Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2015

J. Yi
T. Clausen
LIX, Ecole Polytechnique

July 4, 2014

Collection Tree Protocol for Lightweight On-demand Ad hoc Distance-vector Routing - Next Generation (LOADng-CT) draft-yi-loadngct-02

Abstract

This document describes the Collection Tree Protocol for Lightweight Ad hoc On-Demand - Next Generation (LOADng-CT) to build routes between a designated root and all the other routers in Mobile Ad Hoc NETworks (MANETs). The protocol is an extension of LOADng.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

Collection Tree Protocol for Ligthweight Ad hoc On-Demand - Next Generation (LOADng-CT) is an extension of LOADng protocol [I-D.clausen-lln-loadng] for use in discovering routes from between a designated root and all the other routers in Mobile Ad hoc NETworks (MANETs) by building a Collection Tree.

Compared to LOADng, which builds mainly point-to-point routes, LOADng-CT inherits the information base, message format, basic functions and main characteristics of LOADng, and is extended as follows:

- The root of the expected collection tree can initiate the collection tree building process, so that every LOADng-CT router in the MANETs can discover a route to the root.
- HELLO message and Link Set are introduced to discover the bi-directional neighbors, which guarantees the routes built are bi-directional.
- If required, LOADng-CT can also build the routes from the root to all the other routers in the network.

LOADng-CT is compatible with LOADng [I-D.clausen-lln-loadng], which means:

- LOADng-CT shares the same message format with LOADng.
- A router with LOADng implementation can join the collection tree initiated by a router with LOADng-CT implementation.
- As an extension of LOADng, a router with LOADng-CT implementation has all the functions of LOADng implicitly.

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 [RFC2119].

This document uses the terminology and notation defined in [I-D.clausen-lln-loadng]. Additionally, it defines the following terminology:

Collection Tree - A directed graph that all edges are oriented toward and terminate at one root node.

LOADng-CT Router - A router that implements this LOADng-CT protocol. A LOADng-CT router can be equipped with one or multiple distinct interfaces.

LOADng Router - A router that implements LOADng protocol [I-D.clausen-lln-loadng] without collection tree extension. A LOADng router can be equipped with one or multiple distinct interfaces.

Root - A LOADng-CT Router, which is the root node in the collection tree.

Sensor - A LOADng-CT Router, which is not root node in the collection tree.

Root-to-sensor route - A route from the Root to a sensor. In some documents, it is also called "downward route", or "point-to-multipoint route".

Sensor-to-root route - A route from a sensor to the Root. In some documents, it is also called "upward route", or "multipoint-to-point route".

Sensor-to-sensor route - A route from a sensor to another sensor. In some documents, it is also called "point-to-point route".

3. Applicability Statement

This protocol:

- Is a collection tree protocol for building sensor-to-root routes in MANETs.
- Enables LOADng-CT Routers in the MANETs to discover bi-directional route to the root.
- Can discover the routes on-demand, in case of route failure or participation of new routers, without rebuilding the collection tree.
- Inherits the main characteristics of LOADng, which includes: optimal flooding support, different address lenght support, layer-agnostic, etc.

- Is compatible with LOADng specification. A LOADng Router without this collection tree extension can also join the collection tree.

4. Protocol Overview and Functioning

A LOADng-CT router is able to:

- Initiate a collection tree building process by the Root.
- Identify the bi-directional links between its neighbors by HELLO message exchange.
- Discover routes using only bi-directional links to the root by joining the collection tree.
- Maintain the collection tree on-demand without rebuilding the tree.
- Build routes using only bi-directional links from the root to other routers, if required.

4.1. Overview

To participate in a MANET, a LOADng-CT Router MUST have at least one or more LOADng-CT interfaces. Each LOADng-CT Router MUST be configured with one or more interface addresses.

Each LOADng-CT router performs the following tasks:

- A LOADng-CT router that is expected to be the root of the collection tree initiates the collection tree building by transmitting a Route Request message with COLLECTION_TREE_TRIGGER flag (denoted RREQ_TRIGGER message).
- Upon receiving an RREQ_TRIGGER, the LOADng-CT Router records the address of the sending router interface (i.e., the neighbor, from which it received the RREQ_TRIGGER) in its Link Set, with the status HEARD. If this RREQ_TRIGGER has not been received before, it is retransmitted to its neighbors according to the flooding operation, specified for the network. A HELLO message is also transmitted carrying all the neighbors so that each router can be aware of its bi-directional neighbors. By such RREQ_TRIGGER and HELLO message exchange, every LOADng-CT Router can identify its bi-directional neighbors, marked as SYM.
- After initiating the RREQ_TRIGGER for 2 x NET_TRAVERSAL_TIME, the root generates an RREQ with COLLECTION_TREE_BUILD flag (denoted

RREQ_BUILD message). The RREQ_BUILD is flooded in the network according to the flooding operation. The routers only process and retransmit the RREQ_BUILD from it bi-directional neighbors. On receiving a valid RREQ_BUILD, a new routing tuple to the root is inserted into the Routing Set.

