RPL-RP: RPL with Route Projection for Transversal Routing

Iliar Rabet*, Hossein Fotouhi *, Maryam Vahabi *, Mário Alves†, Mats Björkman*
†*Politécnico do Porto, (ISEP)* Portugal
Email: *{iliar.rabet, hossein.fotouhi, maryam.vahabi, mats.bjorkman}@mdh.se, †mjf@isep.ipp.pt

***Abstract*—Routing Protocol for Low-Power and Lossy Networks (RPL) as the most widely used routing protocol for constrained Internet of Things (IoT) devices optimizes the number of routing states that nodes maintain to minimize resource consumption. Given that the routes are optimized for data collection, this leads to selecting sub-optimal routes, particularly in case of east-west or "transversal" traffic. Additionally, RPL neglects interactions with a central entity in the network for monitoring or managing routes and enabling more flexibility and responsiveness to the system.**

In this paper, we present *RPL with Route Projection (RPL-RP)* that enables collecting siblings' relations at the root node in order to inject routing states to the routers. This backward-compatible RPL extension still favors collection-based traffic patterns but it enriches the way routing protocol handles other flow directions. We address different advantages of RPL-RP in contrast to standard RPL and evaluate its overhead and improvements in terms of end-to-end delay, control overhead and packet delivery ratio. Overall, RPL-RP halves the end-to-end delay and increases network reliability by 5% while increasing network overhead by only 3%.

***Index Terms*—Wireless Sensor Networks, Routing Protocol for Low-Power and Lossy Networks, RPL, Internet of Things, Route Projection**

I. Introduction

The current developments in the field of Internet of Things (IoT) promise to expand the applications with traffic patterns that are more complex than simple data collection that traditional networks are designed for. Hence, bringing ease-of-management and flexibility to the underlying infrastructure on top of which they operate is of utmost importance. However, the common protocol stack implementations barely support such features.

Low-Power and Lossy Networks (LLNs) are key components of IoT and provide wireless communication between sensors and actuators. IPv6 Routing Protocol for Low-power and lossy networks (RPL) [1] has long been adopted by LLNs for its energy efficiency and minimum resource requirements. LLNs are characterized by high loss and fluctuations in the links and applications that mandate low data rates and high scalability. Different IETF working groups have designed a stack of protocols including IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) that defines header compression for IPv6 on top of IEEE 802.15.4 Medium Access schemes. This protocol stack was not initially designed for ease-of-interaction with a central network manager since the nodes were independent entities running a distributed control plane.

RPL maintains a data structure named *Destination Oriented Directed Acyclic Graph* (DODAG) and the functionality starts with a root node transmitting a *DODAG Information Object* (DIO) packet. All the receiving nodes will choose the best parent towards the root based on an objective function and will transmit new DIO packets to further increase the range of DODAG. The upward flow from the nodes to the root can start once a DIO packet is received but for downward communication, the root needs to wait to receive a *Destination Advertisement Object* (DAO) packet.

There are some limitations inherent to the design of the RPL that are biased to the upward collection-based traffic and all the other traffic directions are restricted to transmit only along the DAG resulting in "stretched" routes. For downward communication or any-to-any communication, the nodes in RPL are programmed to be either in *Storing* or *Non-Storing* mode, where the former requires more memory for keeping track of the nodes in the sub-trees of each node. For routing between two non-storing nodes in the network, the packets have to be transmitted all the way up to the root and back down to the destination. Even in the storing mode, they have to find a common ancestor, which is not necessarily the optimal path. Figure 1 exhibits how different modes of RPL stretch the routes while siblings can promote point-to-point routing. Besides the routing mode, the common metrics used by the RPL's objective function prefer the routes that have a better upward connectivity while links can show asymmetric behavior in different directions [2].

Extensively computing all the routes for any-to-any communication does not abide by the resource constraints of the low-power nodes. A reasonable compromise is to have the root node injecting a handful of routing states into some of the in-network nodes when it perceives a partial yet sufficient knowledge of the network topology.

Another common limitation of RPL is the poor performance upon employing mobile nodes [3]–[5]. The Trickle algorithm [6] controls the frequency of transmission of RPL's DIO by trying to adapt its transmission rate to the level of stability in the network. Tuning more frequent transmission of DIO packets can update the routes in a timely manner but it will drain the battery power. In case a controller is aware of the real-time position of a mobile node, for example, a robot controlled by the devices at the edge, RPL has no standard way of manipulating the routing state in the nodes that are

Identify applicable funding agency here. If none, delete this.

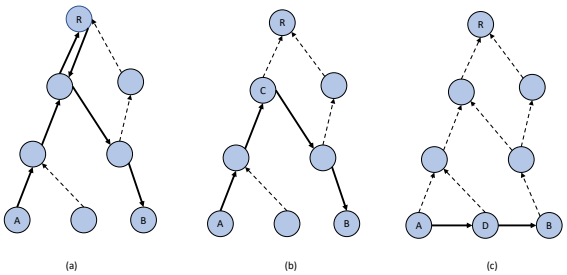

Fig. 1. Examples of data transmission strategies for sending packets from node A to node B. (a) RPL's *non-storing* mode that requires passing through the root. (b) RPL's *storing mode* that permits passing through a common ancestor. (c) A shorter route including siblings.

multiple hops away from the root.

