VOL. 10, NO. 21, NOVEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

INVESTIGATION ON ROUTING ASPECTS TOWARDS RPL OPTIMIZATION

Wan Fariza binti, Wan Abdul Rahman[1], Md. Rafiqul Islam[2] and Aisha Hassan Abdalla[2]
[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia
[2]Department of Electrical and Computer Engineering, Kuliyyah of Engineering, International Islamic University of Malaysia, Malaysia
E-Mail: wfariza@kelantan.uitm.edu.my

ABSTRACT

Routing in Low-power and Lossy Networks (LLNs) requires low overhead on data packets, low routing overhead, minimal memory and computation requirements, and support for sleeping nodes considering battery saving. Most of the devices are distinguished by their low bandwidth, short range, scarce memory capacity, limited processing capability and other attributes of inexpensive hardware. These devices are designed to be compatible with the IEEE 802.15.4 standard. IPv6 over IEEE 802.15.4 has been defined to carry IPv6 packets over IEEE 802.15.4 and similar networks, due to its capability to support routing over possibly various types of interconnected links. The IETF Routing Over Low Power and Lossy Networks (ROLL) working group has designed the IPv6 route-over Routing Protocol for LLNs, known as RPL, which covers the routing requirements of all application domains. However, there are still a number of routing aspects to be tackled in RPL, including memory efficiency, routing overhead and loops occurrence. Therefore, the purpose of this paper is to highlight these issues and investigate the efforts/approaches for solving them.

Keywords: LLN, RPL optimization, loop repair, routing overhead, memory efficiency, asymmetrical links.

INTRODUCTION

Different characteristics of LLNs offer unique challenges to a routing solution. This type of network consist large number of constrained nodes with limited processing power and memory. The routers are interconnected by lossy links, typically supporting only low data rates, which are unstable with relatively low packet delivery rates [1].

The IETF ROLL working group has defined an IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) as a standard routing for LLNs. RPL is based on a Directed Acyclic Graph (DAG) having the property that all edges are oriented in such a way that no cycles exist. All edges are contained in paths oriented toward and terminating at one or more root nodes (DAG root). All DAGs must have at least one DAG root and all paths terminate at a DAG root. The term Destination-Oriented DAG (DODAG) refers to a DAG rooted at a single destination with no outgoing edges. Two important elements in RPL are Rank and Objective Function (OF). A node's Rank defines the node's individual position relative to other nodes with respect to a DODAG root. Rank strictly decreases in the Up direction (from leaf nodes towards DODAG root) and strictly increases in the Down direction (from DODAG root towards leaf nodes). An OF defines how routing metrics, optimization objectives, and related functions are used to compute Rank. Rank is not necessarily a good indication of a distance or path cost to the root. The stability of the Rank determines the stability of the routing topology. RPL nodes construct and maintain these DODAGs through DODAG Information Object (DIO) messages [1].

Nodes advertise their presence, routing cost and related metrics by sending link-local multicast DIO messages to all-RPL-nodes. Nodes listen for DIOs and use their information to join a new DODAG (selecting DODAG parents) or to maintain an existing DODAGs (based on specified OF and Rank of their neighbours) [1].
To establish Downward routes, RPL uses Destination Advertisement Object (DAO) messages to propagate destination information Upward along the DODAG. DAO messages are an optional feature for applications that require point-to-multipoint (P2MP) and or point-to-point (P2P) traffic. The next-hop destinations of these DAO messages are called DAO parents. The Downward traffic can be supported either in Storing (fully stateful) or Non-Storing (fully source routed) mode. In both modes, P2P packets travel Up toward a DODAG root then Down to the final destination (unless the destination is on the Upward route). A simple one-hop P2P optimization is allowed for both modes in which a node may send a P2P packet destined to a one-hop neighbour directly to that node [1].

