A Hybrid Objective Function of ETX and Hop-Count for Optimizing RPL

February 22, 2022

1 Introduction

RPL [1] is the IPv6 routing protocol for low-power and lossy networks (LLNs) standardized by IETF in 2012. Designed to fit the high loss rates, low data rates, and instability characteristic of the LLN, the standard provides a routing mechanism to support multiple types of traffic in LLN. To support the MultiPoint-to-Point (MP2P) type traffic, the RPL uses Objective function (OF) to maintain the upward routes and form a Direction Oriented Directed Acyclic Graph (DODAG). In the current standardization, two OFs are proposed, Objective Function Zero (OF0) [2] and the Minimum Rank with Hysteresis Objective Function (MRHOF) [3].

However, the standards only provision how a routing metric can be converted to rank and how nodes pick their parents to form the DODAG. Which routing metric should be used for OF is completely application and implementation dependent. In the project, we examined the implementation of RPL-lite, the default routing module for Contiki-NG, and explored a hybrid objective function. Our simulation experiments shows that the proposed metric is better than the default Contiki-NG implementation for routing performance in terms of Packet Deliver Rate (PDR) and average latency. The code and simulation files are avaibale at `https://github.com/NKWBTB/MRHOF_logETX_HOP`.

2 Backgrounds

RPL is a distance-vector based proactive routing protocol. RPL uses distance vector rather than link state because the storage of link state information requires a lot of memory resources which a sensor node in LLN is usually not equipped with. The proactive protocol builds and maintains the path before an application data is sent or received.

Without a predefined topology, the RPL discovers links and organizes the network into a topology of Directed Acyclic Graph (DAG), the DAG has a single root which often acts as the LLN border router (LBR). In the standard, the single rooted DAG is called Direction Oriented Directed Acyclic Graph (DODAG). The upward routes refer to the route carrying traffic from a normal node to the root node. Such traffic pattern is common and can be dominant in some applications where sensor nodes periodically report the readings to the control center root. To form the upward route, each normal node in the DODAG maintains a list of candidate parents from its neighbours and chooses a preferred parent to forward the packets towards the root. The selection of the preferred parent is determined by the Objective Function (OF) and the routing metric, and is critical for the LLN to meet up with the application's requirements. The OF also specifies how rank is calculated for each node, the rank tells the relative position of a node to the root and is used for loop avoidance.

In details, the process for building the DODAG follows:

1. A node starts running as DAG root, it will advertise the DAG with DIOs (DODAG Information Object) to its neighbours using link local multicast.
2. A node hearing a DIO might choose to join the DAG. The node adds the sender to the candidate parent list, updates its own rank. The node selects a neighbor as preferred parent after enough probing according to the selected OF and metric specified in the DIO.
3. The nodes then further multicast DIOs to other nodes and eventually all nodes join the DODAG.

RPL-lite used by Contiki-NG implemented the two standardized OFs, OF0 and MRHOF:

- In **OF0**, a node N's rank $R(N)$ is calculated based on the rank of its preferred parent $R(P)$:
  $$R(N) = R(P) + rank\_increase \tag{1}$$
  $$rank\_increase = (R_f * S_p + S_r) * MinHopRankIncrease \tag{2}$$
  where R_f and S_r are normalization factors and S_p is link metric. The OF0 always selects the neighbour with least rank as a node's preferred parent. The OF0 does not specify what metric should be used, the RPL-lite uses Expected Transmission Count (ETX).
- **MRHOF** calculates the path cost adding the link metric and the value advertised in the Metric Container sent with DIO. The node selects the preferred parent with lowest path cost. In addition, the MRHOF introduces hysteresis when switching parents to increase stability of the network. In the RPL-lite implementation, the preferred parent is switched if the new preferred parent passes a rank threshold or time threshold with respect to the old preferred parent. The RPL-lite also uses ETX as the default routing metric for MRHOF.

3 Proposed metric

Both OF0 and MRHOF implemented in RPL-lite use Expected Transmission Count (ETX) as the routing metric. It is the expected number of transmissions a node to make to a destination in order to successfully deliver a packet. The ETX metric is often used for a node to estimate its neighbors' link quality.

