ROLL
Internet-Draft
Intended status: Standards Track
Expires: November 29, 2010

T. Winter, Ed.

P. Thubert, Ed.
Cisco Systems
RPL Author Team
IETF ROLL WG

May 28, 2010

RPL: IPv6 Routing Protocol for Low power and Lossy Networks
draft-ietf-roll-rpl-08

Abstract

Low power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained: LLN routers typically operate with constraints on (any subset of) processing power, memory and energy (battery), and their interconnects are characterized by (any subset of) high loss rates, low data rates and instability. LLNs are comprised of anything from a few dozen and up to thousands of routers, and support point-to-point traffic (between devices inside the LLN), point-to-multipoint traffic (from a central control point to a subset of devices inside the LLN) and multipoint-to-point traffic (from devices inside the LLN towards a central control point). This document specifies the IPv6 Routing Protocol for LLNs (RPL), which provides a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point, as well as point-to-multipoint traffic from the central control point to the devices inside the LLN, is supported. Support for point-to-point traffic is also available.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 29, 2010.

Table of Contents

1. Introduction

Low power and Lossy Networks (LLNs) consist of largely of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery operated). These routers are interconnected by lossy links, typically supporting only low data rates, that are usually unstable with relatively low packet delivery rates. Another characteristic of such networks is that the traffic patterns are not simply point-to-point, but in many cases point-to-multipoint or multipoint-to-point. Furthermore such networks may potentially comprise up to thousands of nodes. These characteristics offer unique challenges to a routing solution: the IETF ROLL Working Group has defined application-specific routing requirements for a Low power and Lossy Network (LLN) routing protocol, specified in [I-D.ietf-roll-building-routing-reqs], [RFC5826], [RFC5673], and

[RFC5548].

This document specifies the IPv6 Routing Protocol for Low power and lossy networks (RPL). Note that although RPL was specified according to the requirements set forth in the aforementioned requirement documents, its use is in no way limited to these applications.

1.1. Design Principles

RPL was designed with the objective to meet the requirements spelled out in [I-D.ietf-roll-building-routing-reqs], [RFC5826], [RFC5673], and [RFC5548].

A network may run multiple instances of RPL concurrently. Each such instance may serve different and potentially antagonistic constraints or performance criteria. This document defines how a single instance operates.

In order to be useful in a wide range of LLN application domains, RPL separates packet processing and forwarding from the routing optimization objective. Examples of such objectives include minimizing energy, minimizing latency, or satisfying constraints. This document describes the mode of operation of RPL. Other companion documents specify routing objective functions. A RPL implementation, in support of a particular LLN application, will include the necessary objective function(s) as required by the application.

A set of companion documents to this specification will provide further guidance in the form of applicability statements specifying a set of operating points appropriate to the Building Automation, Home Automation, Industrial, and Urban application scenarios.

1.2. Expectations of Link Layer Type

In compliance with the layered architecture of IP, RPL does not rely on any particular features of a specific link layer technology. RPL is designed to be able to operate over a variety of different link layers, including but not limited to, low power wireless or PLC (Power Line Communication) technologies.

Implementers may find [RFC3819] a useful reference when designing a link layer interface between RPL and a particular link layer technology.

2. Terminology

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 RFC 2119 [RFC2119].

Additionally, this document uses terminology from [I-D.ietf-roll-terminology], and introduces the following terminology:

DAG: Directed Acyclic Graph. A directed graph having the property that all edges are oriented in such a way that no cycles exist. All edges are contained in paths oriented toward and terminating at one or more root nodes.

DAG root: A DAG root is a node within the DAG that has no outgoing edges. Because the graph is acyclic, by definition all DAGs must have at least one DAG root and all paths terminate at a DAG root.

Destination Oriented DAG (DODAG): A DAG rooted at a single destination, i.e. at a single DAG root (the DODAG root) with no outgoing edges.

DODAG root: A DODAG root is the DAG root of a DODAG.

Rank: The rank of a node in a DAG identifies the nodes position with respect to a DODAG root. The farther away a node is from a DODAG root, the higher is the rank of that node. The rank of a node may be a simple topological distance, or may more commonly be calculated as a function of other properties as described later.

DODAG parent: A parent of a node within a DODAG is one of the

immediate successors of the node on a path towards the DODAG root. The DODAG parent of a node will have a lower rank than the node itself. (See Section 3.5.2.1).

DODAG sibling: A sibling of a node within a DODAG is defined in this specification to be any neighboring node which is located at the same rank within a DODAG. Note that siblings defined in this manner do not necessarily share a common DODAG parent. (See Section 3.5.2.1).

Sub-DODAG The sub-DODAG of a node is the set of other nodes in the DODAG that might use a path towards the DODAG root that contains that node. Nodes in the sub-DODAG of a node have a greater rank than that node itself (although not all nodes of greater rank are necessarily in the sub-DODAG of that node). (See Section 3.5.2.1).

DODAGID: The identifier of a DODAG root. The DODAGID must be unique within the scope of a RPL Instance in the LLN.

DODAG Version: A specific sequence number iteration ("version") of a DODAG with a given DODAGID.

RPL Instance: A set of possibly multiple DODAGs. A network may have more than one RPL Instance, and a RPL node can participate in multiple RPL Instances. Each RPL Instance operates independently of other RPL Instances. This document describes operation within a single RPL Instance. In RPL, a node can belong to at most one DODAG per RPL Instance. The tuple (RPLInstanceID, DODAGID) uniquely identifies a DODAG.

RPLInstanceID: Unique identifier of a RPL Instance.

DODAGVersionNumber: A sequential counter that is incremented by the root to form a new Version of a DODAG. A DODAG Version is identified uniquely by the (RPLInstanceID, DODAGID, DODAGVersionNumber) tuple.

Up: Up refers to the direction from leaf nodes towards DODAG roots, following the orientation of the edges within the DODAG. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper," or "down," and vertices closer to the root are "shallower," or "up."

Down: Down refers to the direction from DODAG roots towards leaf nodes, going against the orientation of the edges within the DODAG. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper," or "down," and vertices closer to the root are "shallower," or "up."

Objective Code Point (OCP): An identifier, used to indicate which Objective Function is in use for forming a DODAG. The Objective Code Point is further described in [I-D.ietf-roll-routing-metrics].

Objective Function (OF): Defines which routing metrics, optimization objectives, and related functions are in use in a DODAG.

Goal: The Goal is a host or set of hosts that satisfy a particular application objective (OF). Whether or not a DODAG can provide connectivity to a goal is a property of the DODAG. For example, a goal might be a host serving as a data collection point, or a gateway providing connectivity to an external infrastructure.

Grounded: A DODAG is said to be grounded, when the root can reach the Goal of the objective function.

Floating: A DODAG is floating if is not Grounded. A floating DODAG is not expected to reach the Goal defined for the OF. Typically, a DAG that is only intended to provide inner connectivity is a Floating DAG.

As they form networks, LLN devices often mix the roles of 'host' and 'router' when compared to traditional IP networks. In this document, 'host' refers to an LLN device that can generate but does not forward RPL traffic, 'router' refers to an LLN device that can forward as well as generate RPL traffic, and 'node' refers to any RPL device, either a host or a router.

3. Protocol Overview

The aim of this section is to describe RPL in the spirit of [RFC4101]. Protocol details can be found in further sections.

3.1. Topology

This section describes how the basic RPL topologies, and the rules by

which these are constructed, i.e. the rules governing DODAG formation.

3.1.1. Topology Identifiers

RPL uses four identifiers to maintain the topology:

- The first is a RPLInstanceID. A RPLInstanceID identifies a set of one or more DODAGs. All DODAGs in the same RPL Instance use the same OF. A network may have multiple RPLInstanceIDs, each of which defines an independent set of DODAGs, which may be optimized for different OFs and/or applications. The set of DODAGs identified by a RPLInstanceID is called a RPL Instance.

- The second is a DODAGID. The scope of a DODAGID is a RPL Instance. The combination of RPLInstanceID and DODAGID uniquely identifies a single DODAG in the network. A RPL Instance may have multiple DODAGs, each of which has an unique DODAGID.

- The third is a DODAGVersionNumber. The scope of a DODAGVersionNumber is a DODAG. A DODAG is sometimes reconstructed from the DODAG root, by incrementing the DODAGVersionNumber. The combination of RPLInstanceID, DODAGID, and DODAGVersionNumber uniquely identifies a DODAG Version.

- The fourth is rank. The scope of rank is a DODAG Version. Rank establishes a partial order over a DODAG Version, defining individual node positions with respect to the DODAG root.

3.2. Instances, DODAGs, and DODAG Versions

Each RPL Instance constructs a routing topology optimized for a certain Objective Function (OF) and routing metrics [I-D.ietf-roll-routing-metrics]. A RPL Instance may provide routes to certain destination prefixes, reachable via the DODAG roots or alternate paths within the DODAG. A single RPL Instance contains one or more Destination Oriented DAG (DODAG) roots. These roots may operate independently, or may coordinate over a non-LLN backchannel.

Each root has a unique identifier, the DODAGID.

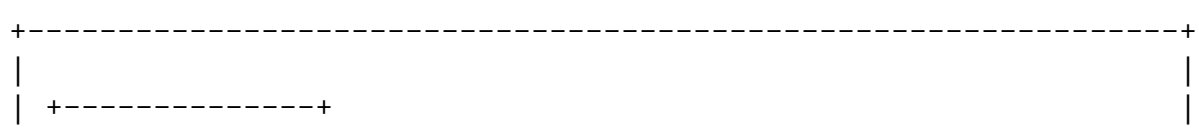
A RPL Instance may comprise:

- a single DODAG with a single root
  - For example, a DODAG optimized to minimize latency rooted at a single centralized lighting controller in a home automation application.

- multiple uncoordinated DODAGs with independent roots (differing DODAGIDs)
  - For example, multiple data collection points in an urban data collection application that do not have an always-on backbone suitable to coordinate to form a single DODAG, and further use the formation of multiple DODAGs as a means to dynamically and autonomously partition the network.
- a single DODAG with a single virtual root coordinating LLN sinks (with the same DODAGID) over some non-LLN backbone
  - For example, multiple border routers operating with a reliable backbone, e.g. in support of a 6LowPAN application, that are capable to act as logically equivalent sinks to the same DODAG.
- a combination of the above as suited to some application scenario.

Traffic is bound to a specific RPL Instance by meta-data that is carried with the packet and associates the packet to a particular RPLInstanceID (Section 8.2). The provisioning or automated discovery of a mapping between a RPLInstanceID and a type or service of application traffic is beyond the scope of this specification.

An example of a RPL Instance comprising a number of DODAGs is depicted in Figure 1. Revision of a DODAG Version (two iterations of the same DODAG) is depicted in Figure 2.



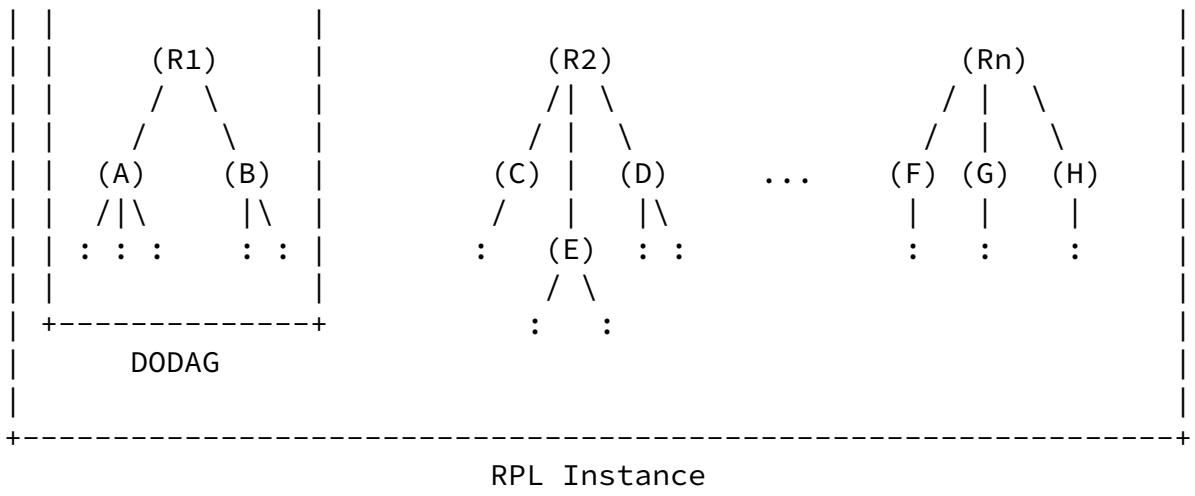


Figure 1: RPL Instance

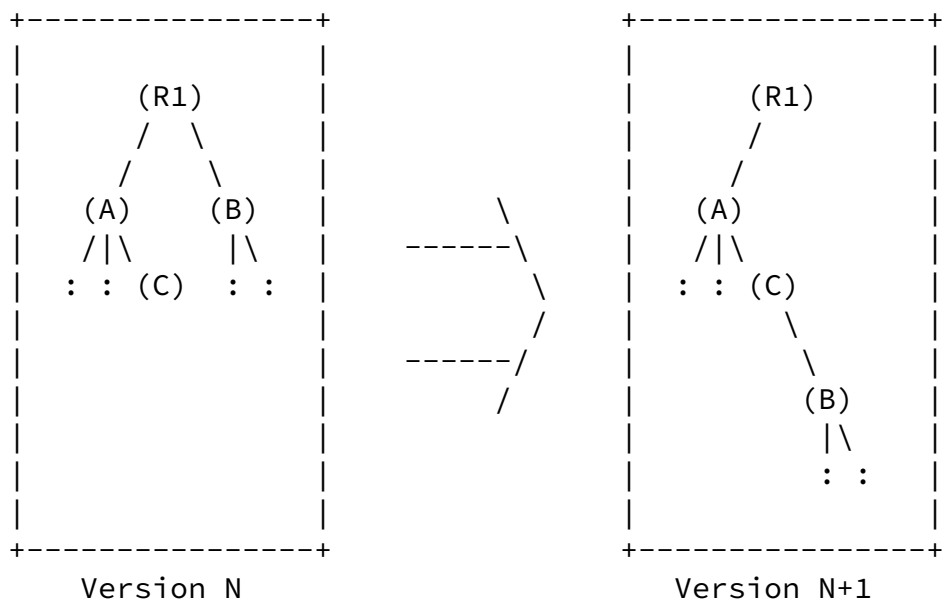


Figure 2: DODAG Version

3.3. Upward Routes and DODAG Construction

RPL provisions routes up towards DODAG roots, forming a DODAG optimized according to the Objective Function (OF) and routing metrics/constraints in use. RPL nodes construct and maintain these

DODAGs through exchange of DODAG Information Object (DIO) messages. Undirected links between siblings are also identified during this process, which can be used to provide additional diversity.

3.3.1. DAG Repair

RPL supports global repair over the DODAG. A DODAG Root may increment the DODAG Version Number, thereby initiating a new DODAG version. This institutes a global repair operation, revising the DODAG and allowing nodes to choose an arbitrary new position within the new DODAG version. Global repair can be seen as a global reoptimization mechanism.

RPL also supports mechanisms which may be used for local repair within the DODAG version. The DIO message specifies the necessary parameters as configured from the DODAG root, as controlled by policy at the root.

3.3.2. Grounded and Floating DODAGs

DODAGs can be grounded or floating. A grounded DODAG offers connectivity to reach a goal. A floating DODAG offers no such connectivity, and provides routes only to nodes within the DODAG.

Floating DODAGs may be used, for example, to preserve inner connectivity during repair.

3.3.3. Administrative Preference

An implementation/deployment may specify that some DODAG roots should be used over others through an administrative preference. Administrative preference offers a way to control traffic and engineer DODAG formation in order to better support application requirements or needs.

3.3.4. Objective Function (OF)

The Objective Function (OF) implements the optimization objectives of route selection within the RPL Instance. The OF is identified by an Objective Code Point (OCP) within the DIO. The OF also specifies the procedure used to select parents and compute rank within a DODAG version along with potentially other DODAG characteristics. Further

details may be found in Section 11, [I-D.ietf-roll-routing-metrics], [I-D.ietf-roll-of0], and related companion specifications.

3.3.5. Distributed Algorithm Operation

A high level overview of the distributed algorithm, which constructs the DODAG, is as follows:

- Some nodes are configured to be DODAG roots, with associated DODAG configuration.
- Nodes advertise their presence, affiliation with a DODAG, routing cost, and related metrics by sending link-local multicast DIO messages.
- Nodes may adjust the rate at which DIO messages are sent in response to stability or detection of routing inconsistencies from both control or data packets (see Section 8.2 for more).
- Nodes listen for DIOs and use their information to join a new DODAG, or to maintain an existing DODAG, as according to the specified Objective Function and rank-based loop avoidance rules.
- Nodes provision routing table entries, for the destinations specified by the DIO, via their DODAG parents in the DODAG version. Nodes MUST provision a DODAG parent as a default route for the associated instance. It is up to the end-to-end application to select the RPL instance to be associated to its traffic (should there be more than one instance) and thus the default route upwards when no longer-match exists.

- Nodes may identify DODAG siblings within the DODAG version to increase path diversity and decrease convergence time during repair.

3.4. Downward Routes and Destination Advertisement

RPL constructs and maintains DODAGs with DIO messages to establish upward routes: it uses Destination Advertisement Object (DAO) messages to establish downward routes along the DODAG as well as other P2P routes. DAO messages are an optional feature for applications that require P2MP or P2P traffic, in either storing

(fully stateful) or non-storing (fully source routed [I-D.hui-6man-rpl-routing-header]) mode.

3.5. Routing Metrics and Constraints Used By RPL

Routing metrics are used by routing protocols to compute shortest paths. Interior Gateway Protocols (IGPs) such as IS-IS ([RFC5120]) and OSPF ([RFC4915]) use static link metrics. Such link metrics can simply reflect the bandwidth or can also be computed according to a polynomial function of several metrics defining different link characteristics. Some routing protocols support more than one metric: in the vast majority of the cases, one metric is used per (sub)topology. Less often, a second metric may be used as a tie-breaker in the presence of Equal Cost Multiple Paths (ECMP). The optimization of multiple metrics is known as an NP complete problem and is sometimes supported by some centralized path computation engine.

In contrast, LLNs do require the support of both static and dynamic metrics. Furthermore, both link and node metrics are required. In the case of RPL, it is virtually impossible to define one metric, or even a composite metric, that will satisfy all use cases.

In addition, RPL supports constrained-based routing where constraints may be applied to both link and nodes. If a link or a node does not satisfy a required constraint, it is 'pruned' from the candidate list, thus leading to a constrained shortest path.

The set of supported link/node constraints and metrics is specified in [I-D.ietf-roll-routing-metrics].

An Objective Function specifies constraints in use, and how these are used, in addition to the objectives used to compute the (constrained) path. Upstream and Downstream metrics may be merged or advertised separately depending on the OF and the metrics. When they are advertised separately, it may happen that the set of DIO parents is different from the set of DAO parents (a DAO parent is a node to

which unicast DAO messages are sent). Yet, all are DODAG parents with regards to the rules for Rank computation.

Example 1: Shortest path: path offering the shortest end-to-end delay

Example 2: Constrained shortest path: the path that does not traverse any battery-operated node and that optimizes the path reliability

3.5.1. Loop Avoidance

RPL guarantees neither loop free path selection nor tight delay convergence times. In order to reduce control overhead, however, such as the cost of the count-to-infinity problem, RPL avoids creating loops when undergoing topology changes. Furthermore, RPL includes rank-based datapath validation mechanisms for detecting loops when they do occur. RPL uses this loop detection to ensure that packets make forward progress within the DODAG version and trigger repairs when necessary.

3.5.1.1. Greediness and Rank-based Instabilities

A node is greedy if it attempts to move deeper in the DODAG version, in order to increase the size of the parent set or improve some other metric. Moving deeper in within a DODAG version in this manner could result in instability and be detrimental to other nodes.