- o If the route from the root to the non-root LOADng-CT Router is desired, the router receiving the RREQ_BUILD MUST unicast an RREP to the root.

4.2. Information Base Overview

In addition to the Routing Set, Local Interface Set, Blacklisted Neighbor Set, Destination Address Set, Pending Acknowledgment Set described in [I-D.clausen-lln-loadng], a Link Set is introduced for LOADng-CT.

Link Set contains tuples, each representing a single link to a router's 1-hop neighbor.

4.3. Signaling Overview

In addition to the Route Request (RREQ), Route Reply (RREP), Route Reply Acknowledgment (RREP_ACK), Route Error (RERR) messages described in [I-D.clausen-lln-loadng], a LOADng-CT router generates and processes following messages:

Route Request with COLLECTION_TREE_TRIGGER flag (RREQ_TRIGGER) - Generated by the root of the collection tree to begin the collection tree building process. An RREQ_TRIGGER is flooded through the network, according to the flooding operation specified for the network.

Route Request with COLLECTION_TREE_BUILD flag (RREQ_BUILD) - Generated by the root of the collection tree and flooded through the network. The LOADng-CT routers receiving RREQ_BUILD can build the routes to the Root.

HELLO - Generated by each LOADng-CT router carrying its 1-hop neighbor information.

5. Parameters and Constants

In addition to the parameters and constants defined in [I-D.clausen-lln-loadng], a LOADng-CT Router uses the parameters and constants described in this section.

RREQ_MAX_JITTER - is the maximum jitter for RREQ message transmission.

HELLO_MIN_JITTER - is the minimum jitter for HELLO message transmission. HELLO_MIN_JITTER MUST be greater than 2 x RREQ_MAX_JITTER.

HELLO_MAX_JITTER - is the maximum jitter for HELLO message transmission.

L_HOLD_TIME - is the minimum time a Link Tuple SHOULD be kept in the Link Set after it was last refreshed.

RREP_REQUIRED - is the flag to define if an RREP message is required on receiving RREQ_BUILD message, to build route from sensor to root.

6. Information Bases

Each LOADng-CT Router maintains an Information Base, containing several information sets. These information sets are given so as to facilitate description of message generation, forwarding and processing rules. In particular, an implementation may chose any representation or structure for when maintaining this information. In addition to the information sets described in [I-D.clausen-lln-loadng], the Link Set is introduced for LOADng-CT extension.

6.1. Link Set

The Link Set records the link status to 1-hop neighbors. It consists of Link Tuples, each of which representing a single link:

(L_neighbor_iface_addr, L_HEARD_time, L_SYM_time, L_time)

NOTE: compared to NHDP link set, I eliminated L_quality, L_pending, L_lost fields.
Do we need them?

where:

L_neighbor_iface_addr - is a network addresses of the MANET interface of the 1-hop neighbor;

L_HEARD_time - is the time up to which the MANET interface of the 1-hop neighbor would be considered heard;

L_SYM_time - is the time up to which the link to the 1-hop neighbor would be considered symmetric;

L_time - specifies when this Tuple expires and MUST be removed.

The link can be in following status:

LOST - If L_HEARD_time and L_SYM_time is expired;

SYM - If L_SYM_time is not expired;

HEARD - If L_HEARD_time is not expired, and L_SYM_time is expired.

7. Protocol Message Content

The packet and message format used by this protocol follows the specification of [I-D.clausen-lln-loadng]. Three additional flags for RREQ and a new HELLO message are introduced for collection tree extension.

7.1. Route Request (RREQ) Message

The fields of RREQ message is defined in [I-D.clausen-lln-loadng]. In addition, LOADng-CT has has three flags:

RREQ.trigger is the COLLECTION_TREE_TRIGGER flag. It is a boolean flag. When set ('1'), it identifies the message is generated by the root of the collection tree to trigger the collection tree building process.

