospf
```

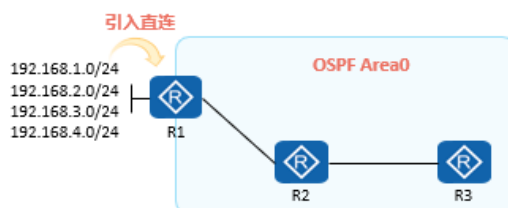
```
[R2-ospf-1] filter-policy ip-prefix in import
```

- R1、R2、R3 运行 OSPF，R1 将 192.168.1.0/24、192.168.2.0/24、192.168.3.0/24 和 192.168.4.0/24 宣告进 OSPF。
- 现在要求 R2 不能访问 R1 上 192.168.1.0/24 网段，但是 R3 可以正常访问。
- 为实现该需求，可以在 R2 上对接收的路由使用 Filter-Policy 进行过滤。

注意：网络基础配置略。

- 通过 Filter-Policy 进行路由过滤，只影响本地的路由表，因此 R3 可以正常学习到 192.168.1.0/24 网段的路由。

对发布的路由进行过滤



R1的配置如下:

```
[R1] ip ip-prefix out index 10 permit 192.168.1.0 24
```

```
[R1] ospf
```

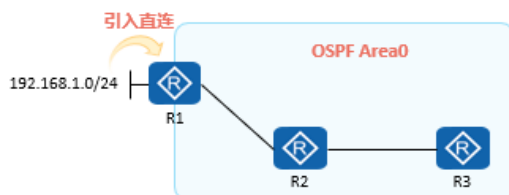
```
[R1-ospf-1] import-route direct
```

```
[R1-ospf-1] filter-policy ip-prefix out export
```

- R1、R2、R3 运行 OSPF，R1 将直连网段 192.168.1.0/24、192.168.2.0/24、192.168.3.0/24 和 192.168.4.0/24 引入 OSPF。
- 现在要求 R2、R3 只能学习到 192.168.1.0/24 网段的路由，学习不到其他三个网段的路由。
- 为实现该需求，可以在 R1 上使用 Filter-Policy 对引入的路由在发布时进行过滤。

注意：网络基础配置略。

修改路由属性



- R1、R2、R3运行OSPF，R1将直连网段192.168.1.0/24引入OSPF。
- 现在要求R2、R3学到的OSPF路由192.168.1.0/24为external-type 1路由（默认为external-type 2路由）。
- 为实现该需求，可以在R1上使用Route-Policy在引入路由时修改外部路由的类型为external-type 1。

R1的配置如下：

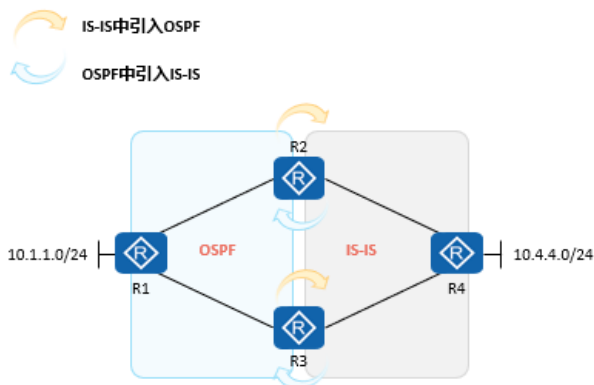
```
[R1] ip ip-prefix external index 10 permit 192.168.1.0 24

[R1] route-policy RP permit node 10
[R1-route-policy] if-match ip-prefix external
[R1-route-policy] apply cost-type type-1
[R1-route-policy] quit