Once a node has joined a DODAG version, RPL disallows certain behaviors, including greediness, in order to prevent resulting instabilities in the DODAG version.

Suppose a node is willing to receive and process a DIO messages from a node in its own sub-DODAG, and in general a node deeper than itself. In this case, a possibility exists that a feedback loop is created, wherein two or more nodes continue to try and move in the DODAG version while attempting to optimize against each other. In some cases, this will result in instability. It is for this reason that RPL limits the cases where a node may process DIO messages from deeper nodes to some forms of local repair. This approach creates an 'event horizon', whereby a node cannot be influenced beyond some limit into an instability by the action of nodes that may be in its own sub-DODAG.

3.5.1.2. DODAG Loops

A DODAG loop may occur when a node detaches from the DODAG and reattaches to a device in its prior sub-DODAG. This may happen in particular when DIO messages are missed. Strict use of the DODAG

Version Number can eliminate this type of loop, but this type of loop may possibly be encountered when using some local repair mechanisms.

3.5.1.3. DAO Loops

A DAO loop may occur when the parent has a route installed upon receiving and processing a DAO message from a child, but the child has subsequently cleaned up the related DAO state. This loop happens when a No-Path (a DAO message that invalidates a previously announced prefix) was missed and persists until all state has been cleaned up. RPL includes an optional mechanism to acknowledge DAO messages, which may mitigate the impact of a single DAO message being missed. RPL includes loop detection mechanisms that may mitigate the impact of DAO loops and trigger their repair.

In the case where stateless DAO operation is used, i.e. source routing specifies the down routes, then DAO Loops should not occur on the stateless portions of the path.

3.5.1.4. Sibling Loops

Sibling loops could occur if a group of siblings kept choosing amongst themselves as successors such that a packet does not make forward progress. This specification limits the number of times that sibling forwarding may be used at a given rank, in order to prevent sibling loops.

3.5.2. Rank Properties

The rank of a node is a scalar representation of the location of that node within a DODAG version. The rank is used to avoid and detect loops, and as such must demonstrate certain properties. The exact calculation of the rank is left to the Objective Function, and may depend on parents, link metrics, and the node configuration and policies.

The rank is not a cost metric, although its value can be derived from and influenced by metrics. The rank has properties of its own that are not necessarily those of all metrics:

Type: The rank is an abstract decimal value.

Function: The rank is the expression of a relative position within a DODAG version with regard to neighbors and is not necessarily a good indication or a proper expression of a distance or a cost to the root.

Stability: The stability of the rank determines the stability of the routing topology. Some dampening or filtering might be applied to keep the topology stable, and thus the rank does not necessarily change as fast as some physical metrics would. A new DODAG version would be a good opportunity to reconcile the discrepancies that might form over time between metrics and ranks within a DODAG version.

Granularity: The portion of the rank that is used to define a node's position in the DAG, DAGRank(node), is coarse grained. A fine granularity would make the selection of siblings difficult, since siblings must have the exact same rank value.

Properties: The rank is strictly monotonic, and can be used to validate a progression from or towards the root. A metric, like bandwidth or jitter, does not necessarily exhibit this property.

Abstract: The rank does not have a physical unit, but rather a range of increment per hop, where the assignment of each increment is to be determined by the Objective Function.

The rank value feeds into DODAG parent selection, according to the RPL loop-avoidance strategy. Once a parent has been added, and a rank value for the node within the DODAG has been advertised, the nodes further options with regard to DODAG parent selection and movement within the DODAG are restricted in favor of loop avoidance.

3.5.2.1. Rank Comparison (DAGRank())

Rank may be thought of as a fixed point number, where the position of the decimal point between the integer part and the fractional part is determined by MinHopRankIncrease. MinHopRankIncrease is the minimum increase in rank between a node and any of its DODAG parents. When an objective function computes rank, the objective function operates on the entire (i.e. 16-bit) rank quantity. When rank is compared, e.g. for determination of parent/sibling relationships or loop detection, the integer portion of the rank is to be used. The integer portion of the Rank is computed by the DAGRank() macro as

follows:

DAGRank(rank) = floor(rank/MinHopRankIncrease)

MinHopRankIncrease is provisioned at the DODAG Root and propagated in the DIO message. For efficient implementation the MinHopRankIncrease

MUST be a power of 2. An implementation may configure a value MinHopRankIncrease as appropriate to balance between the loop avoidance logic of RPL (i.e. selection of eligible parents and siblings) and the metrics in use.

By convention in this document, using the macro DAGRank(node) may be interpreted as DAGRank(node.rank), where node.rank is the rank value as maintained by the node.

A node A has a rank less than the rank of a node B if DAGRank(A) is less than DAGRank(B).

A node A has a rank equal to the rank of a node B if DAGRank(A) is equal to DAGRank(B).

A node A has a rank greater than the rank of a node B if DAGRank(A) is greater than DAGRank(B).

3.5.2.2. Rank Relationships

The computation of the rank MUST be done in such a way so as to maintain the following properties for any nodes M and N that are neighbors in the LLN:

DAGRank(M) is less than DAGRank(N): In this case, the position of M is closer to the DODAG root than the position of N. Node M may safely be a DODAG parent for Node N without risk of creating a loop. Further, for a node N, all parents in the DODAG parent set must be of rank less than DAGRank(N). In other words, the rank presented by a node N MUST be greater than that presented by any of its parents.

DAGRank(M) equals DAGRank(N): In this case the positions of M and N

within the DODAG and with respect to the DODAG root are similar (identical). In some cases, Node M may be used as a successor by Node N, which however entails the chance of creating a loop (which must be detected and resolved by some other means).

DAGRank(M) is greater than DAGRank(N): In this case, the position of M is farther from the DODAG root than the position of N. Further, Node M may in fact be in the sub-DODAG of Node N. If node N selects node M as DODAG parent there is a risk to create a loop.

As an example, the rank could be computed in such a way so as to closely track ETX (Expected Transmission Count, a fairly common routing metric used in LLN and defined in

[I-D.ietf-roll-routing-metrics]) when the objective function is to minimize ETX, or latency when the objective function is to minimize latency, or in a more complicated way as appropriate to the objective function being used within the DODAG.

3.6. Traffic Flows Supported by RPL

3.6.1. Multipoint-to-Point Traffic

Multipoint-to-Point (MP2P) is a dominant traffic flow in many LLN applications ([I-D.ietf-roll-building-routing-reqs], [RFC5826], [RFC5673], [RFC5548]). The destinations of MP2P flows are designated nodes that have some application significance, such as providing connectivity to the larger Internet or core private IP network. RPL supports MP2P traffic by allowing MP2P destinations to be reached via DODAG roots.

3.6.2. Point-to-Multipoint Traffic

Point-to-multipoint (P2MP) is a traffic pattern required by several LLN applications ([I-D.ietf-roll-building-routing-reqs], [RFC5826], [RFC5673], [RFC5548]). RPL supports P2MP traffic by using a destination advertisement mechanism that provisions routes toward destination prefixes and away from roots. Destination advertisements can update routing tables as the underlying DODAG topology changes.

3.6.3. Point-to-Point Traffic

RPL DODAGs provide a basic structure for point-to-point (P2P) traffic. For a RPL network to support P2P traffic, a root must be able to route packets to a destination. Nodes within the network may also have routing tables to destinations. A packet flows towards a root until it reaches an ancestor that has a known route to the destination. As pointed out later in this document, in the most constrained case (when nodes cannot store routes), that common ancestor may be the DODAG root. In other cases it may be a node closer to both the source and destination.

RPL also supports the case where a P2P destination is a 'one-hop' neighbor.

RPL neither specifies nor precludes additional mechanisms for computing and installing potentially more optimal routes to support arbitrary P2P traffic.

4. RPL Instance

Within a given LLN, there may be multiple, logically independent RPL instances. This document describes how a single instance behaves.

A node may belong to multiple RPL Instances.

An instance can be either local to a root or global. When the instance is local, the DAG is a single DODAG that is rooted at the node that owns the DODAGID. This is used in particular for the construction of a temporary DODAG in support of P2P traffic optimization between the root and some other nodes.

Control and Data Packets that traverse a RPL network MUST be tagged in such a fashion that the instance is unambiguously identified (TBD flow label or RPL Hop-by-hop option ([I-D.hui-6man-rpl-option])). The identifiers include the RPLInstanceID and the DODAGID for local instances.

4.1. RPL Instance ID

A global RPLInstanceID MUST be unique to the whole LLN. Mechanisms for allocating and provisioning global RPLInstanceID are out of scope for this document. There can be up to 128 global instance in the whole network, and up 64 local instances per DODAGID.

A global RPLinstanceID is encoded in a RPLinstanceID field as follows:

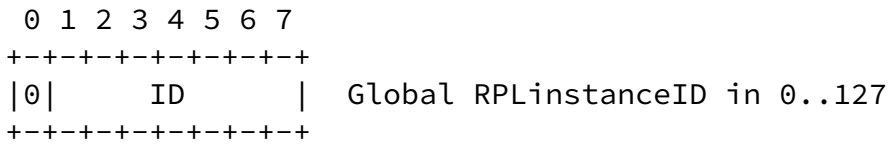


Figure 3: RPL Instance ID field format for global instances

A local RPLInstanceID is autoconfigured by the node that owns the DODAGID and it MUST be unique for that DODAGID. In that case, the DODAGID MUST be a valid address of the root that is used as an endpoint of all communications within that instance.

A local RPLinstanceID is encoded in a RPLinstanceID field as follows:

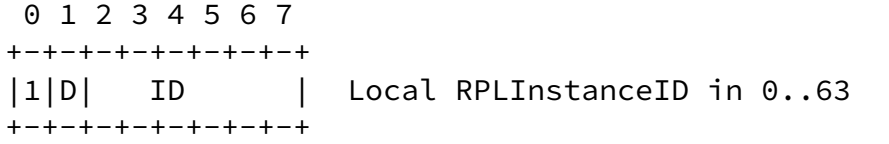


Figure 4: RPL Instance ID field format for local instances

The D flag in a Local RPLInstanceID is always set to 0 in RPL control messages. It is used in data packets to indicate whether the DODAGID is the source or the destination of the packet. If the D flag is set to 1 then the destination address of the IPv6 packet MUST be the DODAGID. If the D flag is clear then the source address of the IPv6

packet MUST be the DODAGID.

5. ICMPv6 RPL Control Message

This document defines the RPL Control Message, a new ICMPv6 message. A RPL Control Message is identified by a code, and composed of a base that depends on the code, and a series of options.

A RPL Control Message has the scope of a link. The source address is a link local address. The destination address is either all routers multicast address (FF02::2) or a link local address.

In accordance with [RFC4443], the RPL Control Message consists of an ICMPv6 header followed by a message body. The message body is comprised of a message base and possibly a number of options as illustrated in Figure 5.

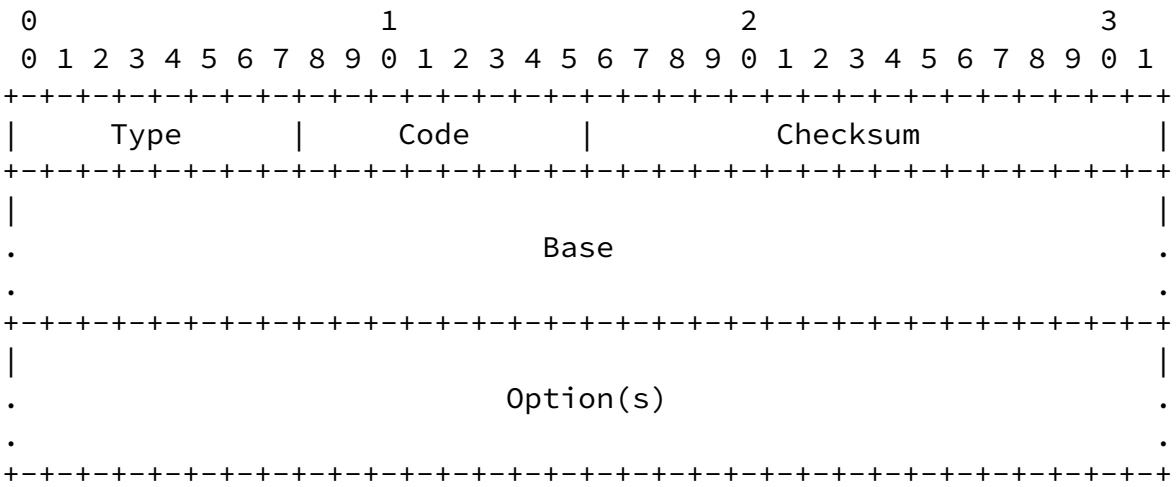


Figure 5: RPL Control Message

The RPL Control message is an ICMPv6 information message with a

requested Type of 155 (to be confirmed by IANA).

The Code field identifies the type of RPL Control Message. This document defines codes for the following RPL Control Message types (all codes are to be confirmed by the IANA Section 15.2):

- 0x00: DODAG Information Solicitation (Section 5.2)
- 0x01: DODAG Information Object (Section 5.3)
- 0x02: Destination Advertisement Object (Section 5.4)
- 0x03: Destination Advertisement Object Acknowledgment (Section 5.5)
- 0x80: Secure DODAG Information Solicitation (Section 5.2.2)
- 0x81: Secure DODAG Information Object (Section 5.3.2)
- 0x82: Secure Destination Advertisement Object (Section 5.4.2)
- 0x83: Secure Destination Advertisement Object Acknowledgment (Section 5.5.2)

The high order bit (0x80) of the code denotes whether the RPL message has security enabled. Secure versions of RPL messages have a modified format to support confidentiality and integrity, illustrated in Figure Figure 6.

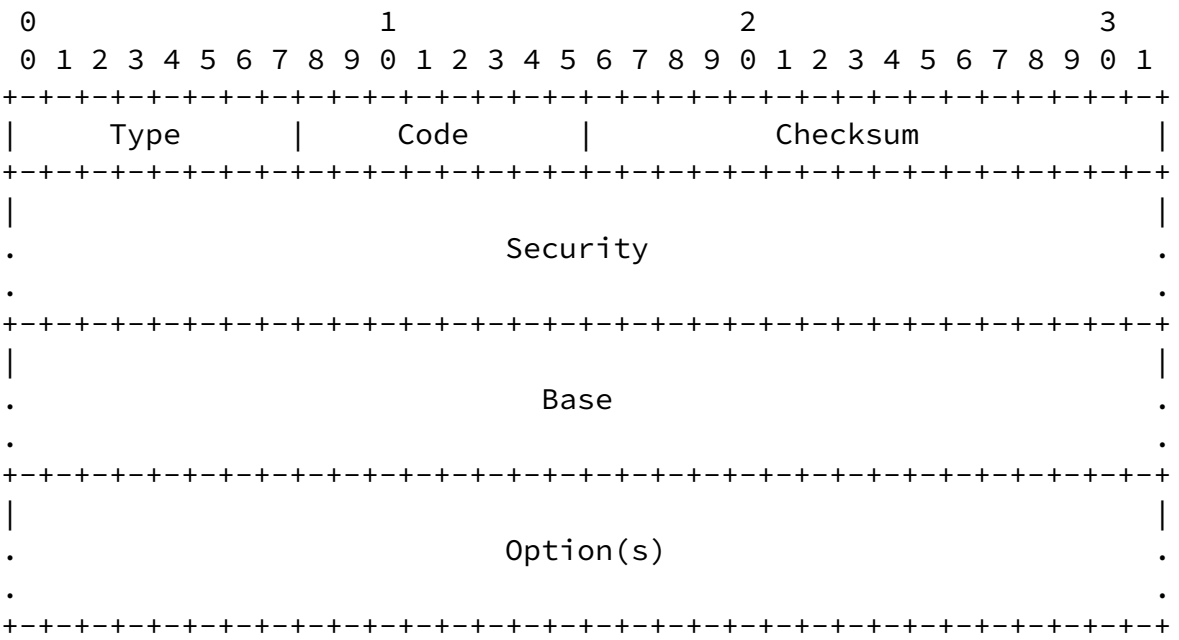


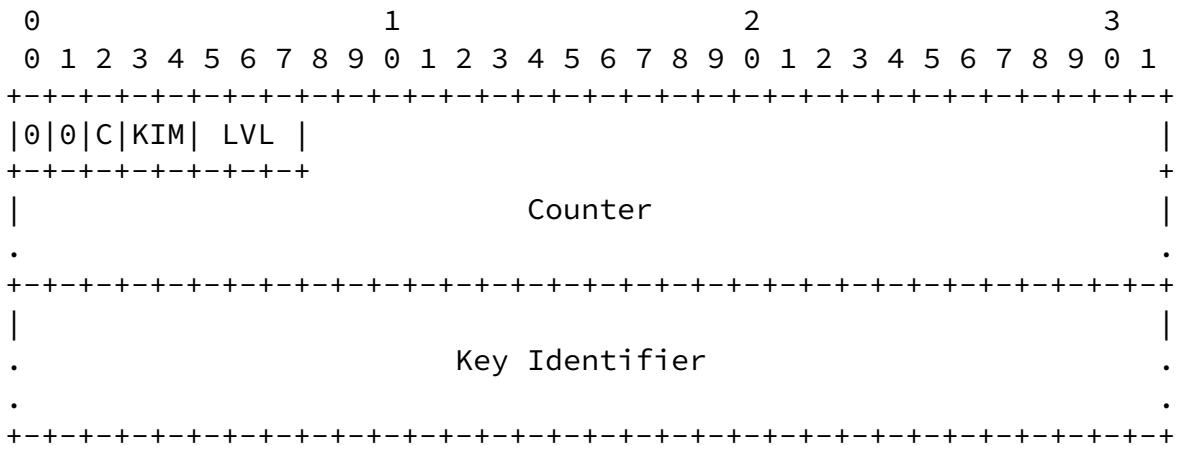
Figure 6: Secure RPL Control Message

The remainder of this section describes the currently defined RPL Control Message Base formats followed by the currently defined RPL Control Message Options.

5.1. RPL Security Fields

Each RPL message has a secure version. The secure versions provide integrity and confidentiality. Because security covers the base message as well as options, in secured messages the security information lies between the checksum and base, as shown in Figure Figure 6.

The format of the security section is as follows:



Security

All fields are considered as packet payload from a security processing perspective. The exact placement and format of message integrity/authentication codes has not yet been determined.

Use of the Security section is further detailed in Section 14.

Security Control Field: The Security Control Field has one flag and two fields:

Counter Compression (C): If the Counter Compression flag is set then the Counter field is compressed from 4 bytes into 1 byte. If the Counter Compression flag is clear then the Counter field is 4 bytes and uncompressed.

