Use cases for DIS Modifications

Georgios Papadopoulos

RAW
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

G. Papadopoulos
IMT Atlantique

March 9, 2020

Use cases for DIS Modifications
draft-papadopoulos-roll-dis-mods-use-cases-00

Abstract

This document presents some of the use-cases which call for DIS flags and options modifications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2020.

Table of Contents

1. Introduction

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Applications

This section details some use cases that require DIS modifications compared to the behaviour currently defined in [RFC6550]. The first use case is thatof a new leaf node joining an established DAG in an energy efficient manner. The second use case describes why node might want to use DIS to identify defunct DAGs for which it still maintains state. The third use case describes the need for adjacency probing and how DIS can used for that.

3.1. A Leaf Node Joining a DAG

This use case is typically of a smart meter being replaced in the field, while a RPL network is operating and stable. The new smart meter must join the network quickly, without draining the energy of the surrounding nodes, be they battery-operated RPL routers or leaf nodes. In this use case, the issues with the current RPL specification are

- Just waiting for a gratuitous DIO may take a long time if the Trickle timers have relaxed to the steady state. A technician who has just installed the new meter needs to positively assess that the meter has joined the network before it leaves the premise. It is not economically viable to ask the technician to standby the meter until a gratuitous DIO has arrived, which may take hours.

- If the meter sends a DIS, it needs to do so using multicast, because it has no knowledge of its surroundings. Sending a

multicast DIS is considered an inconsistency by the nearby RPL routers. They will reset their Trickle timer to the shortest period. This will trigger sending a stream of DIOs until the Trickle timers relax again. The DIOs will be sent in multicast, which will trigger energy expenditure at nearby nodes, which had no need for the DIOs.

A proposed solution could be the following. A new leaf node that joins an established LLN runs an iterative algorithm in which it requests (using multicast DIS) DIOs from routers belonging to the desired DAG.

The DIS message has the "No Inconsistency" flag set to prevent resetting of Trickle timer in responding routers, thereby keeping the aggregated number of transmissions low. It also has the "DIO Type" flag set to make responding routers send unicast DIOs back, thereby not triggering full reception in nearby nodes that have state-of-the-art radio receivers with hardware-based address filtering.

The DIS message can include a Response Spreading option prescribing a suitable spreading interval based on the expected density of nearby routers and on the expected Layer 2 technology.

The DIS will likely include a Metric Container listing the routing constraints that the responding routers must satisfy in order to be allowed to respond [RFC6551].

At each iteration, the node multicasts such a DIS and waits for forthcoming DIOs. After a time equal to the spreading interval, the node considers the current iteration to be unsuccessful. The node consequently relaxes the routing constraints somewhat and proceeds to the next iteration.

The cycle repeats until the node receives one or more DIOs or until it has relaxed the constraints to the lowest acceptable values.

This algorithm has been proven in the field to be extremely energy-efficient, especially when routers have a wide communication range.

3.2. Identifying A Defunct DAG

A RPL node may remove a neighbor from its parent set for a DAG for a number of reasons:

- The neighbor is no longer reachable, as determined using a mechanism such as Neighbor Unreachanility Detection (NUD) [RFC4861], Bidirectional Forwarding Detection (BFD) [RFC5881] or L2 triggers [RFC5184]; or

- The neighbor advertises an infinite rank in the DAG; or
- Keeping the neighbor as a parent would required the node to increase its rank beyond L + DAGMaxRankIncrease, where L is the minimum rank the node has had in this DAG; or
- The neighbor advertises membership in a different DAG within the same RPL Instance, where a different DAG is recognised by a different DODAGID or a different DODAGVersionNumber.

Even if the conditions listed above exist, a RPL node may fail to remove a neighbor from its parent set because:

- The node may fail to receive the neighbor's DIOs advertising an increased rank or the neighbor's membership in a different DAG;
- The node may not check, and hence may not detect, the neighbor's unreachability for a long time. For example, the node may not have any data to send to this neighbor and hence may not encounter any event (such as failure to send data to this neighbor) that would trigger a check for the neighbor's reachability.

