Eliding and Querying RPL Information
draft-thubert-roll-eliding-dio-information-02

Abstract

This document presents a method to safely elide a group of RPL options in a DIO message by synchronizing the state associated with each of these options between parent and child using a new sequence counter in DIO or DAO messages. A child that missed a DIO message with an update of any of those protected options detects it by the change of sequence counter and queries the update with a DIS Message. The draft also provides a method to fully elide the options in a DAO message.

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 27 April 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Classical Link State protocol synchronize their Link State Database (LSDB) by sequencing every change. Each interested node maintains the last sequence of the LSDB it is synchronizing with. If the last known sequence number is older than the current, the node needs to learn one by one all the state changes between the last known and the current state.
[RPL] does not operate that way. With RPL, the routing information is repeated over and over in DODAG Information Object (DIO) and Destination Advertisement Object (DAO) messages. There is no concept of synchronization. The most recent information overrides a previous one and a stale state eventually times out.

The RPL way was designed to enable routing from most nodes to most nodes most of the time in a Low-Power Lossy Network (LLN) where the quality of the links and the cost of communications does not permit to maintain a permanent synchronization.

This principle was applied to both the routing and non-routing information such as configuration settings, prefix information, and node capabilities.

This non-routing state is carried in RPL Messages as options. Some of the DIO options may be needed to decide whether a node can join a network as a leaf or as a router, and may affect the parent selection or the address selection. It is thus critical that each node maintains its state to the freshest and selects parents that are also synchronized to the freshest.

[RPL] allows a parent to elide options in the DIO messages that it sends repeatedly, to conserve battery and save bandwidth. When it does so, a newcomer child that missed DIOs that contained the configuration option may operate on default or partial information. If it is pessimistic, it may query all possible the information even when it is not needed. Conversely, a node that slept may have missed a DIO with a change in some critical information and may not be even aware of it, so it may fail to query for the update and operate on deprecated parameters.

This document uses a new sequence counter called RCSS to synchronize the state in a child node with that of its parent, and recursively with that of the whole network, to the latest setting from the Root.

The protected options are:

1. The Route Information Option (RIO) defined in section 6.7.5 of [RPL]
2. The DODAG Configuration Option (DCO) defined in section 6.7.6 of [RPL]
3. The Prefix Information Option (PIO) defined in section 6.7.10 of [RPL]
4. The Extended MOP Option (MOPex) defined in [MOPEX-CAP]
5. The Global Capabilities Option (GCO) defined in [MOPEX-CAP]

Any change in those options causes an increment of the RCSS and enables a network-wide synchronization to the new state. If the change impacts the routing substantially, the Root should decide to increment the Version Number at the same time to fully rebuild the DODAG with the new settings of the options. It must be noted that the operation of the Version Number in itself provides no guarantee that the non-routing state is fully resynchronized everywhere unless all the options are present in all the DIO messages.

2. Terminology

2.1. BCP 14

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. References

The Terminology used in this document is consistent with and incorporates that described in Terms Used in Routing for Low-Power and Lossy Networks [RFC7102].

Other terms in use in LLNs are found in Terminology for Constrained-Node Networks [RFC7228].

A glossary of classical RPL acronyms is given in Section 2.3.

The term "byte" is used in its now customary sense as a synonym for "octet".

"RPL", "RPL Packet Information" (RPI) and "RPL Instance", DIO, DAO and DIS messages are defined in the "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC7102] specification.

This document uses the terms RPL-Unaware Leaf (RUL) and RPL Aware Leaf (RAL) consistently with [USE_OF_RPL_INFO].