Key Identifier Mode (KIM): The Key Identifier Mode field indicates whether the key used for packet protection is determined implicitly or explicitly and indicates the particular representation of the Key Identifier field. The Key Identifier Mode is set one of the non-reserved values from the table below:

Mode	KIM	Meaning	Key Identifier Length (octets)
0	00	Peer-to-peer key determined implicitly from originator and recipient of packet. Key Source is not present. Key Index is not present.	0
1	01	Group key determined implicitly from Key Index and side information. Key Source is not present. Key Index is present.	1
2	10	Signature key used; group key determined explicitly if encryption used. Key Source may be present. Key Index may be present.	0/9
3	11	Group key determined explicitly from Key Source Identifier and Key Index. Key Source is present. Key Index is present.	9

Key Identifier Mode (KIM) Encoding

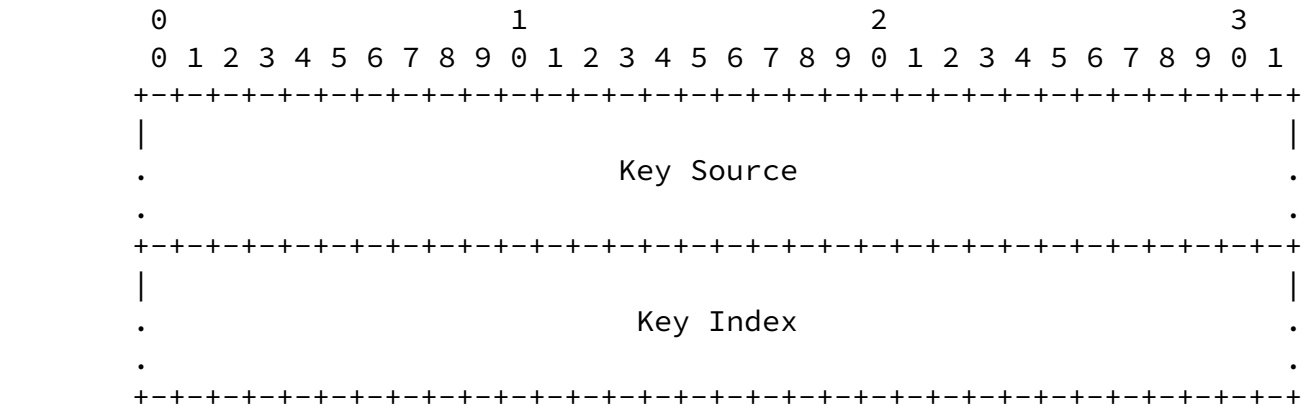
Security Level (LVL): The Security Level field indicates the provided packet protection. This value can be adapted on a per-packet basis and allows for varying levels of data authenticity and, optionally, for data confidentiality. When nontrivial protection is provided, replay protection is always provided. The Security Level is set to one of the non-reserved values in the table below:

		Without Signatures		With Signatures	
ID	LVL	Attributes	Auth Len	Attributes	Sig Len
0	000	None	0	None	37
1	001	MIC-32	4	Sign-32	37
2	010	MIC-64	8	Sign-64	45
3	011	Rsvd	N/A	Rsvd	N/A
4	100	ENC	0	ENC	37
5	101	ENC-MIC-32	4	ENC-Sign-32	41
6	110	ENC-MIC-64	8	ENC-Sign-64	45
7	111	Rsvd	N/A	Reserved	N/A

Security Level (LVL) Encoding

Counter: The Counter field indicates the non-repeating value (nonce) used with the cryptographic mechanism that implements packet protection and allows for the provision of semantic security. This value is compressed from 4 octets to 1 octet if the Counter Compression field of the Security Control Field is set to one.

Key Identifier: The Key Identifier field indicates which key was used to protect the packet. This field provides various levels of granularity of packet protection, including peer-to-peer keys, group keys, and signature keys. This field is represented as indicated by the Key Identifier Mode field and is formatted as follows:



Key Identifier

Key Source: The Key Source field, when present, indicates the logical identifier of the originator of a group key. When present this field is 8 bytes in length.

Key Index: The Key Index field, when present, allows unique identification of different keys with the same originator. It is the responsibility of each key originator to make sure that actively used keys that it issues have distinct key indices and that all key indices have a value unequal to 0x00. When present this field is 1 byte in length.

Unassigned bits of the Security section are reserved. They MUST be

set to zero on transmission and MUST be ignored on reception.

5.2. DODAG Information Solicitation (DIS)

The DODAG Information Solicitation (DIS) message may be used to solicit a DODAG Information Object from a RPL node. Its use is analogous to that of a Router Solicitation as specified in IPv6 Neighbor Discovery; a node may use DIS to probe its neighborhood for nearby DODAGs. Section 6.3 describes how nodes respond to a DIS.

5.2.1. Format of the DIS Base Object

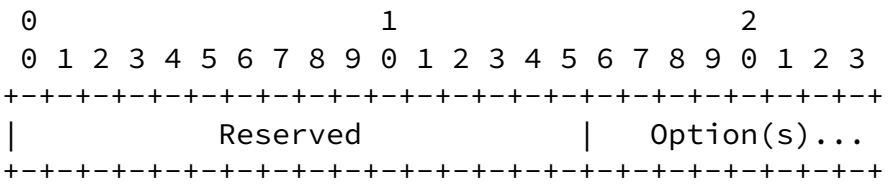


Figure 7: The DIS Base Object

Unassigned bits of the DIS Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.2.2. Secure DIS

A Secure DIS message follows the format in Figure Figure 6, where the base format is the DIS message shown in Figure Figure 7.

5.2.3. DIS Options

The DIS message MAY carry valid options.

This specification allows for the DIS message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x07 Solicited Information

5.3. DODAG Information Object (DIO)

The DODAG Information Object carries information that allows a node to discover a RPL Instance, learn its configuration parameters, select a DODAG parent set, and maintain the upward routing topology.

5.3.1. Format of the DIO Base Object

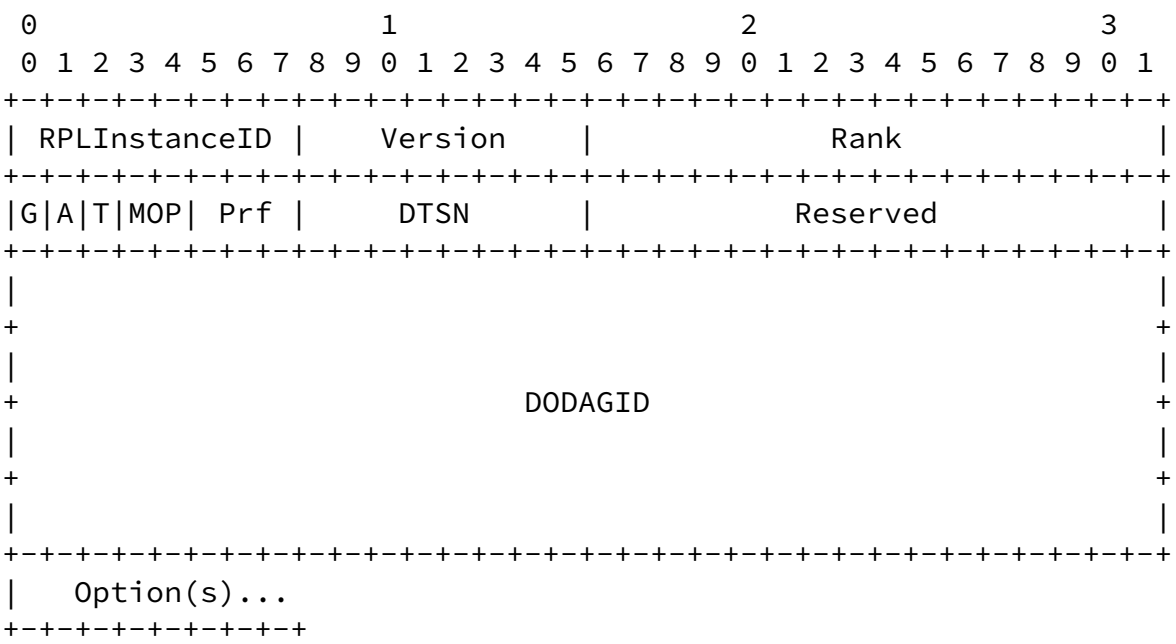


Figure 8: The DIO Base Object

Control Field: The DAG Control Field has three flags and two fields:

Grounded (G): The Grounded (G) flag indicates whether the upward routes this node advertises provide connectivity to the set of addresses which are application-defined goals. If the flag is set, the DODAG is grounded and provides such connectivity. If the flag is cleared, the DODAG is floating and may not provide such connectivity.

Destination Advertisement Supported (A): The Destination Advertisement Supported (A) flag indicates whether the root of this DODAG can collect and use downward route state. If the flag is set, nodes in the network are enabled to exchange destination advertisements messages

to build downward routes (Section 7). If the flag is cleared, destination advertisement messages are disabled and the DODAG maintains only upward routes.

Destination Advertisement Trigger (T): The Destination Advertisement Trigger (T) flag indicates a complete refresh of downward routes. If the flag is set, then a refresh of downward route state is to take place over the entire DODAG. If the flag is cleared, the downward route maintenance is in its normal mode of operation. The further details of this process are described in Section 7.

Mode of Operation (MOP): The Mode of Operation (MOP) field identifies the mode of operation of the RPL Instance as administratively provisioned at and distributed by the DODAG Root. All nodes who join the DODAG must be able to honor the MOP in order to fully participate as a router, or else they must only join as a leaf. MOP is encoded as in the table below:

MOP	Meaning
00	Non-storing
01	Storing
10	Reserved for future specification of mixed-mode
11	Reserved

Mode of Operation (MOP) Encoding

DODAGPreference (Prf): A 3-bit unsigned integer that defines how preferable the root of this DODAG is compared to other DODAG roots within the instance. DAGPreference ranges from 0x00 (least preferred) to 0x07 (most preferred). The default is 0 (least preferred). Section 6.2 describes how DAGPreference affects DIO processing.

Version Number: 8-bit unsigned integer set by the DODAG root. Section 6.2 describes the rules for version numbers and how they affect DIO processing.

Rank: 16-bit unsigned integer indicating the DODAG rank of the node sending the DIO message. Section 6.2 describes how Rank is set and how it affects DIO processing.

RPLInstanceID: 8-bit field set by the DODAG root that indicates which RPL Instance the DODAG is part of.

Destination Advertisement Trigger Sequence Number (DTSN): 8-bit unsigned integer set by the node issuing the DIO message. The Destination Advertisement Trigger Sequence Number (DTSN) flag is used as part of the procedure to maintain downward routes. The details of this process are described in Section 7.

DODAGID: 128-bit unsigned integer set by a DODAG root which uniquely identifies a DODAG. Possibly derived from the IPv6 address of the DODAG root.

Unassigned bits of the DIO Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.3.2. Secure DIO

A Secure DIO message follows the format in Figure Figure 6, where the base format is the DIS message shown in Figure Figure 8.

5.3.3. DIO Options

The DIO message MAY carry valid options.

This specification allows for the DIO message to carry the following options:

0x00 Pad1
0x01 PadN
0x02 Metric Container

0x03 Routing Information
0x04 DODAG Configuration
0x09 Prefix Information

5.4. Destination Advertisement Object (DAO)

The Destination Advertisement Object (DAO) is used to propagate destination information upwards along the DODAG. The DAO message is unicast by the child to the selected parent(s). The DAO message may optionally, upon explicit request or error, be acknowledged by the parent with a Destination Advertisement Acknowledgement (DAO-ACK) message back to the child.

5.4.1. Format of the DAO Base Object

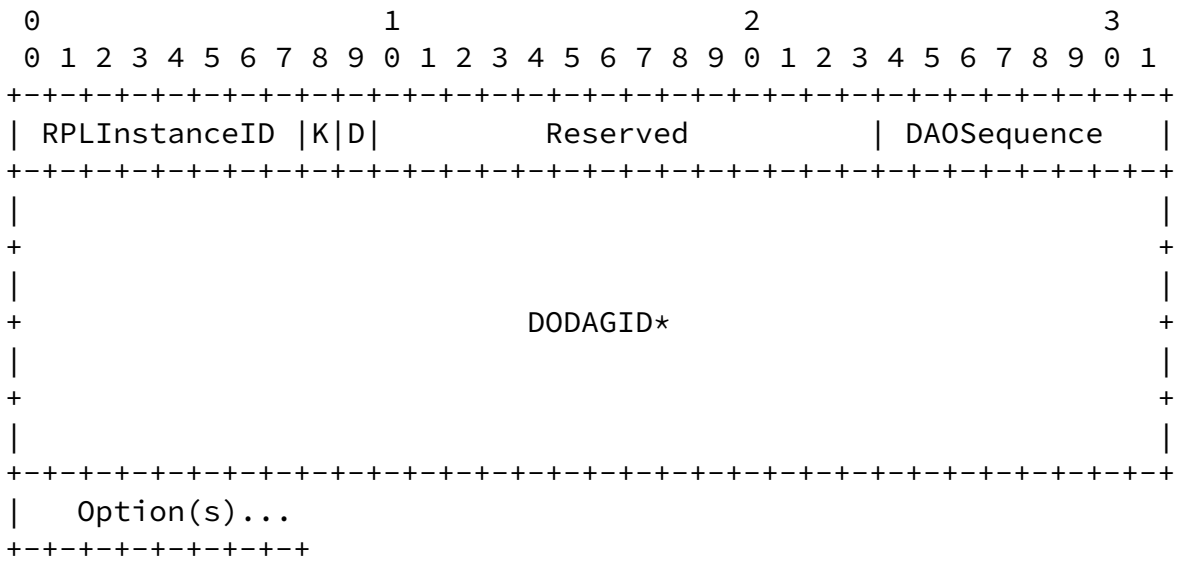


Figure 9: The DAO Base Object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

K: The 'K' flag indicates that the parent is expected to send a DAO-ACK back.

D: The 'D' flag indicates that the DODAGID field is present. This would typically only be set when a local RPLInstanceID is used.

DAOSequence: Incremented at each unique DAO message, echoed in the DAO-ACK message.

DODAGID*: 128-bit unsigned integer set by a DODAG root which uniquely identifies a DODAG. This field is only present when the 'D' flag is set. This field is typically only present when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID. When a global RPLInstanceID is in use this field need not be present.

Unassigned bits of the DAO Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.4.2. Secure DAO

A Secure DAO message follows the format in Figure Figure 6, where the base format is the DAO message shown in Figure Figure 9.

5.4.3. DAO Options

The DAO message MAY carry valid options.

This specification allows for the DAO message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x06 Transit Information

A special case of the DAO message, termed a No-Path, is used to clear downward routing state that has been provisioned through DAO operation. The No-Path carries a RPL Transit Information option, which identifies the destination to which the DAO is associated, with a lifetime of 0x00000000 to indicate a loss of reachability.

5.5. Destination Advertisement Object Acknowledgement (DAO-ACK)

The DAO-ACK message is sent as a unicast packet by a DAO parent in response to a unicast DAO message from a child.

5.5.1. Format of the DAO-ACK Base Object

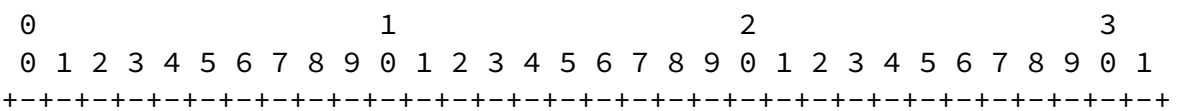


Figure 10: The DAO ACK Base Object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

DAOSequence: Incremented at each DAO message from a given child, echoed in the DAO-ACK by the parent. The DAOSequence serves in the parent-child communication and is not to be confused with the Transit Information option Sequence that is associated to a given target down the DODAG.

Status: Indicates the completion. 0 is unqualified acceptance, above 128 are rejection code indicating that the node should select an alternate parent.

Unassigned bits of the DAO-ACK Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.5.2. Secure DAO-ACK

A Secure DAO-ACK message follows the format in Figure Figure 6, where the base format is the DAO-ACK message shown in Figure Figure 10.

5.5.3. DAO-ACK Options

This specification does not define any options to be carried by the DAO-ACK message.

5.6. RPL Control Message Options

5.6.1. RPL Control Message Option Generic Format

RPL Control Message Options all follow this format:

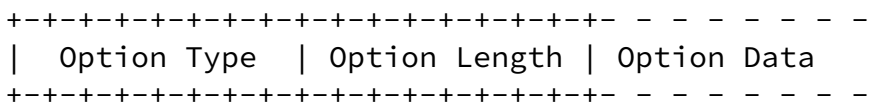


Figure 11: RPL Option Generic Format

Option Type: 8-bit identifier of the type of option. The Option Type values are to be confirmed by the IANA Section 15.4.

Option Length: 8-bit unsigned integer, representing the length in octets of the option, not including the Option Type and Length fields.

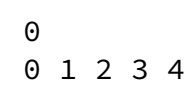
Option Data: A variable length field that contains data specific to the option.

When processing a RPL message containing an option for which the Option Type value is not recognized by the receiver, the receiver MUST silently ignore the unrecognized option and continue to process the following option, correctly handling any remaining options in the message.

RPL message options may have alignment requirements. Following the convention in IPv6, options with alignment requirements are aligned in a packet such that multi-octet values within the Option Data field of each option fall on natural boundaries (i.e., fields of width n octets are placed at an integer multiple of n octets from the start of the header, for n = 1, 2, 4, or 8).

5.6.2. Pad1

The Pad1 option may be present in DIS, DIO, DAO, and DAO-ACK messages, and its format is as follows:



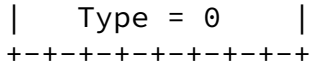


Figure 12: Format of the Pad 1 Option

The Pad1 option is used to insert one or two octets of padding into the message to enable options alignment. If more than one octet of padding is required, the PadN option should be used rather than multiple Pad1 options.

NOTE! the format of the Pad1 option is a special case - it has neither Option Length nor Option Data fields.

5.6.3. PadN

The PadN option may be present in DIS, DIO, DAO, and DAO-ACK messages, and its format is as follows:

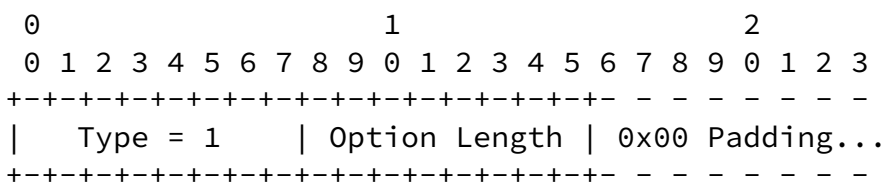


Figure 13: Format of the Pad N Option

The PadN option is used to insert two or more octets of padding into the message to enable options alignment. PadN Option data MUST be ignored by the receiver.

Option Type: 0x01 (to be confirmed by IANA)

Option Length: For N (N > 1) octets of padding, the Option Length field contains the value N-2.

Option Data: For N (N > 1) octets of padding, the Option Data consists of N-2 zero-valued octets.

5.6.4. Metric Container

The Metric Container option may be present in DIO messages, and its format is as follows:

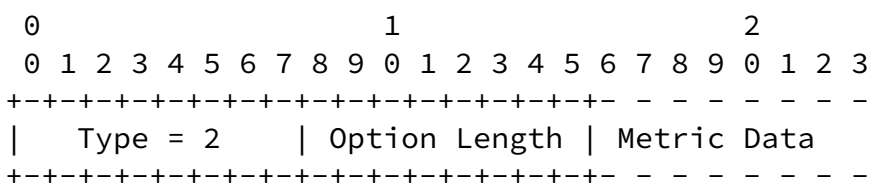


Figure 14: Format of the Metric Container Option

The Metric Container is used to report metrics along the DODAG. The Metric Container may contain a number of discrete node, link, and aggregate path metrics and constraints specified in [I-D.ietf-roll-routing-metrics] as chosen by the implementer.

The processing and propagation of the Metric Container is governed by implementation specific policy functions.

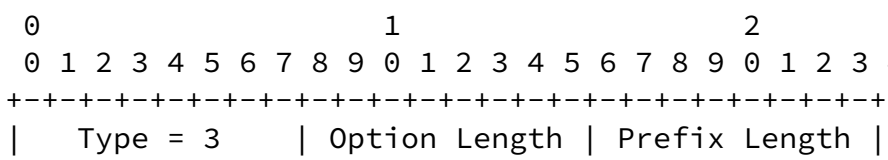
Option Type: 0x02 (to be confirmed by IANA)

Option Length: The Option Length field contains the length in octets of the Metric Data.

Metric Data: The order, content, and coding of the Metric Container data is as specified in [I-D.ietf-roll-routing-metrics].