The default implementation of RPL-lite calculates the path cost by summing the ETX of each edge in the path. This raises some questions simply using the addition of ETX for path selection. The ETX only represents the expected transmission number for a single edge, the sum of ETXs along the path does not represent the expected transmission needed for the path. The path selected by the accumulative ETX may introduce long single hop [4] which can be a bottleneck for the network. In the work [4], they provide an example: from node A to B, there are two paths where the first path has an accumulative ETX of

$$ETX_{AB1} = 2.2 + 2.2 + 2.1 = 6.5$$

and the second path has

$$ETX_{AB2} = 3 + 3 = 6$$

In this case, the OF will pick the second path $AB2$, however the first path provides a better average link quality. The work [4] provides a simple workaround by using the average ETX along the path. However, another work [5] points out that using the average ETX violates the monotonicity of the path cost. This means a child node's path cost can be smaller than its parent, this could lead to excessive churn in the network (parent child relationship reverse frequently).

In this project, we explore to improve the usage of ETX in the RPL without the violation of monotonicity in path cost. The ETX can be converted to the packet error probability e_{pt}:

$$ETX = \frac{1}{1 - e_{pt}}$$

This gives the probability of a successful transmission over an edge i without an error:

$$P_i = 1 - e_{pt_i}$$

For a path l consists of edge $1...n$, under independence assumption, the probability of the successful transmission over the path is therefore:

$$P_l = \prod_{i=1}^{n} P_i$$

Maximizing path transmission success probability P_l is to minimize $1/P_l$:

$$\frac{1}{P_l} = \prod_{i=1}^{n} ETX_i$$

Following the derivation, we can see that to select path with best link quality is actually to find path with minimum product of ETXs.

Directly using the product of ETX for path selection is not practically for two reasons. First, the OF0 and MRHOF are not designed for multiplicative metrics. Second, the product grows exponentially as the number of hops increases, the metric could quickly drain the field used for storing the metric. Therefore in the implementation we propose the sum of logarithm ETX for maximizing path link quality:

$$Metric_l = \log(\prod_{i=1}^{n} ETX_i) = \sum_{i=1}^{n} \log ETX_i$$

In addition, using path link quality alone could lead to selecting paths with large number of hops, this is not desired if latency is required by applications. So we further incorporate hop count in the designed metric:

$$Metric_l = \sum_{i=1}^{n} (\log ETX_i + HopIncrease)$$

where $HopIncrease$ is an adjustable non-negative constant for weighting the importance of hop count in the metric, we use 1 for simplicity. This metric is clearly monotonic because:

$$\begin{aligned} ETX_i &\geq 1 \\ \log ETX_i &\geq 0 \\ HopIncrease &\geq 0 \end{aligned}$$

4 Simulations

We evaluate the proposed metric in a simulation environment using cooja [6] simulator.

4.1 Setup

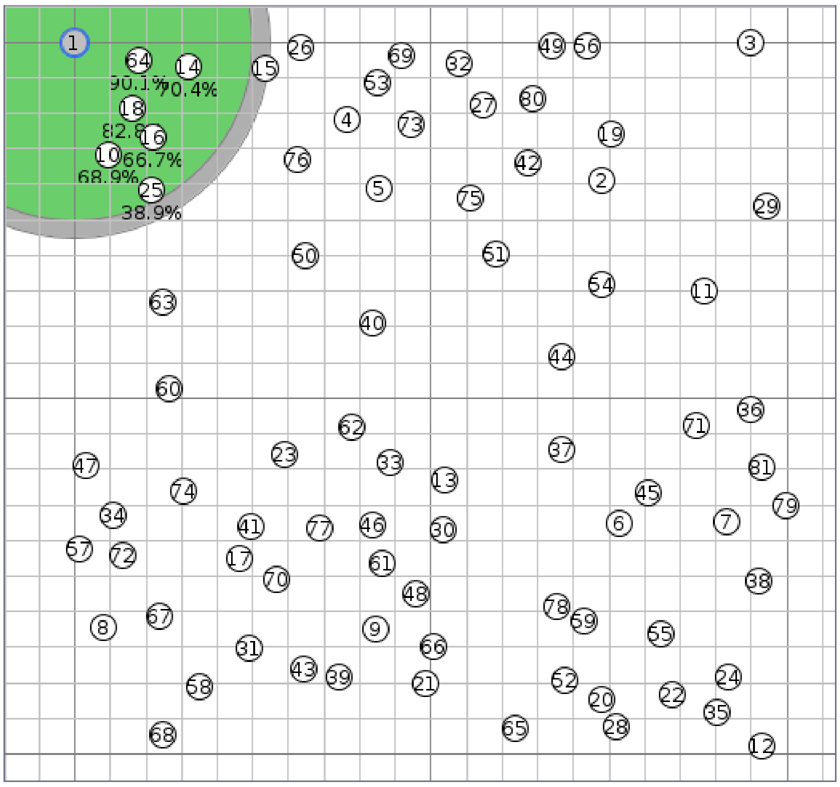

Figure 1: Physical layout of the LLN for simulation.

