Loop-Free Routing in Low-Power and Lossy Networks

Jianlin Guo, Chuan Han, Philip Orlik, Jinyun Zhang
Mitsubishi Electric Research Laboratories
Cambridge, USA
{guo, chan, porlik, jzhang}@merl.com

Koichi Ishibashi
Communication Systems Technology Department
Mitsubishi Electric Corporation IT R&D Center
Ofuna, Japan
Ishibashi.Koichi@ce.MitsubishiElectric.co.jp

***Abstract*—IPv6 based Low-Power and Lossy Networks (LLNs) are emerging. Internet Engineering Task Force (IETF) has developed an IPv6 Routing Protocol for LLNs (RPL), which is widely considered as a feasible routing protocol for LLNs. However, routing loops and lack of a loop-free local route repair mechanism are two major open issues to be addressed in RPL. Based on the framework of RPL, this paper proposes a Loop-Free Routing Protocol for LLNs (LRPL). We provide an innovative rank computation method and a loop-free local route repair mechanism to eliminate routing loops in RPL. Simulation results show that the proposed LRPL performs much better than conventional routing protocols in terms of packet delivery rate, end-to-end packet delay, and routing overhead.**

Keywords-loop-free routing; loop-free local route repair; low power and lossy network; routing protocol for low power and lossy network; directed acyclic graph; destination oriented directed acyclic graph; bidirectional routes

I. Introduction

Low-Power and Lossy Networks (LLNs) are a class of networks in which routers and their communication links are constrained. LLN routers typically operate with constrains on processing power, memory, power consumption, and lifetime. Their communication links are characterized by high loss rate, low data rate, low transmission power, and short transmission range. There can be from a few dozen up to thousands of nodes within a LLN. The characteristics of LLN require that routing overhead must be much less than application data. Therefore, routing in LLN is different from routing in mobile ad-hoc networks. Conventional routing protocols, such as Ad-hoc On-demand Distance Vector (AODV) [1] and Dynamic Source Routing (DSR) [2], designed for mobile ad-hoc networks are not suitable for routing in LLNs because of high routing overhead. IETF has developed an IPv6 Routing Protocol for LLNs (RPL) [3].

Based on routing metrics and constraints, RPL builds Directed Acyclic Graph (DAG) topology to establish bidirectional routes for LLNs. RPL routes are optimized for traffic to or from one or more roots that act as sinks. A DAG is partitioned into one or more Destination Oriented DAGs (DODAGs), one DODAG per sink. DODAG is basic logic structure in RPL. The sink in a DODAG is called the DODAG root. RPL supports multipoint-to-point traffic (from nodes inside the LLN to the DODAG root) and point-to-multipoint traffic (from the DODAG root to nodes inside the LLN). Support for point-to-point traffic is also available. The traffic of LLN flows along the edges of DODAG, either upwards to the DODAG root or downwards from the DODAG root.

Upward routes, having the DODAG root as destination, are provided by the DODAG construction mechanism using the DODAG Information Object (DIO) messages. The DODAG root configures the DODAG parameters such as RPLInstanceID, DODAG Version Number, DODAGID, Rank, DTSN, etc. and advertises these parameters in DIO messages. To join a DODAG, a node selects a set of parents on the routes towards the DODAG root and configures its own rank. It also selects a preferred parent as next hop for upward traffic. Upon joining a DODAG, a node transmits the DIO messages to advertise the DODAG parameters.

Downward routes, from the DODAG root to other nodes, are provided by these nodes transmitting the Destination Advertisement Object (DAO) messages. A node selects a subset of its parents as its DAO parents. Three modes of operation for downward routes are specified in RPL:

1) No downward routes maintained by RPL.
2) Storing mode of operation in which each router maintains downward routing tables to all nodes in its sub-DODAG, i.e. nodes that are deeper down in the DODAG. The DAO messages propagate from the nodes towards the root, where each intermediate node adds its reverse routing stack to the DAO message.
3) Non-Storing mode of operation in which only the DODAG root stores routes to all nodes in the network. Each node unicasts DAO messages to the root, which then calculates routes to all destinations by piecing together the information collected from DAO messages. In non-storing mode, downward traffic is sent by way of source routing.

RPL has been implemented and evaluated by researchers. It has been shown that IPv6 with the RPL routing has a battery lifetime of years [4]. RPL based routing for advanced metering infrastructure in smart grid has been proposed [5], in which an expected transmission time based rank computation method has been provided and evaluated. Some considerations in RPL implementation are presented in [6].

RPL is widely considered as a feasible routing protocol for LLNs. However, there are several important issues left unresolved. RPL is not a loop-free routing protocol. Experiment shows that loops occur frequently and in 74.14% of the 4114 snapshots, at least one loop was observed [7]. Even though RPL provides mechanism to resolve loops, researchers have shown that the mechanism may cause even worse turmoil than the routing loops themselves [8]. There is no local route repair mechanism provided in RPL.

In this paper, we present an innovative rank computation method for loop-free routing in LLNs. We also provide a

 ISBN: 978-1-61208-207-3

method for local route repair without causing any routing loop. The proposed local route repair method applies to both Storing mode and Non-Storing mode of operation in RPL. Based on the proposed rank computation method, a node can discover multiple bidirectional routes towards the DODAG root. Simulation results show the proposed Loop-Free Routing Protocol for LLNs (LRPL) achieves almost 100% of packet delivery rate with low end-to-end delay and frequent packet transmission in large scale LLNs. It performs much better than the conventional routing protocols.

II. RANK DEFINITION AND RANK SPLIT OPERATION

Rank plays very important role in the DODAG construction and maintenance. The rank of a node defines a position of the node relative to other nodes with respect to the DODAG root. Each node maintains its own rank. Nodes maintain their ranks based on parent-child relationship such that a child must have a rank strictly greater than ranks of all its parents. The DODAG root has no parent and therefore has the lowest rank. The acyclic structure of a DODAG is maintained as long as the rank of any node is strictly greater than ranks of all its parents. It is safe for a node to decrease its rank, as long as its new rank remains greater than ranks of its parents. However, rank increase can cause routing loops within a DODAG. RPL allows rank increase which is the source of routing loops in RPL.