5.6.5. Route Information

The Route Information option may be present in DIO messages, and is equivalent in function to the IPv6 ND Route Information option as defined in [RFC4191]. The format of the option is modified slightly (Type, Length) in order to be carried as a RPL option as follows:



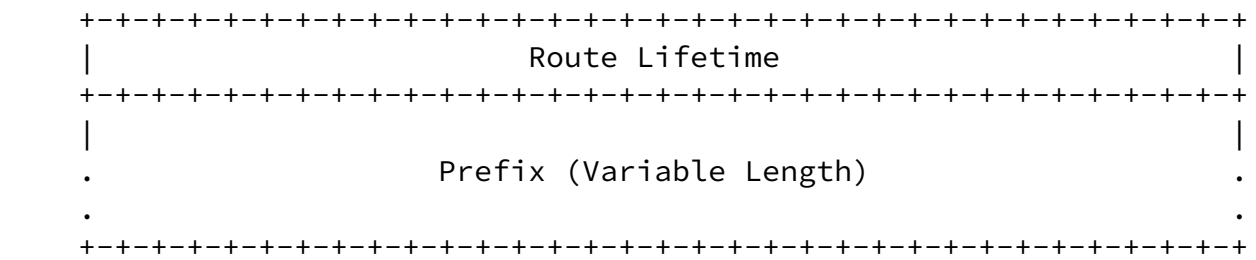


Figure 15: Format of the Route Information Option

The Route Information option is used to indicate that connectivity to the specified destination prefix is available from the DODAG root.

In the event that a RPL Control Message may need to specify connectivity to more than one destination, the Route Information option may be repeated.

[RFC4191] should be consulted as the authoritative reference with respect to the Route Information option. The field descriptions are transcribed here for convenience:

Option Type: 0x03 (to be confirmed by IANA)

Option Length: Variable, length of the option in octets excluding the Type and Length fields. Note that this length is expressed in units of single-octets, unlike in IPv6 ND.

Prefix Length 8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128. The Prefix field is 0, 8, or 16 octets depending on Length.

Prf: 2-bit signed integer. The Route Preference indicates whether to prefer the router associated with this prefix over others, when multiple identical prefixes (for different routers) have been received. If the Reserved (10) value is received, the Route Information Option MUST be ignored.

Resvd: Two 3-bit unused fields. They MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Route Lifetime 32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for route determination. A value of all one bits (0xffffffff) represents infinity.

Prefix Variable-length field containing an IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be initialized to zero by the sender and ignored by the receiver.

Unassigned bits of the Route Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.6.6. DODAG Configuration

The DODAG Configuration option may be present in DIO messages, and its format is as follows:

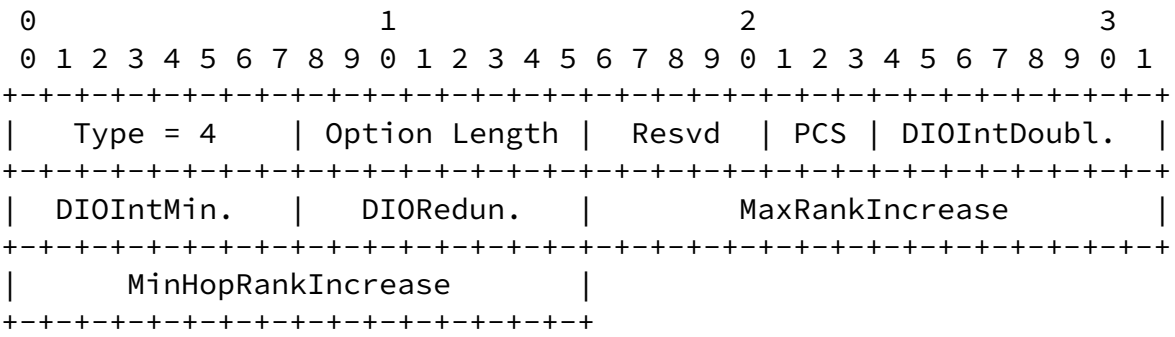


Figure 16: Format of the DODAG Configuration Option

The DODAG Configuration option is used to distribute configuration information for DODAG Operation through the DODAG.

The information communicated in this option is generally static and unchanging within the DODAG, therefore it is not necessary to include in every DIO. This information is configured at the DODAG Root and distributed throughout the DODAG with the DODAG Configuration Option.

Nodes other than the DODAG Root MUST NOT modify this information when

propagating the DODAG Configuration option. This option MAY be included occasionally by the DODAG Root (as determined by the DODAG Root), and MUST be included in response to a unicast request, e.g. a unicast DODAG Information Solicitation (DIS) message.

Option Type: 0x04 (to be confirmed by IANA)

Option Length: 8 bytes

Path Control Size (PCS): 3-bit unsigned integer used to configure the number of bits that may be allocated to the Path Control field (see Section 7.1.4.2).

DIOIntervalDoublings: 8-bit unsigned integer used to configure Imax of the DIO trickle timer (see Section 6.3.1).

DIOIntervalMin: 8-bit unsigned integer used to configure Imin of the DIO trickle timer (see Section 6.3.1).

DIORedundancyConstant: 8-bit unsigned integer used to configure k of the DIO trickle timer (see Section 6.3.1).

MaxRankIncrease: 16-bit unsigned integer used to configure DAGMaxRankIncrease, the allowable increase in rank in support of local repair. If DAGMaxRankIncrease is 0 then this mechanism is disabled.

MinHopRankInc 16-bit unsigned integer used to configure MinHopRankIncrease as described in Section 3.5.2.1.

5.6.7. RPL Target

The RPL Target option may be present in DAO messages, and its format is as follows:

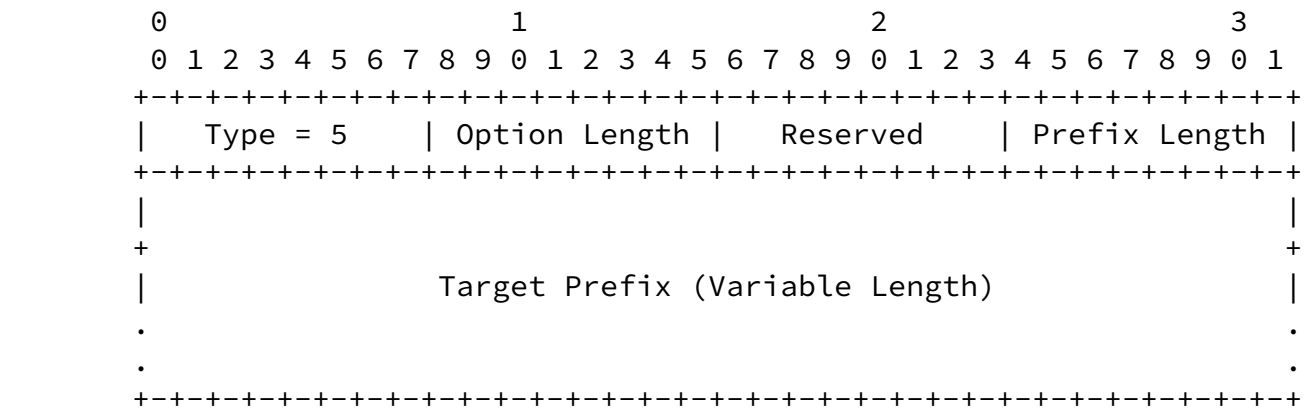


Figure 17: Format of the RPL Target Option

The RPL Target Option is used to indicate a target IPv6 address, prefix, or multicast group that is reachable or queried along the DODAG. It is used in DIO to identify a resource that the root is trying to reach, and in a DAO to indicate reachability. It is used in a DAO message to indicate reachability. A set of one or more Transit Information options MAY directly follow the Target option in a DAO message in support of constructing source routes in a non-storing mode of operation [I-D.hui-6man-rpl-routing-header]. When the same set of Transit Information options apply equally to a set of DODAG Target options, the group of Target options MUST appear first, followed by the Transit Information options which apply to those Targets.

The RPL Target option may be repeated as necessary to indicate multiple targets.

Option Type: 0x05 (to be confirmed by IANA)

Option Length: Variable, length of the option in octets excluding the Type and Length fields.

Prefix Length: 8-bit unsigned integer. Number of valid leading bits in the IPv6 Prefix.

Target Prefix: Variable-length field identifying an IPv6 destination address, prefix, or multicast group. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be set to zero on transmission and MUST be ignored on receipt.

5.6.8. Transit Information

The Transit Information option may be present in DAO messages, and its format is as follows:

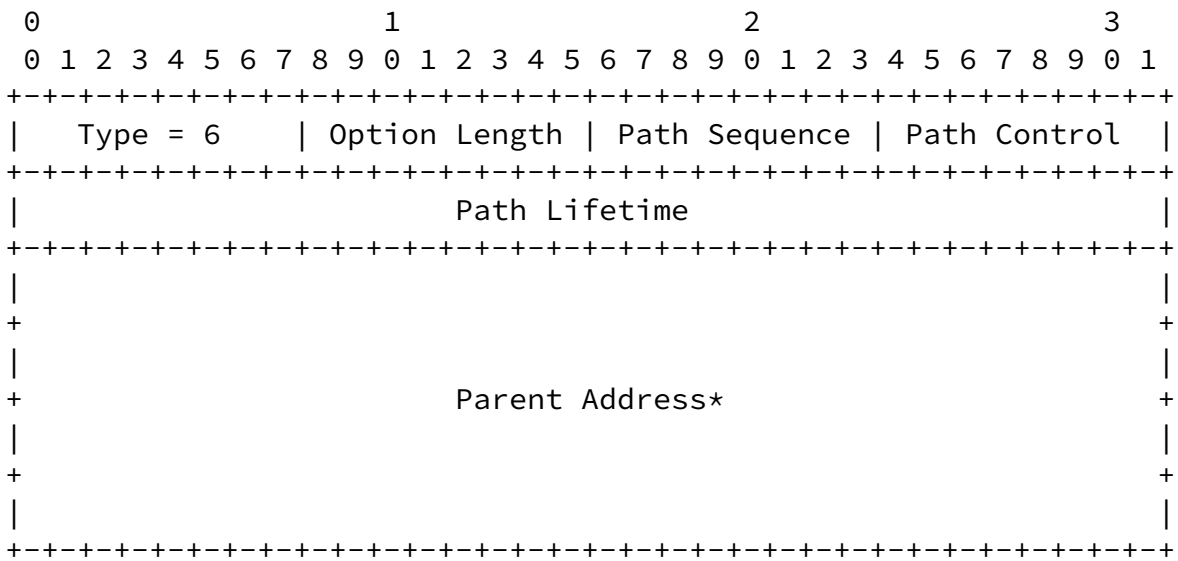


Figure 18: Format of the Transit Information option

The Transit Information option is used for a node to indicate attributes for a path to one or more destinations. The destinations are indicated as by one or more Target options that immediately precede the Transit Information option(s).

The Transit Information option can used for a node to indicate its DODAG parents to an ancestor that is collecting DODAG routing information, typically for the purpose of constructing source routes. In the non-storing mode of operation this ancestor will be the DODAG Root, and this option is carried by the DAO message. The option length is used to determine whether the Parent Address is present or not.

A non-storing node that has more than one DAO parent MAY include a

Transit Information option for each DAO parent as part of the non-storing Destination Advertisement operation. The node may code the Path Control field in order to signal a preference among parents.

One or more Transit Information options MUST be preceded by one or more RPL Target options. In this manner the RPL Target option indicates the child node, and the Transit Information option(s) enumerate the DODAG parents.

A typical non-storing node will use multiple Transit Information options, and it will send the DAO thus formed to only one parent that will forward it to the root. A typical storing node with use one Transit Information option with no parent field, and will send the DAO thus formed to multiple parents.

Option Type: 0x06 (to be confirmed by IANA)

Option Length: Variable, depending on whether or not Parent Address is present.

Path-Sequence: 8-bit unsigned integer. When a RPL Target option is issued by the node that owns the Target Prefix (i.e. in a DAO message), that node sets the Path-Sequence and increments the Path-Sequence each time it issues a RPL Target option.

Path Control: 8-bit bitfield. The Path Control field limits the number of DAO-Parents to which a DAO message advertising connectivity to a specific destination may be sent, as well as providing some indication of relative preference. The limit provides some bound on overall DAO fan-out in the LLN. The leftmost bit is associated with a path that contains a most-preferred link, and the subsequent bits are ordered down to the rightmost bit which is least preferred.

Path Lifetime: 32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for route determination. A value of all one bits (0xFFFFFFFF) represents infinity. A value of all zero bits (0x00000000) indicates a loss of reachability. This is referred as a No-Path in this document.

Parent Address (optional): IPv6 Address of the DODAG Parent of the node originally issuing the Transit Information Option. This field may not be present, as according to the DODAG Mode of Operation and indicated by the Transit Information option length.

Unassigned bits of the Transit Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.6.9. Solicited Information

The Solicited Information option may be present in DIS messages, and its format is as follows:

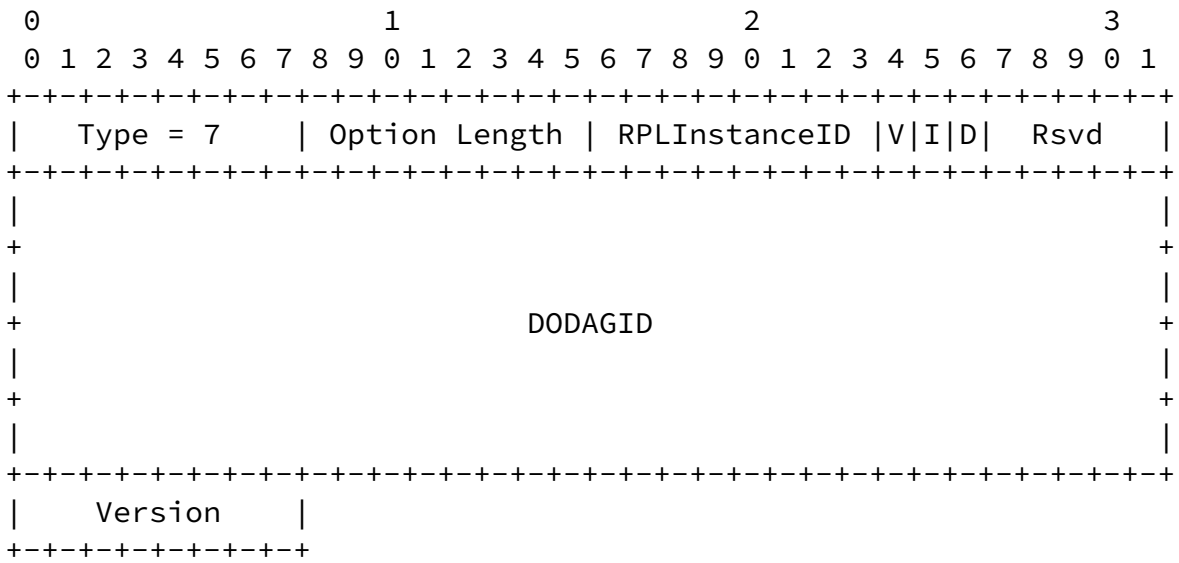


Figure 19: Format of the Solicited Information Option

The Solicited Information option is used for a node to request a subset of neighboring nodes that meet the specified criteria to respond to a DIS message.

The Solicited Information option may specify a number of predicate criteria to be matched by a receiving node. If a node receiving a

multicast DIS message containing a Solicited Information option matches ALL of the predicates, then it MUST reset its trickle timer in order to trigger a DIO response to the DIS message. When a node receives a DIS message containing a Solicited information option, and the DIS message is unicast OR the node does not match ALL the predicates, then the node MUST NOT reset the trickle timer.

Option Type: 0x07 (to be confirmed by IANA)

Option Length: 19 bytes

Control Field: The Solicited Information option Control Field has three flags:

V: If the V flag is set then the Version field is valid and a node should only respond if its DODAGVersionNumber matches the requested version. If the V flag is clear then the Version field is not valid and the Version field MUST be set to zero on transmission and ignored upon receipt.

I: If the I flag is set then the RPLInstanceID field is valid and a node should only respond if it matches the requested RPLInstanceID. If the I flag is clear then the RPLInstanceID field is not valid and the RPLInstanceID field MUST be set to zero on transmission and ignored upon receipt.

D: If the D flag is set then the DODAGID field is valid and a node should only respond if it matches the requested DODAGID. If the D flag is clear then the DODAGID field is not valid and the DODAGID field MUST be set to zero on transmission and ignored upon receipt.

Version: 8-bit unsigned integer containing the DODAG Version number that is being solicited when valid.

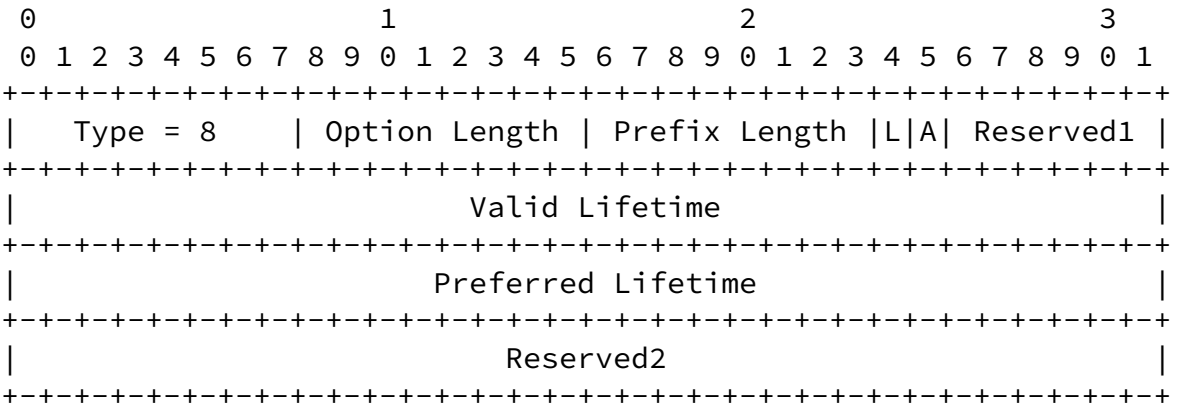
RPLInstanceID: 8-bit unsigned integer containing the RPLInstanceID that is being solicited when valid.

DODAGID: 128-bit unsigned integer containing the DODAGID that is being solicited when valid.

Unassigned bits of the Solicited Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

5.6.10. Prefix Information

The Prefix Information option may be present in DIO messages, and is equivalent in function to the IPv6 ND Prefix Information option as defined in [RFC4861]. The format of the option is modified slightly (Type, Length) in order to be carried as a RPL option as follows:



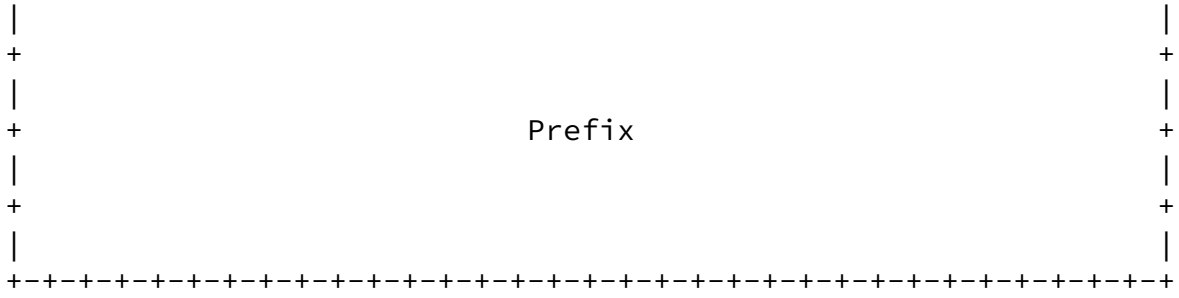


Figure 20: Format of the Prefix Information Option

The Prefix Information option may be used to distribute the prefix in use inside the DODAG, e.g. for address autoconfiguration.

[RFC4861] should be consulted as the authoritative reference with respect to the Prefix Information option. The field descriptions are transcribed here for convenience:

Option Type: 0x08 (to be confirmed by IANA)

Option Length: 30. Note that this length is expressed in units of single-octets, unlike in IPv6 ND.

Prefix Length 8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128. The prefix length field provides necessary information for on-link determination (when combined with the L flag in the prefix information option). It also assists with address autoconfiguration as specified in [RFC4862], for which there may be more restrictions on the prefix length.