RREQ.build is the COLLECTION_TREE_BUILD flag. It is a boolean flag. When set ('1'), it identifies the message is generated by the root of the collection tree to build routes to the root.

RREQ.ct-rrep is the COLLECTION_TREE_RREP flag. It is a boolean flag. When set ('1'), it identifies the RREQ_BUILD message requires the receiving sensors send an RREP message to the root, to build root-to-sensor routes.

7.2. HELLO Message

A HELLO message has the following fields:

HELLO.addr-length is an unsigned integer field, encoding the length of the originator and destination addresses as follows:

HELLO.addr-length := the length of an address in octets - 1

HELLO.originator is an identifier of HELLO.addr-length + 1 octets, specifying the address for which this RREP was generated

HELLO.validity-time represents L_HOLD_TIME for the transmitting MANET interface.

HELLO.1hop-list represents a list of 1-hop neighbor. It consists of 1-hop tuples:

(H_1hop_address, H_1hop_status)

where:

H_1hop_address - is a network address of 1-hop neighbors from the Link Set of this MANET interface (i.e., L_neighbor_iface_addr).

H_1hop_status - is the associated 1-hop status (L_status) of H_1hop_address.

8. Route Request for Collection Tree Triggering (RREQ_TRIGGER)

The RREQ_TRIGGER is an RREQ message with COLLECTION_TREE_TRIGGER flag set ('1'). It is generated by a LOADng-CT root when a collection tree is desired or needs to be rebuilt. It will trigger the collection tree building process.

8.1. RREQ_TRIGGER Generation

A packet with RREQ_TRIGGER message is generated by a LOADng-CT router that wants to be a root of a collection tree, according to Section 7.1 with the following content:

- RREQ.addr-length set to the length of the address in octets -1;
- RREQ.metric-type set to the desired metric type;
- RREQ.route-metric := 0;
- RREQ.seq-num set to the next unused sequence number, maintained by this LOADng-CT Router;

- RREQ.hop-count := 0;
- RREQ.hop-limit := MAX_HOP_LIMIT;
- RREQ.originator := one address of the LOADng-CT Interface of the LOADng Router that generates the RREQ. If the LOADng-CT Router is generating RREQ on behalf of a host connected to this LOADng-CT Router, the source address of the data packet, generated by that host, is used;
- RREQ.destination := RREQ.originator;
- RREQ.trigger := true.

8.2. RREQ_TRIGGER Processing

On receiving an RREQ_TRIGGER message, a LOADng-CT Router MUST process the message according to this section:

1. A received RREQ_TRIGGER is invalid, and MUST be discarded without further processing, if any of the following conditions are true:
    - The address length specified by this message (i.e., MSG.addr-length + 1) differs from the length of the address(es) of this LOADng Router.
    - There is a tuple in the Routing Set where:
        - R_dest_addr = MSG.originator
        - R_seq_num >= MSG.seq-num
2. Add the heard neighbor to the Link Set:
    - Find the the Link Tuple (henceforth, matching Link Tuple) where:
        - L_neighbor_iface_addr = previous-hop
    - The matching Neighbor Tuple is updated with the following information. If there is no matching Neighbor Tuple found, a new Neighbor Tuple is created with:
        - L_neighbor_iface_addr := previous-hop;
        - L_HEARD_time := current_time + L_HOLD_TIME;

+ L_local_iface_addr := the interface address through which the RREQ_TRIGGER was received;

+ L_time := current_time + L_HOLD_TIME;

3. If the address contained in MSG.originator is an address of this LOADng Router, the message is processed according to Section 11.2 of [I-D.clausen-lln-loadng]. All the Routing Tuples created or updated MUST set the R_bidirectional to FALSE. Otherwise, the message is discarded without further processing.

4.

5. If hop-count is less than MAX_HOP_COUNT and hop-limit is greater than 0, the message is considered for forwarding according to Section 8.3

6. A HELLO message is scheduled according to Section 9.

8.3. RREQ_TRIGGER Forwarding

An RREQ_TRIGGER message considered for forwarding follows Section 12.3: RREQ Forwarding of [I-D.clausen-lln-loadng].

8.4. RREQ_TRIGGER Transmission

An RREQ_TRIGGER message is transmitted according to Section 12.4: RREQ Transmission of [I-D.clausen-lln-loadng].