The term RPL-Unaware Leaf (RUL) is used to refer to a node that uses a RPL router (without necessarily knowing it) as 6LR and depends on that router to obtain reachability for its addresses inside the RPL domain. On the contrary, the term RPL-Aware Node (RAN) is used to refer to a RAL or a RPL router that participates to RPL and advertises its addresses of prefixes by itself.
2.3. Glossary

This document often uses the following acronyms:

DODAG  Destination-Oriented Directed Acyclic Graph
LLN  Low-Power and Lossy Network
RPI  RPL Packet Information (an Option in the Hop-By_Hop Header)
RAL  RPL-Aware Leaf
RAN  RPL-Aware Node, a RPL router or a RPL-Aware Leaf
RS  Router Solicitation
RCSS  RPL Configuration State Sequence
RPL  IPv6 Routing Protocol for LLNs (pronounced ripple)
RUL  RPL-Unaware Leaf

3. Updating RFC 6550

This document adds a new field called RCSS to the DIO message. The RCSS is a sequence counter set by the Root and operated as specified in Section 7 of [RPL], more in Section 5.

This document also introduces a new RPL Control Message Option called the Abbreviated Option Option (AOO). The AOO is the compressed replacement of a protected option that indicates the RCSS of the last change of that option, but elides its content, more in Section 4.3.

This document modifies the DIS Base Object to enable the individual query of the protected options by a node that missed a change, more in Section 4.2.

This document also enables to abbreviate a full DAO message when all the options are unchanged from the previous DAO message that was positively acknowledged. In that case the DAO is resent with the same DAOSquence and all the options are elided. A new flag in the DAO Base Object indicates that this is an abbreviated DAO message, more in Section 7.

4. Message Formats
4.1. Updated DIO Base Object

The format of the DIO Base Object is defined in section 6.3.1 of [RPL]. This specification uses a 8th octet that was reserved in [RPL] to transport the RCSS.

```
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RPLInstanceID | Version Number |             Rank              |
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G|O| MOP | Prf |     DTSN      |     Flags     |      RCSS     |
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               +
|                                                               |
+                            DODAGID                            +
|                                                               |
+                                                               +
|                                                               +
|                                                               |
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option(s)...   |
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Updated DIO Base Object

Updated fields:

RCSS
One Byte, the RPL Configuration State Sequence

4.2. Updated DIS Base Object

The DIS Base Object is used by a child to query from a parent the most recent changes in protected options. This specification adds flags to indicate which options are requested and the freshest RCSS to which the querying node was synchronized.

```
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|D|P|M|O| Flg | LastSync RCSS | Option(s)... |
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Updated DIS Base Object

Updated fields:
R
One Bit, indicates that the RIO is requested

D
One Bit, indicates that the DCO is requested

P
One Bit, indicates that the PIO(s) is(are) requested

M
One Bit, indicates that the MOPex is requested

O
One Bit, indicates that the GCO is requested

Last Synchronized RCSS
One Byte, indicates the freshest RCSS to which the querier was synchronized

4.3. New Abbreviated Option Option

The abbreviated version of an option is a replacement for any option. It does not advertise the content of the option but indicates the sender’s value for the RCSS of that option. It is transported in a 4-bytes long Abbreviated Option Option (AOO) with a format as below:

```
  0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  | Option Length | Abbrev. opt.  | Last Mod RCSS |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Abbreviated Option Option Format

Option fields:

Option Type
One byte indicating "Abbreviated Option", see Table 3 in Section 9.3

Option Length
MUST be set to 2 indicating Option data of 2 bytes

Abbreviated Option
The Option Type of the option being abbreviated

Last Modification RCSS
The RCSS at which the option was last modified
4.4. Updated DAO Base Object

The format of the DAO Base Object is defined in section 6.4.1 of [RPL]. This specification adds a ‘A’ flag in to indicate the DAO options eliding.

