Independent Submission
Internet-Draft
Expires: March 24, 2007

M. Stenberg
O. Troan
cisco Systems, Inc.

September 20, 2006

IPv6 Prefix Delegation routing state maintenance approaches
draft-stenberg-pd-route-maintenance-00

Status of this Memo

Copyright Notice

Abstract

The maintenance of Prefix Delegation (PD) routing state is an issue that people have discussed in the IETF DHC WG, and there have been drafts on the topic. However, as the pros and cons of the different routing state maintenance solutions have not been examined thoroughly, this text attempts to shed some light on both the actual problem and the various alternative solutions.

1. Introduction

A prefix delegation deployment consists of Requesting Routers (RR), Delegating Routers (DR) and possibly a backend provisioning system (see Figure 1). The delegated prefix has to be routed in the network. This document explores various alternatives for how the route for the delegated prefix can be injected in the network, and how the routing state can be maintained.

```
   /~~~~~~~~~\
   | Network |
   \~~~~~~~~~/
    |
    |-----------------------------------------
    |                                         \
    | +-----------------------------+-------------------+
    | | Backend provisioning system | DR 4 (integrated) |
    | +-----------------------------+-------------------+
    |    |                       |             |
    | +------+                +------+     +------+
    |-| DR 1 |                | DR 3 |     | RR 3 |
    \ +------+                +------+     +------+
     --- | --------------------/ |
      +------+                +------+
      | DR 2 |                | RR 2 |
      +------+                +------+
         |
      +------+
      | RR 1 |
      +------+
```

Figure 1: Possible prefix delegation deployment cases.

Prefix delegation is a stateful protocol. The RR needs to maintain state so that it can sub-delegate prefixes to downstream links. The RR maintains soft-state which can be recovered by redoing the prefix request (for example, using Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [1] with the Prefix Delegation options defined in [2]).

If the DR should do route injection on behalf of the RR, it needs to maintain state. The backend provisioning system must maintain a list of prefixes delegated, as a prefix delegation is a long-lived entity (lifetime of a customer relationship, as in months or years). The backend system and DR might run on the same router. This document focuses on the case where the backend system and DR are separate and the DR has little or no persistent storage. Therefore the DR 4 case (in Figure 1) is not covered here, as it is trivial - the backend has nonvolatile storage for the prefixes, and it can re-inject the routes

when the integrated DR 4 restarts.

The DR's routing state that needs to be maintained can be divided into two distinct categories: local routing state (that is, a local RIB entry containing the next-hop and the interface the assigned prefix is connected to), and global (AS-wide) routing state which requires advertising the route via a routing protocol.

Advertising a route per delegation from the DR can be avoided if there is an aggregate prefix covering the delegation. This requires stringent address allocation procedures and prohibits an RR from moving to a different DR.

2. Different approaches for maintaining routing state

As any router (or the backend system for that matter) can go offline and come back up later, it is necessary for the system to recover from these intermittent failures.

The problem is how to delegate responsibility for route maintenance to one (or more) of the three components of the system, and letting it take care of maintaining the required routing state in place for the RR's prefix.

2.1. Backend provisioning system responsible for routing state

Considering the backend provisioning system is the only component in the system that actually requires significant amount of nonvolatile storage, from data system point of view it would be ideal to have the backend provisioning system responsible for maintaining the routing state as well.

It would mean that the backend provisioning system should, when DR restarts, (securely) re-inject the local or global routing state into the DRs.

In practise, this is infeasible:

o There is no standard way of detecting when the DR is restarted.

o Redundancy of DR, or the links between DR and backend system, makes it difficult for the backend system to judge the state of the DR accurately without significant extra configuration data about the deployed configuration.

o None of the current routing protocols are suitable for altering remote router's local routing information, and therefore some protocol development would be in order for this approach to be usable. There are also security implications with this solution.

o Lack of scalability; the benefit of having 'backend' provisioning system disappears as it will need to take care of maintaining routes of every one of its DRs.

o The backend may lack the information to identify the DR to the routing system. With multiple DRs, if the delegation protocol does not contain everything needed to re-inject the route later on to the specific DR, it won't work. For example, DHCPv6 does not uniquely identify the relays. And if interim DRs do not have backend provisioning system-addressible addresses, there is a problem. All DRs may not have global unicast addresses, and this

is problematic especially in configurations spanning multiple administrative domains.