STORING VERSUS NON-STORING MODE

In Storing mode, packet is directed Down towards the destination by a common ancestor of the source and the destination prior to reaching a DODAG root. In Non-Storing mode, the packet has to travel all the way to a DODAG root before travelling Down. In Storing mode, the DAO message is unicast by a child to the selected parent(s). A node must not address unicast DAO messages to nodes that are not its DAO parents. In Non-Storing mode, the DAO message is unicast to the DODAG root. Standard RPL defined in [1] stated that no implementation is expected to support both Storing and Non-Storing modes.

In Storing mode, all non-root, non-leaf nodes store Downward routing tables for their sub-DODAG and destinations learned from DAOs. Each hop on the Downward route in a Storing network examines its routing table to decide on the next hop. In Non-Storing mode, nodes do not store Downward routing tables. Only the

DODAG root should store source routing table entries for destinations learned from DAOs. The DODAG root therefore must be able to generate source routes for those destinations. Downward packets are routed with source routes populated by a DODAG root [1].

The Transit Information option in DAO message is used for a node to indicate its DODAG parents to an ancestor that is collecting DODAG routing information for the purpose of constructing source routes. In Non-Storing mode, the ancestor will be the DODAG root, and this option is carried by the DAO message. A Non-Storing node that has more than one DAO parent may include a Transit Information option for each DAO parent with a preference among parents. The preference may influence the decision of the DODAG root when selecting among the alternate parents for constructing Downward routes. When a node removes a node from its DAO parent set, it may generate a new DAO message with an updated Transit Information option. In the Storing mode, the DODAG Parent Address subfield is not needed because the DAO message is sent directly to the parent. When a Storing mode node removes a node from its DAO parent set, it should send a No-Path DAO message to that removed DAO parent to invalidate the existing route [1].
In Non-Storing mode, the root builds a strict source routing header, hop-by-hop, by recursively looking up one-hop information that ties a Target and a transit address together [1].

All nodes joining the DODAG must be able to honour the Mode of Operation (MOP) as administratively provisioned at and distributed by the DODAG root in order to fully participate as a router, or else they must only join as a leaf. The DIO message (on occasion) of a leaf node must be advertised with an INFINITE_RANK (in order to avoid other nodes from selecting this node as their parents) [1].

ROUTING PATHOLOGY

Although initial RPL standard does not support mixed-mode of operation where some nodes source route and other store routing tables, routing pathology issues in a mixed network of Storing and Non-Storing nodes has been discussed in [2].

In practical, as the size of LLN deployments increase, a homogeneous Non-Storing mode network will introduce a high level of communication overhead, and a homogeneous Storing mode network will require too much memory resources. The primary advantage of the Non-Storing mode is that it requires very little memory for Storing routing states on the resource-constrained embedded devices with limited processing and storage capabilities. However, the Non-Storing mode requires a source routing header to be attached to all packets which not only increase the packet size but also cause the packet size to be variable depending on the path length. This in turn decreases the effective maximum transmission unit (MTU) of the packet. On the other hand, the Storing mode does not have the long-route problem since it does not need source routing header. However, each RPL node must store route information to all destinations in its own subtree, which may be too demanding for the limited memory constraints of small embedded devices. A More Memory-efficient Storing mode RPL; MERPL has been proposed in [3] by ensuring that the number of routing table entry stored in a node does not exceed a pre-specified factor of N. When the number of routing table entries to be stored is larger than N, the node will transfer part of its responsibility to the selected child (based on the number of routing table entries maintained at child node). However, no clear approach on how to determine the value of N has been defined.

It is stated in [2] that a more efficient network can be achieved by allowing a mixed of computationally powerful nodes with route Storing capabilities and low-cost nodes that do not need to maintain a routing table. However, allowing a mix of nodes operating in Storing and Non-Storing modes to form a single network can cause a routing pathology. Routing pathology can partition the network due to the scenarios where nodes cannot send packets to the root and the root cannot send packets to the nodes even though they have plenty of multi-hop physical connectivity in the network.