Figure 1 shows an example of RPL routing loop in which the DODAG consists of 10 routers N_1 to N_{10} and the root. The integers are the respective ranks. The DODAG structure is shown by directed edges. If the route from N_1 to the root is broken, N_1 can poison the broken route by advertising a rank of infinity. If this infinity rank advertisement is lost, N_2 still has N_1 as its parent. N_3 then advertises its rank equal to 3, N_1 receives the advertisement from N_3 and selects N_3 as its parent. Loop N_1-N_3-N_2-N_1 is created. The cause of this loop is that N_1 increased its rank to infinity.

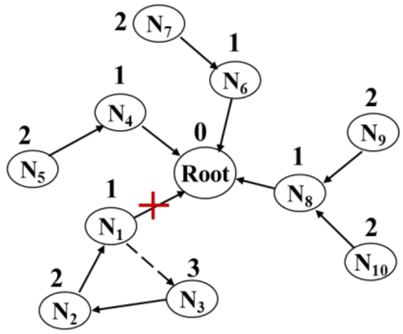

Figure 1. Routing Loop Example in the RPL

The routing loops can be avoided if nodes do not increase their ranks. In order to meet this requirement, we define the rank R as a proper fraction such that:

$$R = \frac{m}{n} \qquad (1)$$

where *m* and *n* are integers such that $0 \le m < n$.

Even though the rank is defined as proper fraction, it is maintained as two integers, numerator *m* and denominator *n*. The fractional value of rank is only used in rank operations such as rank comparison.

The principle of this innovative rank definition is that there are an infinite number of proper fractions between any two proper fractions. This principle guarantees that given any two ranks, there always exists a rank in between them. However, integer rank does not possess such property because there is no integer existing between any two consecutive integers.

For any two ranks $R_1 = m/n$ and $R_2 = p/q$, the rank split operation is defined as:

$$sp(R_1, R_2) = \frac{m+p}{n+q} \qquad (2)$$

It can be shown that if $R_1 < R_2$, then $R_1 < sp(R_1,R_2) < R_2$.

In this paper, we define the root rank as 0/1 and the infinite rank as 1/1. The infinite rank can not be advertised in the DIO messages.

III. DODAG CONSTRUCTION

In RPL, a node may act as a router or a leaf node. To construct a new DODAG, the DODAG root transmits a DIO message containing new (RPLInstanceID, DODAGID) tuple. To construct a new DODAG Version, the DODAG root transmits a DIO message with an increased DODAG Version Number. The DODAG Version Number is monotonically incremented by the DODAG root. The DIO message is transmitted via link-local multicasting to all-RPL-nodes. Nodes obtain the DODAG parameters configured by the DODAG root in received DIO messages. A node must keep the DODAG parameters unchanged except Rank and DTSN.

In this paper, we use symbols such as N_i, N_j, N_k, etc. to denote nodes and use $R(N_i)$ to denote the rank of node N_i. For simplicity, we assume RPLInstanceID and DODAGID are fixed. To construct and maintain a DODAG, a node N_i maintains following state parameters:

TABLE 1. Node State Parameters

$R(N_i)$	Rank of node N_i as proper fraction *m/n*
$P(N_i)$	Parent set of node N_i
$p(N_i)$	Preferred parent of node N_i
$c(N_i)$	The minimum cost from node N_i to the DODAG root
$c(N_i,N_j)$	Cost from node N_i to node N_j
$VN(N_i)$	DODAG Version Number maintained by node N_i
DR-SN(N_i)	DODAG repair sequence number of node N_i
T_p	Parent threshold

The cost can be hop count, expected transmission time, and other options. For a node, if the number of parents is less than T_p, the node can add more parents into its parent set if such parents are available. A node N_i maintains its parent set $P(N_i)$ such that for each parent $N_p \in P(N_i)$, $R(N_i) > R(N_p)$.

Initially, all nodes do not belong to any DODAG and do not transmit the DIO messages because a node can transmit the DIO messages only if the node joins a DODAG. The DODAG root initiates a new DODAG construction process by configuring the DODAG parameters and transmitting the DIO messages to advertise the DODAG parameters.

In response to receiving a DIO message, a node can update its state parameters only if one of the following conditions holds:

(1) The node wants to join a DODAG
(2) The DODAG Version Number in the DIO message is greater than the DODAG Version Number maintained by receiving node
(3) The DODAG Version Number in the DIO message equals the DODAG Version Number of receiving node, and the rank in the DIO message is lower than the rank of receiving node.

Upon receiving a DIO message transmitted by the DODAG root containing new (RPLInstanceID, DODAGID) tuple or new DODAGVersionNumber, the first hop nodes of the DODAG root may join new DODAG or new DODAG Version. To do so, the first hop nodes add the DODAG root into their parent set and store the DODAG parameters. The first hop nodes keep all DODAG parameters unchanged except the rank. The first hop nodes set their ranks such that their ranks > 0/1 and their ranks <= sp(0/1, 1/1) = 1/2. Upon joining a new DODAG or a new DODAG Version, the first hop routers generate and transmit the DIO messages to advertise the DODAG parameters.

Upon receiving the DIO messages transmitted by the first hop routers, the second hop nodes of the DODAG root that want to join new DODAG or new DODAG Version perform similar procedure as the first hop nodes do. However, in this case, the second hop nodes may receive multiple DIO messages from the first hop routers. The second hop nodes use received DIO messages to calculate their ranks and select a subset of the DIO message senders as their parents. To calculate its rank, a second hop node find the maximum rank, Rank_Max, among all ranks of its parents and sets its rank such that its rank > Rank_Max and its rank <= sp(Rank_Max, 1/1). The second hop routers then generate and transmit the DIO messages same as the first hop router do.

A first hop node of the DODAG root may also receive the DIO messages transmitted by other first hop routers. The first hop node may perform same procedure as the second hop nodes do to select more parents.

This DIO message propagation process continues until all nodes in network receive the DIO messages, store the DODAG parameters, select parents and determine ranks.

Figure 2 shows the process of DODAG construction, where router N_j transmitted the DIO message containing $VN(N_j)$, $R(N_j)$, $c(N_j)$, etc. and node N_i receives the DIO message. $VN(N_i)$, $R(N_i)$, $P(N_i)$, and $p(N_i)$ are state parameters maintained by node N_i.

