ROLL
Internet-Draft
Intended status: Standards Track
Expires: September 18, 2021

R. Jadhav, Ed.

P. Thubert
Cisco
M. Richardson
Sandelman Software Works
R. Sahoo
Juniper

March 17, 2021

RPL Capabilities
draft-ietf-roll-capabilities-08

Abstract

This draft enables the discovery, advertisement and query of capabilities for RPL nodes.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RPL [RFC6550] specifies a proactive distance-vector based routing scheme. The protocol creates a DAG-like structure which operates with a given "Mode of Operation" (MOP) determining the minimal and mandatory set of primitives to be supported by all the participating nodes.

This document adds a notion of capabilities, through which a node in the network could inform its peers about its additional capabilities. Using capabilities, a node could determine whether the target node supports the required feature before utilizing the feature. This document highlights the differences between capabilities and Mode of Operation and explains the necessity for the former.

1.1. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

MOP: Mode of Operation. Identifies the MOP of the RPL Instance as administratively provisioned at and distributed by the DODAG root.

MOPex: Extended MOP: As defined in [I-D.ietf-roll-mopex].

Capabilities: Additional features or capabilities that are supported by the node.

Cap: Abbreviated term used for Capability.

Caps: Abbreviated term used for Capabilities.

DAO: DODAG Advertisement Object. A RPL (pronounced ripple) message used to advertise the target information in order to establish routing adjacencies.

DIO: DODAG Information Object. A RPL message initiated by the root and is used to advertise the network configuration information.

Current parent: Parent 6LR node before switching to the new path.

NPDAO: No-Path DAO. A DAO message that contains a Transit Information Option with lifetime equal to 0.

Upstream path/direction: Path or direction from the node to the Root in a DAG.

Downstream path/direction: Path or direction to the node from the Root in a DAG.

This document uses terminology described in [RFC6550]. For the sake of readability all the known relevant terms are repeated in this section.

1.2. What are Capabilities?

Currently, RPL specification does not have a mechanism whereby a node can signal the set of features that are available on its end. Such a mechanism could help the root to advertise its capabilities and in response also determine some advanced information about the capabilities of the joining nodes. This document defines Capabilities and corresponding messaging handshakes that could be supported by the nodes. Capabilities are embedded as an RPL Control Message Option as defined in Section 6.7 of [RFC6550].

2. Requirements for this document

Following are the requirements considered for this documents:

REQ1: Optional capabilities handshake. Capabilities are features, possibly optional, which could be indicated between the nodes and the root within an RPL Instance.

REQ2: Capabilities handshake could be optionally added with existing MOPs. Capabilities, being optional, could be put to use with existing MOPs. Capabilities and MOP-extension are mutually independent i.e. a DIO can have a capabilities option, MOP-extension option, or both in the same message.

REQ3: Capabilities could be explicitly queried.

2.1. How are Capabilities different from existing RPL primitives?

The Mode of Operation (MOP) field in RPL mandates the operational requirement for the nodes joining as routers. MOP and DIO Configuration Option is strictly controlled by the Root node in RPL. Intermediate 6LRs cannot modify these fields. Also, the MOP never changes for the lifetime of the RPL Instance. Changes in DIO Configuration Option are possible but are rare. Capabilities, on the other hand, might change more dynamically.

RPL DIO message also carries routing metrics and constraints as specified in [RFC6551]. Metrics and constraints are used in addition to an objective function to determine a node's rank calculation. A router may use capabilities carried in DIO messages as additional

metrics/constraints. However, capabilities have a larger scope and might be carried in messages other than DIO and can flow in either direction (upstream and downstream).

3. Capabilities

Handling of Capabilities MUST be supported if the network uses MOPex [I-D.ietf-roll-mopex].

Note that capabilities and MOPex are mutually exclusive and it is possible for an implementation to support either or both of the options.

3.1. Capability Control Message Option

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   Type = TODO | Option Length | Capabilities TLVs
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Capabilities Option

Multiple capabilities can be sent in the same message. The length field allows the message parser to skip the capability TLV parsing.

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   CapType     |      Len      |J|I|C|  Flags  |     ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Capabilities TLV

Every capability is identified by its type and it may have an optional Capability Info. Note that a given capability may or may not be disseminated with additional information depending on the scope of the capability indicated by the I bit.