9. HELLO message

HELLO messages are generated by a LOADng-CT Router after receiving RREQ_TRIGGER for a certain periode of time. (TODO: define the time here) HELLO message carries the router's 1-hop neighbor information. Each LOADng-CT Router that receives the HELLO messages are thus able to determine its bi-directional 1-hop neighbors.

9.1. HELLO Generation

A packet with HELLO message is generated after receiving a fresh RREQ_TRIGGER message, and MUST be jittered according to MIN_HELLO_JITTER and MAX_HELLO_JITTER, in which MIN_HELLO_JITTER MUST be greater than 2 x MAX_RREQ_JITTER. A new HELLO message is created with following content:

o HELLO.addr-length set to the length of the address in octets -1;

- HELLO.originator := the address of the LOADng-CT interface that generates the HELLO message;
- HELLO.validity-time := L_HOLD_TIME;
- HELLO.1hop-list includes all the 1-hop neighbors in Link Set, other than those from Link Tuples with L_status = LOST. Each 1hop tuple is set with following content:
  - H_1hop_address := L_neigbhor_iface_addr;
  - H_1hop_status := L_status.

9.2. HELLO Processing

On receiving a HELLO message, a LOADng-CT Router MUST process the message according to this section:

1. Find the 1-hop neighbor address in the HELLO message's 1-hop neighbor List (henceforth, matching Neighbor Address) where:
   - H_1hop_address = any address in Local Interface Set
2. If no matching Neighbor Address is found, the originator of the HELLO message MUST be blacklisted by creating a Blacklist tuple:
   - B_neighbor_address := HELLO.originator;
   - B_valid_time := current_time + B_HOLD_TIME.
3. For the matching Neighbor Address, find the Link Tuple in the LOADng-CT Router's Link Set (henceforth, matching Link Tuple) where:
   - L_neighbor_iface_addr = matching Neighbor Address

   If the matching Link Tuple exists, update the matching Link Tuple as:
   - L_SYS_time := current_time + L_HOLD_TIME;
   - L_time := current_time + L_HOLD_TIME

9.3. HELLO Forwarding

The HELLO message is not considered for forwarding.

9.4. HELLO Transmission

The initially generated HELLO messages are sent to all neighbor LOADng-CT Routers.

10. Route Request for Collection Tree Building (RREQ_BUILD)

The RREQ_BUILD is an RREQ message with COLLECTION_TREE_BUILD flag set ('1'). It is generated by a LOADng-CT root after 2 x NETWORK_TRAVERSAL_TIME of the RREQ_TRIGGER message. The sensors receiving the RREQ_BUILD messages will build the route to the root.

10.1. RREQ_BUILD Generation

A packet with RREQ_TRIGGER message is generated by the root, following the generation of RREQ_TRIGGER, according to Section 7.1 with the following content:

- RREQ.addr-length set to the length of the address in octets -1;
- RREQ.metric-type set to the desired metric type;
- RREQ.route-metric := 0;
- RREQ.seq-num set to the next unused sequence number, maintained by this LOADng-CT Router;
- RREQ.hop-count := 0;
- RREQ.originator := one address of the LOADng-CT Interface of the LOADng Router that generates the RREQ. If the LOADng-CT Router is generating RREQ on behalf of a host connected to this LOADng-CT Router, the source address of the data packet, generated by that host, is used;
- RREQ.destination := RREQ.originator;
- RREQ.build := true;
- If the root requires root-to-sensor routes, RREQ.ct-rrep := true.

10.2. RREQ_BUILD Processing

On receiving an RREQ_BUILD message, the root MUST discard it silently without further processing. A sensor MUST process the message according to this section:

1. If the message is invalid for processing, as defined in Section 11.1: Identifying Invalid RREQ or RREP Messages of [I-D.clausen-lln-loadng], the message MUST be discarded without further processing. The message is not considered for forwarding.

2. Find the Link Tuple in the Link Set where:

   * L_neighbor_iface_address = previous-hop; AND

   * L_status = SYM

   If no tuple is found, the message MUST be discarded without further processing. The message is not considered for forwarding.

3. The message is processed according to Section 11.2: RREQ and RREP Message Processing of [I-D.clausen-lln-loadng]. All the Route Tuples created or updated during processing MUST set R_bidirectional to TRUE.

4. If hop-count is less than MAX_HOP_COUNT and hop-limit is greater than 0, the message is considered for forwarding according to Section 8.3

5. If the downward route from the root to the sensor is required (i.e., RREQ.ct-rrep is set, or REPLY_REQUIRED is true), an RREP is generated and unicast to the root.