Three problems in a mixed network are; (i) ‘cannot route through leaf problem’, (ii) ‘no source route header problem’, (iii) ‘DAO not processed problem’. As stated in standard RPL, when a node attaches itself as a leaf (due to different MOPs), the node cannot act a router. Leaf nodes may send their data to their next hop, but may never accept data for forwarding. This is not an issue if the node is at the fringe of the network. However, the problem occurs when the node is somewhere in the middle of a forwarding path and the node has a subtree of nodes that needs to connect to itself for routing. This is known as ‘cannot route through leaf’ problem [4].

In case of Downward routing, ‘cannot route through leaf’ problem also occurs with two additional problems. Consider a mixed network with a Storing mode root. A Non-Storing node attached itself as a leaf (in the middle of this network) is not only incapable of forwarding the packets from root to its subtree because it is a leaf, but also due to no knowledge about of any routes to its subtrees. The Non-Storing node neither received nor processed any DAO messages from its subtrees and thus never stored any routing table entries for the nodes in its subtree. This problem is known as ‘no source route header’ problem plus ‘DAO not processed’ problem [4].

In another situation, consider a mixed network with a Non-Storing root. The packets (going Downward) from the root has a source routing header. However, a Storing node joined the network as a leaf, does not process those routing headers, refraining the root from reaching its subtree. The Storing mode node ignores the source routing header because this element is not used in Storing mode operations. This problem is known as ‘ignore source route’ problem [4].

To eliminate the network partition problem and preserve the high bidirectional data delivery performance, a few solutions have been proposed in [2] and [4]. The

first solution is to enhance the capability of a leaf as a router. In this 'leaf as a router' scheme, Storing mode node in a network with Non-Storing root, and Non-Storing mode node in a network with Storing root will no longer use infinity as its Rank. However, implementing a single 'leaf as a router' scheme does not solve the whole problem in a mixed network. There are another four enhancements suggested; (i) modified DAO transmission, (ii) modified DAO format, (iii) modified source routing header support, and (iv) Storing mode flag.

Modified DAO transmission requires the Non-Storing mode nodes to send DAOs hop-by-hop, giving a chance to all Storing mode nodes along the path towards the root to store the DAO information. This allows all DAO messages, from both Storing and Non-Storing mode nodes to be processed at intermediate nodes, thus solves the 'DAO not processed' problem [4].

It is stated in RPL standard that the Parent Address field in the Transit Information option is optional for Storing mode operation. However, that information is needed in Non-Storing mode operation so that the root can construct the routing paths and include it in the source routing header. Thus, modified DAO format requires all DAOs from both Storing and Non-Storing mode nodes to include the Transit Information option's address field in order to resolve the 'no source routing header' problem for packets being sent by Storing mode nodes [4].

Apart from the above modified DAO format, to solve the problem of 'ignore source route' and 'no source route', all Storing mode nodes should implement source routing header support by understanding and following the routing information included in the source routing header (if present) [4].

Another enhancement (optional) is Storing mode flag. This 1-bit flag in DAO message is set when DAO initiating node supports Storing mode, and cleared otherwise. A Storing node receiving this DAO message is required to store this flag information. The purpose is to construct a more efficient Downward routes. The source routing header may be constructed only up to the next Storing node rather than constructing a full end-to-end routing path which can be longer [4].

DATA-PATH VALIDATION AND LOOP OCCURRENCE

The occurrence of loop is another concerned issue in RPL. Avoiding loop is important to prevent packets from being forwarded between two nodes without any progress. Although RPL tries to avoid creating loops when undergoing topology changes, practically it guarantees neither loop-free path selection. However, RPL can detect and repair a loop using Rank-based data path validation [1].

RPL uses on-demand loop detection using data packets due to the low-power and lossy nature of LLNs. Maintaining a routing topology that is constantly up-to-date with the physical topology can waste energy. Thus, transient and infrequent changes in connectivity need not be addressed by RPL until there is data to send [1].