L 1-bit on-link flag. When set, indicates that this prefix can be used for on-link determination. When not set the advertisement makes no statement about on-link or off-link properties of the prefix. In other words, if the L flag is not set a host MUST NOT conclude that an address derived from the prefix is off-link. That is, it MUST NOT update a previous

indication that the address is on-link.

A 1-bit autonomous address-configuration flag. When set indicates that this prefix can be used for stateless address

configuration as specified in [RFC4862].

Reserved1 6-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Valid Lifetime 32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. The Valid Lifetime is also used by [RFC4862].

Preferred Lifetime 32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred [RFC4862]. A value of all one bits (0xffffffff) represents infinity. See [RFC4862]. Note that the value of this field MUST NOT exceed the Valid Lifetime field to avoid preferring addresses that are no longer valid.

Reserved2 This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Prefix An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and MUST be initialized to zero by the sender and ignored by the receiver. A router SHOULD NOT send a prefix option for the link-local prefix and a host SHOULD ignore such a prefix option.

Unassigned bits of the Prefix Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6. Upward Routes

This section describes how RPL discovers and maintains upward routes. It describes the use of DODAG Information Objects (DIOs), the messages used to discover and maintain these routes. It specifies how RPL generates and responds to DIOs. It also describes DODAG Information Solicitation (DIS) messages, which are used to trigger DIO transmissions.

6.1. DIO Base Rules

1. If the 'A' flag of a DIO Base is cleared, the 'T' flag MUST also be cleared.

2. For the following DIO Base fields, a node that is not a DODAG root MUST advertise the same values as its preferred DODAG parent (defined in Section 6.2.1). Therefore, if a DODAG root does not change these values, every node in a route to that DODAG root eventually advertises the same values for these fields. These fields are:
    1. Grounded (G)
    2. Destination Advertisement Supported (A)
    3. Destination Advertisement Trigger (T)
    4. DAGPreference (Prf)
    5. Version
    6. RPLInstanceID
    7. DODAGID

3. A node MAY update the following fields at each hop:
    1. Destination Advertisements Stored (S)
    2. DAGRank
    3. DTSN

4. The DODAGID field each root sets MUST be unique within the RPL Instance.

6.2. Upward Route Discovery and Maintenance

Upward route discovery allows a node to join a DODAG by discovering neighbors that are members of the DODAG of interest and identifying a set of parents. The exact policies for selecting neighbors and parents is implementation-dependent and driven by the OF. This section specifies the set of rules those policies must follow for interoperability.

6.2.1. Neighbors and Parents within a DODAG Version

RPL's upward route discovery algorithms and processing are in terms of three logical sets of link-local nodes. First, the candidate neighbor set is a subset of the nodes that can be reached via link-local multicast. The selection of this set is implementation-dependent and OF-dependent. Second, the parent set is a restricted subset of the candidate neighbor set. Finally, the preferred parent, a set of size one, is an element of the parent set that is the preferred next hop in upward routes.

More precisely:

1. The DODAG parent set MUST be a subset of the candidate neighbor set.

2. A DODAG root MUST have a DODAG parent set of size zero.

3. A node that is not a DODAG root MAY maintain a DODAG parent set of size greater than or equal to one.

4. A node's preferred DODAG parent MUST be a member of its DODAG parent set.

5. A node's rank MUST be greater than all elements of its DODAG parent set.

6. When Neighbor Unreachability Detection (NUD), or an equivalent mechanism, determines that a neighbor is no longer reachable, a RPL node MUST NOT consider this node in the candidate neighbor set when calculating and advertising routes until it determines that it is again reachable. Routes through an unreachable neighbor MUST be removed from the routing table.

These rules ensure that there is a consistent partial order on nodes within the DODAG. As long as node ranks do not change, following the above rules ensures that every node's route to a DODAG root is loop-free, as rank decreases on each hop to the root. The OF can guide candidate neighbor set and parent set selection, as discussed in [I-D.ietf-roll-routing-metrics] and [I-D.ietf-roll-of0].

6.2.2. Neighbors and Parents across DODAG Versions

The above rules govern a single DODAG version. The rules in this section define how RPL operates when there are multiple DODAG versions:

6.2.2.1. DODAG Version

1. The tuple (RPLInstanceID, DODAGID, DODAGVersionNumber) uniquely defines a DODAG Version. Every element of a node's DODAG parent set, as conveyed by the last heard DIO message from each DODAG parent, MUST belong to the same DODAG version. Elements of a

node's candidate neighbor set MAY belong to different DODAG Versions.

2. A node is a member of a DODAG version if every element of its DODAG parent set belongs to that DODAG version, or if that node is the root of the corresponding DODAG.

3. A node MUST NOT send DIOs for DODAG versions of which it is not a member.

4. DODAG roots MAY increment the DODAGVersionNumber that they advertise and thus move to a new DODAG version. When a DODAG root increments its DODAGVersionNumber, it MUST follow the conventions of Serial Number Arithmetic as described in [RFC1982].

5. Within a given DODAG, a node that is a not a root MUST NOT advertise a DODAGVersionNumber higher than the highest DODAGVersionNumber it has heard. Higher is defined as the greater-than operator in [RFC1982].

6. Once a node has advertised a DODAG version by sending a DIO, it MUST NOT be member of a previous DODAG version of the same DODAG (i.e. with the same RPLInstanceID, the same DODAGID, and a lower DODAGVersionNumber). Lower is defined as the less-than operator in [RFC1982].

Within a particular implementation, a DODAG root may increment the DODAGVersionNumber periodically, at a rate that depends on the deployment, in order to trigger a global reoptimization of the DODAG. In other implementations, loop detection may be considered sufficient to solve routing issues by triggering local repair mechanisms, and the DODAG root may increment the DODAGVersionNumber only upon administrative intervention. Another possibility is that nodes within the LLN have some means by which they can signal detected routing inconsistencies or suboptimalities to the DODAG root, in order to request an on-demand DODAGVersionNumber increment (i.e. request a global repair of the DODAG). Note that such a mechanism is for further study and out of the scope of this document.

When the DODAG parent set becomes empty on a node that is not a root, (i.e. the last parent has been removed, causing the node to no longer be associated with that DODAG), then the DODAG information should not be suppressed until after the expiration of an implementation-specific local timer in order to observe if the DODAGVersionNumber has been incremented, should any new parents appear for the DODAG. This will help protect against the possibility of loops that may occur of that node were to inadvertently rejoin the old DODAG version in its own prior sub-DODAG.

As the DODAGVersionNumber is incremented, a new DODAG Version spreads outward from the DODAG root. Thus a parent that advertises the new DODAGVersionNumber cannot possibly belong to the sub-DODAG of a node that still advertises an older DODAGVersionNumber. A node may safely add such a parent, without risk of forming a loop, without regard to

its relative rank in the prior DODAG Version. This is equivalent to jumping to a different DODAG.

As a node transitions to new DODAG Versions as a consequence of following these rules, the node will be unable to advertise the previous DODAG Version (prior DODAGVersionNumber) once it has committed to advertising the new DODAG Version.

During transition to a new DODAG Version, a node may decide to forward packets via 'future parents' that belong to the same DODAG (same RPLInstanceID and DODAGID), but are observed to advertise a more recent (incremented) DODAGVersionNumber. In that case, the node MUST act as a leaf with regard to the new version for the purpose of loop detection as specified in Section 8.2.

6.2.2.2. DODAG Roots

1. A DODAG root that does not have connectivity to the set of addresses described as application-level goals, MUST NOT set the Grounded bit.

2. A DODAG root MUST advertise a rank of ROOT_RANK.

3. A node whose DODAG parent set is empty MAY become the DODAG root of a floating DODAG. It MAY also set its DAGPreference such that it is less preferred.

An LLN node that is a goal for the Objective Function is the root of its own grounded DODAG, at rank ROOT_RANK.

In a deployment that uses a backbone link to federate a number of LLN roots, it is possible to run RPL over that backbone and use one router as a "backbone root". The backbone root is the virtual root of the DODAG, and exposes a rank of BASE_RANK over the backbone. All the LLN roots that are parented to that backbone root, including the backbone root if it also serves as LLN root itself, expose a rank of ROOT_RANK to the LLN, and are part of the same DODAG, coordinating DODAGVersionNumber and other DODAG root determined parameters with the virtual root over the backbone.

6.2.2.3. DODAG Selection

The DODAGPreference (Prf) provides an administrative mechanism to engineer the self-organization of the LLN, for example indicating the most preferred LBR. If a node has the option to join a more preferred DODAG while still meeting other optimization objectives, then the node will generally seek to join the more preferred DODAG as determined by the OF. All else being equal, it is left to the

implementation to determine which DODAG is most preferred, possibly based on additional criteria beyond Prf and the OF.

6.2.2.4. Rank and Movement within a DODAG Version

1. A node MUST NOT advertise a rank less than or equal to any member of its parent set within the DODAG Version.

2. A node MAY advertise a rank lower than its prior advertisement within the DODAG Version.

3. Let L be the lowest rank within a DODAG version that a given node has advertised. Within the same DODAG Version, that node MUST NOT advertise an effective rank higher than L + DAGMaxRankIncrease. INFINITE_RANK is an exception to this rule: a node MAY advertise an INFINITE_RANK at any time. (This rule corresponds to a limited rank increase for the purpose of local repair within the DODAG Version.)

4. A node MAY, at any time, choose to join a different DODAG within a RPL Instance. Such a join has no rank restrictions, unless that different DODAG is a DODAG Version of which this node has previously been a member, in which case the rule of the previous bullet (3) must be observed. Until a node transmits a DIO indicating its new DODAG membership, it MUST forward packets along the previous DODAG.

5. A node MAY, at any time after hearing the next DODAGVersionNumber Version advertised from suitable DODAG parents, choose to migrate to the next DODAG Version within the DODAG.

Conceptually, an implementation is maintaining a DODAG parent set within the DODAG Version. Movement entails changes to the DODAG parent set. Moving up does not present the risk to create a loop but moving down might, so that operation is subject to additional constraints.

When a node migrates to the next DODAG Version, the DODAG parent and sibling sets need to be rebuilt for the new version. An implementation could defer to migrate for some reasonable amount of time, to see if some other neighbors with potentially better metrics but higher rank announce themselves. Similarly, when a node jumps into a new DODAG it needs to construct new DODAG parent/sibling sets for this new DODAG.

When a node moves to improve its position, it must conceptually abandon all DODAG parents and siblings with a rank larger than itself. As a consequence of the movement it may also add new

siblings. Such a movement may occur at any time to decrease the rank, as per the calculation indicated by the OF. Maintenance of the parent and sibling sets occurs as the rank of candidate neighbors is observed as reported in their DIOs.

If a node needs to move down a DODAG that it is attached to, causing the rank to increase, then it MAY poison its routes and delay before moving as described in Section 6.2.2.5.

6.2.2.5. Poisoning a Broken Path

1. A node MAY poison, in order to avoid being used as an ancestor by

the nodes in its sub-DODAG, by advertising an effective rank of INFINITE_RANK and resetting the associated DIO trickle timer to cause this INFINITE_RANK to be announced promptly.

2. The node MAY advertise an effective rank of INFINITE_RANK for an arbitrary number of DIO timer events, before announcing a new rank.

3. As per Section 6.2.2.4, the node MUST advertise INFINITE_RANK within the DODAG version in which it participates, if its revision in rank would exceed the maximum rank increase.

An implementation may choose to employ this poisoning mechanism when a node loses all of its current parents, i.e. the set of DODAG parents becomes depleted, and it can not jump to an alternate DODAG. An alternate mechanism is to form a floating DODAG.

The motivation for delaying announcement of the revised route through multiple DIO events is to (i) increase tolerance to DIO loss, (ii) allow time for the poisoning action to propagate, and (iii) to develop an accurate assessment of its new rank. Such gains are obtained at the expense of potentially increasing the delay before portions of the network are able to re-establish upwards routes. Path redundancy in the DODAG reduces the significance of either effect, since children with alternate parents should be able to utilize those alternates and retain their rank while the detached parent re-establishes its rank.

Although an implementation may advertise INFINITE_RANK for the purposes of poisoning, it is not expected to be equivalent to setting the rank to INFINITE_RANK, and an implementation would likely retain its rank value prior to the poisoning in some form, for purpose of maintaining its effective position within (L + DAGMaxRankIncrease).

6.2.2.6. Detaching

1. A node unable to stay connected to a DODAG within a given DODAG version MAY detach from this DODAG version. A node that detaches becomes root of its own floating DODAG and SHOULD immediately

advertise this new situation in a DIO as an alternate to poisoning.

6.2.2.7. Following a Parent

1. If a node receives a DIO from one of its DODAG parents, indicating that the parent has left the DODAG, that node SHOULD stay in its current DODAG through an alternative DODAG parent, if possible. It MAY follow the leaving parent.

A DODAG parent may have moved, migrated to the next DODAG Version, or jumped to a different DODAG. A node should give some preference to remaining in the current DODAG, if possible via an alternate parent, but ought to follow the parent if there are no other options.

6.2.3. DIO Message Communication

When an DIO message is received, the receiving node must first determine whether or not the DIO message should be accepted for further processing, and subsequently present the DIO message for further processing if eligible.

1. If the DIO message is malformed, then the DIO message is not eligible for further processing and MUST be silently discarded. A RPL implementation MAY log the reception of a malformed DIO message.

2. If the sender of the DIO message is a member of the candidate neighbor set, then the DIO is eligible for further processing.

6.2.3.1. DIO Message Processing

As DIO messages are received from candidate neighbors, the neighbors may be promoted to DODAG parents by following the rules of DODAG discovery as described in Section 6.2. When a node places a neighbor into the DODAG parent set, the node becomes attached to the DODAG through the new DODAG parent node.

The most preferred parent should be used to restrict which other nodes may become DODAG parents. Some nodes in the DODAG parent set may be of a rank less than or equal to the most preferred DODAG parent. (This case may occur, for example, if an energy constrained device is at a lesser rank but should be avoided as per an

optimization objective, resulting in a more preferred parent at a greater rank).

6.3. DIO Transmission

RPL nodes transmit DIOs using a Trickle timer ([I-D.ietf-roll-trickle]). A DIO from a sender with a lower DAGRank that causes no changes to the recipient's parent set, preferred parent, or Rank SHOULD be considered consistent with respect to the Trickle timer.

The following packets and events MUST be considered inconsistencies with respect to the Trickle timer, and cause the Trickle timer to reset:

- When a node detects an inconsistency when forwarding a packet, as detailed in Section 8.2.
- When a node receives a multicast DIS message whose constraints (Solicited Information) it satisfies.
- When a node joins a new DODAG Version (e.g. by updating its DODAGVersionNumber, joining a new RPL Instance, etc.)

Note that this list is not exhaustive, and an implementation MAY consider other messages or events to be inconsistencies.

If a node receives a unicast DIS message whose constraints (Solicited Information) it satisfies, it MUST unicast a DIO in response, and this DIO MUST include the RPL instance's DODAG Configuration object.

6.3.1. Trickle Parameters

The configuration parameters of the trickle timer are specified as follows:

Imin: learned from the DIO message as (2^DIOIntervalMin)ms. The default value of DIOIntervalMin is DEFAULT_DIO_INTERVAL_MIN.

Imax: learned from the DIO message as DIOIntervalDoublings. The default value of DIOIntervalDoublings is DEFAULT_DIO_INTERVAL_DOUBLINGS.

k: learned from the DIO message as DIORedundancyConstant. The default value of DIORedundancyConstant is DEFAULT_DIO_REDUNDANCY_CONSTANT. In RPL, when k has the value of 0x00 this is to be treated as a redundancy constant of infinity in RPL, i.e. Trickle never suppresses messages.

6.4. DODAG Selection

The DODAG selection is implementation and OF dependent. Nodes SHOULD prefer to join DODAGs for RPLInstanceIDs advertising OCPs and destinations compatible with their implementation specific objectives. In order to limit erratic movements, and all metrics being equal, nodes SHOULD keep their previous selection. Also, nodes SHOULD provide a means to filter out a parent whose availability is detected as fluctuating, at least when more stable choices are available.

When connection to a grounded DODAG is not possible or preferable for security or other reasons, scattered DODAGs MAY aggregate as much as possible into larger DODAGs in order to allow connectivity within the LLN.

A node SHOULD verify that bidirectional connectivity and adequate link quality is available with a candidate neighbor before it considers that candidate as a DODAG parent.

6.5. Operation as a Leaf Node

In some cases a RPL node may attach to a DODAG as a leaf node only. One example of such a case is when a node does not understand the RPL Instance's OF or advertised path metric. A leaf node does not extend DODAG connectivity but still needs to advertise its presence using DIOs. A node operating as a leaf node must obey the following rules:

1. It MUST NOT transmit DIOs containing the DAG Metric Container.

2. Its DIOs must advertise a DAGRank of INFINITE_RANK.

3. It MAY transmit unicast DAOs as described in Section 7.1.

4. It MAY transmit multicast DAOs to the '1 hop' neighborhood as described in Section 7.1.9.

6.6. Administrative Rank

In some cases it might be beneficial to adjust the rank advertised by a node beyond that computed by the OF based on some implementation specific policy and properties of the node. For example, a node that

has limited battery should be a leaf unless there is no other choice, and may then augment the rank computation specified by the OF in order to expose an exaggerated rank.

7. Downward Routes

This section describes how RPL discovers and maintains downward routes. The use of messages containing the Destination Advertisement Object (DAO), used to construct downward routes, are described. The downward routes are necessary in support of P2MP flows, from the DODAG roots toward the leaves. It specifies non-storing and storing behavior of nodes with respect to DAO messaging and DAO routing table entries. Nodes, as according to their resources and the implementation, may selectively store routing table entries learned from DAO messages, or may instead propagate the DAO information upwards and independently source local topology information in a new DAO message. information. A further optimization is described whereby DAO messages may be used to populate routing table entries for the '1-hop' neighbors, which may be useful in some cases as a shortcut for P2P flows.

7.1. Downward Route Discovery and Maintenance

7.1.1. Overview

Destination Advertisement operation produces DAO messages that flow up the DODAG, provisioning downward routing state for destination prefixes available in the sub-DODAG of the DODAG root, and possibly other nodes. The routing state provisioned with this mechanism is in the form of soft-state routing table entries. DAO operation is presently defined in two distinct modes of operation, non-storing and storing, and allowance is made for future expansion.

Destination Advertisement may or may not be enabled over a DODAG rooted at a DODAG root. This is an a priori configuration determined by the implementation/deployment and not generally changed during the operation of the RPL LLN.

Destination Advertisement may be configured to operate in either a

storing or non-storing mode, as reported in the MOP in the DIO message. Every node in the network participating in Destination Advertisement must behave consistently with that configured mode of operation, or alternately behave only as a leaf node. Hybrid or mixed-mode operation is not currently specified.

When Destination Advertisement is enabled:

1. The RPL Instance will be configured a priori as appropriate to satisfy the application to operate in either non-storing or storing mode.

2. All nodes who join the DODAG MUST abide with the MOP setting from the root. Nodes that would not have the capability to fully participate as a router (e.g. to operate as a storing node) can still join as a leaf (i.e. host).

3. In storing mode operation, all non-root nodes are expected to either store routing table entries for ALL destinations learned from DAO operation, or to act as a leaf node only.

4. In non-storing mode operation, no node other than the DODAG Root is expected to store routing table entries learned from DAO messages. Each node is only responsible to report its own set of parents to the DODAG Root.

5. DODAG roots nodes SHOULD be capable to store routing table entries learned from DAO operation when the RPL Instance is operated in a non-storing mode.

6. The mode of operation in the RPL Instance is signaled from the DODAG Root in the MOP control field of the DIO message.

7.1.2. Mode of Operation

- DAO Operation may not be required for all use cases.

- Some applications may only need support for collection/upward/MP2P flow with no acknowledgement/reciprocal traffic.

- Some DODAGs may not support DAO Operation, which could mean that DAO Operation is wasteful overhead.

- As a special case, multicast DAO operation may be used to populate 'one-hop' neighborhood routing table entries, and is distinct from the unicast DAO operation used to establish downward routes along the DODAG. This special case is an exception to the RPL Instance mode of operation as well.