Upon receiving the DIO message, node N_i first checks if the received DIO message is malformed or was received already. If yes, N_i discards the DIO message. If no, N_i checks if N_j equals N_i. If yes, N_i discards the DIO message, because N_i just received its own DIO message. Otherwise, N_i processes the DIO message further.

N_i checks if a new DODAG is advertised in the DIO message. If yes, N_i joins new DODAG. It initializes its state parameters as $VN(N_i) = VN(N_j)$, $P(N_i) = \{N_j\}$, $p(N_i) = N_j$, $R(N_i) = sp(R(N_j),1/1)$, $c(N_i) = c(N_j) + c(N_i, N_j)$, and stores other DODAG parameters. N_i then resets its trickle timer to transmit the DIO message and schedules a DAO message transmission if N_j is also selected as its DAO parent. Otherwise, the DIO processing goes to next step.

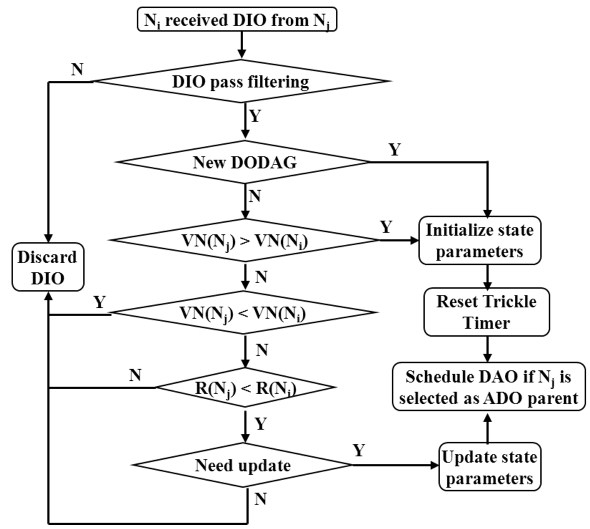

Figure 2. The DODAG Construction Process

N_i checks if the $VN(N_j) > VN(N_i)$. If yes, N_i joins new DODAG Version. It initializes its state parameters same as joining new DODAG. N_i also clears downward routing tables if the mode of operation is Storing. N_i then resets its trickle timer to transmit the DIO message and schedules a DAO message transmission if N_j is also selected as its DAO parent. Otherwise, the DIO processing goes to next step.

N_i checks if $VN(N_j) < VN(N_i)$. If yes, it discards the DIO message. If $VN(N_j) = VN(N_i)$ and $R(N_j) \geq R(N_i)$, N_i discards the DIO message. If $VN(N_j) = VN(N_i)$ and $R(N_j) < R(N_i)$, N_i checks if it is necessary to update its state parameters by using received the DIO message. If no, N_i discards the DIO message. If yes, N_i updates state parameters. If N_j is not in its parent set $P(N_i)$ and $|P(N_i)| < T_p$, N_i adds N_j into its parent set such that $P(N_i) = P(N_i) \cup \{N_j\}$ and updates its preferred parent as

$$p(N_i) = \arg \min_{N_k \in P(N_i)} \{c(N_i, N_k) + c(N_k)\} \quad (3)$$

and the minimum cost as

$$c(N_i) = c(N_i, p(N_i)) + c(p(N_i)) \quad (4)$$

If there are multiple parents that have the same minimum cost, N_i can randomly pick one preferred parent. N_i then schedules a DAO message transmission if N_j is also added into its DAO parent set. If N_j is already in DAO parent set, N_i makes necessary updates without scheduling the DAO message transmission.

A node can receive multiple DIO messages from neighbors within the same DODAG. These DIO messages can be used to select a subset of the DIO message transmitters as its parents and determine its rank. Among all its parents, the node selects

 ISBN: 978-1-61208-207-3

one parent with the minimum cost as its preferred parent to be used as the next hop along upward routes to the root.

Figure 3 shows an example of the DODAG construction. Initially, nodes N_1 – N_6 are not members of any DODAG version, and their parent sets are empty. The DODAG root sets its rank to 0/1, the DODAG version number to 1, and its parent set to empty.

The root transmits the DIO message carrying its DODAG version number 1, and rank 0/1. Nodes N_1, N_2 and N_3 receive the DIO message. Because nodes N_1, N_2 and N_3 are not members of the newly advertised DODAG, N_1, N_2 and N_3 joins the DODAG and set their DODAG version numbers to 1, ranks to sp(1/1, 0/1) = 1/2, and select the root as their preferred parent.

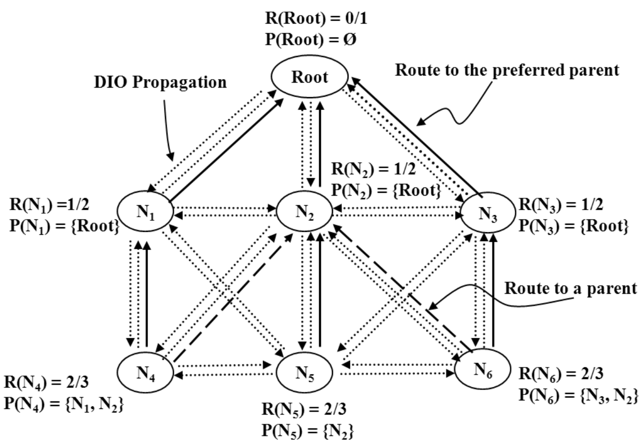

Figure 3. The DODAG Construction Example

Upon joining the DODAG, nodes N_1, N_2, and N_3 transmit the DIO messages with DODAG version number 1 and rank 1/2. The DIO messages from routers N_1, N_2, and N_3 are discarded by the root because the DODAG version number in the DIO messages equals the DODAG version number of the root, and the rank in the DIO messages is greater than the rank of the root.

N_1 discards the DIO message from N_2 because the DODAG version number in the DIO message equals the DODAG version number of N_1, and the rank in the DIO message equals N_1's rank. Similarly, N_2 discards the DIO messages from N_1 and N_3, and N_3 discards the DIO message from N_2.

