

JUNOS 'upto' v. 'through' Route-filter Match-type

Stephen Gill
E-mail: gillsr@cymru.com
Revision: 1.0, 05/21/2001

Contents

Introduction	2
Explanation	2
Example	3
Reinforcement	4

Introduction

A director at Juniper Networks once asked me of a good way to explain the "through" match type on a route-filter within a policy statement. After thinking about it carefully I realized that I didn't have a good explanation for how the option worked either, only that one should only use it for minimal corner cases. Naturally the question made me devote some thought to the subject.

Explanation

Route filters are used to specify groups of prefixes in policy statements so as to perform certain types of actions on them. For the most part, how each of the match types function is fairly self-explanatory. Unfortunately it is a little bit more difficult to determine the difference between the "upto" and "through" match-types because they seem so similar when in fact they are quite distinctive. If one could guess without any outside help what the latter of these two options actually accomplishes I would be thoroughly impressed.

The main difference is that with the "through" match-type one must specify a network **and** a prefix mask for the second argument, whereas with the "upto" keyword, only a prefix mask is necessary. Upto works exactly like Cisco in which for a given netblock, all of the prefix sizes will match from point A to point B. The "through" keyword is not just looking at the initial network prefix and then all the sub-prefixes that may come after it. Rather, it is starting at the source prefix and marking the traversal to the destination prefix tracing the actual path that it takes to get there. One should think of "through" as a way to group a contiguous set of exact matches with varying netblocks.

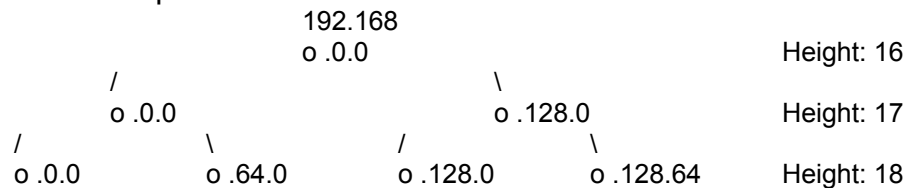
To trace the path from point A to point B one simply needs to traverse the binary tree (Jtree) and mark the steps at each hop. The first prefix dictates at what height in the tree to start, while the second one tells us how low to go. After each step down the tree is marked, the end result will be the outline of the path it took to get there, which is the summary of all the matches for the "through" statement.

Example

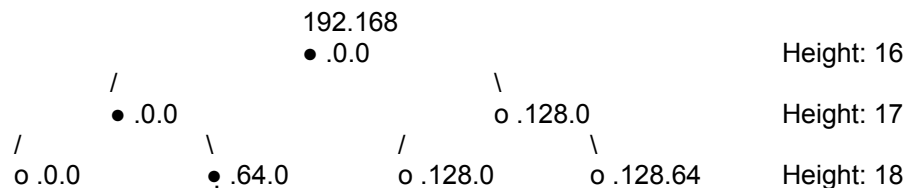
Consider the following route-filter example:

```
route-filter 192.168.0.0/16 through 192.168.64.0/18
```

To figure out what matches actually occur, let's take a look at the tree that represents the above prefixes.



Now, if we were to trace the path that it took to get from point A (192.168.0.0/16) to point B (192.168.64.0/18) it would look like this:



Thus, the matches would consist of the following:

```
route-filter 192.168.0.0/16 exact
route-filter 192.168.0.0/17 exact
route-filter 192.168.64.0/18 exact
```

We could have specified these three netblocks individually in a policy-statement and the net effect would have been the same. What differs is the amount of typing involved in the initial setup. As it likely appears, there are perhaps very few circumstances in which a match like "through" would come in handy. Supposedly some individuals have managed to allocate IP addresses in their network in such a way that this comes in handy. Unless it saved hours of typing, I would rather type in a few extra keystrokes and make my configuration more readable.

That said, if you do decide to use the "through" statement, here are a few general rules of thumb:

- Prefix A must be equal to or less than prefix B.
- Prefix A and prefix B will always match.
- The most matches you can have are 32+1 as show in examples #2 and #3.

- Each match is considered an exact match.
- Numbers in the binary tree "restart" at each octet (every 8 bits or 8 nodes high).

Reinforcement

Following are a few more examples that may or may not help to get you started. Drawing a binary tree might help understand the examples a bit better, but I will leave that exercise to the reader.

1. `route-filter 192.168.0.0/16 through 192.168.16.0/20 accept`

$[\text{Mask B} - \text{Mask A}] + 1 = [20 - 16] + 1 = 5$. Therefore it will match 5 times. Prefix A and prefix B will always match as shown below by the first and last filters:

```
route-filter 192.168.0.0/16 exact accept
route-filter 192.168.0.0/17 exact accept
route-filter 192.168.0.0/18 exact accept
route-filter 192.168.0.0/19 exact accept
route-filter 192.168.16.0/20 exact accept
```

2. `route-filter 0.0.0.0/0 through 0.0.0.0/32 accept`

$[\text{Mask B} - \text{Mask A}] + 1 = [32 - 0] + 1 = 33$. Therefore it will match 33 times.

```
route-filter 0.0.0.0/0 exact accept
route-filter 0.0.0.0/1 exact accept
route-filter 0.0.0.0/n exact accept
```

3. `route-filter 0.0.0.0/0 through 255.255.255.255/32 accept;`

$[\text{Mask B} - \text{Mask A}] + 1 = [32 - 0] + 1 = 33$. Therefore it will match 33 times.

```
route-filter 0.0.0.0/0 exact accept
route-filter 128.0.0.0/1 exact accept
route-filter 192.0.0.0/2 exact accept
route-filter 224.0.0.0/3 exact accept
route-filter 240.0.0.0/4 exact accept
route-filter 248.0.0.0/5 exact accept;
route-filter 252.0.0.0/6 exact accept
route-filter 254.0.0.0/7 exact accept
route-filter 255.0.0.0/8 exact accept
route-filter 255.128.0.0/9 exact accept
route-filter 255.192.0.0/10 exact accept
route-filter 255.224.0.0/11 exact accept
route-filter 255.240.0.0/12 exact accept
route-filter 255.248.0.0/13 exact accept
route-filter 255.252.0.0/14 exact accept
route-filter 255.254.0.0/15 exact accept
route-filter 255.255.0.0/16 exact accept
route-filter 255.255.128.0/17 exact accept
```

```
route-filter 255.255.192.0/18 exact accept
route-filter 255.255.224.0/19 exact accept
route-filter 255.255.240.0/20 exact accept
route-filter 255.255.248.0/21 exact accept
route-filter 255.255.252.0/22 exact accept
route-filter 255.255.254.0/23 exact accept
route-filter 255.255.255.0/24 exact accept
route-filter 255.255.255.128/25 exact accept
route-filter 255.255.255.192/26 exact accept
route-filter 255.255.255.224/27 exact accept
route-filter 255.255.255.240/28 exact accept
route-filter 255.255.255.248/29 exact accept
route-filter 255.255.255.252/30 exact accept
route-filter 255.255.255.254/31 exact accept
route-filter 255.255.255.255/32 exact accept
```