The Rank is used to avoid and detect loops. If the Rank of node X is less than the Rank of node Y, the position of node X is closer to the DODAG root than the position of node Y. In this case, node X may safely be a DODAG parent for node Y without risk of creating a loop. If the Rank of node X equals to the Rank of node Y, their positions with respect to the root are identical. Routing through a node with equal (or greater) Rank may cause a routing loop. Particularly, the Rank of the nodes must monotonically decrease towards the DODAG destination. An inconsistency between the routing decision of a packet (Upward or Downward) and the Rank relationship between the source and destination nodes indicates a possible loop. A local repair is triggered on receiving such a packet [1].

A DODAG loop may occur when a node detaches from the DODAG and reattaches to a device in its prior sub-DODAG. This may happen when DIO messages are missed. Consider a local repair mechanism that allows a node to detach from the DODAG, advertise a Rank of infinity (in order to poison its route and inform its sub-DODAG) and then reattach to the DODAG. However, the poisoning may fail if the INFINITE_RANK advertisements are lost. The detached node may reattach to its own prior sub-DODAG, thus causing a DODAG loop. The Rank-based data-path validation mechanism can be used to detect and trigger correction of the loop [1].

A DAO loop may occur when a parent has a route installed upon receiving and processing a DAO message from a child, but the child has subsequently cleaned up the related DAO state. This is due to the missing No-Path (a DAO message that invalidates a previously announced prefix). To mitigate the impact of a single DAO message being missed, an optional mechanism to acknowledge DAO messages is included in RPL [1].

DAG INCONSISTENCY LOOP DETECTION

RPL includes a reactive loop detection technique that triggers repair of broken paths. The RPL Packet Information that is transported within the data packet is used to detect loop. A receiver detects a DODAG inconsistency if the direction of a packet does not match the Rank relationship based on Down 'O' and Sender Rank fields in the RPL Packet Information. Down 'O' is 1-bit flag indicating whether the packet is expected to progress Up or Down. The 'O' flag is set when the packet is expected to progress Down, and cleared when forwarding toward the DODAG root. A SenderRank is 16-bit field set to zero by the source and to DAGRank(rank) by a router that forwards inside the RPL network. There is an inconsistency if a packet is received with either the 'O' bit set from a node of a higher Rank, or the 'O' bit cleared from a node of a lower Rank [1].

DODAG GLOBAL REPAIR

A global repair is initiated by the DODAG root by incrementing the DODAGVersionNumber. The increment in DODAGVersionNumber results in a different DODAG topology and a new DODAG Version spreads

outward from the DODAG root. A parent that advertises the new DODAGVersionNumber cannot belong to the sub-DODAG of a node advertising an older DODAGVersionNumber. Therefore, a node can safely add a parent of any Rank with a newer DODAGVersionNumber without forming a loop [1].

Consider a situation where a node has left a DODAG with DODAGVersionNumber N. The node had a sub-DODAG and did attempt to poison that sub-DODAG by advertising a Rank of INFINITE_RANK, but those advertisements may have become lost. Then, if the node did observe a candidate neighbour advertising a position in that previous DODAG with DODAGVersionNumber N, that candidate neighbour could possibly has been in the node's former sub-DODAG, and there is a possible case where adding that candidate neighbour as a parent could cause a loop. However, if that candidate neighbour is observed to advertise a DODAGVersionNumber N + 1, then that candidate is certain to be safe, as it has been able to increment the DODAGVersionNumber by listening from the DODAG root while the original node was detached [1].

A temporary Rank discontinuity may form between the next DODAG Version and the prior DODAG Version when the DODAG root increments the DODAGVersionNumber. This is due to the decision of nodes in adjusting their Rank in the next DODAG Version and deferring their migration into the next DODAG Version [1].

When a router that is still a member of the prior DODAG Version forward a packet to a (future) parent in the next DODAG Version, the parent may detect an inconsistency because the Rank ordering in the prior DODAG Version is not necessarily the same as in the next DODAG Version. If the sending router is aware that the chosen successor has already joined the next DODAG Version, it must update the SenderRank to INFINITE_RANK as it forwards the packet to avoid false detection of Rank inconsistency [1].