N_4 receives DIO messages from N_1 and N_2. Because N_4 is not a member of the advertised DODAG, N_4 joins the DODAG and sets its DODAG version number to 1, its rank to sp(1/1, 1/2) = 2/3, and select N_1 as the preferred parent and N_2 as parent. Similarly, N_6 receives the DIO messages from N_2 and N_3, joins the DODAG, sets its DODAG version number to 1, rank to sp(1/1, 1/2) = 2/3, adds N_2 and N_3 into its parent set, and selects N_3 as the preferred parent. N_5 receives the DIO messages from N_1, N_2, and N_3. Because N_5 is not a member of the advertised DODAG, N_5 joins the DODAG and sets its DODAG version number to 1, its rank to sp(1/1, 1/2) = 2/3. However, N_5 only selects N_2 as its parent and preferred parent even though N_2 may select N_1, N_2, and N_3 as parents.

Upon joining the DODAG, nodes N_4, N_5 and N_6 also transmit their DIO messages. These DIO messages are discarded by their neighbors because the DODAG version number in the DIO messages equals the DODAG version number of the neighbors, and the rank of N_4, N_5 and N_6 are not lower than ranks of the neighbors.

IV. DODAG LOCAL REPAIR

The DODAG local repair is performed by using two new RPL control messages, the DODAG Repair Request (DR-REQ) message and the DODAG Repair Reply (DR-REP) message.

The DR-REQ message consists of N_q, $R(N_q)$, $VN(N_q)$, DR-SN(N_q), NL-REQ, and other fields. The N_q is the identifier of node generating DR-REQ message, $R(N_q)$ is the rank of N_q, $VN(N_q)$ is the DODAG Version Number of N_q, DR-SN(N_q) is the DODAG repair sequence number of N_q, NL-REQ is the node list traveled through by DR-REQ message and present only in Non-Storing mode. In addition, the DR-REQ message may also have a hop count field and a maximum hop count field. Once hop count reaches the maximum hop count, the DR-REQ message is discarded.

The DR-REP message consists of N_q, $R(N_q)$, DR-SN(N_q), D, $R(N_p)$, c, $VN(N_p)$, NL-REP, and other fields. N_q, $R(N_q)$ and DR-SN(N_q) are same as in the DR-REQ message. N_q is destination of DR-REP message. D indicates the travel direction of DR-REP message, $R(N_p)$ is the rank of router generating the DR-REP message if D = UP and is the rank of router transmitting the DR-REP message if D = DOWN, c is the minimum cost of link(s) from the router transmitting the DR-REP message to the DODAG root, $VN(N_p)$ is the DODAG Version Number of DR-REP message generator, and NL-REP is combination of NL-REQ in the DR-REQ message and node list travelled by upward DR-REP message. D and NL-REP are present only in Non-Storing mode.

When a node detects a broken route by using mechanisms provided in RPL, it may need to discover new parents. The DODAG is locally repaired by node transmitting a DR-REQ message. The DR-REQ message is transmitted by the DR-REQ message generator via link-local multicasting to all-RPL-nodes.

Upon receiving a DR-REQ message, a link-local neighbor discards the DR-REQ message if it does not have a route to the DODAG root. If the link-local neighbor is the DODAG root or a router that has a route to the DODAG root and a rank lower than the rank carried in the DR-REQ message, this neighbor generates a DR-REP message. If the link-local neighbor has route to the DODAG root and its rank is greater than or equal to the rank carried in the DR-REQ message, this neighbor forwards the DR-REQ message to its preferred parent.

In Storing mode, the DR-REP message generator transmits the DR-REP message to node N_q by using downward routing tables. Route entry is added into downward tables while the DR-REQ message is processed. In Non-Storing mode, the DR-REP message is forwarded up to the DODAG root, which then transmits the DR-REP message to node N_q by using source routing.

ISBN: 978-1-61208-207-3

A. DODAG Local Repair in Storing Mode

In Storing mode, if the route from node N_q to its parent N_{qp} is broken, N_q removes N_{qp} from its parent set such that $P(N_q) = \{N_k \mid N_k \in P(N_q) / \{N_{qp}\}\}$. If the updated parent set $P(N_q)$ is empty, N_q transmits a DR-REQ message to discover new parents. If the updated parent set $P(N_q)$ is not empty, N_q checks if N_{qp} is its preferred parent $p(N_q)$. If yes, N_q selects a new preferred parent $p(N_q)$ as shown in equation (3) and updates $c(N_q)$ as shown in equation (4). If N_{qp} is also in N_q's DAO parent set, N_q schedules a No-Path DAO message transmission.

Whether or not N_{qp} is N_q's preferred parent, N_q can transmit a DR-REQ message to discover additional parents if $|P(N_q)| < T_p$. To construct a DR-REQ message in Storing mode, N_q increases DR-SN(N_q) by 1 and uses N_q, $R(N_q)$, $VN(N_q)$, and DR-SN(N_q) to fill the fields in the DR-REQ message.

A.1 DR-REQ Message Processing

Figure 4 shows the procedure of processing the DR-REQ message when router N_i receives a DR-REQ message from N_j in which $VN(N_q)$, N_q, $R(N_q)$ and DR-SN(N_q) are the parameters carried in the DR-REQ message, and $VN(N_i)$, R (N_i), and $P(N_i)$ are state parameters of N_i.

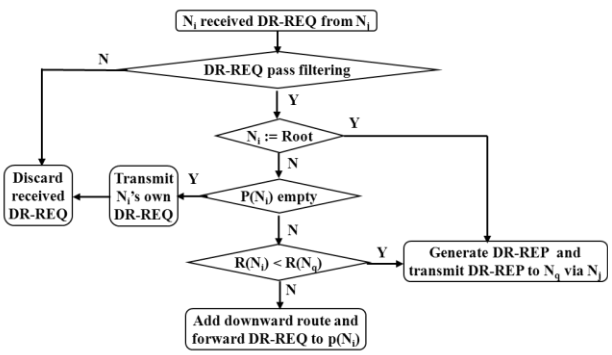

Figure 4. The DR-REQ Processing in Storing Mode

Router N_i first performs the filtering process. The DR-REQ message is discarded if this DR-REQ message is received already by checking N_q and DR-SN(N_q) or if the $VN(N_q)$ is not equal to $VN(N_i)$ or if the DR-REQ message is transmitted by N_i's parent or if the DR-REQ message is generated by N_i's parent or by N_i itself.