[R1] ospf
[R1-ospf-1] import-route direct route-policy RP
```

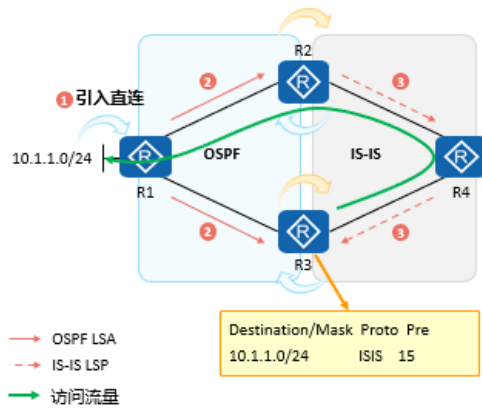
注意：网络基础配置略。

双点双向路由重发布



- 在边界路由器上把两个路由域的路由相互引入，称之为双向路由重发布。
- 两个路由域存在两个边界路由器，并且都执行双向路由重分发，此时称为双点双向路由重发布。
- 双点双向路由重发布是一种经典的路由模型，因单点的双向路由重发布缺乏冗余性，一旦单点的边界路由器故障，那么两个路由域之间的通信可能就会出现异常，因此在大型网络部署中一般采用双点双向路由重发布。
- 双点双向路由重发布虽然增强了网络的可靠性，但是容易引发：次优路径、路由环路等问题。

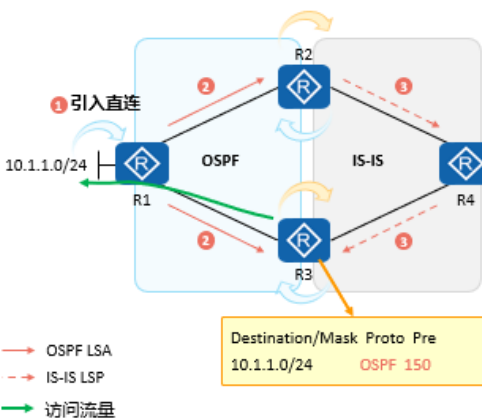
次优路径问题



以10.1.1.0/24为例：

- R1将直连路由10.1.1.0/24引入到OSPF中。
- R2、R3执行双向路由重发布，R2先将10.1.1.0/24重发布到IS-IS中，R3将会学习到来自R4的IS-IS路由。
- 对R3而言，IS-IS路由（优先级15）优于OSPF外部路由（优先级150），因此优选来自R4的IS-IS路由。后续R3访问10.1.1.0/24网段的路径为：R3->R4->R2->R1，这是次优路径。

解决次优路径问题 (1)

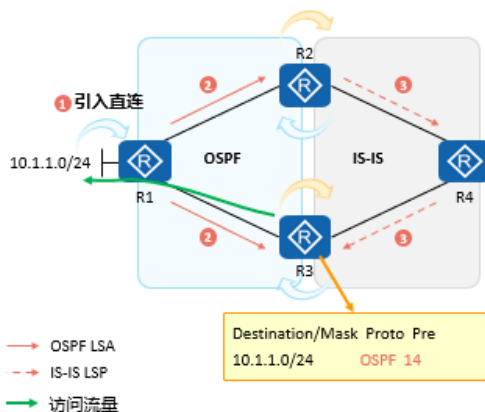


- 解决方案一：在R3的IS-IS进程内，通过Filter-Policy禁止来自R4的10.1.1.0/24路由加入本地路由表。
- 在R3上执行以下操作：

```
[R3] acl 2001
[R3-acl-basic-2001] rule 5 deny source 10.1.1.0 0
[R3-acl-basic-2001] rule 10 permit

[R3] isis
[R3-isis-1] filter-policy 2001 import
```


解决次优路径问题 (2)



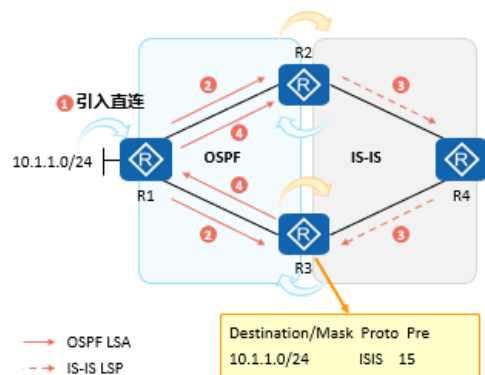
- 解决方案二：R3通过ACL匹配10.1.1.0/24路由，在Route-Policy中调用该条ACL，将匹配这条ACL的路由的优先级设置为14（优于IS-IS）。在OSPF视图下使用