A packet with one inconsistency detected along the path may still continue as it is not considered as a critical error. However, if there is a second detection along the path of the same packet, the packet must be dropped. This is done based on the Rank-Error bit associated with the packet. When an inconsistency is detected, if the Rank-Error bit was not set, then the bit is set. If the Rank-Error bit was set already, then the packet will be discarded [1].

DAO INCONSISTENCY RECOVERY

DAO inconsistency loop recovery applies to Storing mode only. A packet can be used to recursively explore and clean up the obsolete DAO states along a sub-DODAG. When DAO inconsistency is detected, the router should send the packet back to the parent that passed it with the Forwarding-Error 'F' bit set. Forwarding-Error 'F' bit is 1-bit flag indicating that this node cannot forward the packet further towards the destination. Otherwise, the router must silently discard the packet [1].

Upon receiving a packet with a Forwarding-Error bit set, the (parent) node removes the routing states that caused forwarding to that neighbour, clears the Forwarding-Error bit and attempts to send the packet again via an alternate neighbour. If the alternate neighbour still has an inconsistent DAO state via this node, the previous process will recursively repeats. In Non-Storing mode, the packets are source routed to the destination. Therefore, DAO inconsistencies are not corrected locally. Instead, an ICMP error with a new code "Error in Source Routing Header" is sent back to the root [1].

LOOP FREE LOCAL REPAIR

DODAG loops caused by local DODAG repair mechanism are an issue to be addressed. As mentioned above, the cause of DODAG loops comes from Rank increase by DODAG local repair mechanism. The idea of loop prevention, detection and avoidance has been emphasized in [5]. However, each approach involved with certain costs. The loop prevention approach requires a node to wait for a sequence number update by the DODAG root (global repair) before increasing its Rank in order to choose new parents. The loop detection approach puts costs on the already small available space for carrying the data by requiring the node tags to be carried in the packet. On the other hand, the loop avoidance requires dismantling the sub-DAG rooted at the node performing the Rank increase which can be too pricy if resolved quickly with a minor change in DAG structure. Using this loop avoidance approach, whenever any node needs to increase its Rank, it starts a wait timer and generates a new DIO advertising an infinite Rank, thereby detaching itself from the DAG. As the children node receives this DIO, they either remove the node from their parent set or detach from the DAG themselves. New DIOs will be generated by the children node within specific interval (wait time) to advertise their new status. Therefore, if the wait time is large enough, new Rank can be chosen correctly without creating any loops. However, the loop may still occur if the DIOs lost or the wait time is not large enough. Based on simulations performed in [5], loop avoidance in a DAG based routing protocol is not recommended due to the turmoil caused by dismantling of the sub-DAGs in order to increase Ranks may be much more than what the routing loops themselves will cause. This is due to the generation of large number DIOs during the stabilization times resulting from large number of affected nodes.

As mentioned earlier, the reason for RPL to repair loops only when detected is to reduce control overhead. However, when repairing loops is done only when detected, the forward progress of data packets would be delayed by the triggered local repair mechanism, thus, increasing end-to-end delay. In addition, the data packet has to be buffered during repair. The first problem will give a bad impact for real-time application such as alarm signals in which increased delay may be undesirable. The second issue, buffering incoming packets during the route repair process may not be possible for all incoming data packets, thus, leading to dropped packets. This is due to

the nature of RPL which is supposed to run on LLN routers with memory constraint [6].

Furthermore, a global repair (by increasing the DODAGVersionNumber) in Non-Storing mode leads to an increased control overhead in the network caused by DIO messages. Thus, results in a possible energy drain of the routers and congestion of the channel. According to [6], the above mentioned effects of loop detection in RPL may be minimized if carefully implemented with respect to avoiding loops before they occur. Another method for repairing DODAG locally without causing any DODAG loops has been proposed in [7]. Instead of increasing the Rank as in standard RPL, the method proposed does not increase the Rank. Thus, it is loop free. Similar to standard RPL, when a DODAG parent becomes unreachable, a node may switch to another DODAG parent for upward traffic. However, the node first has to transmit a DODAG Repair Request (DRQ) message via link-local multicasting to all-RPL-nodes to locally repair the DODAG [7].