Len: 8-bit unsigned integer, representing the length in octets of the TLV, not including the CapType, Length and Flags fields.

J = Join only as leaf if capability not understood.

I = Ignore the message if this capability is not understood.

C = Flag indicating that the capability MUST be copied in the downstream message.

3.2. Capabilities Handshake

The root node can advertise the set of capabilities it supports in the DIO message. A node can take advantage of the knowledge that the root supports a particular capability. Similarly, a node can advertise its capabilities in the DAO message using the capability control message option defined in this document. Capabilities advertised by non-root nodes is strictly a subset of the capabilities advertised by the root.

In storing MOP, the DAO message from the 6LR can contain multiple target options because of the DAO-Aggregation. The targets of the capabilities option are indicated by one or more Target options that precede the Capabilities Option. This handling is similar to the Transit Information Option as supported in Section 6.7.8. of [RFC6550].

4. Querying Capabilities

Nodes may be interested in knowing the capabilities of another node before taking an action. For example, consider [I-D.ietf-roll-dao-projection], in which the Root may want to know the capabilities of the nodes along a network segment before it initiates a projected DAO to install the routes along that segment.

Caps can be carried in existing RPL Control messages as Control Options, however, Caps can also be queried explicitly. This section provides a way for a node to query the capability set of another node. The capability query and subsequent response messages are directly addressed between the two peers.

4.1. Capability Query (CAPQ)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RPLInstanceID |       Flags    |   reserved    | CAPQSequence  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option(s)...
+-+-+-+-+-+-+-+-+-+
```

Figure 3: CAPQ base object

CAPQSequence: One byte, Sequence number sent by the CAPQ sender and reflected back by the responder in the CAPS message.

Flags: One byte, set to zero by sender, ignored by receiver.

reserved: One byte, set to zero by sender, ignored by receiver.

The CAPQ base object may be followed by one or more options. The Capability Type List Control Option (see Figure 4) is used to carry a set of capability types to query about.

If the sender does not send a Capability Type List Control Option, this indicates that the node intends to query the Capability Type List supported by the target node.

4.1.1. Capability Type List Control Option

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Type = TODO | Option Length |  CapType1     |  CapType2     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  CapType3     |  .....
     +-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Capability Type List Control Option

4.1.2. Secure CAPQ

A Secure CAPQ message follows the format in [RFC6550] Figure 7, where the base message format is the CAPQ message shown in Figure 3.

4.1.3. Base rules for CAPQ handling

A CAPQ message may get dropped or lost in the transit. The sender of CAPQ MAY retry the CAPQ message after some delay. The delay SHOULD NOT be less than 1 second.

4.2. Capability Set Response (CAPS)

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | RPLInstanceID |    Flags      |  Reserved     |  CAPQSequence |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Option(s)...
     +-+-+-+-+-+-+-+-+-+
```

Figure 5: CAPS base object

Flags: One byte, set to zero by sender, ignored by receiver.

reserved: One byte, set to zero by sender, ignored by receiver.

CAPQSequence: One byte, Sequence number copied from CAPQSequence received in the CAPQ message.

CAPS message SHOULD contain the capability set Figure 1 queried by the CAPQ sender. If the target node does not support a subset of the queried capabilities then the Capability Type List with the unsupported cap-types SHOULD be sent back indicating the queried capabilities not-supported by the target node. For example, check Appendix A.3

If the CAPQ message does not contain any Capability Type List option then the receiver MUST respond with the cap types it supports using a Capability Type List Option (see Figure 4).

If the capability set cannot be transmitted in a single message (for e.g., because of MTU limitations) then multiple CAPS messages could be used. All the CAPS messages MUST use the same CAPQSequence number copied from the corresponding CAPQ message.

4.2.1. Secure CAPS

A Secure CAPS message follows the format in [RFC6550] Figure 7, where the base message format is the CAPS message shown in Figure 5.

5. Guidelines for defining new capabilities

This section provides guidelines/recommendations towards defining new capabilities. Note that the capabilities might be carried as part of the multicast messaging such as DIO and hence the set should be used sparingly.

5.1. Handling Capability flags

A node MUST drop or discard the message with an unknown capability with the 'D' flag set. The message MUST be discarded silently.