Having considered the backend provisioning system as the responsible component, it is clearly NOT the way to go. That leaves the DR and RR components.

2.2. Delegating router responsible for routing state

As the DR is part of the local routing infrastructure, placing the responsibility for routing state in the DR seems sensible. With that design decision, the next problem is _when_ the routing information is updated after the DR restarts:

2.2.1. Approach 1: On-demand lease query

In the on-demand lease query case as defined in [3], the routing state maintenance problem is assumed to be local, and therefore the DR will receive packets both from the network at large as well as the RR even after a loss of local state caused by a restart.

When traffic arrives to DR either from the RR, or from the network to the DR for a prefix without local routing information, the DR will perform lease query, acquire the allocated prefix, and update the routing information appropriately.

This approach, while simple to specify, has some major issues:

- It depends on the aggregated prefix to cause the inbound traffic to wind up in the DR. This assumption may not be valid, depending on the address assignment policies of the organization. Geographical or network topological hierarchical address assignment at large seems to be a failure, and it is unclear if all deployments can really implement this.

- It requires the incoming traffic both from the RR and the network, for which no route exists, to trigger the lease query. This has two negative side effects: it requires support from the fast path hardware in the DR, and potentially causes large amount of spurious requests to the backend provisioning system (up to the desired rate that is considered harmful to the system).

- It requires simulated ordering of the unordered transaction stream, to ensure that the routing state is maintained correctly. The DR cannot be argued to be particularly stateless anymore.

2.2.2. Approach 2: Anticipatory lease query

Anticipatory, or bulk lease query, solves the routing state problem by requesting ALL prefix information from the backend provisioning system at the DR restart time. There are two different ways: The first approach is asynchronous, that is, the old state is fetched while handling the delegation requests, requiring synchronization algorithm between the bulk data retrieved from the backend system, and the requests served during that. For synchronization, some sort of ordering of the transaction stream is needed. The second alternative is synchronous: the bulk query is performed first, and only then the RRs' requests are handled.

Bulk query has several advantages over the on-demand case:

- No need for triggering based on either inbound or outbound traffic for the prefix.

- If DR handles the query synchronously, we can avoid the ordering of the transaction stream and the associated complexity rising from it.

- Given reasonable TCP transport scheme, the transfer of the state is more efficient than the on-demand case in terms of total number of packets.

- Does not require changes to fast path hardware, as no new triggers are needed from the traffic. Instead, simple additional code in the system initialization is enough.

But, unfortunately it has also some disadvantages:

- It causes more uneven load on the backend provisioning system than the on-demand case. If the prefix is not being actively used at the time, it will not cause traffic in the on-demand case, but it will in the bulk case.

- Synchronization is non-trivial if the DR serves RR requests during the bulk retrieval of the data.

- Doesn't work very well with virtual interfaces - it is hard to retrieve state at boot time if the interfaces themselves get up only at some point, and with their transient nature mapping a DUID to individual customers is difficult.

2.2.3. Approach 3: Persistent storage

It is possible for the DR to store the route information to be injected either locally, or on some adjacent storage node. The clear advantage of this is the lack of traffic on the wire.

Unfortunately, it has also some problems - the data being possibly outdated due to lack of synchronization, and the management overhead when the customers for example move around would be significant.

However, in most deployment scenarios persistent storage at or near all routers is not desirable or possible in the first place, so this is listed simply for the sake of completeness.

2.3. Requesting router responsible for routing state

The most interested party in the routing state of the given prefix is the RR itself; therefore, giving the responsibility for maintaining its routing state to it seemed to be idea worth considering.

Due to the operators wariness of the systems not under their direct control, even with the RR responsible for maintenance of the state, the real route injection should be handled by the DR.