If N_i is the DODAG root, N_i accepts the DR-REQ message, generates a DR-REP message by copying N_q, $R(N_q)$, DR-SN(N_q) from the DR-REQ message, and setting $R(N_p)$ = R(Root), c = 0, $VN(N_p)$ = VN(Root), and transmits the DR-REP message to node N_q via next hop node N_j.

If N_i is not the DODAG root, the processing of DR-REQ message is as follows. If N_i's parent set $P(N_i)$ is empty, N_i discards the DR-REQ message and transmits a its own DR-REQ message. If N_i's parent set $P(N_i)$ is not empty and $R(N_i) < R(N_q)$, N_i accepts the DR-REQ message and generates a DR-REP message by copying N_q, $R(N_q)$, DR-SN(N_q) from the DR-REQ message, and setting $R(N_p) = R(N_i)$, $c = c(N_i)$, $VN(N_p) = VN(N_i)$. N_i transmits the DR-REP message to node N_q via next hop node N_j. If N_i's parent set $P(N_i)$ is not empty and $R(N_i) \geq R(N_q)$, N_i adds a downward routing entry to node N_q into its downward routing table, and forwards the DR-REQ message to its preferred parent $p(N_i)$.

A.2 DR-REP Message Processing

Figure 5 shows the procedure of processing the DR-REP message when node N_i receives a DR-REP message from router N_j in which $VN(N_p)$, N_q, $R(N_p)$, DR-SN(N_q) and $R(N_q)$ are the parameters carried in the DR-REP message, and $VN(N_i)$, $R(N_i)$, $P(N_i)$, $p(N_i)$, $c(N_i)$, and T_p are state parameters of node N_i.

If $VN(N_p)$ is not equal to $VN(N_i)$ or this DR-REP message is received already, node N_i discards the DR-REP message. Otherwise, N_i processes the DR-REP message further.

If N_i is the DR-REQ message generator and N_j is not in N_i's parent set $P(N_i)$, N_i adds N_j into $P(N_i)$ if $|P(N_i)| < T_p$ and updates $p(N_i)$ according to equation (3) and $c(N_i)$ according to equation (4). N_i then schedules a DAO message transmission if N_j is also added into its DAO set.

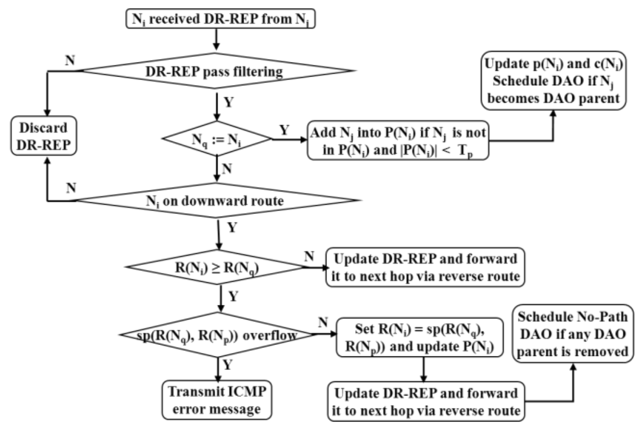

Figure 5. The DR-REP Processing in Storing Mode

If N_i is not the DR-REQ message generator, the processing of the DR-REP message is as follows. If N_i is not on the downward route, N_i discards the DR-REP message. Otherwise, if $R(N_i) \geq R(N_q)$, N_i decreases its rank $R(N_i)$ as

$$R(N_i) = sp(R(N_q) + R(N_p)) \quad (5)$$

and updates its parent set $P(N_i)$ as

$$P(N_i) = \{N_k \mid R(N_k) < R(N_i), N_k \in P(N_i)\} \quad (6)$$

If the preferred parent $p(N_i)$ is removed due to its rank decrease, N_i selects a new $p(N_i)$ according to equation (3) and updates $c(N_i)$ according to equation (4). N_i then updates the DR-REP message by setting $R(N_p) = R(N_i)$ and $c = c(N_i)$, forwards the DR-REP message to next hop node obtained from downward routing table. N_i schedules a No-Path DAO message transmission if any DAO parent is removed.

If $R(N_i) < R(N_q)$, N_i updates the DR-REP message by setting $R(N_p) = R(N_i)$ and $c = c(N_i)$, forwards it to next hop node obtained from downward routing table. In Storing mode, $R(N_i) < R(N_q)$ occurs if N_i is on multiple DODAG repair routes. When N_i receives a DR-REP message, it may decrease its rank. Therefore, subsequent DR-REP messages may carry a

 ISBN: 978-1-61208-207-3

rank $R(N_q)$ greater than or equal to $R(N_i)$. If N_i is only on a single DODAG repair route, $R(N_i) \geq R(N_q)$ must be true based on the DR-REQ message processing procedure.

By the definition of rank split operation, it is easy to show that rank $R(N_p)$ in the DR-REP message is the maximum rank of routers on the route from the DR-REP message generator to the DR-REP message transmitter. $R(N_p)$ is always less than $R(N_q)$. Therefore, when the DR-REP message reaches the DR-REQ message generator N_q, rank $R(N_p)$ in the DR-REP message must be less than $R(N_q)$. Therefore, the rank monotonically increases along a route from the DE-REP message generator to the DR-REQ message generator. This guarantees that rank increases monotonically along the route from the DODAG root to any node.

B. DODAG Local Repair in Non-Storing Mode

The processing of upward route failure from node N_q to its parent N_{qp} in Non-Storing mode is mostly similar to that in Storing mode. The first difference is that after removing a DAO parent, the node schedules a transmission of DAO message instead of No-Path DAO message. The second difference is that NL-REQ field is present in the DR-REQ message; D and NL-REP fields are present in the DR-REP message. The third difference is that the DR-REP message is first forwarded upwards to the DODAG root, which then sends the DR-REP message downwards to node N_q.

B.1 DR-REQ Message Processing

Figure 6 shows the procedure of processing the DR-REQ message when N_i receives a DR-REQ message from N_j in which $VN(N_q)$, N_q, $R(N_q)$, $DR\text{-}SN(N_q)$, and NL-REQ are the parameters in the DR-REQ message and $VN(N_i)$, $R(N_i)$, and $P(N_i)$ are state parameters of N_i.