Following a similar simulation settings in [7], our simulation includes 80 client nodes and one server node as the DODAG root. The physical layout is shown in Figure 1. The root node is labeled as 1 and placed at the top left $(0, 0)$ position. The 80 client nodes are randomly placed with both x and y coordinates ranging from 0 to 200.

The application running in the simulation is modified from an official example provided by the Contiki-NG repository. The root acts as a UDP server and the clients run a UDP client. After an initial delay time of $65s$, the client starts to send packets periodically to the server with an interval of $8s$. The application UDP packet simply includes a counter as payload. When receiving a packet, the server silently writes the packet payload to log without making any response. The initial delay is set so there is enough time for the network to converge. A random delay or prepone of $\pm 1s$ is applied when sending the packets of clients.

Parameter	Value
Mote Type	Cooja mote
MAC	IEEE 802.15.4 CSMA/CA
RPL_CONF_MOP	RPL_MOP_NO_DOWNWARD_ROUTES

Table 1: Other simulation settings

The Unit Disk Graph Model (UDGM) is used to introduce distance loss on the radio medium. The TX range is set at 50 while the interference range is 55. We keep the TX Ratio at 100% while changing RX Ratio level from 30% to 100% to test the effect of different level of lossiness.

We set the simulation time for an hour. See Table 1 for other simulation settings. Once the simulation is done, we calculate the performance metrics parsed from the log file generated from the simulator using a Python script. The code and automated simulation scripts are available at `https://github.com/NKWBTB/MRHOF_logETX_HOP`.

4.2 Performance evaluation and baselines

By parsing the simulation log, we are able to evaluate the average Packet Delivery Rate (PDR) and the average latency in the network. Each packet in the network can be uniquely identified by a tuple (Sender address, count). The total number of packet sent is obtained by counting the tuples from the client nodes, and the total number of packet received can be obtained similarly by counting the tuples from the server node. Thus, the average PDR is calculated as:

$$AveragePDR = \frac{\#\{\text{Total packets received}\}}{\#\{\text{Total packets sent}\}}$$

The log also contained the timestamp for the packets sent and received. The average latency can be obtained by:

$$AverageLatency = \frac{1}{|S|}\sum_{k\in S}(RecvTime(k) - SentTime(k))$$

where S is the set of received packets.

We compare the proposed OF with the default implementation of RPL-lite, namely, **OF0** and **MRHOF-ETX** which both use accumulative ETX. The RPL-lite also provide an option to use accumulative square of ETX (MRHOF-ETX^2) to penalize links with bad quality. We include the results of **MRHOF-ETX^2** for comparison. Our proposed OF, called **MRHOF-logETX+Hop**, is also based on MRHOF. We are further interested the effect of each individual component in our metric, so the results for **MRHOF-Hop** using hop count alone, and **MRHOF-logETX** using the logarithm of ETX alone are included for study.

5 Results

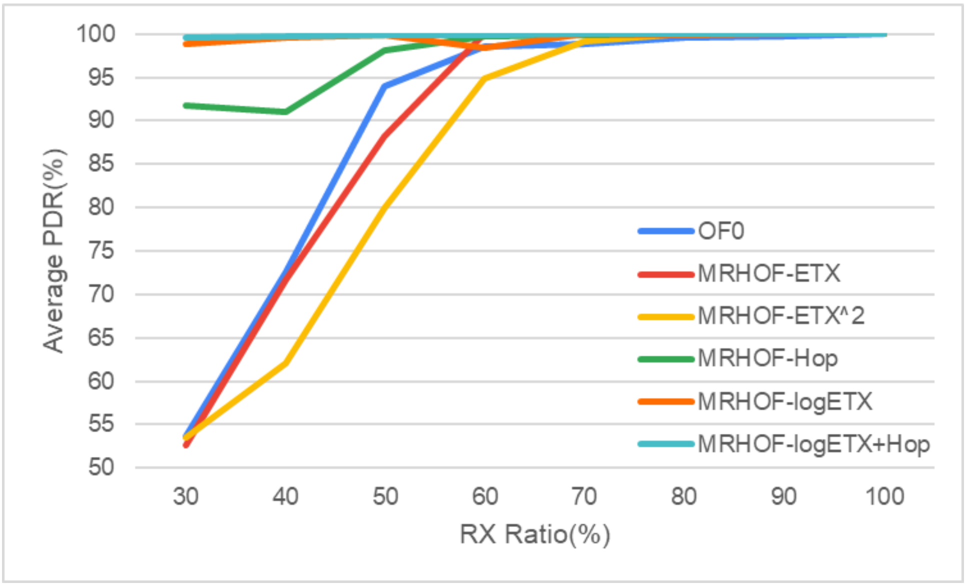

Figure 2: Average Packet Delivery Rate.

