85th IETF – Atlanta, USA

draft-asghar-pim-explicit-rpf-vector-00

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Asghar</td>
<td><a href="mailto:jasghar@cisco.com">jasghar@cisco.com</a></td>
</tr>
<tr>
<td>IJ. Wijnands</td>
<td><a href="mailto:ice@cisco.com">ice@cisco.com</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:sowkrish@cisco.com">sowkrish@cisco.com</a></td>
</tr>
<tr>
<td>V. Arya</td>
<td><a href="mailto:varya@directv.com">varya@directv.com</a></td>
</tr>
</tbody>
</table>
Problem Statement

• This draft documents a solution to build multicast trees via an explicitly configured path sent in the PIM join

• Describes a special use of the Reverse Path Forwarding (RPF) Vector TLV as defined in [RPC 5496]
Motivation behind this draft

- A stack of RPF vectors can be specified to route PIM Joins semi-explicitly using the neighbor addresses:
  1. However, upon a link/node failure the addresses within a stack of RPF vectors could be unreachable
  2. In this case, router will perform a RIB unicast source reachability lookup and route the PIM Join around the link/node failover and not use the desired RPF vector stack path
  3. In a live-live multicast network or Ring topology, both disjoint multicast trees could be routed along the same path, and not longer be disjoint
- Our draft addresses these issues by proposing a new encoding method that allows to explicitly route PIM Joins using Explicit PIM Vector TLV Stack:
  - `draft-asghar-pim-explicit-rpf-vector-00`
Solution Example (this draft): Explicit Path Vector TLV Stack

1. Multicast Source IP: $S = 10.0.0.1$
   - R2: 12.0.0.1
   - R3: 13.0.0.1
   - R4: 14.0.0.1

Explicitly routed path for PIM Join using RPF vector TLV stack

RIB RPF computed path for PIM Join
Solution (this draft)

• Multicast join path R4->R3->R2->R1, where the forwarding states are installed hop-by-hop dynamically

```
<--- (S,G) Join ---
[S]---(R1)--(R2)---(R3)--(R4)---[R]
   |       |       |
   (R5)---(R6)
```

• Multicast join path R4->R3->R6->R5->R2->R1, where the multicast JOIN is explicitly routed to the source hop-by-hop using the explicit RPF vector list

```
[S]---(R1)--(R2)---(R3)--(R4)---[R]
   <--- |       |       ---
   |   |       |       |
   |   (R5)---(R6) |
   - (S,G) Join -
```
Explicit RPF Vector Attribute TLV Format

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|S| Type | Length | Value
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+.....

F bit
-----
Forward Unknown TLV. If this bit is set the TLV is forwarded regardless of whether the router understands the Type. If the TLV is known the F bit is ignored.

S bit
-----
Bottom of Stack. If this bit is set then this is the last TLV in the stack.

Type
----
The Vector Attribute type is 1.

Length
-----
Length depending on Address Family of Encoded-Unicast address.

Value
-----
Encoded-Unicast address.
Moving forward

• Looking for your feedback
• We are open to co-authoring