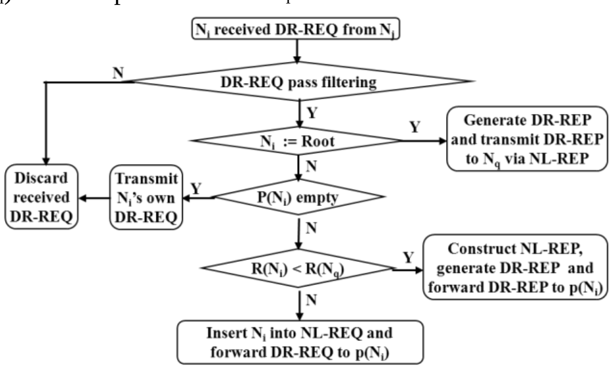

Figure 6. The DR-REQ Processing in Non-Storing Mode

The DR-REQ message is discarded if this DR-REQ message is received already or if $VN(N_q)$ is not equal to $VN(N_i)$ or if the DR-REQ message is transmitted by N_i's parent or if the DR-REQ message is generated by N_i's parent or by N_i itself.

If N_i is the DODAG root, N_i accepts the DR-REQ message, and generates a DR-REP message similarly as in Storing mode. However, in this case, the DODAG root sets D to DOWN, NL-REP field in DR-REP message to NL-REQ field in DR-REQ message, and transmits the DR-REP message to node N_q via the route provided by NL-REP field.

If N_i is not the DODAG root, the processing of the DR-REQ message is as follows.

If N_i's parent set $P(N_i)$ is empty, N_i discards the received DR-REQ message and transmits its own DR-REQ message. If N_i's parent set $P(N_i)$ is not empty and $R(N_i) < R(N_q)$, N_i accepts the DR-REQ message, and generates a DR-REP message similar as the DODAG root does. However, N_i sets D = UP, NL-REP = NL-REQ $\bigcup \{N_i\}$, and forwards the DR-REP message to its preferred parent $p(N_i)$. If N_i's parent set $P(N_i)$ is not empty and $R(N_i) \geq R(N_q)$, N_i updates the DR-REQ message by inserting N_i in NL-REQ such that NL-REQ = NL-REQ $\bigcup \{N_i\}$, and forwards the DR-REQ message to its preferred parent $p(N_i)$.

B.2 DR-REP Message Processing

Figure 7 shows that N_i receives a DR-REP message from N_j in which $VN(N_p)$, N_q, $R(N_p)$, $DR\text{-}SN(N_q)$, D, $R(N_q)$ and NL-REP are the parameters in the DR-REP message, $VN(N_i)$, $R(N_i)$, $P(N_i)$, $p(N_i)$, $c(N_i)$, and T_p are state parameters of N_i.

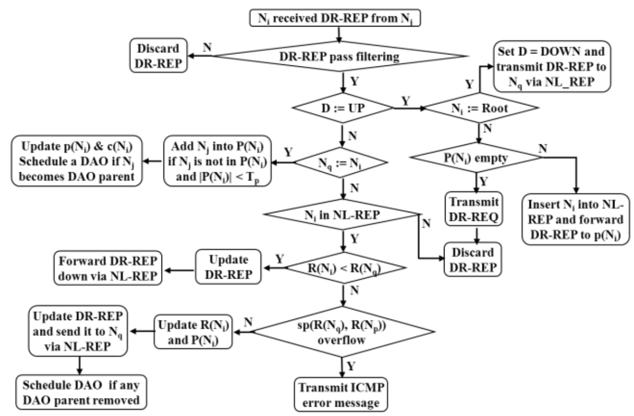

Figure 7. The DE-REP Processing in Non-Storing Mode

If $VN(N_p)$ is not equal to $VN(N_i)$ or this DR-REP message is received already, N_i discards the DR-REP message.

If D = UP, the DR-REP message is transmitted upwards. If N_i is the DODAG root, N_i updates the DR-REP message by changing D = DOWN, $R(N_p) = R(Root)$, c = 0, and transmits DR-REP message down to node N_q via the route provided by NL-REP field. If N_i is not the DODAG root and its parent set $P(N_i)$ is not empty, N_i updates DR-REP message such that NL-REP = NL-REP $\bigcup \{N_i\}$, and forwards the DR-REP message to its preferred parent $p(N_i)$. If N_i is not the DODAG root and its parent set $P(N_i)$ is empty, N_i discards the received DR-REP message.

If D = DOWN, the DR-REP message is transmitted downwards. If N_i is DR-REQ message generator, N_j is not in its parent set $P(N_i)$ and $|P(N_i)| < T_p$, N_i adds N_j into $P(N_i)$ and updates $p(N_i)$ according to equation (3) and $c(N_i)$ according to equation (4). N_i then schedules a DAO message transmission if N_j is also added into its DAO parent set. If N_i is not the DR-REQ message generator and is not on the downward route, N_i discards the DR-REP message. Otherwise, if $R(N_i) < R(N_q)$, N_i updates the DR-REP message by setting $R(N_p) = R(N_i)$, c = $c(N_i)$, and forwards the DR-REP message to node N_q via the

 ISBN: 978-1-61208-207-3

route provided by NL-REP field. If $R(N_i) \geq R(N_q)$, N_i decreases its rank $R(N_i)$ to $sp(R(N_q), R(N_p))$ and updates its parent set $P(N_i)$, the preferred parent $p(N_i)$ and cost $c(N_i)$ according to equations (5), (6), (3) and (4) respectively. N_i then updates the DR-REP message by setting $R(N_p) = R(N_i)$, $c = c(N_i)$, and forwards the DR-REP message to node N_q via the route provided by NL-REP field. Furthermore, N_i schedules a DAO message transmission if any DAO parent is removed due to its rank decrease.

By definition of the rank split operation, it can also been shown that rank $R(N_p)$ in the downward DR-REP message is the maximum rank of routers on the route from the root to DR-REP transmitter. $R(N_p)$ is always less than $R(N_q)$. Therefore, when the DR-REP message reaches the DR-REQ message generator, the rank $R(N_p)$ in the DR-REP message must be less than $R(N_q)$, which is the rank of the DR-REQ message generator. Hence, the rank monotonically increases from the root to the DR-REQ message generator. This guarantees that rank increases monotonically along a route from the root to any node.