```
preference ase
```

命令调用Route-Policy修改外部路由的优先级。
- 在R3上执行以下操作：

```
[R3]acl 2000
[R3-acl-basic-2000] rule permit source 10.1.1.0 0
[R3-acl-basic-2000] quit
[R3]route-policy hcip permit node 10
[R3-route-policy] if-match acl 2000
[R3-route-policy] apply preference 14
[R3-route-policy] quit
[R3]ospf 1
[R3-ospf-1] preference ase route-policy hcip
```

路由环路问题

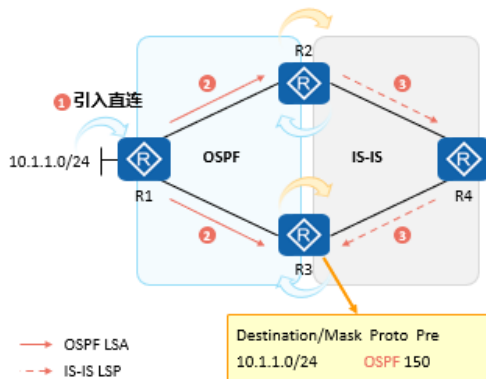


场景描述：

- R1将直连路由10.1.1.0/24引入到OSPF中。
- R1、R2、R3运行OSPF协议，10.1.1.0/24网段路由在全OSPF域内通告。
- R2执行了双向路由重发布。
- R2、R3、R4运行IS-IS协议，10.1.1.0/24网段路由在全IS-IS域内通告。
- R3执行了双向路由重发布。
- 10.1.1.0/24网段路由再次被通告进OSPF域内，形成路由环路。



解决路由环路问题 (1)



- 解决方案一：在R3的OSPF中引入IS-IS路由时，通过Route-Policy过滤掉10.1.1.0/24路由。
- 在R3上执行以下操作：

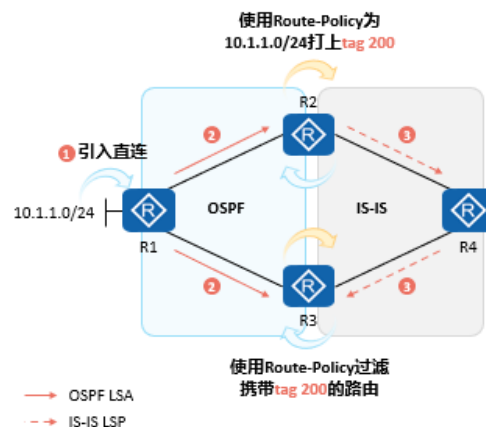
```
[R3] acl 2001
[R3-acl-basic-2001] rule 5 deny source 10.1.1.0 0
[R3-acl-basic-2001] rule 10 permit

[R3] route-policy RP permit node 10
[R3-route-policy] if-match 2001
[R3-route-policy] quit

[R3] ospf
[R3-ospf-1] import-route isis 1 route-policy RP
```



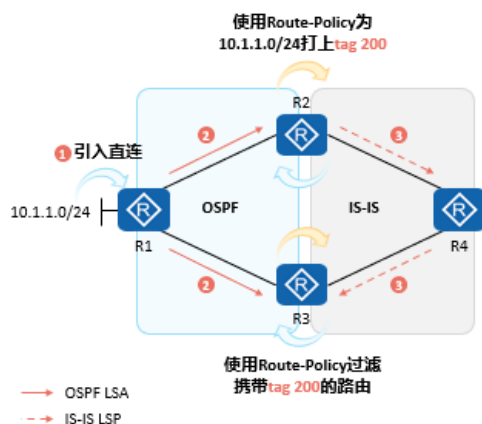
解决路由环路问题 (2)



- 解决方案二：使用Tag实现有选择性地将路由引入，在R2上将路由10.1.1.0/24从OSPF引入到IS-IS中时打上Tag 200，在R3上将IS-IS引入到OSPF中时，过滤携带Tag 200的路由。
- 在R2上执行如下操作：

```
[R2] acl 2000
[R2-acl-basic-2000] rule permit source 10.1.1.0 0
[R2-acl-basic-2000] quit
[R2] route-policy hcip permit node 10
[R2-route-policy] if-match acl 2000
[R2-route-policy] apply tag 200
[R2-route-policy] quit
[R2] isis 1
[R2-isis-1] import-route ospf route-policy hcip
```

解决路由环路问题 (3)



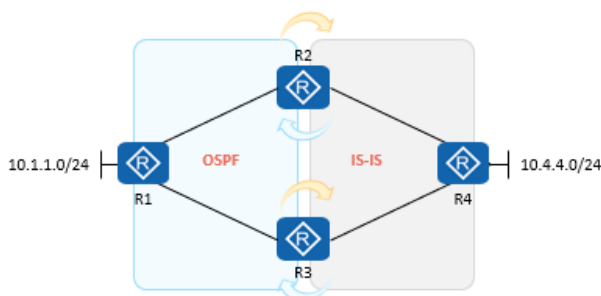
- 在R3上执行如下操作:

```
[R3]route-policy hcip deny node 10
[R3-route-policy]if-match tag 200
[R3-route-policy]quit
[R3]route-policy hcip permit node 20

[R3]ospf 1
[R3-ospf-1]import-route isis route-policy hcip
```

在路由重发布的实际应用中，通过IP前缀进行路由匹配固然可行，但当网络规模较大时，配置工作量较大；通过Tag进行路由匹配可以极大简化配置工作量。

场景思考



在R1上将10.1.1.0/24引入到OSPF，在R4上将10.4.4.0/24引入到IS-IS，R2、R3上执行路由双向路由重发布，该场景下如何使用匹配Tag的方式防止路由环路？

思考题：

- (简答题) Filter-Policy export 在 OSPF、BGP 中的作用分别是？
- (简答题) Route-Policy 多个节点之间的逻辑关系为？一个节点内多个条件语句的逻辑关系为？

答案：

- 在 OSPF 中的作用为过滤从其他路由协议引入到 OSPF 中的路由条目；在 BGP 中的作用为限制本地对外发布的路由

条目。

- 节点之间的逻辑关系为或，条件语句间的逻辑关系为与。
-