1. The 'A' flag in the DIO as conveyed from the DODAG root serves to enable/disable DAO operation over the entire DODAG. This flag should be administratively provisioned a priori at the DODAG root as a function of the implementation/deployment and not tend to change.

2. When DAO Operation is disabled, a node MUST NOT emit DAO messages.

3. When DAO Operation is disabled, a node MAY ignore the MOP field.

4. When DAO Operation is disabled, a node MAY ignore received DAO messages.

7.1.3. Destination Advertisement Parents

- Nodes will select a subset of their DODAG Parents to whom DAO messages will be sent
  - This subset is the set of 'DAO Parents'
  - Each DAO parent MUST be a DODAG Parent. (Not all DODAG parents need to be DAO parents).
- The selection of DAO parents is implementation specific and may be based on selecting the DODAG Parents that offer the best upwards cost (as opposed to downwards or mixed), as determined by the metrics in use and the Objective Function.
- When DAO messages are unicast to the DAO Parent, the identity of the DAO Parent (DODAGID and DODAGVersionNumber) combined with the RPLInstanceID in the DAO message unambiguously associates the DAO

message, and thus the particular destination prefix, with a DODAG Version.

7.1.4. DAO Operation on Storing Nodes

7.1.4.1. DAO Routing Table Entry

A DAO Routing Table Entry conceptually contains the following elements:

- Advertising Neighbor Information
  - IPv6 Address
  - Interface ID
- To which DAO Parents has this entry been reported
- Retry Counter
- Logical equivalent of DAO Content:
  - DAO Sequence
  - DAO Lifetime
  - DAO Path Control (as learned from each child)
  - Destination Prefix (or Address or Mcast Group)

The DAO Routing Table Entry is logically associated with the following states:

CONNECTED This entry is 'owned' by the node - it is manually configured and is considered as a 'self' entry for DAO Operation

REACHABLE This entry has been reported from a neighbor of the node. This state includes the following substates:

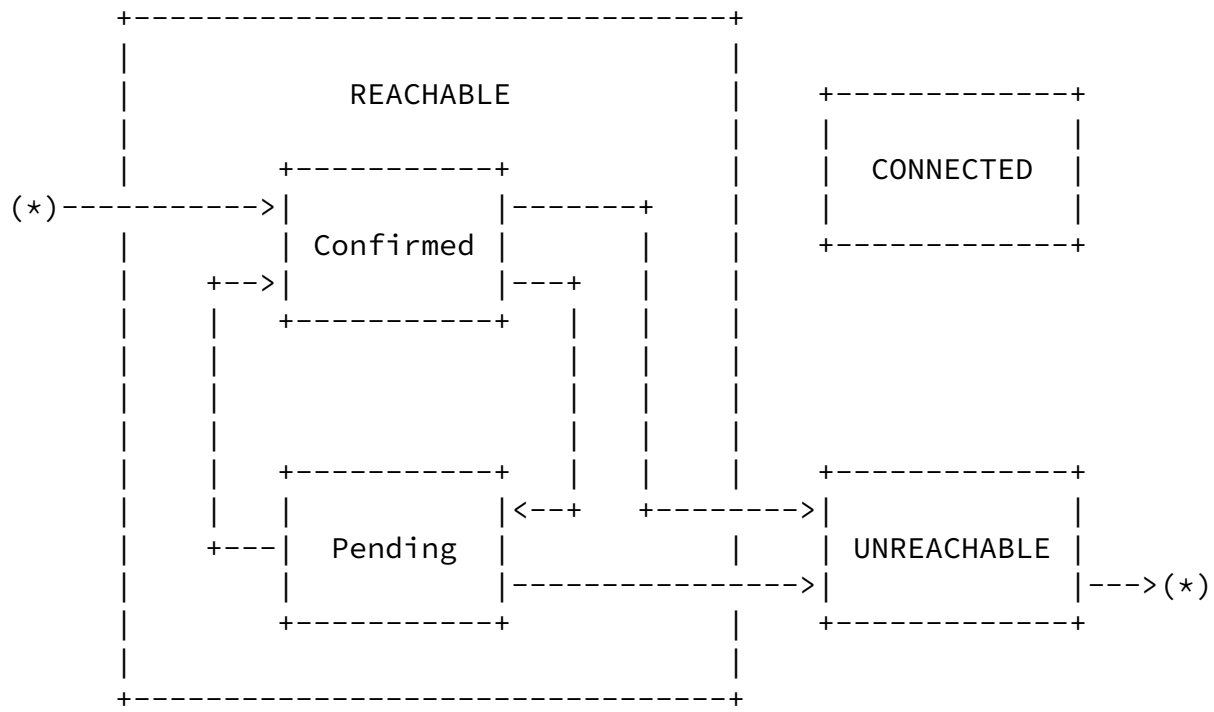
CONFIRMED This entry is active, newly validated, and usable

PENDING This entry is active, awaiting validation, and usable. A Retry Counter is associated with this substate

UNREACHABLE This entry is being cleaned up. This entry may be suppressed when the cleanup process is complete.

When an attempt is to be made to report the DAO entry to DAO Parents, the DAO Entry record is logically marked to indicate that an attempt has not yet been made for each parent. As the unicast attempts are completed for each parent, this mark may be cleared. This mechanism may serve to limit DAO entry updates for each parent to a subset that needs to be reported.

7.1.4.1.1. DAO Routing Table Entry Management



DAO Routing Table Entry FSM

7.1.4.1.1.1. Operation in the CONNECTED state

1. CONNECTED DAO entries are to be provisioned outside of the context of RPL, e.g. through a management API. An implementation SHOULD provide a means to provision/manage CONNECTED DAO entries,

including whether they are to be redistributed in RPL.

7.1.4.1.1.2. Operation in the REACHABLE state

1. When a REACHABLE(*) entry times out, i.e. the DAO Lifetime has elapsed, the entry MUST be placed into the UNREACHABLE state and No-Path SHOULD be scheduled to send to the node's DAO Parents.

2. When a No-Path for a REACHABLE(*) entry is received with a newer DAO Sequence Number, the entry MUST be placed into the UNREACHABLE state and No-Path SHOULD be scheduled to send to the node's DAO Parents.

3. When a REACHABLE(*) entry is to be removed because NUD or equivalent has determined that the next-hop neighbor is no longer reachable, the entry MUST be placed into the UNREACHABLE state and No-Path SHOULD be scheduled to send to the node's DAO Parents.

4. When a REACHABLE(*) entry is to be removed because an associated Forwarding Error has been returned by the next-hop neighbor, the entry MUST be placed into the UNREACHABLE state and No-Path SHOULD be scheduled to send to the node's DAO Parents.

5. When a DAO (or No-Path) for a REACHABLE(*) entry is received with an older or unchanged DAO Sequence Number, then the DAO (or No-Path) SHOULD be ignored and the associated entry MUST NOT be updated with the stale information.

7.1.4.1.1.2.1. REACHABLE(Confirmed)

1. When a DAO for a previously unknown (or UNREACHABLE) destination is received and is to be stored, it MUST be entered into the routing table in the REACHABLE(Confirmed) state, and a DAO SHOULD be scheduled to send to the node's DAO Parents.

2. When a DAO for a REACHABLE(Confirmed) entry is received with a newer DAO Sequence Number, the entry MUST be updated with the logical equivalent of the DAO contents and a DAO SHOULD be scheduled to send to the node's DAO Parents.

3. When a DAO for a REACHABLE(Confirmed) entry is expected, e.g. because a DIO to request a DAO refresh is sent, then the DAO entry MUST be placed in the REACHABLE(Pending) state and the associated Retry Counter MUST be set to 0.

7.1.4.1.1.2.2. REACHABLE(Pending)

1. When a DAO for a REACHABLE(Pending) entry is received with a newer DAO Sequence Number, the entry MUST be updated with the logical equivalent of the DAO contents and the entry MUST be placed in the REACHABLE(Confirmed) state.

2. When a DAO for a REACHABLE(Pending) entry is expected, e.g. because DAO has (again) been triggered with respect to that neighbor, then the associated Retry Counter MUST be incremented.

3. When the associated Retry Counter for a REACHABLE(Pending) entry reaches a maximum threshold, the entry MUST be placed into the UNREACHABLE state and No-Path SHOULD be scheduled to send to the node's DAO Parents.

7.1.4.1.1.3. Operation in the UNREACHABLE state

1. An implementation SHOULD bound the time that the entry is allocated in the UNREACHABLE state. Upon the equivalent expiry of the related timer (RemoveTimer), the entry SHOULD be suppressed.

2. While the entry is in the UNREACHABLE state a node SHOULD make a reasonable attempt to report a No-Path to each of the DAO parents.

3. When the node has completed an attempt to report a No-Path to each of the DAO parents, the entry SHOULD be suppressed.

7.1.4.2. Storing Mode DAO Message and Path Control

In the storing mode of operation, a DAO message from a node will contain one or more Target Options, each Target Option specifying either a CONNECTED destination or a destination in the sub-DODAG of the node.

For each attempt made to report the DAO entry to a set of DAO parents, the Path Control field will be constructed as follows:

1. The size of the path control field will be specified by the PCS control field of the DODAG Configuration Option. The default value is DEFAULT_PATH_CONTROL_SIZE.

2. For each unique destination to be reported that is CONNECTED, the logical equivalent of a path control bitmap that is the size of the path control field shall be initialized with the leftmost bits set, where the number of leftmost bits corresponds to the size of the path control field as specified by PCS.

3. For each unique destination to be reported that is not CONNECTED, i.e. that destination is contained in the node's sub-DODAG, the logical equivalent of a path control bitmap that is the size of the path control field shall be initialized by ORing the content of all of the Path Control fields received in DAO messages from the node's children for that destination.

4. For each DAO Parent that the node shall attempt an update to, the node shall exclusively allocate 1 or more set bits from the path control bitmap to that DAO Parent. The path control bits SHOULD be allocated in order of preference, such that the most significant bits, or groupings of bits, are allocated to the most preferred DAO parents as determined by the node. Once a bit from the path control bitmap has been allocated to a DAO Parent for this attempt, the corresponding bit MUST be set in the Path Control field in the DAO message sent to that DAO Parent, and that bit MUST NOT be allocated to any other DAO Parent.

5. A unicast DAO message may be sent for DAO Parents that have a non-zero Path Control field.

6. If any DAO Parent is left without any bits set in its Path Control field, then that a unicast DAO message MUST NOT be sent to that DAO parent for this attempt.

7.1.5. Operation of DAO Non-storing Nodes

1. In the non-storing mode of operation, each node sending a DAO message to its DODAG Parents will include a RPL Target option to describe itself, followed by RPL Transit Information option(s) to describe its parents. This information is sufficient for the DODAG Root to collect the DODAG topology and construct source routes in the downward direction.

2. In the non-storing mode of operation, each node receiving a DAO message will arrange to pass the content of the DAO message along to the DODAG Root. When possible the content of DAO messages may

be aggregated.

3. When a DAO is received from a child by a node who will not store a routing table entry for the DAO, the node MUST schedule to pass the DAO contents along to its DAO parents.

7.1.6. Scheduling to Send DAO (or No-Path)

1. An implementation SHOULD arrange to rate-limit the sending of DAOs.

2. When scheduling to send a DAO, an implementation SHOULD equivalently start a timer (DelayDAO) to delay sending the DAO. If the DelayDAO timer is already running then the DAO may be considered as already scheduled, and implementation SHOULD leave the timer running at its present duration.

o When computing the delay before sending a DAO, in order to increase the effectiveness of aggregation, an implementation MAY allow time to receive DAOs from its sub-DODAG prior to emitting DAOs to its DAO Parents.

  * Suppose there is an implementation parameter DAO_LATENCY which represents the maximum expected time for a DAO operation to traverse the LLN from the farthest node to the root. The scheduled delay in such cases may be, for example, such that DAO_LATENCY/DAGRank(self_rank) <= DelayDAO < DAO_LATENCY/ DAGRank(parent_rank), where DAGRank() is defined as in Section 3.5.2, such that nodes deeper in the DODAG may tend to report DAO messages first before their parent nodes will report DAO messages. Note that this suggestion is intended as an optimization to allow efficient aggregation -- it is not required for correct operation in the general case.

7.1.7. Triggering DAO Message from the Sub-DODAG

Triggering DAO messages from the Sub-DODAG occurs by using the following control fields with the rules described below:

The DTSN field from the DIO is a sequence number that is part of the mechanism to trigger DAO messages. The motivation to use a sequence number is to provide some means of reliable signaling to the sub-

DODAG. Whereas a control flag that is activated for a short time may be unobserved by the sub-DODAG if the triggering DIO messages are lost, the DTSN increment may be observed later even if some intervening DIO messages have been lost.

The 'T' flag provides a way to signal the refresh of DAO information over the entire DODAG version. Whereas a DTSN increment may only trigger a DAO refresh as far as the next storing node (because a storing node will not increment its own DTSN in response, as described in the rules below), the assertion of the 'T' flag in conjunction with an incremented DTSN will result in a DAO refresh from the entire DODAG.

The control fields are used to trigger DAO messages as follows:

1. A DAO Trigger Sequence Number (DTSN) MUST be maintained by each node per RPL Instance. The DTSN, in conjunction with the 'T' flag from the DIO message, provides a means by which DAO messages may be reliably triggered in the event of topology change.

2. The DTSN MUST be advertised by the node in the DIO message.

3. A node keeps track of the DTSN that it has heard from the last DIO from each of its DAO Parents. Note that there is one DTSN maintained per DAO Parent- each DAO Parent may independently increment it at will.

4. DAO Transmission SHOULD be scheduled when a new parent is added to the DAO Parent set.

5. A node that receives a newly incremented DTSN from a DAO Parent MUST schedule a DAO transmission.

- In storing mode operation, when a node sees a DTSN increment, it is caused to reissue its entire set of routing table entries learned from DAO messages (or an aggregated subset thereof), but will not need to increment its own DTSN.

- In either storing or non-storing modes of operation, when a node sees a DTSN increment AND the 'T' flag is set, it does increment its own DTSN as well. The 'T' flag 'punches through' all nodes, causing all routing state from the entire sub-DODAG to be

refreshed.

7.1.8. Sending DAO Messages to DAO Parents

1. DAO Messages sent to DAO Parents MUST be unicast.

   * The IPv6 Source Address is a link local address of the node sending the DAO message.

   * The IPv6 Destination Address is a link local address of the DAO parent.

2. A node MUST send the DAO with the same sequence to all its DAO parents that are to be used on the way back to the DAO target.

3. When using source routing, a Destination that builds the DAO also indicates its parent in the DAO as a Transit Information option. If the node has multiple DAO parents, it MAY include one Transit Information Option per parent and pass the DAO to one or more

parent. The Transit Information option indicates the preference for that parent encoded in the Path Control bitfield.

4. When the appointed time arrives (DelayDAO) for the transmission of DAO messages (with jitter as appropriate) for the requested entries, the implementation MAY aggregate the the entries into a reduced numbers of DAOs to be reported to each parent, and perform compression if possible.

5. Note: it is NOT RECOMMENDED that a DAO Transmission (No-Path) be scheduled when a DAO Parent is removed from the DAO Parent set.

6. A node MAY set the K flag in a unicast DAO message to solicit a unicast DAO-ACK in response in order to confirm the attempt. A node receiving a unicast DAO message with the K flag set SHOULD respond with a DAO-ACK. A node receiving a DAO message without the K flag set MAY respond with a DAO-ACK, especially to report an error condition.

7.1.9. Multicast Destination Advertisement Messages

A special case of DAO operation, distinct from unicast DAO operation,

is multicast DAO operation which may be used to populate '1-hop' routing table entries.

1. A node MAY multicast a DAO message to the link-local scope all-nodes multicast address FF02::1.

2. A multicast DAO message MUST be used only to advertise information about self, i.e. prefixes directly connected to or owned by this node, such as a multicast group that the node is subscribed to or a global address owned by the node.

3. A multicast DAO message MUST NOT be used to relay connectivity information learned (e.g. through unicast DAO) from another node.

4. Information obtained from a multicast DAO MAY be installed in the routing table and MAY be propagated by a node in unicast DAOs.

5. A node MUST NOT perform any other DAO related processing on a received multicast DAO, in particular a node MUST NOT perform the actions of a DAO parent upon receipt of a multicast DAO.

- The multicast DAO may be used to enable direct P2P communication, without needing the RPL routing structure to relay the packets.

- The multicast DAO does not presume any DODAG relationship between the emitter and the receiver.

8. Packet Forwarding and Loop Avoidance/Detection

8.1. Suggestions for Packet Forwarding

When forwarding a packet to a destination, precedence is given to selection of a next-hop successor as follows:

1. This specification only covers how a successor is selected from the DODAG version that matches the RPLInstanceID marked in the IPv6 header of the packet being forwarded. Routing outside the instance can be done as long as additional rules are put in place such as strict ordering of instances and routing protocols to protect against loops.

2. If a local administrative preference favors a route that has been

learned from a different routing protocol than RPL, then use that successor.

3. If there is an entry in the routing table matching the destination that has been learned from a multicast destination advertisement (e.g. the destination is a one-hop neighbor), then use that successor.

4. If there is an entry in the routing table matching the destination that has been learned from a unicast destination advertisement (e.g. the destination is located down the sub-DODAG), then use that successor. If there are DAO Path Control bits associated with multiple successors, then consult the Path Control bits to order the successors by preference when choosing.

5. If there is a DODAG version offering a route to a prefix matching the destination, then select one of those DODAG parents as a successor according to the OF and routing metrics.

6. Any other as-yet-unattempted DODAG parent may be chosen for the next attempt to forward a unicast packet when no better match exists.

7. If there is a DODAG version offering a route to a prefix matching the destination, but all DODAG parents have been tried and are temporarily unavailable (as determined by the forwarding procedure), then select a DODAG sibling as a successor (after appropriate packet marking for loop detection as described in Section 8.2.

8. Finally, if no DODAG siblings are available, the packet is dropped. ICMP Destination Unreachable may be invoked (an inconsistency is detected).

TTL must be decremented when forwarding. If the packet is being forwarded via a sibling, then the TTL may be decremented more aggressively (by more than one) to limit the impact of possible loops.

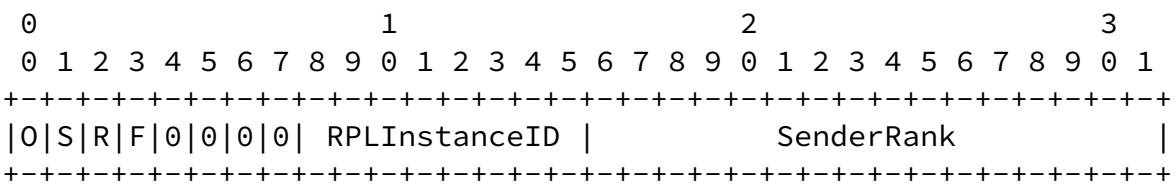
Note that the chosen successor MUST NOT be the neighbor that was the predecessor of the packet (split horizon), except in the case where it is intended for the packet to change from an up to an down flow,

such as switching from DIO routes to DAO routes as the destination is neared.

8.2. Loop Avoidance and Detection

RPL loop avoidance mechanisms are kept simple and designed to minimize churn and states. Loops may form for a number of reasons, from control packet loss to sibling forwarding. RPL includes a reactive loop detection technique that protects from meltdown and triggers repair of broken paths.

RPL loop detection uses information that is placed into the packet. A future version of this specification will detail how this information is carried with the packet (e.g. a hop-by-hop option ([I-D.hui-6man-rpl-option]) or summarized somehow into the flow label). For the purpose of RPL operations, the information carried with a packet is constructed follows:



RPL Packet Information

Down 'O' bit: 1-bit flag indicating whether the packet is expected to progress up or down. A router sets the 'O' bit when the packet is expect to progress down (using DAO routes), and resets it when forwarding towards the root of the DODAG version. A host or RPL leaf node MUST set the bit to 0.