If the receiving the DRQ message, a link-local neighbouring router will act accordingly based on four cases. First, if a link-local neighbouring router is the DODAG root, it accepts the DRQ message and generates a DODAG Repair Reply (DRP) message. Second, a link-local neighbouring router (which is not the DODAG root) discards the DRQ message if it does not have any DODAG parent. Third, if the link-local neighbouring router is not the DODAG root, has non-empty DODAG parent set and its Rank is lower than Rank of the node generating the DRQ message (RankQ), it accepts the DRQ message and generates a DRP message. Fourth, if the link-local neighbouring router is not the DODAG root, has non-empty DODAG parent set and its Rank is greater or equal to RankQ, it forwards the DRQ message to its preferred DODAG parent. The forwarding process continues until the DRQ message reaches a node with the first, second or third condition above [7].

The DRP message is unicast. In Storing mode, the DRP message is transmitted to the DRP message generator by using a Downward routing table, whereas in Non-Storing mode, it is transmitted by source routing approach via Path option [7].

Upon receiving a DRP message, a node first performs filtering process. Only if the DRP message passes the filtering process, it will be further processed. The DRP message will be discarded if, (i) the RPLInstanceID or DODAGVersionNumber or DODAGID of the DRP message is different from the value maintained by the receiving node, (ii) the DRP message was already received, or (iii) the receiving node is a leaf node and is not a DRQ message generator [7].

If the receiving node is the DRQ message generator and the DRP message sender is not in its DODAG parent set, it may add the DRP message sender into its DODAG parent set and select a new preferred parent. If the DRP message is received by a router with a Rank lower than RankQ which has a Downward route entry to node DRPID, the router updates the Rank of the node transmitting the DRP message (RankP) to its own Rank and forwards the DRP message Downward to the next-hop node. In another case, if the receiving router has a Downward route entry to node DRPID but its Rank is greater than or equal to RankQ, it checks whether it can decrease its Rank such that RankQ > its Rank > RankP. If no, it just discards the DRP message. If yes, the receiving router decreases its Rank to an appropriate value, and adds the DRP message sender into its DODAG parent set if the sender is not in its DODAG parent set. Any DODAG parent whose Rank is greater than or equal to its new Rank, will be removed. If its preferred parent is removed, a new preferred parent will be selected. The receiving router than updates the RankP in the DRP message to its new Rank and forwards the DRP message to the next hop. However, if the receiving router is not the DRQ message generator, and it has no route entry to a node DRPID in its Downward routing table, it discards the DRP message [7].

Using NS-2 simulator, the loop-free local repair method proposed in [7] is claimed to have lower routing overhead and shorter end-to-end delay. The performance could be further verified in future by considering some common real-time applications requirements.

ASYMMETRICAL LINKS

Apart from loop avoidance issue, a RPL node should verify that bidirectional connectivity and adequate link quality is available with a candidate neighbour before considering that candidate as a DODAG parent [1]. A link is bidirectional when traffic confirmed possible in both direction. A topology must be very good for both Upwards and Downwards traffic; otherwise traffic between two nodes in the instance may suffer. However, a perfect symmetry is rarely present in LLNs, whether links are based on radios or power-line. A link is asymmetric if it is bidirectional, but exhibits significant differences in link characteristics for both directions [8].

An asymmetrical link can only be used for traffic in one direction, whereby cannot contribute to the routing topology. This results in an unoptimized use of bandwidth and/or reduction of the possible path diversity [8]. In order to fully utilize the network resources (i.e. available path), the asymmetrical links should also be considered in routing, but with extra efforts.

A single DAG is adequate for both Upwards and Downwards traffic when the link properties do not widely differ between the Upwards and Downwards directions. In handling the asymmetrical links, two DAGs; one for Upwards traffic and one for Downwards traffic should be constructed [8].