10.3. RREQ_BUILD Forwarding

An RREQ_BUILD message considered for forwarding follows Section 12.3 RREQ Forwarding of [I-D.clausen-lln-loadng].

10.4. RREQ_BUILD Transmission

An RREQ_BUILD message is transmitted according to Section 12.4 RREQ Transmission of [I-D.clausen-lln-loadng].

11. Collection Tree Maintenance and Local Repair

The tuples in the Routing Set are maintained by mechanisms specified in Section 9: Route Maintenance in [I-D.clausen-lln-loadng]. For the collection tree, LOADng-CT Routers are able to:

o Repair a broken route to the root.

- Participate in an existed collection tree, as a newly joined LOADng-CT Router, or a LOADng router that does not support collection tree extension.

If a link break occurs during data packet transmission, or a new LOADng-CT Router wishes to participate in the collection tree, the sensor initiate a new RREQ message without COLLECTION_TREE_TRIGGER and COLLECTION_TREE_BUILD flag set (denoted RREQ_NORMAL message) and <destination> field set to one of the interface addresses of the root. The RREQ_NORMAL MUST be processed and forwarded according to Section 12: Route Requests (RREQs) in [I-D.clausen-lln-loadng].

The collection tree can also be re-built by triggering a new RREQ_TRIGGER message by the root. How and when this RREQ_TRIGGER message should initiated would depend on network implementations. Some of the possible solutions can be:

- The root initiates an RREQ_TRIGGER message periodically. The interval SHOULD be depend on the frequency of topology changes.

- The root initiates an RREQ_TRIGGER message when it receives large amount of RREQ message destined to itself.

To avoid normal RREQ messages (without TRIGGER or BUILD flag) being broadcast through the whole network, and take benefits from the fact that "most of other neighbor routers might have an available route the to root", a Smart Route Request scheme MAY be employed:

- On receiving an RREQ_NORMAL message, the intermediate LOADng-CT Router checks if it has an available Route Tuple to the destination. If no Route Tuple is found, the message is forwarded according to the RREQ forward procedure.

- If an available Route Tuple to the destination is found, the intermediate router jut unicasts the RREQ_NORMAL to the destination according to the Route Tuple.

The LOADng Routers cannot join the collection tree in the collection tree building phase, because they cannot generate HELLO message and thus cannot be verified as bi-directional neighbor. For those routers, they can join the collection tree by initiating a new normal RREQ message.

12. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this

Internet-Draft, and based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

There is currently one publicly-known implementation of LOADng-CT specified in this document.

12.1. Implementation of Ecole Polytechnique

This implementation is developed by the Networking Group at Ecole Polytechnique and applied to LOADng [I-D.clausen-lln-loadng]. It can run over real network interfaces, and can also be integrated with the network simulator NS2. It is a Java implementation, and can be used on any platform that includes a Java virtual machine.

The implementation is based on -00 revision of this document, and includes only about 100 lines of additional code to the LOADng implementation. Primary simulation results have been published in [IEEE_WiCOM2012]. Results show that LOADng-CT extension can greatly reduce the overhead for collection tree building, compared to LOADng core specification.

13. Security Considerations

As an extension of LOADng protocol, LOADng-CT inherits the vulnerabilities of LOADng discussed in section 18 of [I-D.clausen-lln-loadng].

In addition, the collection tree building process relies on strictly ordered message sequences: RREQ_TRIGGER message for triggering the building process, then HELLO message for bi-direction neighbor check, and RREQ_BUILD message for collection tree build in the end. The message emission is controlled by router parameters like

NET_TRAVERSAL_TIME, RREQ_JITTER, and HELLO_JITTER.

The receiving order can be expected if those parameters are set correctly -- however, in real implementations, there might exist mis-configured routers, or even compromised routers that emit messages out of order. For example, if a router sends a HELLO message before it receives all the RREQ_TRIGGER messages from its neighbours, or an RREQ_BUILD message is received before the HELLO message exchange finished, the router cannot identify its bi-directional neighbours correctly -- thus is not able to join the collection tree as expected.

14. IANA Considerations

IANA is requested to ....

15. References

15.1. Normative References

[I-D.clausen-lln-loadng]
Clausen, T., Verdiere, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenu, C., Lys, T., and J. Dean, "The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-clausen-lln-loadng-10 (work in progress), October 2013.