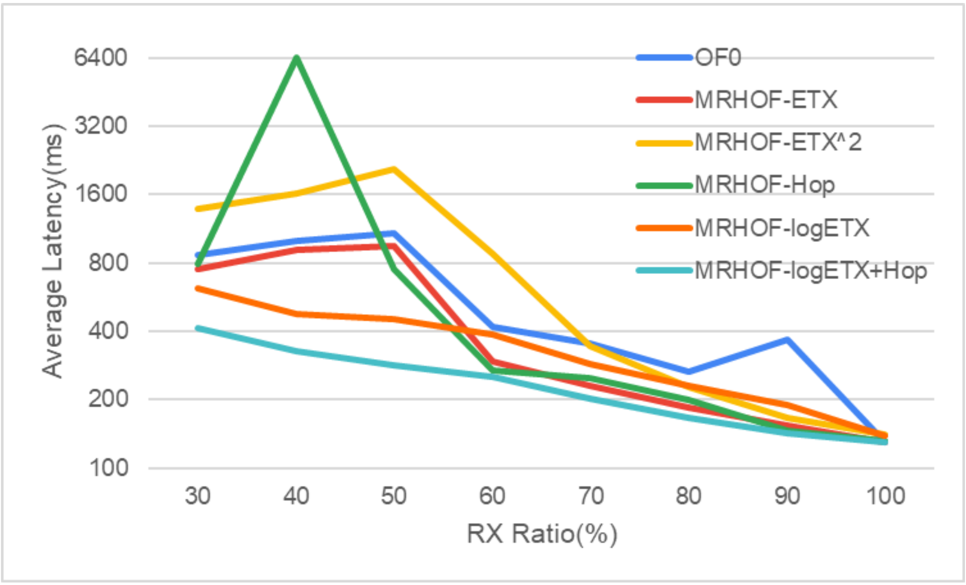

Figure 3: Average Latency.

The results for simulation is shown at Figure 2 and Figure 3. In terms of average PDR, all methods performs similarly when RX Ratio $\geq 80\%$. However the performance of OF0, MRHOF-ETX and MRHOF-ETX^2 start to degrade when the RX Ratio drops below 70%. The MRHOF-Hop keeps an average PDR $\geq 90\%$, moreover, the MRHOF-logETX and MRHOF-logETX+Hop are able to maintain a PDR $\geq 98\%$ at all test conditions.

For the average latency, the MRHOF-logETX+Hop, MRHOF-Hop and MRHOF-ETX performs best when the RX Ratio is $\geq 60\%$. When the RX Ratio keeps to drop, only MRHOF-logETX and MRHOF-logETX+Hop can keep a latency below 700ms. Further more the MRHOF-logETX+Hop is able to keep the average latency around 400ms when the RX Ratio is only 30%.

Finally, we can tell from the results the logETX metric can provide high average PDR, and provide a good resistance to link quality drop considering latency. While the hop count metric can provide low latency when the link quality is good. Combining the two metrics, the proposed MRHOF-logETX+Hop outperforms all other baselines in terms of average PDR and latency.

6 Conclusions

In this project, we explored using a hybrid object function involving ETX and hop count in RPL. The cooja simulation results show that the proposed OF outperforms current implementation of RPL-lite, the default routing module of Contiki-NG, with respect to average PDR and latency. For further work, the evaluation on a real testbed should be conducted. The scalability when the number of nodes expands can also be considered.

References

[1] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks." RFC 6550, Mar. 2012.

[2] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)." RFC 6552, Mar. 2012.

[3] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function." RFC 6719, Sept. 2012.

[4] W. Xiao, J. Liu, N. Jiang, and H. Shi, "An optimization of the object function for routing protocol of low-power and lossy networks," in *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, pp. 515–519, 2014.

[5] B. Ghaleb, A. Y. Al-Dubai, E. Ekonomou, A. Alsarhan, Y. Nasser, L. M. Mackenzie, and A. Boukerche, "A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1607–1635, 2019.

[6] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pp. 641–648, 2006.

[7] H. Ali, "A performance evaluation of rpl in contiki: A cooja simulation based study," *School of Computing, Blekinge Institute of Technology*, 2012.