```
+---------------+---------------+---------------+---------------+
| RPLInstanceID |K|D|A| Flags   |   Reserved    | DAOSequence   |
+---------------+---------------+---------------+---------------+
| +---------------+---------------+---------------+---------------+
| + DODAGID*     |
| +---------------+---------------+---------------+---------------+
| Option(s)...   |
| +---------------+---------------+---------------+---------------+
```

Figure 4: Updated DAO Base Object

Updated fields:

A

One Bit, indicates DAO in abbreviated version

5. RCSS Operation

Settings and updates to network-wide parameters are initiated by the Root and propagated down the DODAG in RPL Control Message Options in DIO messages. The DIO messages arrive asynchronously via different parents and may confuse a child. The RCSS allows the child to keep synchronized to the latest settings network-wide parameters that are propagated in protected options.

The RCSS is a sequence number that is operated as specified in section 7.2 of [RPL]. The scope of an RCSS is one DODAG within one RPL Instance. The RCSS applies to a DIO Message and a same value of the RCSS can be used in DIO messages that are sent consecutively with no change in the protected options.

The Root of the DODAG is authoritative to set and update the RCSS and the options that it protects. The RCSS and the protected options are propagated together down the DODAG without a change, more in Section 5.1.
The RCSS allows a child node to recognize the fresher DIO Message(s) as received from one or more advertising parents and to use only parents with a consistent state of network-wide parameters, more in Section 5.2.

By extension, the RCSS is also defined for each protected option. A child associates an option with the values of the RCSS indicated in DIO Messages in which the option is advertised and uses it to assess the relative freshness of different versions of an option, more in Section 5.3.

Unchanged options may be sent in full, elided, or in the abbreviated form specified in Section 4.3. Which form to use depends on the RCSS, more in Section 5.3.

If the link MTU does not permit to send a single DIO message with all the options packaged then the options may be spread over multiple consecutive DIO messages with the same RCSS that are sent in a rapid sequence.

5.1. Updating the RCSS

The RCSS is incremented by the Root using a lollipop technique as specified in section 7.2 of [RPL]. RCSS values are comparable if they are within a window of comparison of SEQUENCE_WINDOW increments or one indicates a reboot.

A reboot of the Root is detected when the RCSS moves from the circular to the straight part of the lollipop. In order to maximize the chances of detection, the straight part should be kept very short with a RECOMMENDED initialization at 252 or above.

During the straight part of the lollipop, a second reboot of the Root might not be recognized and a same value of the RCSS may reappear with different settings in the protected options. For that reason the protected options MUST be provided in full with each increment on the RCSS during the straight part of the lollipop.

When a field is modified in one of the protected options, the Root MUST send a DIO with an incremented RCSS and the modified protected option(s) in full. The Root MAY also update the Version Number to form a new DODAG altogether.

The Root SHOULD jump rapidly away from the straight part once the network has sufficiently settled by resetting the RCSS to 0, which places the RCSS in the circular region of the lollipop, where the protected options MAY be elided or abbreviated.
5.2. RCSS Freshness and Parent selection

A child node maintains the freshest RCSS received from its parents in each of the RPL Instances that it participates to, and uses that RCSS for its own DIO messages once it has all the protected options synchronized to it.

A child and a candidate parent are out-of-sync when the RCSS values that they maintain for a RPL Instance are not comparable. A child MUST NOT use a parent that is out-of-sync unless no other parent is available, in which case it MAY align its RCSS and resynchronize to that parent.

When a child receives from a candidate parent a DIO with an RCSS that is fresher than the one it is using, the child MUST synchronize the state relative to the protected options with that parent. The child node MUST refrain from using that parent and the new state including the RCSS, until it has synchronized all of the protected options to that RCSS. When it is fully synchronized, the child may then use that parent and the new RCSS.

Using a back-level parent may cause packets to be dropped, misunderstood or misrouted. The child SHOULD refrain from using a parent that exposes an older RCSS if the change causes an incompatibility issue.

5.3. RCSS of an Option