Figure 8 illustrates how the broken route in Figure 1 is handled by the proposed DODAG local repair method. The fractions are the ranks of nodes and the root, respectively. After the route to the root is broken, N_1 removes the root from its parent set $P(N_1)$ and transmits a DR-REQ message with $N_q = N_1$ and $R(N_q) = R(N_1) = 1/2$. N_2 discards the DR-REQ message because this DR-REQ message is transmitted by its parent N_1. N_3 forwards the DR-REQ message to N_2 because $R(N_q)$ in the DR-REQ message is smaller than its rank $R(N_3) = 3/4$. However, the DR-REQ message forwarded by N_3 is discarded by N_2 because the DR-REQ message is generated by N_2's parent N_1. N_5 forwards the DR-REQ message to N_4 because $R(N_q)$ is smaller than $R(N_5) = 2/3$. N_4 forwards the DR-REQ message to the root because the rank $R(N_q)$ equals its rank $R(N_4) = 1/2$. The root generates a DR-REP message with $R(N_p) = R(Root) = 0/1$ and transmits the DE-REP message back to N_1.

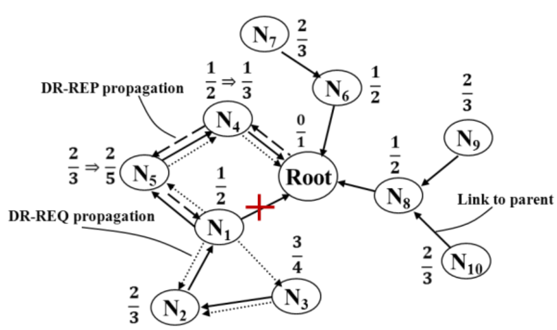

Figure 8. Example of the DODAG Local Repair

Upon receiving this DR-REP message, N_4 decreases its rank $R(N_4)$ to 1/3 because its old $R(N_4) = 1/2$, which equals to $R(N_q)$. N_4 then sets $R(N_p)$ to its new rank $R(N_4) = 1/3$ and forwards the DR-REP message to N_5. Upon receiving the DR-REP message, N_5 decreases its rank to 2/5 because its old rank $R(R_5) = 2/3$, which is greater than $R(N_q) = 1/2$. N_5 then sets $R(N_p)$ to its new rank $R(N_5) = 2/5$ and forwards DR-REP message to N_1. Upon receiving the DR-REP message from N_5, N_1 selects N_5 as its parent and transmits DIO message without changing its rank. The DODAG local repair process initiated by N_1 is completed.

V. SIMULATIONS

The performance of AODV and DSR has been evaluated considerably. The NS2 simulator is used to simulate AODV and DSR in [10 - 17. Unfortunately, most of simulation results are obtained with a small number of nodes, less or equal to 50 nodes [11-17]. Another common fact is that all simulations are performed using IEEE 802.11 wireless network instead of IEEE 802.15.4 wireless network, which is designed for LLNs. RPL has been implemented and simulated in [5]. However, the simulation was also done over IEEE 802.11 wireless networks.

We used NS2 simulator with IEEE 802.15.4 to simulate the performance of proposed routing protocol in large scale LLNs. Nodes are randomly displaced in a rectangle with the DODAG root in the middle of rectangle. In the simulation, transmission range is 30 meters and data rate is 100kbps. The CBR traffic is employed with 50 bytes of payload. TwoRayGround channel model and Shadowing channel model [8] are used. Performance metrics are data packet delivery rate (PDR), data average end-to-end delay (AED) and routing overhead (ROH) per data packet.

TABLE 2. TwoRayGround Channel Model with 1000 Nodes

Metrics	CBR Interval = 5 Minutes		CBR Interval = 2 Minutes	
	AODV	LRPL	AODV	LRPL
PDR	56.78%	100%	13.8%	100%
AED	920ms	140ms	2310ms	150ms
ROH	5.96	0.22	4.42	0.09

Tables 2 shows simulation results using TwoRayGround channel model, 1000 nodes and 24 hours simulation time. 1000 nodes are randomly deployed in a 320m by 320m rectangle. LRPL achieves 100% of packet delivery rate. AODV only achieves 56.78% of packet delivery rate for 5-minute CBR Interval and drops 82.6% of data packet for 2-minute CBR interval. For 5-minute CBR interval, LRPL is 6.6 times faster than AODV. For 2-minute CBR interval, LRPL is 15.4 times faster than AODV. For 5-minute CBR interval, LRPL's routing overhead is 27 times lower than AODV outing overhead. For 2-minute CBR interval, LRPL's routing overhead is 49 times lower than AODV routing overhead.

TABLE 3. Shadowing Channel Model with 500 Nodes

Metrics	PLE = 2.0		PLE = 2.5		PLE = 3.0	
	AODV	LRPL	AODV	LRPL	AODV	LRPL
PDR	36.7%	99.98%	34.1%	99.83%	32.5%	99.99%
AED	1530ms	177ms	1680ms	184ms	1840ms	166ms
ROH	168.75	0.44	188.75	0.43	550.5	0.43

Table 3 shows the performance comparison with Shadowing channel model and 500 nodes, which are randomly deployed in a 250m by 200m rectangle. The shadowing deviation is 4dB, CBR interval is 30 minutes and simulation time is 24 hours. Table 3 illustrates performance variation of routing protocols as path loss exponent (PLE) changes. LRPL almost achieves 100% of packet delivery rate. However, AODV drops more than 63% of packets. LRPL is about 10

 ISBN: 978-1-61208-207-3

times faster than AODV. The routing overhead of LRPL is at least 380 times lower than that of AODV.

TABLE 4. Shadowing Channel Model with 500 Nodes

Metrics	PLE = 2.0	PLE = 2.5	PLE = 3.0	PLE = 3.5	PLE = 4.0
PDR	99.98%	99.83%	99.99%	99.94%	99.99%
AED	177ms	184ms	166ms	205ms	194ms
ROH	0.44	0.43	0.43	0.58	0.55

Table 4 illustrates a more complete performance of LRPL with Shadowing channel model and 500 nodes. It can be seen that the overall performance of LRPL is excellent. LRPL maintains its performance as path loss exponent increases from 2.0 to 4.0, especially the packer delivery rate, which is almost 100%. The end-to-end packet delay and the routing overhead tend to increase; the change however is very small.