The nice thing about some of the RR-oriented solutions is that they can be deployed without any changes to the rest of the infrastructure.

2.3.1. Approach 1: Layer-2 detection of link state

If the RR implementation gets notifications about the state of the link layer, it can actually detect the state of the network link going down and coming back up; performing reconfiguration to ensure that the routing state is still up seems like a trivial solution in this case.

This solution can be the best one when operating over connection-oriented media (PPPoE, L2TP) but it doesn't work on say, Ethernet without direct connection between the RR and DR.

2.3.2. Approach 2: Keepalive

If the RR doesn't have L2 way of detecting DR being restarted, it can maintain a keepalive mechanism using, for example, Bidirectional Forwarding Detection (BFD - [4]) to send self-addressed echo packets to the DR and waiting for their replies. The implementations SHOULD do this only if there is no traffic from the network within a desired period of time - see IPv6 Neighbor Unreachability Detection (NUD)'s

definition of forward progress detection as a way to send keepalives only when truly necessary in [5].

Assuming sub-second round-trips (reasonable assumption in most modern network environments), the longest factor for the determining the keepalive timeout is the recovery speed of the DR (by orders of magnitude), as it can take from some seconds (hot standby) to minutes (non-HA restart, or cold standby with huge configurations). The initial keepalive timer should be some fraction of the highest delay in the system, that is, the DR recovery time. The subsequent retries if no reply is received within reasonable timeframe should be calculated based on the link delay, and jitter, to ensure that the reply is unlikely to be coming back by the time the keepalive message is re-sent.

As far as overhead is concerned, assuming the cold standby/restart taking minute(s), with a keepalive per 60 seconds for example, the QoS would remain roughly same as with faster intervals (as the DR going down would cause interruption in the routing in the order of minute(s) in any case). This value would cause overhead of 0.017pps per RR, and it is unlikely to be the straw that breaks camel's back for the DR. With any traffic, even NUD packets should outnumber the keepalive traffic.

As far as resource utilization is concerned, this solution involves only routing plane of the RR, the data link between RR and DR, and fast path of DR which bounces the packets back. Therefore it can be argued to be fairly lightweight general-purpose solution.

2.3.3. Approach 3: Short lifetimes

The current best practice for maintaining the routing state is to set short configuration lifetimes (DHCP T1/T2 values). It causes extra traffic and load on the whole DHCP infrastructure. That is because during every reconfiguration, even with the DR constantly up and running, the backend system is queried. The transaction involves all three components. Due to that, every RR will cause constant load on the backend system itself over the time, making the solution simply not scale well.

2.3.4. Approach 4: Routing protocol to the requesting router

The final RR-based approach consists of the RR actually running a routing protocol; this way, the RR router can simply advertise the prefix as it receives it, and everything just works. Or not, as it may be.

The downside is the security, or complete lack of it. The DR

accepting arbitrary RR-advertised prefixes (assuming no state at the DR) should be acceptable only if DR and RR are within the same administrative domain. For that case, this is probably the cleanest solution of all.

If administrative boundaries are crossed, the DR will not take prefix advertisement at face value. The DR will have extra overhead of checking the backend provisioning system for AAA purposes before actually doing anything with the prefix. This can imply look-up for validity using the prefix and the interface the advertisement came from, including the DUID or some other identifier within the route advertisement message, or using some real AAA mechanism if the routing protocol supports one. If minimal changes to the routing protocol implementation are desired, it is also possible to ignore the advertisement itself, and just trigger on-demand lease query, thereby using the routing protocol just as an alternative keepalive mechanism the with most of the logic shoved in DR instead of RR.

```
                  /~~~~~~~~~~~\
                  |  Network |
                  \~~~~~~~~~~~/
                      |   |
             ---------/   \---------
            /                       \
+----------------------+   +----------------------+
| Delegating router 1 |   | Delegating router 2 |
+----------------------+   +----------------------+
            \                       /
             -----------+-----------
                        |
             +----------------------+
             | Requesting router   |
             +----------------------+
```

Figure 2: Multihomed deployment.