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

[RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, February 2009.

[RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, March 2009.

15.2. Informative References

[IEEE_WiCOM2012]
Yi, J., Clausen, T., and A. Colin de Verdiere, "Efficient Data Acquisition in Sensor Networks: Introducing (the) LOADng Collection Tree Protocol", Proceedings of IEEE WiCOM2012, IEEE International Conference on Wireless

Communications, Networking and Mobile Computing, 2012.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

Appendix A. LOADng-CT Control Messages using RFC5444

This section presents how the abstract LOADng-CT messages, used throughout this specification, are mapped into [RFC5444] messages.

A.1. RREQ_TRIGGER and RREQ_BUILD Messages Encoding Considerations

This protocol makes use of RREQ message defined in [I-D.clausen-lln-loadng]. Therefore, it reuses the RREQ Message Type defined in [I-D.clausen-lln-loadng], and defines three additional flags: RREQ.trigger, RREQ.build and RREQ.ct-rrep. Table 1 describes how those flags are mapped into [RFC5444].

RREQ Element	RFC5444-Element	Considerations
RREQ.trigger	FLAGS Message TLV	Encoded by way of a Message-Type-specific Message TLV of type FLAGS, defined in Table 4
RREQ.build	FLAGS Message TLV	Encoded by way of a Message-Type-specific Message TLV of type FLAGS, defined in Table 4
RREQ.ct-rrep	FLAGS Message TLV	Encoded by way of a Message-Type-specific Message TLV of type FLAGS, defined in Table 4

Table 1: RREQ Message Elements for LOADng-CT

A.2. HELLO Message Encoding Considerations

This protocol makes use of HELLO Message Type. It is compatible with HELLO Message defined in [RFC6130], i.e., the HELLO message generated by this protocol is also valid for [RFC6130]. Table 2 describes how

HELLO messages are mapped in to [RFC5444] elements.

HELLO Element	RFC5444-Element	Considerations
HELLO.addr-length	<msg-addr-length>	Supports addresses from 1-16 octets
HELLO.originator	<msg-orig-addr>	MUST be included in a HELLO message
HELLO.validity-time	VALIDITY_TIME Message TLV (Type 1)	Exactly one VALIDITY_TIME TLV MUST be included in a HELLO message. The value is calculated by [RFC5497]
HELLO.1hop-list	LINK_STATUS Address Block TLV (Type 3)	With type extension 0. Specifies the status of the link from the indicated network address (LOST = 0, SYMMETRIC = 1, or HEARD = 2)

Table 2: HELLO Message Elements for LOADng-CT

Appendix B. RFC5444-Specific IANA Considerations

This specification uses two message types: RREQ and HELLO, which has been allocated in "Message Types" namespace of [RFC5444], in [I-D.clausen-lln-loadng] and [RFC6130] respectively.

IANA is requested to add a RREQ Message-Type-specific Message TLV Type, in accordance with Section 6.2.1 of [RFC5444], with allocation policies as specified in Table 3.

Type	Description	Allocation Policy
129	FLAGS	
130-223	Unassigned	Expert Review

Table 3: RREQ Message-Type-specific TLV Type for LOADng-CT

Allocation of the FLAGS TLV from the RREQ Message-Type-specific Message TLV Types in Table 3 will create a new Type Extension registry, with type extension 0, as illustrated in Table 4.

Name	Type	Type Extension	Bit	Description
FLAGS	129	0	0	NOTE: 0 is recommended for smart-rreq flag. To be defined in another document
FLAGS	129	0	1	RREQ.trigger flag (i.e. the RREQ message is RREQ_TRIGGER when it is set to 1)
FLAGS	129	0	2	RREQ.build flag (i.e. the RREQ message is RREQ_BUILD when it is set to 1)
FLAGS	129	0	3	RREQ.ct-rrep flag (i.e. the receiving sensors are required to send RREP when it is set to 1)
FLAGS	129	0	4-7	reserved for future use
FLAGS	129	1-255		Unassigned

Table 4: Message TLV Type assignment: FLAGs

Appendix C. LOADng-CT Control Packet Illustrations

TODO

Authors' Addresses

Jiazi Yi
LIX, Ecole Polytechnique

Phone: +33 1 6933 4031
Email: jiazi@jiaziyi.com
URI: http://www.jiaziyi.com/

Thomas Clausen
LIX, Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33-6-6058-9349
Email: T.Clausen@computer.org
URI: http://www.thomasclausen.org