TABLE 5. Shadowing Channel Model with 1000 Nodes

Metrics	PLE = 2.0	PLE = 2.5	PLE = 3.0	PLE = 3.5	PLE = 4.0
PDR	99.60%	99.79%	99.29%	99.28%	99.54%
AED	271ms	249ms	369ms	354ms	303ms
ROH	0.90	1.02	1.16	1.61	1.24

Tables 5 shows simulation results of LRPL using Shadowing channel model and 1000 nodes. It can also be seen that the overall performance of LRPL is also excellent. LRPL achieves also 100% of packet delivery rate for all cases. As path loss exponent increases from 2.0 to 4.0, the end-to-end packet delay and the routing overhead tend to increase.

Tables 4 and 5 show that packet delivery rate of LRPL is almost same for 500 nodes and 1000 nodes. However, the end-to-end delay increases for about 55% and the routing overhead however increases about 150%. The routing overhead increase is mostly contributed by the DODAG local repair packets. It indicates that as the number of nodes increases, communication interference also increases. Therefore, the communication link breaks more often.

To compare the proposed LRPL with RPL, we refer to the results in [5], which simulated RPL using 802.11 wireless network. The performance of RPL was evaluated with smaller shadowing deviation of 1dB and 2dB. For shadowing deviation of 2dB, RPL only achieves a 97.9% of packet delivery rate. On the other hand, LRPL achieves more than 99% of packet delivery rate with shadowing deviation of 4dB. It can be seen that even with lower data rate of 802.15.4 and larger shadowing fading effect, LRPL performs better than RPL.

VI. CONCLUSION

In this paper, we present a loop-free routing protocol in LLNs based on IETF RPL framework. The proposed routing protocol defines rank as proper fraction to guarantee no routing loops can be created. A DODAG local repair method is also proposed for fast route repair. The proposed routing protocol is simulated by using NS2 simulator with a large number of nodes over IEEE 802.15.4 low power and lossy wireless networks. Simulation results show that the proposed routing protocol performs much better than conventional routing protocols. It achieves almost 100% of packet delivery rate with much shorter end-to-end delay and lower routing overhead. Therefore, it is a desired routing protocol for LLNs, especially when network scale is large and message generation rate is high. We are planning to implement RPL in 802.15.4 wireless network. The results will be reported in the future.

REFERENCES

[1] C.E. Perkins, E.M. Royer, and S.R. Das, “Ad-hoc On-demand Distance Vector (AODV) routing”, RFC 3561, July 2003

[2] D. Johnson, Y. Hu, and D. Maltz, “The Dynamic Source Routing protocol (DSR) for mobile Ad Hoc networks for IPv4”, RFC 4728, February 2007

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Stuick, JP. Vasseur, and R. Alexander “RPL: IPv6 Routing Protocol for Low power and Lossy Networks”, RFC 6550, March 2012

[4] N. Tsiftes, J. Eriksson, and A. Dunkels, “Poster Abstract: Low-Power Wireless IPv6 Routing with ContikiRPL”, The 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, Stockholm, Sweden, April 2010

[5] D. Wang, Z. Tao, J. Zhang, and A. Abouzeid, “RPL Based Routing for Advanced Metering Infrastructure in Smart Grid”, IEEE International Workshop on Smart Grid Communications, May 2010

[6] T. Clausen and U. Herberg, “Some Considerations on Routing in Particular and Lossy Environment”, Proceedings of the 1st IAB Interconnecting Smart Objects with the Internet Workshop, Prague, Czech Republic, March 2011

[7] T. Clausen, J. Yi, and U. Herberg, “Experieces with RPL: IPv6 Routing Protocol for Low power and Lossy Networks”, the 83rd IETF Plenary Meeting, Paris, France, March 2012

[8] W. Xie, M. Goyal, H. Mosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi, “Routing Loops in DAG-based Low Power and Lossy Networks”, Proc. IEEE AINA 2010, 2010

[9] K. Fall and K. Varadhan, “The Network Simulator Manual”, http://www.isi.edu/nsnam/ns/ns-documentation.html, May 2010

[10] A. Goel and A. Sharma, “Performance Analysis of Mobile Ad-hoc Network Using AODV Protocl”, International Journal of Computer Science and Security (IJCSS), Volume (3): Issue (5), 2009

[11] D. Singh, P. Trivedi, and J.D. Lal, “Performance Evaluation of DSR and AODV Networking Protocol With Varying Pause Time”, Proceedings of SPIT-IEEE Colloquium and International Conference, Mumbai, India, Vol. 3, 190, 2007

[12] A.H.A. Rahman and Z.A. Zukarnain, “Performance Comparison of AODV, DSDV and I-DSDV Routing Protocols in Mobile Ad Hoc Networks”, European Journal of Scientific Research ISSN 1450-216X Vol.31 No.4 (2009), pp.566-576

[13] N.P. Bobade and N.N. Mhala, “Performance Evaluation of Ad Hoc On Demand Distance Vector in MANETs with varying Network Size using NS-2 Simulation”, International Journal on Computer Science and Engineering, Vol. 02, No. 08, 2010, 2731-2735

[14] A.K. Gupta, H. Sadawarti, and A.K. Verma, “Performance analysis of AODV, DSR & TORA Routing Protocols”, International Journal of Engineering and Technology, Vol.2, No.2, April 2010, ISSN: 1793-8236

[15] S. Shah, A. Khandre, M. Shirole, and G. Bhole, “Performance Evaluation of Ad Hoc Routing Protocols Using NS2 Simulation”, Mobile and Pervasive Computing (CoMPC–2008)

[16] C.E. Perkins and P. Bhagwat, “Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers”, In Proceedings of the SIGCOMM 94, August 1994

[17] S.S. Tyagi and R.K. Chauhan, “Performance Analysis of Proactive and Reactive Routing Protocols for Ad hoc Networks”, International Journal of Computer Applications, Vol. 1, No. 4, 2010