The 'J' (join) flag can be set in context to a capability either by a 6LR or the root. The 'J' flag indicates that if the capability is not supported by a node then it can join the instance only as a 6LN (or do not join as 6LR).

The 'C' (copy) flag is set by the node indicating that the capabilities MUST be copied downstream by the node even if the node does not understand the capability.

5.1.1. Rules to handle capabilities flag

How should a node react to capability it does
not support before joining the Instance?
On receiving a capability it does not support, the node MUST check the 'J' flag of the capability before joining the Instance. If the 'J' flag is set then it can only join as a 6LN.

How should a node react to capability it does not support after joining the Instance?
If the node is operating as 6LR and subsequently it receives a capability from its preferred parent which it does not understand with 'J' flag set, then the node has to switch itself to 6LN mode. During switching, the node needs to inform its downstream peers of its changed status by sending a DIO with infinite rank as mentioned in RFC6550. Alternatively, a node may decide to switch to another parent with compatible and known capabilities.

When to use and when not to use Capabilities?

Capabilities are used to indicate a feature that is supported by the node. Capabilities are not meant for configuration management for e.g., setting a threshold.

6. Node Capabilities

6.1. Capability Indicators

Capability Indicators indicate the capabilities supported by the node in the form of simple flags. Capabilities that do not need additional information to be specified can make use of these flags to indicate their support.

6.1.1. Format of Capability Indicators

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CapType=0x01  |      Len       |J|I|C|   Flags |T|..Indicators..
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Capability Indicators TLV

Flags: LRs MUST set it to 0. I bit will always be set to 0.

T flag (Bit 1): Indicates whether the node supports 6LoRH [RFC8138].

6.2. Routing Resource Capability

Storing Mode of Operation requires each intermediate router in the LLN to maintain routing state information in the routing table. LLN routers typically operate with constraints on processing power, memory, and energy (battery power). Memory limits the size of the routing state an LR and BR can maintain. When the routing table of an LR or BR is full, it will either reject the new DAO messages received or will use some replacement policy to remove a routing entry and add the new one. Rejection of DAO messages will lead to an increase in DAO message transmission that impacts the energy and network convergence time. Routing state replacement leads to downward path downtime.

One possible way to solve problems due to routing table size constraint is to use this information to add neighbors to the DAO parent set. Routing resource capability can be used by LR and BR to advertise their current routing table usage details in the network. LR or LNs in LLN can use this information in the selection of the DAO parent set. PCE can use this information to select intermediate routers for the projected routes. Routing Resource is an optional capability.

Routing resource capabablity sent in DIO message has link local scope and it MUST NOT be forwarded. The 'C' bit of this capability MUST be set to 0.

6.2.1. Format of Routing Resource Capability

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | CapType=0x02  |    Len=3      |J|I|C|  Flags  |  Reserved     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        Total Capacity         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Routing Resource Capability TLV

Type: 0x02.

Flags: I bit MUST be set to 0. C bit MUST be set to 0.

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

Resvd: 8-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Total Capacity: 16 bit unsigned integer representing the routing table size.

7. Acknowledgements

Thanks to Georgios Papadopoulos, Li Zhao for early review and feedback.

8. IANA Considerations

IANA is requested to allocate new codes for the CAPQ and CAPS messages from the RPL Control Codes registry.

Code	Description	Reference
TBD1	Capability Query	This document
TBD2	Capability Response	This document
TBD3	Secure Capability Query	This document
TBD4	Secure Capability Response	This document

New RPL Control Messages

The MSB of the codes allocated to "Secure" messages above should be set.

8.1. New option: Capabilities

New entry is required for supporting new Capabilities option and new Capability Type List Option in the "RPL Control Message Options" space [RFC6550].

Value	Meaning	Reference
TODO	Capability Option	This document
TODO	Capability Type List Option	This document

New options

8.2. Capability Sub-Type

IANA is requested to create a registry for the Capabilities Type as described in Figure 2 of this document. This registry should be located in TODO. New Capabilities types may be allocated only by an IETF review.