In such cases, a node would continue to consider itself attached to a DAG even if all its parents in the DAG are unreachable or have moved to different DAGs. Such a DAG can be characterized as being defunct from the node's perspective. If the node maintains state about a large number of defunct DAGs, such state may prevent a considerable portion of the total memory in the node from being available for more useful purposes.

To alleviate the problem described above, a RPL node may invoke the following procedure to identify a defunct DAG and delete the state it maintains for this DAG. Note that, given the proactive nature of RPL protocol, the lack of data traffic using a DAG can not be considered a reliable indication of the DAG's defunction. Further, the Trickle timer based control of DIO transmissions means the possibility of an indefinite delay in the receipt of a new DIO from a functional DAG parent. Hence, the mechanism described here is based on the use of a DIS message to solicit DIOs about a DAG suspected of defunction. Further, a multicast DIS is used so as to avoid the need to query each parent individually and also to discover other neighbor routers that may serve as the node's new parents in the DAG.

When a RPL node has not received a DIO from any of its parents in a DAG for more than a locally configured time duration:

- The node generates a multicast DIS message with:

* the "No Inconsistency" flag set so that the responding routers do not reset their Trickle timers.

* the "DIO Type" flag not set so that the responding routers send multicast DIOs and other nodes in the vicinity do not need to invoke this procedure.

* a Solicited Information option to identify the DAG in question. This option must have the I and D flags set and the RPLInstanceID/DODAGID fields must be set to values identifying the DAG. The V flag inside the Solicited Information option should not be set so as to allow the neighbors to send DIOs advertising the latest version of the DAG.

* a Response Spreading option specifying a suitable time interval over which the DIO responses may arrive.

o After sending the DIS, the node waits for the duration specified inside the Response Spreading option to receive the DIOs generated by its neighbors. At the conclusion of the wait duration:

* If the node has received one or more DIOs advertising newer version(s) of the DAG, it joins the latest version of the DAG, selects a new parent set among the neighbors advertising the latest DAG version and marks the DAG status as functional.

* Otherwise, if the node has not received a DIO advertising the current version of the DAG from a neighbor in the parent set, it removes that neighbor from the parent set. As a result, if the node has no parent left in the DAG, it marks the DAG as defunct and schedule the deletion of the state it has maintained for the DAG after a locally configured "hold" duration. (This is because, as per RPL specification, when a node no longer has any parents left in a DAG, it is still required to remember the DAG’s identity (RPLInstanceID, DODAGID, DODAGVersionNumber), the lowest rank (L) it has had in this DAG and the DAGMaxRankIncrease value for the DAG for a certain time interval to ensure that the node does not join an earlier version of the DAG and does not rejoin the current version of the DAG at a rank higher than L + DAGMaxRankIncrease.)

3.3. Adjacencies probing with RPL

RPL avoids periodic hello messaging as compared to other distance vector protocols. It uses trickle timer based mechanism to update configuration parameters. This significantly reduces the RPL control overhead. One of the fallout of this design choice is that, in the

absence of regular traffic, the adjacencies could not be tested and repaired if broken.

RPL provides a mechanism in the form of unicast DIS to query a particular node for its DIO. A node receiving a unicast DIS MUST respond with a unicast DIO with Configuration Option. This mechanism could as well be made use of for probing adjacencies and certain implementations such as Contiki uses this. The periodicity of the probing is implementation dependent, but the node is expected to invoke probing only when

- o There is no data traffic based on which the links could be tested.
- o There is no L2 feedback. In some case, L2 might provide periodic beacons at link layer and the absence of beacons could be used for link tests.

3.3.1. Deliberations

- o Should the probing scheme be standardized? In some cases using multicast based probing may prove advantageous.
- o In some cases using multicast based probing may prove advantageous. Currently RPL does not have multicast based probing. Multicast DIS/DIO may not be suitable for probing because it could possibly lead to change of states.

4. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC5184] Teraoka, F., Gogo, K., Mitsuya, K., Shibui, R., and K. Mitani, "Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover", RFC 5184, DOI 10.17487/RFC5184, May 2008, <https://www.rfc-editor.org/info/rfc5184>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <https://www.rfc-editor.org/info/rfc6551>.

Author's Address

Georgios Z. Papadopoulos
IMT Atlantique
Office B00 - 102A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 04
Email: georgios.papadopoulos@imt-atlantique.fr