By extension, the RCSS of an option is maintained by all nodes and is defined for all but the root as the freshest RCSS indicated by a DIO message from a candidate parent in which the option was present, in the abbreviated form or in full. A child maintains a state for the RCSS of each of the protected options and synchronizes its state for the options by comparing that RCSS with the one found in new DIO messages for the option.

Protected options may be sent in full, elided, or in the abbreviated form. Which form to use depends on the RCSS of the option that a parent maintains:

* A parent MAY use either form when the RCSS is not changed from a previous DIO; eliding options is PREFERRED in stable conditions to save resources.

* When a protected option is updated, the RCSS is mechanically incremented, and the new option MUST be sent in full on the first DIO that advertises that new RCSS and the corresponding AOO SHOULD NOT be added.
When the RCSS is updated but a protected option is unchanged, the parent SHOULD NOT fully elide the option as it may cause multiple children to resynchronize it to no avail. The use of AOO is RECOMMENDED unless it may cause a desynchronization for that option, in which case the option SHOULD be placed in full more in Section 5.3.

When a child receives a DIO from a candidate parent, for each option:

If the Option is advertised in the abbreviated form then the RCSS that the DIO advertises for the option is the Last Modification RCSS of the AOO, else

If the Option is advertised in full then the RCSS that the DIO advertises for the option is the RCSS of the DIO, else

If the Option is elided then the RCSS is unspecified but it is at most as fresh as the RCSS of the DIO, and the RCSS of the DIO is assumed for the comparison.

This means that if an Option is advertised in both the abbreviated form and in full in a same DIO message then the RCSS in the AOO has precedence.

To keep the RCSS comparable for each option, the RCSS of an option must lazily progress along with the global RCSS even if there was no change in the options. Each parent including the Root MUST advertise a new RCSS for each of the protected options at least once within a sliding window of SEQUENCE_WINDOW increments.

When an option was not changed for a new RCSS, one parent may advertise it in the abbreviated form while another sends the option in full only, e.g., in response to a DIS message. A fresher RCSS indicates that the option is either the same or carries a more recent update than the one with an older RCSS.

The RCSS of an option may be obtained from a DIO message that carries the option in full even if the RCSS of the DIO is not the freshest across parents, as long as the RCSS of the DIO is fresher than the current one for that option.

If the current value of the maintained RCSS for a given option is not fresher or as fresh as that advertised in a DIO message, then the child MUST update its state for that option as specified in Section 6.
6. Synchronizing Options

As the value of the RCSS progresses, it is NOT RECOMMENDED for a child to attempt to resynchronize to an intermediate value of a RCSS with a parent that is already back level vs. the lastest known RCSS or the RCSS that this child is currently using, or a parent that is out-of-sync with self. The child MAY only do so if it lost reachability to all the candidate parents that advertise a fresher and not out-of-sync value of RCSS.

A child can resynchronize any of the protected options to the latest RCSS by sending a DIS Message to a candidate parent that advertises that RCSS in DIO messages. The child MUST set the desired combination of ‘R’, ‘D’, ‘P’, ‘M’ and ‘O’ flags to indicate the option(s) that it needs updated. The child MUST signal in the Last Synchronized RCSS field of the DIS the freshest value of RCSS for which it was fully synchronized, or a conventional value of OUT-OF-SYNC-RCSS of 129 if it was never synchronized or is out-of-sync with the parent.

The DIO message that is sent in response MUST contain in full all the options that are requested and that were updated since the Last Synchronized RCSS in the DIS Message. This means all of the protected options of the child was never synchronized or is out-of-sync with the parent. The other options MUST be added in the abbreviated form. The options MAY be spread over more than one DIO message sent in a quick sequence. It is possible that the DIS is not received by the parent or that a DIO that carries all or subset of the requested options is lost in return. In that case the child MUST resend a DIS with the bits associated to the options that are still missing after a reasonable technology-dependent time before it retries the request. The child MAY use any parent that advertises the RCSS to get any of the options up to that level.

7. Abbreviating the DAO Message