There is also a case where the RR HAS to run its own routing protocol; in multihomed situation like Figure 2, with the same routable prefix advertised via two different DRs, there is no other practical way to get the system working. Of course, static route configuration is always an alternative but it is seldom desirable.

The routing-protocol-based solutions all require a significant level of trust between RR and DR; regrettably the current routing protocols are not designed with AAA (or security for that matter) in mind, and therefore when crossing administrative boundaries, the alternatives are either using them as-is as a hint that something needs to be done, or significantly extending the protocols in the AAA direction.

Adding extra complexity to the DR's routing protocol implementation or configuration is not desirable in general.

Finally, the current prefix delegation solution (DHCPv6 PD) does not provide the information about which routing protocol to use, and there is no routing protocol auto-negotiation protocol. Therefore the auto-configuration of the RR with arbitrary routing protocol cannot be done currently.

3. Security Considerations

The backend-oriented solution detailed in Section 2.1 implies a significant level of trust between the DR and the backend provisioning system. The system's configuration is simpler if the backend provisioning system can inject arbitrary routes to the DR, but allowing injection of routes for only specific sub-prefixes of a specific prefix is considerably more secure solution. Unfortunately it requires advance configuration of the prefix(es) involved.

The delegating router-based solutions detailed in Section 2.2 do not have any security issues, assuming the delegation protocol itself is secured, or can be assumed to be used only within a trusted network.

The requesting router-based solutions in Section 2.3, even incorrectly implemented, at most just cause extra load to the DR. As noted in Section 2.3.4, even when running routing protocol from the RR, ideally the DR should consider the advertisements only a hint at best if not part of the same adminstrative domain. This may not be ideal if the routing protocol information should be propagated as-is onward, as in the the multihoming cases. Unfortunately, those cases also most likely cross administrative boundaries (the requesting router being part of one domain, and connected to delegating routers in most likely more than one), the providers will not most likely trust the routing protocol to be used as-is at the delegating routers, and their complexity will increase due to the required AAA/policy checks. This is a potential security risk in a critical part of the network infrastructure.

4. IANA Considerations

As this document is informational in nature and only summarizes current best practices, it does not require action from IANA.

5. Summary

The backend provisioning system should not be assigned the responsibility for the maintenance of the route. As seen in Section 2.1, that approach has significant obstacles without any clear benefits.

If the link layer state can be used to detect the (potential) restart of delegating router, the requesting router-based simple reconfiguration described in Section 2.3.1 seems to be the best choice.

When link layer state is not available, there is no clear 'best' solution. The tradeoff seems to be between increasing the complexity of the delegation protocol and the delegating router/backend system (as described in the lease query cases in Section 2.2.1 and Section 2.2.2), decreasing scalability of the system significantly by using low lifetimes for configuration (as described in Section 2.3.3), or small overhead of the keepalive (as described in Section 2.3.2).

Only in multihoming cases, given some extensions to the current prefix delegation protocol, should routing protocol on the requesting router be considered, as described in Section 2.3.4. Multihoming solution itself is challenging to do securely, as noted in Section 3, due to lack of AAA support in routing protocols.

6. References

[1] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[2] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[3] Brzozowski, J., "DHCPv6 Leasequery", draft-ietf-dhc-dhcvp6-leasequery-00 (work in progress), August 2006.

[4] Katz, D. and D. Ward, "Bidirectional Forwarding Detection", draft-ietf-bfd-base-05 (work in progress), June 2006.

[5] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

Appendix A. Acknowledgements

Thanks to Bernie Volz for feedback during writing of the document.

Authors' Addresses

Markus Stenberg
cisco Systems, Inc.
Shinjuku Mitsui Building, 2-1-1, Nishi-Shinjuku
Shinjuku-Ku, Tokyo-to 1630409
JP

Email: mstenber@cisco.com

Ole Troan
cisco Systems, Inc.
Shinjuku Mitsui Building, 2-1-1, Nishi-Shinjuku
Shinjuku-Ku, Tokyo-to 1630409
JP

Email: ot@cisco.com

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.