Sibling 'S' bit: 1-bit flag indicating whether the packet has been forwarded via a sibling at the present rank, and denotes a risk of a sibling loop. A host or RPL leaf node MUST set the bit to 0.

Rank-Error 'R' bit: 1-bit flag indicating whether a rank error was

detected. A rank error is detected when there is a mismatch in the relative ranks and the direction as indicated in the 'O' bit. A host or RPL leaf node MUST set the bit to 0.

Forwarding-Error 'F' bit: 1-bit flag indicating that this node can not forward the packet further towards the destination. The 'F' bit might be set by sibling that can not forward to a parent a packet with the Sibling 'S' bit set, or by a child node that does not have a route to destination for a packet with the down 'O' bit set. A host or RPL leaf node MUST set the bit to 0.

RPLInstanceID: 8-bit field indicating the DODAG instance along which the packet is sent.

SenderRank: 16-bit field set to zero by the source and to DAGRank(rank) by a router that forwards inside the RPL network.

8.2.1. Source Node Operation

If the source is aware of the RPLInstanceID that is preferred for the packet, then it MUST set the RPLInstanceID field associated with the packet accordingly, otherwise it MUST set it to the RPL_DEFAULT_INSTANCE.

8.2.2. Router Operation

8.2.2.1. Instance Forwarding

Instance IDs are used to avoid loops between DODAGs from different origins. DODAGs that constructed for antagonistic constraints might contain paths that, if mixed together, would yield loops. Those loops are avoided by forwarding a packet along the DODAG that is associated to a given instance.

The RPLInstanceID is associated by the source with the packet. This RPLInstanceID MUST match the RPL Instance onto which the packet is placed by any node, be it a host or router. For traffic originating outside of the RPL domain there may be a mapping occurring at the gateway into the RPL domain, possibly based on an encoding within the flow label. This aspect of RPL operation is to be clarified in a future version of this specification.

When a router receives a packet that specifies a given RPLInstanceID and the node can forward the packet along the DODAG associated to that instance, then the router MUST do so and leave the RPLInstanceID value unchanged.

If any node can not forward a packet along the DODAG associated to the RPLInstanceID, then the node SHOULD discard the packet and send an ICMP error message.

8.2.2.2. DAG Inconsistency Loop Detection

The DODAG is inconsistent if the direction of a packet does not match the rank relationship. A receiver detects an inconsistency if it receives a packet with either:

- the 'O' bit set (to down) from a node of a higher rank.
- the 'O' bit reset (for up) from a node of a lesser rank.
- the 'S' bit set (to sibling) from a node of a different rank.

When the DODAG root increments the DODAGVersionNumber a temporary rank discontinuity may form between the next version and the prior version, in particular if nodes are adjusting their rank in the next version and deferring their migration into the next version. A router that is still a member of the prior version may choose to forward a packet to a (future) parent that is in the next version. In some cases this could cause the parent to detect an inconsistency because the rank-ordering in the prior version is not necessarily the same as in the next version and the packet may be judged to not be making forward progress. If the sending router is aware that the chosen successor has already joined the next version, then the sending router MUST update the SenderRank to INFINITE_RANK as it forwards the packets across the discontinuity into the next DODAG version in order to avoid a false detection of rank inconsistency.

One inconsistency along the path is not considered as a critical error and the packet may continue. But a second detection along the path of a same packet should not occur and the packet is dropped.

This process is controlled by the Rank-Error bit associated with the packet. When an inconsistency is detected on a packet, if the Rank-Error bit was not set then the Rank-Error bit is set. If it was set the packet is discarded and the trickle timer is reset.

8.2.2.3. Sibling Loop Avoidance

When a packet is forwarded along siblings, it cannot be checked for forward progress and may loop between siblings. Experimental evidence has shown that one sibling hop can be very useful and is generally sufficient to avoid loops. Based on that evidence, this

specification enforces the simple rule that a packet may not make 2 sibling hops in a row.

When a host issues a packet or when a router forwards a packet to a non-sibling, the Sibling bit in the packet must be reset. When a router forwards to a sibling: if the Sibling bit was not set then the Sibling bit is set. If the Sibling bit was set then then the router SHOULD return the packet to the sibling that that passed it with the Forwarding-Error 'F' bit set and the 'S' bit left untouched.

8.2.2.4. DAO Inconsistency Loop Detection and Recovery

A DAO inconsistency happens when router that has an down DAO route via a child that is a remnant from an obsolete state that is not matched in the child. With DAO inconsistency loop recovery, a packet can be used to recursively explore and cleanup the obsolete DAO states along a sub-DODAG.

In a general manner, a packet that goes down should never go up again. If DAO inconsistency loop recovery is applied, then the router SHOULD send the packet back to the parent that passed it with the Forwarding-Error 'F' bit set and the 'O' bit left untouched. Otherwise the router MUST silently discard the packet.

8.2.2.5. Forward Path Recovery

Upon receiving a packet with a Forwarding-Error bit set, the node MUST remove the routing states that caused forwarding to that neighbor, clear the Forwarding-Error bit and attempt to send the packet again. The packet may be sent to an alternate neighbor. If that alternate neighbor still has an inconsistent DAO state via this node, the process will recurse, this node will set the Forwarding-Error 'F' bit and the routing state in the alternate neighbor will be cleaned up as well.

9. Multicast Operation

This section describes further the multicast routing operations over an IPv6 RPL network, and specifically how unicast DAOs can be used to relay group registrations up. Wherever the following text mentions Multicast Listener Discovery (MLD), one can read MLDv1 ([RFC2710]) or

MLDv2 ([RFC3810]).

As is traditional, a listener uses a protocol such as MLD with a router to register to a multicast group.

Along the path between the router and the DODAG root, MLD requests are mapped and transported as DAO messages within the RPL protocol; each hop coalesces the multiple requests for a same group as a single DAO message to the parent(s), in a fashion similar to proxy IGMP, but

recursively between child router and parent up to the root.

A router might select to pass a listener registration DAO message to its preferred parent only, in which case multicast packets coming back might be lost for all of its sub-DODAG if the transmission fails over that link. Alternatively the router might select to copy additional parents as it would do for DAO messages advertising unicast destinations, in which case there might be duplicates that the router will need to prune.

As a result, multicast routing states are installed in each router on the way from the listeners to the root, enabling the root to copy a multicast packet to all its children routers that had issued a DAO message including a DAO for that multicast group, as well as all the attached nodes that registered over MLD.

For unicast traffic, it is expected that the grounded root of an DODAG terminates RPL and MAY redistribute the RPL routes over the external infrastructure using whatever routing protocol is used in the other routing domain. For multicast traffic, the root MAY proxy MLD for all the nodes attached to the RPL domain (this would be needed if the multicast source is located in the external infrastructure). For such a source, the packet will be replicated as it flows down the DODAG based on the multicast routing table entries installed from the DAO message.

For a source inside the DODAG, the packet is passed to the preferred parents, and if that fails then to the alternates in the DODAG. The packet is also copied to all the registered children, except for the one that passed the packet. Finally, if there is a listener in the external infrastructure then the DODAG root has to further propagate the packet into the external infrastructure.

As a result, the DODAG Root acts as an automatic proxy Rendezvous Point for the RPL network, and as source towards the Internet for all multicast flows started in the RPL LLN. So regardless of whether the root is actually attached to the Internet, and regardless of whether the DODAG is grounded or floating, the root can serve inner multicast streams at all times.

10. Maintenance of Routing Adjacency

The selection of successors, along the default paths up along the DODAG, or along the paths learned from destination advertisements down along the DODAG, leads to the formation of routing adjacencies that require maintenance.

In IGPs such as OSPF [RFC4915] or IS-IS [RFC5120], the maintenance of a routing adjacency involves the use of Keepalive mechanisms (Hellos) or other protocols such as BFD ([I-D.ietf-bfd-base]) and MANET Neighborhood Discovery Protocol (NHDP [I-D.ietf-manet-nhdp]). Unfortunately, such an approach is not desirable in constrained environments such as LLN and would lead to excessive control traffic in light of the data traffic with a negative impact on both link loads and nodes resources. Overhead to maintain the routing adjacency should be minimized. Furthermore, it is not always possible to rely on the link or transport layer to provide information of the associated link state. The network layer needs to fall back on its own mechanism.

Thus RPL makes use of a different approach consisting of probing the neighbor using a Neighbor Solicitation message (see [RFC4861]). The reception of a Neighbor Advertisement (NA) message with the "Solicited Flag" set is used to verify the validity of the routing adjacency. Such mechanism MAY be used prior to sending a data packet. This allows for detecting whether or not the routing adjacency is still valid, and should it not be the case, select another feasible successor to forward the packet.

11. Guidelines for Objective Functions

An Objective Function (OF) allows for the selection of a DODAG to join, and a number of peers in that DODAG as parents. The OF is used to compute an ordered list of parents. The OF is also responsible to compute the rank of the device within the DODAG version.

The Objective Function is indicated in the DIO message using an Objective Code Point (OCP), as specified in [I-D.ietf-roll-routing-metrics], and indicates the method that must be used to construct the DODAG. The Objective Code Points are specified in [I-D.ietf-roll-routing-metrics], [I-D.ietf-roll-of0], and related companion specifications.

11.1. Objective Function Behavior

Most Objective Functions are expected to follow the same abstract behavior:

- The parent selection is triggered each time an event indicates that a potential next hop information is updated. This might happen upon the reception of a DIO message, a timer elapse, all DODAG parents are unavailable, or a trigger indicating that the state of a candidate neighbor has changed.

- An OF scans all the interfaces on the device. Although there may typically be only one interface in most application scenarios, there might be multiple of them and an interface might be configured to be usable or not for RPL operation. An interface can also be configured with a preference or dynamically learned to be better than another by some heuristics that might be link-layer dependent and are out of scope. Finally an interface might or not match a required criterion for an Objective Function, for instance a degree of security. As a result some interfaces might be completely excluded from the computation, while others might be more or less preferred.

- An OF scans all the candidate neighbors on the possible interfaces to check whether they can act as a router for a DODAG. There might be multiple of them and a candidate neighbor might need to pass some validation tests before it can be used. In particular, some link layers require experience on the activity with a router to enable the router as a next hop.

- An OF computes self's rank by adding to the rank of the candidate a value representing the relative locations of self and the candidate in the DODAG version.
  - The increase in rank must be at least MinHopRankIncrease.
  - To keep loop avoidance and metric optimization in alignment, the increase in rank should reflect any increase in the metric value. For example, with a purely additive metric such as ETX, the increase in rank can be made proportional to the increase in the metric.
  - Candidate neighbors that would cause self's rank to increase are not considered for parent selection
- Candidate neighbors that advertise an OF incompatible with the set of OF specified by the policy functions are ignored.
- As it scans all the candidate neighbors, the OF keeps the current best parent and compares its capabilities with the current candidate neighbor. The OF defines a number of tests that are critical to reach the objective. A test between the routers determines an order relation.
  - If the routers are equal for that relation then the next test is attempted between the routers,
  - Else the best of the two routers becomes the current best parent and the scan continues with the next candidate neighbor

  - Some OFs may include a test to compare the ranks that would result if the node joined either router
- When the scan is complete, the preferred parent is elected and self's rank is computed as the preferred parent rank plus the step in rank with that parent.
- Other rounds of scans might be necessary to elect alternate parents and siblings. In the next rounds:
  - Candidate neighbors that are not in the same DODAG are ignored

* Candidate neighbors that are of greater rank than self are ignored

* Candidate neighbors of an equal rank to self (siblings) are ignored for parent selection

* Candidate neighbors of a lesser rank than self (non-siblings) are preferred

12. RPL Constants and Variables

Following is a summary of RPL constants and variables.

BASE_RANK This is the rank for a virtual root that might be used to coordinate multiple roots. BASE_RANK has a value of 0.

ROOT_RANK This is the rank for a DODAG root. ROOT_RANK has a value of MinHopRankIncrease (as advertised by the DODAG root), such that DAGRank(ROOT_RANK) is 1.

INFINITE_RANK This is the constant maximum for the rank. INFINITE_RANK has a value of 0xFFFF.

RPL_DEFAULT_INSTANCE This is the RPLInstanceID that is used by this protocol by a node without any overriding policy. RPL_DEFAULT_INSTANCE has a value of 0.

DEFAULT_PATH_CONTROL_SIZE TBD (To be determined)

DEFAULT_DIO_INTERVAL_MIN TBD (To be determined)

DEFAULT_DIO_INTERVAL_DOUBLINGS TBD (To be determined)

DEFAULT_DIO_REDUNDANCY_CONSTANT TBD (To be determined)

DEFAULT_MIN_HOP_RANK_INCREASE TBD a power of two (To be determined)

DIO Timer One instance per DODAG that a node is a member of. Expiry triggers DIO message transmission. Trickle timer with variable interval in [0, DIOIntervalMin..2^DIOIntervalDoublings]. See Section 6.3.1

DAG Version Increment Timer Up to one instance per DODAG that the node is acting as DODAG root of. May not be supported in all implementations. Expiry triggers increment of DODAGVersionNumber, causing a new series of updated DIO message to be sent. Interval should be chosen appropriate to propagation time of DODAG and as appropriate to application requirements (e.g. response time vs. overhead).

DelayDAO Timer Up to one instance per DAO parent (the subset of DODAG parents chosen to receive destination advertisements) per DODAG. Expiry triggers sending of DAO message to the DAO parent. See Section 7.1.6

RemoveTimer Up to one instance per DAO entry per neighbor (i.e. those neighbors that have given DAO messages to this node as a DODAG parent) Expiry triggers a change in state for the DAO entry, setting up to do unreachable (No-Path) advertisements or immediately deallocating the DAO entry if there are no DAO parents. See Section 7.1.4.1.1.3

13. Manageability Considerations

The aim of this section is to give consideration to the manageability of RPL, and how RPL will be operated in LLN beyond the use of a MIB module. The scope of this section is to consider the following aspects of manageability: fault management, configuration, accounting and performance.

13.1. Control of Function and Policy

13.1.1. Initialization Mode

When a node is first powered up, it may either choose to stay silent and not send any multicast DIO message until it has joined a DODAG, or to immediately root a transient DODAG and start sending multicast DIO messages. A RPL implementation SHOULD allow configuring whether the node should stay silent or should start advertising DIO messages.

Furthermore, the implementation SHOULD to allow configuring whether or not the node should start sending an DIS message as an initial probe for nearby DODAGs, or should simply wait until it received DIO messages from other nodes that are part of existing DODAGs.

13.1.2. DIO Base option

RPL specifies a number of protocol parameters.

A RPL implementation SHOULD allow configuring the following routing protocol parameters, which are further described in Section 5.3:

DAGPreference
RPLInstanceID
DAGObjectiveCodePoint
DODAGID
Routing Information
Prefix Information
DIOIntervalDoublings
DIOIntervalMin
DIORedundancyConstant

DAG Root behavior: In some cases, a node may not want to permanently act as a DODAG root if it cannot join a grounded DODAG. For example a battery-operated node may not want to act as a DODAG root for a long period of time. Thus a RPL implementation MAY support the ability to configure whether or not a node could act as a DODAG root for a configured period of time.

DODAG Table Entry Suppression A RPL implementation SHOULD provide the ability to configure a timer after the expiration of which logical equivalent of the DODAG table that contains all the records about a DODAG is suppressed, to be invoked if the DODAG parent set becomes empty.

13.1.3. Trickle Timers

A RPL implementation makes use of trickle timer to govern the sending of DIO message. Such an algorithm is determined a by a set of configurable parameters that are then advertised by the DODAG root along the DODAG in DIO messages.

For each DODAG, a RPL implementation MUST allow for the monitoring of the following parameters, further described in Section 6.3.1:

I
T
C
I_min
I_doublings

A RPL implementation SHOULD provide a command (for example via API, CLI, or SNMP MIB) whereby any procedure that detects an inconsistency may cause the trickle timer to reset.

13.1.4. DAG Version Number Increment

A RPL implementation may allow by configuration at the DODAG root to refresh the DODAG states by updating the DODAGVersionNumber. A RPL implementation SHOULD allow configuring whether or not periodic or event triggered mechanism are used by the DODAG root to control DODAGVersionNumber change.

13.1.5. Destination Advertisement Timers

The following set of parameters of the DAO messages SHOULD be configurable:

- The DelayDAO timer
- The Remove timer

13.1.6. Policy Control

DAG discovery enables nodes to implement different policies for selecting their DODAG parents.

A RPL implementation SHOULD allow configuring the set of acceptable or preferred Objective Functions (OF) referenced by their Objective Codepoints (OCPs) for a node to join a DODAG, and what action should be taken if none of a node's candidate neighbors advertise one of the configured allowable Objective Functions.

A node in an LLN may learn routing information from different routing protocols including RPL. It is in this case desirable to control via

administrative preference which route should be favored. An implementation SHOULD allow for specifying an administrative preference for the routing protocol from which the route was learned.

13.1.7. Data Structures

Some RPL implementation may limit the size of the candidate neighbor list in order to bound the memory usage, in which case some otherwise

viable candidate neighbors may not be considered and simply dropped from the candidate neighbor list.

A RPL implementation MAY provide an indicator on the size of the candidate neighbor list.

13.2. Information and Data Models

The information and data models necessary for the operation of RPL will be defined in a separate document specifying the RPL SNMP MIB.

13.3. Liveness Detection and Monitoring

The aim of this section is to describe the various RPL mechanisms specified to monitor the protocol.

As specified in Section 3.1, an implementation is expected to maintain a set of data structures in support of DODAG discovery:

- The candidate neighbors data structure
- For each DODAG:
  - A set of DODAG parents

13.3.1. Candidate Neighbor Data Structure

A node in the candidate neighbor list is a node discovered by the some means and qualified to potentially become of neighbor or a sibling (with high enough local confidence). A RPL implementation SHOULD provide a way monitor the candidate neighbors list with some metric reflecting local confidence (the degree of stability of the neighbors) measured by some metrics.

A RPL implementation MAY provide a counter reporting the number of times a candidate neighbor has been ignored, should the number of candidate neighbors exceeds the maximum authorized value.

13.3.2. Directed Acyclic Graph (DAG) Table

For each DAG, a RPL implementation is expected to keep track of the following DODAG table values:

- DODAGID
- DAGObjectiveCodePoint

- A set of prefixes offered upwards along the DODAG
- A set of DODAG Parents
- timer to govern the sending of DIO messages for the DODAG
- DODAGVersionNumber

The set of DODAG parents structure is itself a table with the following entries:

- A reference to the neighboring device which is the DAG parent
- A record of most recent information taken from the DAG Information Object last processed from the DODAG Parent
- A flag reporting if the Parent is a DAO Parent as described in Section 7

13.3.3. Routing Table

For each route provisioned by RPL operation, a RPL implementation MUST keep track of the following:

- Routing Information (prefix, prefix length, ...)

- Lifetime Timer
- Next Hop
- Next Hop Interface
- Flag indicating that the route was provisioned from one of:
  - Unicast DAO message
  - DIO message
  - Multicast DAO message

13.3.4. Other RPL Monitoring Parameters

A RPL implementation SHOULD provide a counter reporting the number of a times the node has detected an inconsistency with respect to a DODAG parent, e.g. if the DODAGID has changed.

A RPL implementation MAY log the reception of a malformed DIO message along with the neighbor identification if avialable.

13.3.5. RPL Trickle Timers

A RPL implementation operating on a DODAG root MUST allow for the configuration of the following trickle parameters:

- The DIOIntervalMin expressed in ms
- The DIOIntervalDoublings
- The DIORedundancyConstant

A RPL implementation MAY provide a counter reporting the number of times an inconsistency (and thus the trickle timer has been reset).

13.4. Verifying Correct Operation

This section has to be completed in further revision of this document to list potential Operations and Management (OAM) tools that could be used for verifying the correct operation of RPL.