Value	Meaning	Reference
0x01	Capability Indicators	This document
0x02	Routing Resource Capability	This document

Type

8.3. New Registry for CAPQ Flags

IANA is requested to create a registry for the Capabilities flags as described in Section 4.1 of this document. This registry should be located in TODO. New Capabilities flags may be allocated only by an IETF review. Currently no flags are defined by this document. Each value is tracked with the following qualities:

- Flag
- Description
- Defining RFC

8.4. New Registry for Capabilities Flags

IANA is requested to create a registry for the Capabilities flags as described in Section 2.1 of this document. This registry should be located in TODO. New Capabilities flags may be allocated only by an IETF review. Currently no flags are defined by this document. Each value is tracked with the following qualities:

- Flag
- Description
- Defining RFC

8.5. New Registry for Capabilities Indicators

IANA is requested to create a registry for the Capabilities Indicators as described in Section 6.1 of this document. This registry should be located in TODO. New Capabilities indicators may be allocated only by an IETF review. Each value is tracked with the following qualities:

- Flag
- Description

o Defining RFC

9. Security Considerations

The options defined in this document are carried in the base message objects as defined in [RFC6550]. The RPL control message options are protected by the same security mechanisms that protect the base messages.

Capabilities flag can reveal that the node has been upgraded or is running a old feature set. This document assumes that the base messages that carry these options are protected by RPL security mechanisms and thus are not visible to a malicious node.

[TODO] implications of malicious attack involving setting the capability flags.

10. References

10.1. Normative References

[I-D.ietf-roll-mopex]
Jadhav, R., Thubert, P., and M. Richardson, "Mode of Operation extension", draft-ietf-roll-mopex-02 (work in progress), September 2020.

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

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

[RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <https://www.rfc-editor.org/info/rfc8138>.

10.2. Informative References

[I-D.ietf-roll-dao-projection]
Thubert, P., Jadhav, R., and M. Gillmore, "Root initiated routing state in RPL", draft-ietf-roll-dao-projection-16 (work in progress), January 2021.

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

Appendix A. Capability Handshake Example

A.1. Query supported Cap Types

```
                 Root                             6LR/6LN
                 |                                    |
                 |  CAPQ(seq=1, opts=nil)             |
                 |----------------------------------->|
                 |                                    |
                 |                                    |
                 |  CAPS(seq=1, opts={CapTypeList})   |
                 |<-----------------------------------|
                 |                                    |
```

Figure 8: Query supported Cap Types

CAPQ message with no CapTypeList Option results in the peer responding with a CAPS message with CapTypeList Option indicating all the capability set it supports.

A.2. Query specific Cap Set

```
                 Root                             6LR/6LN
                 |                                    |
                 |  CAPQ(seq=2,                       |
                 |   opts={CapTypeList=[Cap1, Cap2]})|
                 |----------------------------------->|
                 |                                    |
                 |                                    |
                 |  CAPS(seq=2,                       |
                 |       opts={Cap1=Cap1Value,        |
                 |             Cap2=Cap2Value})       |
                 |<-----------------------------------|
                 |                                    |
```

Figure 9: Query specific Cap Set

This flow indicates the case where the Root probes for specific Capabilities of the peer node and the peer node responds with the value of indicated Capability set.

A.3. CAPS with partial Cap Set

```
                 Root                             6LR/6LN
                 |                                   |
                 |  CAPQ(seq=3,                      |
                 |   opts={CapTypeList=[Cap1, Cap2,  |
                 |                      Cap3, Cap4]})|
                 |---------------------------------->|
                 |                                   |
                 |                                   |
                 |  CAPS(seq=3,                      |
                 |    opts={Cap2=Cap2Value,          |
                 |          Cap3=Cap3Value,          |
                 |          CapTypeList=[Cap1,Cap4]})|
                 |<----------------------------------|
                 |                                   |
```

Partial Capability Set handshake

Assume that Root queries for capabilities {Cap1, Cap2, Cap3, Cap4} from the peer node. However the peer node does not support or does not understand capability {cap1, cap4}. In this case the peer node will respond back with value of Cap2 and Cap3 (which it understands) and set the CapTypeList option with {Cap1, Cap4} type.

Authors' Addresses

Rahul Arvind Jadhav (editor)
Marathahalli
Bangalore, Karnataka 560037
India

Email: rahul.ietf@gmail.com

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
France

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

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Rabi Narayan Sahoo
Juniper

Email: rabinarayans0828@gmail.com