When a node receives a positive DAO-ACK upon a DAO message for a given DAOSequence, The DAO-ACK indicates that the DAO was fully processed by its destination (parent or Root). Until there’s a change in one of the DAO options since that DAOSequence, the next DAO messages merely refresh the lifetime of the routes. In that case, increasing the DAOSequence creates undesirable churn up the DODAG for no added value.

This specification enables a node to refresh the state in a destination that is associated to one or more DAO message(s) that were acknowledged by that destination without resending the DAO message(s) in full. Instead, the node MAY use a single abbreviated
DAO message that is sent to the same destination and with the same DAOSequence as the DAO message(s) that it refreshes, and with the ‘A’ flag set (see Section 4.4) to signal it is an abbreviated DAO.

This can be more than one message if the node could not package all its state in a single message, e.g., due to MTU restrictions. In that case the DAO state that is refreshed is the aggregation of the DAO messages that were acknowledged for the provided DAOSequence by that destination.

Upon the abbreviated DAO, the destination refreshes the state associated to the original DAO message(s) received with that DAOSequence, typically by extending the lifetimes of the routes that were advertised with the same duration.

A node MAY also unset ‘K’ flag in the abbreviated DAO message and not expect a DAO-ACK, if the node can assume the risk that the DAO is lost, e.g., if the routes will be refreshed again before the lifetime expires.

Only the DAO message(s) with the last (freshest) DAOSequence can be a abbreviated. A nod MUST NOT use an abbreviated DAO with a DAOSequence that is not the freshest and it MUST NOT use the abbreviated form of the DAO until the destination has acknowledged all state associated with that DAOSequence. If a destination receives an abbreviated DAO with a DAOSequence that is not the freshest from that node, or the destination does not have a state for that node, then MUST send a DAO-ACK with a DAO Status indicating an error. The destination MAY use a new Status of ‘Out-of-Sync’ in which case the node MUST resent the DAO Message(s) in full with its freshest DAOSequence and the destination resynchronizes to that level.

It is RECOMMENDED to use an abbreviated DAO messages whenever possible, because a smaller DAO message consumes less energy and bandwidth and has better chances of delivery. In Non-Storing Mode the benefits increases with the number of hops to the Root, and in Storing Mode with the amount of state that is implicitly refreshed.

8. Security Considerations

TBD

9. IANA Considerations
9.1. New DODAG Information Solicitation Flags

5 new bits are allocated in the Registry for the DODAG Information Solicitation (DIS) Flags defined for [RPL].

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Capability description</th>
<th>Defining RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>‘R’ bit &quot;RIO requested&quot;</td>
<td>THIS RFC</td>
</tr>
<tr>
<td>1</td>
<td>‘D’ bit &quot;DCO requested&quot;</td>
<td>THIS RFC</td>
</tr>
<tr>
<td>2</td>
<td>‘P’ bit &quot;PIO(s) requested&quot;</td>
<td>THIS RFC</td>
</tr>
<tr>
<td>3</td>
<td>‘M’ bit &quot;MOPex requested&quot;</td>
<td>THIS RFC</td>
</tr>
<tr>
<td>4</td>
<td>‘O’ bit &quot;GCO requested&quot;</td>
<td>THIS RFC</td>
</tr>
</tbody>
</table>

Table 1: New DIS Flags

9.2. New DODAG Advertisement Object Flag

1 new bit is allocated in the Registry for the Destination Advertisement Object (DAO) Flags defined for [RPL].

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Capability description</th>
<th>Defining RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>‘A’ bit &quot;DAO abbreviated&quot;</td>
<td>THIS RFC</td>
</tr>
</tbody>
</table>

Table 2: New DAO Flag

9.3. New RPL Control Message Option

A new entry is required for the new option of type "Abbreviated Option", from the "RPL Control Message Options" space defined for [RPL].

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Defining RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD IANA</td>
<td>Abbreviated Option</td>
<td>THIS RFC</td>
</tr>
</tbody>
</table>

Table 3: New Option Type
10. Acknowledgments

11. Normative References


12. Informative References
Authors’ Addresses

Pascal Thubert (editor)
Cisco Systems, Inc
Building D, 45 Allee des Ormes - BP1200
06254 Mougins - Sophia Antipolis
France

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

Rahul Arvind Jadhav
Huawei Tech
Kundalahalli Village, Whitefield,
Bangalore 560037
Karnataka
India

Phone: +91-080-49160700
Email: rahul.ietf@gmail.com

Li Zhao
China
200233
SHANGHAI
Xinsi Building, No. 926 Yi Shan Rd
Cisco Systems, Inc

Email: liz3@cisco.com

Dominique Barthel
Orange Labs
28 chemin du Vieux ChÂ*ne
38243 Meylan
France

Email: dominique.barthel@orange.com