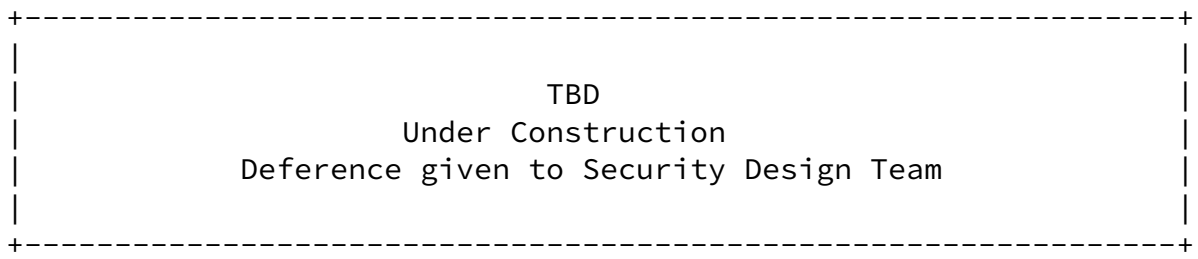
13.5. Requirements on Other Protocols and Functional Components

RPL does not have any impact on the operation of existing protocols.

13.6. Impact on Network Operation

To be completed.

14. Security Considerations



14.1. Overview

From a security perspective, RPL networks are no different from any other network. They are vulnerable to passive eavesdropping attacks and potentially even active tampering when physical access to a wire is not required to participate in communications. The very nature of ad hoc networks and their cost objectives impose additional security

constraints, which perhaps make these networks the most difficult environments to secure. Devices are low-cost and have limited capabilities in terms of computing power, available storage, and power drain; and it cannot always be assumed they have neither a trusted computing base nor a high-quality random number generator aboard. Communications cannot rely on the online availability of a fixed infrastructure and might involve short-term relationships between devices that may never have communicated before. These constraints might severely limit the choice of cryptographic algorithms and protocols and influence the design of the security architecture because the establishment and maintenance of trust relationships between devices need to be addressed with care. In

addition, battery lifetime and cost constraints put severe limits on the security overhead these networks can tolerate, something that is of far less concern with higher bandwidth networks. Most of these security architectural elements can be implemented at higher layers and may, therefore, be considered to be outside the scope of this standard. Special care, however, needs to be exercised with respect to interfaces to these higher layers.

The security mechanisms in this standard are based on symmetric-key and public-key cryptography and use keys that are to be provided by higher layer processes. The establishment and maintenance of these keys are outside the scope of this standard. The mechanisms assume a secure implementation of cryptographic operations and secure and authentic storage of keying material.

The security mechanisms specified provide particular combinations of the following security services:

Data confidentiality: Assurance that transmitted information is only disclosed to parties for which it is intended.

Data authenticity: Assurance of the source of transmitted information (and, hereby, that information was not modified in transit).

Replay protection: Assurance that a duplicate of transmitted information is detected.

Timeliness (delay protection): Assurance that transmitted information was received in a timely manner.

The actual protection provided can be adapted on a per-packet basis and allows for varying levels of data authenticity (to minimize security overhead in transmitted packets where required) and for optional data confidentiality. When nontrivial protection is required, replay protection is always provided.

Replay protection is provided via the use of a non-repeating value (nonce) in the packet protection process and storage of some status information for each originating device on the receiving device, which allows detection of whether this particular nonce value was used previously by the originating device. In addition, so-called

delay protection is provided amongst those devices that have a loosely synchronized clock on board. The acceptable time delay can be adapted on a per-packet basis and allows for varying latencies (to facilitate longer latencies in packets transmitted over a multi-hop communication path).

Cryptographic protection may use a key shared between two peer devices (link key) or a key shared among a group of devices (group key), thus allowing some flexibility and application-specific tradeoffs between key storage and key maintenance costs versus the cryptographic protection provided. If a group key is used for peer-to-peer communication, protection is provided only against outsider devices and not against potential malicious devices in the key-sharing group.

Data authenticity may be provided using symmetric-key based or public-key based techniques. With public-key based techniques (via signatures), one corroborates evidence as to the unique originator of transmitted information, whereas with symmetric-key based techniques data authenticity is only provided relative to devices in a key-sharing group. Thus, public-key based authentication may be useful in scenarios that require a more fine-grained authentication than can be provided with symmetric-key based authentication techniques alone, such as with group communications (broadcast, multicast), or in scenarios that require non-repudiation.

14.2. Functional Description of Packet Protection

14.2.1. Transmission of Outgoing Packets

This section describes the transmission of secured RPL control packets. Give an outgoing RPL control packet and required security protection, this section describes how RPL generates the secured packet to transmit. It describes the order of cryptographic operations to provide the required protection.

A RPL node MUST set the security section in the RPL packet to describes the required protection level.

The Counter field of the security header MUST be an increment of the last Counter field transmitted.

If the RPL packet is not a response to a Consistency Check message,

the node MAY set the Counter Compression field of the security option. If the packet is a response to a Consistency Check message, the node MUST clear the Counter Compression field.

A node sets the Key Identifier Mode (KIM) of the packet based on its understanding of what keys destinations have.

A node MUST replaced the original packet payload with that payload encrypted using the security protection, key, and nonce specified in the security section.

14.2.2. Reception of Incoming Packets

This section describes the reception of a secured RPL packet. Given an incoming RPL packet, this section describes now RPL generates an unencrypted version of the packet and validates its integrity.

The receiver uses the security control field of the security section to determine what processing to do. If the described level of security does not meet locally maintained security policies, a node MAY discard the packet without further processing. These policies can include security levels, keys used, or source identifiers.

Using a nonce derived from the Counter field and other information (as described in Section Figure 21), the receiver checks the integrity of the packet by comparing the received MAC with the computed MAC. If this integrity check does not pass, a node MUST discard the packet.

RPL uses the key information described in a RPL message to decrypt its contents as necessary. Once a message has passed its integrity checks and been successfully decrypted, the node can update its local security information, such as the source's expected counter value for counter compression. A node MUST NOT update security information on receipt of a message that fails security policy checks, integrity checks, or decryption.

14.2.3. Cryptographic Mode of Operation

The cryptographic mode of operation used is based on the CCM mode of operation specified with [TBDREF] and the block-cipher AES-128 [TBDREF]. This mode of operation is widely supported by existing implementations and coincides with the CCM* mode of operation specified with [TBDREF].

14.2.3.1. Nonce

The so-called nonce is constructed as follows:

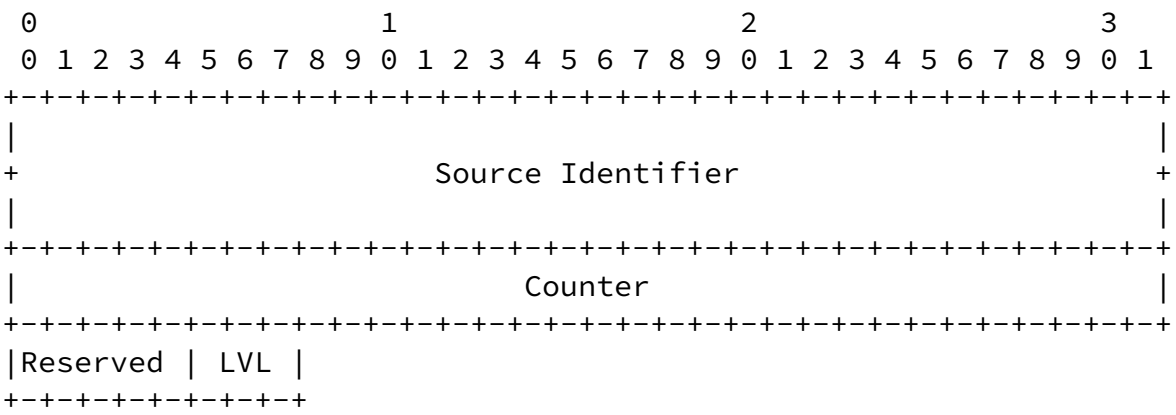


Figure 21: CCM* Nonce

Source Identifier: 8 bytes. Source Identifier is set to the logical identifier of the originator of the protected packet.

Counter: 4 bytes. Counter is set to the (uncompressed) value of the corresponding field in the Security option of the RPL control message.

Security Level (LVL): 3 bits. Security Level is set to the value of the corresponding field in the Security option of the RPL control message.

Unassigned bits of the nonce are reserved. They MUST be set to zero when constructing the nonce.

All fields of the nonce shall be represented is most-significant-octet and most-significant-bit first order.

14.3. Protecting RPL ICMPv6 messages

For a RPL ICMPv6 message, the entire packet is within the scope of RPL security. The message authentication code is calculated over the entire IPv6 packet. This calculation is done before any compression

that lower layers may apply. The IPv6 and ICMPv6 headers are never encrypted. The body of the RPL ICMPv6 message MAY be encrypted, starting from the first byte after the security information and continuing to the end of the packet.

14.4. Security State Machine

A DAG root starting a DODAG sets the RPL routing security policy for the entire DODAG.

A member of a secure DODAG MUST conform to the policy set by the DAG root. When starting a secure DODAG, the DAG root will send secure DIO messages. A node attempting to join the DODAG will send a secure Authentication Request (AREQ) to the DAG root. Nodes that are not authenticated in a secure DODAG will be unable to generate properly constructed secured RPL packets. These nodes are in state "unauthenticated". A member of a secure DODAG MUST forward an AREQ packet to the DAG root, and MUST NOT forward any other type of packet from an unauthenticated node.

The DAG root may choose to respond to the AREQ with an ARSP packet. This packet will provide the authenticating node with the cryptographic materials necessary to participate in RPL routing. Some authentication flows may involve the exchange of more than one AREQ or ARSP packets.

The simplest authentication flow will involve the use of a single pre-installed network-wide authentication key. The installation of this key is out of scope of this document. The authenticating node will use the pre-installed key to calculate a MIC for the AREQ packet. The DODAG root will verify the authenticity of the authenticating node using the same key. The DODAG root, having previously chosen a single random instance-wide shared key, will send this key, encrypted and authenticated with the pre-installed key, in the ARSP packet. The authenticating node, decoding this packet with the pre-installed key, will verify the authenticity of the DODAG root.

It is assumed that additional authentication and key exchange mechanisms will be included in future drafts of the document.

Periodic key updates will use the secure KU packet code. The responsibility for initiating key update will reside with the DODAG root, and is out of scope of this document.

15. IANA Considerations

15.1. RPL Control Message

The RPL Control Message is an ICMP information message type that is to be used carry DODAG Information Objects, DODAG Information Solicitations, and Destination Advertisement Objects in support of

RPL operation.

IANA has defined an ICMPv6 Type Number Registry. The suggested type value for the RPL Control Message is 155, to be confirmed by IANA.

15.2. New Registry for RPL Control Codes

IANA is requested to create a registry, RPL Control Codes, for the Code field of the ICMPv6 RPL Control Message.

New codes may be allocated only by an IETF Consensus action. Each code should be tracked with the following qualities:

- Code
- Description
- Defining RFC

Three codes are currently defined:

Code	Description	Reference
0x00	DODAG Information Solicitation	This document
0x01	DODAG Information Object	This document

0x02	Destination Advertisement Object	This document
0x80	Secure DODAG Information Solicitation	This document
0x81	Secure DODAG Information Object	This document
0x82	Secure Destination Advertisement Object	This document
0x83	Secure Destination Advertisement Object Acknowledgment	This document

RPL Control Codes

15.3. New Registry for the Mode of Operation (MOP) DIO Control Field

IANA is requested to create a registry for the Mode of Operation (MOP) DIO Control Field, which is contained in the DIO Base.

New fields may be allocated only by an IETF Consensus action. Each

field should be tracked with the following qualities:

- Mode of Operation
- Capability description
- Defining RFC

Two values are currently defined:

MOP	Description	Reference
00	Non-Storing mode of operation	This document
01	Storing mode of operation	This document

DIO Base Flags

15.4. RPL Control Message Option

IANA is requested to create a registry for the RPL Control Message Options

Value	Meaning	Reference
0	Pad1	This document
1	PadN	This document
2	DAG Metric Container	This Document
3	Routing Information	This Document
4	DAG Timer Configuration	This Document
5	RPL Target	This Document
6	Transit Information	This Document
7	Solicited Information	This Document
8	Prefix Information	This Document

RPL Control Message Options

16. Acknowledgements

The authors would like to acknowledge the review, feedback, and comments from Roger Alexander, Emmanuel Baccelli, Dominique Barthel, Yusuf Bashir, Phoebus Chen, Mathilde Durvy, Manhar Goindi, Mukul Goyal, Anders Jagd, JeongGil (John) Ko, Quentin Lampin, Jerry Martocci, Matteo Paris, Alexandru Petrescu, Joseph Reddy, and Don

Sturek.

The authors would like to acknowledge the guidance and input provided by the ROLL Chairs, David Culler and JP Vasseur.

The authors would like to acknowledge prior contributions of Robert Assimiti, Mischa Dohler, Julien Abeille, Ryuji Wakikawa, Teco Boot, Patrick Wetterwald, Bryan Mclaughlin, Carlos J. Bernardos, Thomas Watteyne, Zach Shelby, Caroline Bontoux, Marco Molteni, Billy Moon, and Arsalan Tavakoli, which have provided useful design considerations to RPL.

17. Contributors

RPL is the result of the contribution of the following members of the RPL Author Team, including the editors, and additional contributors as listed below:

JP Vasseur
Cisco Systems, Inc
11, Rue Camille Desmoulins
Issy Les Moulineaux, 92782
France

Email: jpv@cisco.com

Thomas Heide Clausen
LIX, Ecole Polytechnique, France

Phone: +33 6 6058 9349
EMail: T.Clausen@computer.org
URI: http://www.ThomasClausen.org/

Philip Levis
Stanford University
358 Gates Hall, Stanford University
Stanford, CA 94305-9030
USA

Email: pal@cs.stanford.edu

Richard Kelsey
Ember Corporation
Boston, MA

USA

Phone: +1 617 951 1225
Email: kelsey@ember.com

Jonathan W. Hui

Arch Rock Corporation
501 2nd St. Ste. 410
San Francisco, CA 94107
USA

Email: jhui@archrock.com

Kris Pister
Dust Networks
30695 Huntwood Ave.
Hayward,   94544
USA

Email: kpister@dustnetworks.com

Anders Brandt
Sigma Designs
Emdrupvej 26A, 1.
Copenhagen, DK-2100
Denmark

Email: abr@sdesigns.dk

Stephen Dawson-Haggerty
UC Berkeley
Soda Hall, UC Berkeley
Berkeley, CA 94720
USA

Email: stevedh@cs.berkeley.edu

18. References

18.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

18.2. Informative References

[I-D.hui-6man-rpl-option]
Hui, J. and J. Vasseur, "RPL Option for Carrying RPL Information in Data-Plane Datagrams", draft-hui-6man-rpl-option-00 (work in progress), March 2010.

[I-D.hui-6man-rpl-routing-header]
Hui, J., Vasseur, J., and D. Culler, "A Source Routing Header for RPL", draft-hui-6man-rpl-routing-header-00 (work in progress), May 2010.

[I-D.ietf-bfd-base]
Katz, D. and D. Ward, "Bidirectional Forwarding Detection", draft-ietf-bfd-base-11 (work in progress), January 2010.

[I-D.ietf-manet-nhdp]
Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", draft-ietf-manet-nhdp-12 (work in progress), March 2010.

[I-D.ietf-roll-building-routing-reqs]
Martocci, J., Riou, N., Mil, P., and W. Vermeylen, "Building Automation Routing Requirements in Low Power and Lossy Networks", draft-ietf-roll-building-routing-reqs-09 (work in progress), January 2010.

[I-D.ietf-roll-of0]
Thubert, P., "RPL Objective Function 0", draft-ietf-roll-of0-01 (work in progress), February 2010.

[I-D.ietf-roll-routing-metrics]
Vasseur, J., Kim, M., Networks, D., and H. Chong, "Routing Metrics used for Path Calculation in Low Power and Lossy Networks", draft-ietf-roll-routing-metrics-06 (work in progress), April 2010.

[I-D.ietf-roll-terminology]
Vasseur, J., "Terminology in Low power And Lossy Networks", draft-ietf-roll-terminology-03 (work in progress), March 2010.

[I-D.ietf-roll-trickle]
Levis, P., Clausen, T., Hui, J., and J. Ko, "The Trickle Algorithm", draft-ietf-roll-trickle-01 (work in progress), April 2010.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4101] Rescorla, E. and IAB, "Writing Protocol Models", RFC 4101, June 2005.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, June 2007.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to

Intermediate Systems (IS-ISs)", RFC 5120, February 2008.

[RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.

Appendix A. Requirements

A.1. Protocol Properties Overview

RPL demonstrates the following properties, consistent with the requirements specified by the application-specific requirements documents.

A.1.1. IPv6 Architecture

RPL is strictly compliant with layered IPv6 architecture.

Further, RPL is designed with consideration to the practical support and implementation of IPv6 architecture on devices which may operate under severe resource constraints, including but not limited to memory, processing power, energy, and communication. The RPL design does not presume high quality reliable links, and operates over lossy links (usually low bandwidth with low packet delivery success rate).

A.1.2. Typical LLN Traffic Patterns

Multipoint-to-Point (MP2P) and Point-to-multipoint (P2MP) traffic flows from nodes within the LLN from and to egress points are very common in LLNs. Low power and lossy network Border Router (LBR) nodes may typically be at the root of such flows, although such flows are not exclusively rooted at LBRs as determined on an application-specific basis. In particular, several applications such as building

or home automation do require P2P (Point-to-Point) communication.

As required by the aforementioned routing requirements documents, RPL supports the installation of multiple paths. The use of multiple paths include sending duplicated traffic along diverse paths, as well as to support advanced features such as Class of Service (CoS) based routing, or simple load balancing among a set of paths (which could be useful for the LLN to spread traffic load and avoid fast energy depletion on some, e.g. battery powered, nodes). Conceptually, multiple instances of RPL can be used to send traffic along different topology instances, the construction of which is governed by different Objective Functions (OF). Details of RPL operation in

support of multiple instances are beyond the scope of the present specification.

A.1.3. Constraint Based Routing

The RPL design supports constraint based routing, based on a set of routing metrics and constraints. The routing metrics and constraints for links and nodes with capabilities supported by RPL are specified in a companion document to this specification, [I-D.ietf-roll-routing-metrics]. RPL signals the metrics, constraints, and related Objective Functions (OFs) in use in a particular implementation by means of an Objective Code Point (OCP). Both the routing metrics, constraints, and the OF help determine the construction of the Directed Acyclic Graphs (DAG) using a distributed path computation algorithm.

A.2. Deferred Requirements

NOTE: RPL is still a work in progress. At this time there remain several unsatisfied application requirements, but these are to be addressed as RPL is further specified.

Appendix B. Outstanding Issues

This section enumerates some outstanding issues that are to be addressed in future revisions of the RPL specification.

B.1. Additional Support for P2P Routing

In some situations the baseline mechanism to support arbitrary P2P traffic, by flowing upwards along the DODAG until a common ancestor is reached and then flowing down, may not be suitable for all application scenarios. A related scenario may occur when the down paths setup along the DODAG by the destination advertisement mechanism are not the most desirable downward paths for the specific application scenario (in part because the DODAG links may not be symmetric). It may be desired to support within RPL the discovery and installation of more direct routes 'across' the DAG. Such mechanisms need to be investigated.

B.2. Address / Header Compression

In order to minimize overhead within the LLN it is desirable to perform some sort of address and/or header compression, perhaps via labels, addresses aggregation, or some other means. This is still under investigation.

B.3. Managing Multiple Instances

A network may run multiple instances of RPL concurrently. Such a network will require methods for assigning and otherwise managing RPLInstanceIDs. This will likely be addressed in a separate document.

Authors' Addresses

Tim Winter (editor)

Email: wintert@acm.org

Pascal Thubert (editor)
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

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

RPL Author Team
IETF ROLL WG

Email: rpl-authors@external.cisco.com