Having two DAGs is quite challenging since it would penalize peer-to-peer traffic that would have to go through the root in order to leave the Upwards instance and then re-enter at the root in order to join the Downwards instance. Going through the root stretches the path in Storing mode, but it is not an issue in Non-Storing mode since the packet already has to go through the root to load the routing header [8].

To avoid extra stretch through the root in Storing mode, it is required to allow Upward traffic to be

transferred from one instance to the next before reaching the root. Therefore, the two instances; Upwards and Downwards can be bound together with a parent-child relationship between the two instances, to transfer traffic from an instance onto another. The relationship is needed to ensure traffic that goes down does not generally go back up again [8].

A new flag bit is defined in DIO message to indicate that the DAG is directional. An OF that supports directional links should favor directional links over non directional links. It is recommended in [8] that the 'D' flag should be accounted for in the selection of the preferred parent; that is before considering parent that causes the lesser resulting Ranks of the node. However, there is an exception. In case there is no next hop for a packet going down, the packet will be dropped or sent back with an error along the wrong direction. To avoid this situation, the constraints that are applied to build the topology can be lowered. Although this is less efficient, it gives chance for traffic to still be transferred.

CONCLUSIONS

This paper highlighted several routing aspects to be tackled in *RPL*. Memory efficiency and less overhead could be achieved by considering a mixed mode network of *Storing* and Non-*Storing* nodes. However, the routing pathology should be handled wisely to ensure a smooth interoperability. For this purpose, a few elements need to modified for both *Storing and Non-Storing* nodes.

Apart from that, loop occurrence in *RPL* should be considered. If handled properly, loop avoidance could enhance the performance of *RPL* in terms of end-to-end delay, memory consumption for packet buffering, and routing overhead. Types of loop in *RPL* and the solutions proposed have been detailed. The issue of asymmetrical links has also been touched since considering only symmetrical links would lead to an inefficient use of network resources. Thus, proper use of asymmetrical links should be considered. It is hoped that the issues highlighted and approaches presented in this paper would be beneficial for further research in optimizing *RPL* in the future.

REFERENCES

[1] Winter T., Thubert P., Brandt A., Hui J., Kelsey R., Levis P., Pister K., Struik R., Vasseur J. and Alexander R. 2012. RFC 6550 – RPL, pp. 1–157.

[2] Ko J., Jeong J., Park J., Jun J., Kim N. and Gnawali O. 2014. RPL Routing Pathology in a Network with a Mix of Nodes Operating in Storing and Non-Storing Modes. draft-ko-roll-mix-network-pathology-04. pp. 1–8.

[3] Gan W., Shi Z., Zhang C., Sun L. and Ionescu D. 2013. MERPL: A More Memory-efficient Storing Mode in RPL. 19th IEEE International Conference on Networks (ICON) 2013, pp. 1–5.

[4] Ko J., Jeong J., Park J., Jun J., Gnawali O and Paek J. 2015. DualMOP-RPL: Supporting Multiple Modes of Downward Routing in a Single RPL Network. ACM Transactions on Sensor Networks, Vol. 11, No. 2, Article Vol. 39, pp. 1–20.

[5] Guo J., Orlik P. and Bhatti G. 2013. Loop Free DODAG Local Repair. draft-guo-roll-loop-free-dodag-repair-01, pp. 1–19.

[6] Clausen, T., Colin de Verdiere, A., Yi, J., Herberg, U. and Igarashi, Y. 2015. Observations of RPL: IPv6 Routing Protocol for Low Power and Lossy Networks. draft-clausen-lln-rpl-experiences-09, pp. 1–31.

[7] Xie W., Goyal M., Hosseini H., Martocci J., Bashir, Y., Baccelli, E. and Durresi, A. 2010. Proceedings - International Conference on Advanced Information Networking and Applications, AINA, pp. 1–8.

[8] Thubert P. 2012. RPL Adaptation for Asymmetrical Links. draft-thubert-roll-asymlink-01, pp. 1–9.