A recent effort by IETF ROLL group [7] has also focused on the above-mentioned problems. The draft on the "root initiated routing state in RPL" defines new control packet types and control packet options that provide backward-compatible mechanisms for the RPL root to collect more information and install the so-called *projected routes* on the selected nodes. The projected routes are of a higher priority to the nodes.

In this paper, we present RPL-RP, an open-source implementation of the RPL with route projection, over Contiki-NG operating system along with a dashboard for monitoring the links. We also analyze its applicability to some of the scenarios that are challenging to the traditional RPL.

The paper is organized as the following: Section II reviews the related work and outlines the gaps in the literature. In section III, we explain some of the details of the implemented system and then evaluate it in terms of overheads and improvements in section IV. Finally, section V concludes the paper and outlines prominent research directions.

II. Related Work

A substantial body of knowledge belongs to centralized management in constrained networks. Sensor OpenFlow [8] is one of the earliest efforts for offloading the network control in order to bring about more flexibility against dynamic policies and ease of management. In a similar approach, SDN-WISE [9] defines its own Topology Discovery layer in the constrained nodes to enable cross-layer operations. An integration with the ONOS [10] controller, allows MAC layer scheduling as specified by TSCH mode of IEEE 802.15.4 and increases connectivity for mobile nodes [11].

In μSDN [12], the authors argue that a centralized controller needs to be compatible with legacy RPL nodes and introduces some optimizations that allow μSDN to overcome the constraints that are common to IoT nodes. Optimizations can include avoiding packet fragmentation, source-routed control packets, configuring update timers. Hydro [13] was another effort to improve the distributed DAG formation with centralized primitives.

Not all the efforts to support any-to-any routing in RPL use a centralized entity. Authors of ORPL [14] combine RPL with opportunistic routing which means that traffic can be forwarded to any node based on the information about its sub-tree. Bitmaps and bloom filter are used to represent this information in a compressed format to avoid memory overflow. Bacceli et al. [15] introduced on-demand mechanisms to discover routes based on flooding control packets. RFC6997 [16] documents a standardized version of P2P-RPL that defines a new operation mode in which the *Origin* creates a temporary DAG along the main DODAG.

In an earlier work [17], we introduced an extension to RPL in which a centralized entity monitors the link qualities for a mobile node and defines some thresholds and timers to update the routes accordingly. Such solutions can also enable the networks to benefit from the computation capacity in the edge nodes to implement tracking algorithms such as particle filter and Unscented Kalman Filter to predict the future position of the mobile nodes and starting the handover process prior to link disconnection [18].

Thubert et. al [19] proposes a framework for an SDN-based TSCH scheduler that meets the requirements of deterministic networking. The authors claimed that the key to improving reliability and mitigating interference is diversity. Diversity can be achieved in different domains as spatial diversity is leveraged with multi-path routing, temporal diversity by retransmissions, and frequency diversity using channel hopping.

The IETF draft on DAO projection [7] defines some primitives to involve the central border router in the distributed operation of RPL and classifies route projection into *Storing Mode Projected Route* (SMPR) and *Non-Storing Mode Projected Routes* (NMPR). The mode for projected routes is independent of RPL's operation mode, meaning that the network can consist of storing mode RPL working with non-storing route projection or vice versa. NMPR uses source routing for the data packets but in SMPR root node asks the source node to update the routing state in all the intermediate nodes. The ROLL working group is currently actively working on this document and to the best of our knowledge there is not any available implementation to compare with. In both modes of DAO projection, getting acknowledgement from either source or destination would suffice. We suggest a reform to put all the intermediate nodes in direct connection with the controller rather than getting an acknowledgement only from source or destination of the path. This will ease troubleshooting since the controller gets to know which link in the path is troublesome.

III. Design and implementation of RPL-RP

In this section, we explain how RPL-RP extends the RPL protocol to fulfill the following requirements:

- Installing point-to-point routes to optimize the path length
- Collecting sibling information besides parent-child relations in the RPL root to be used in a topology viewer dashboard or a controller
- Designing real-time interaction with a manual or automatic controller.
- Reducing the routing header by eliminating the source routing header or loose source routing.

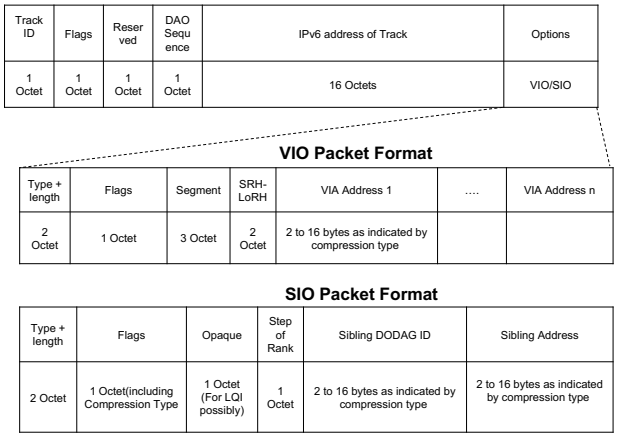

Fig. 2. The packet format for DAO packet including VIO option for P-DAO or SIO for upward DAO packets.

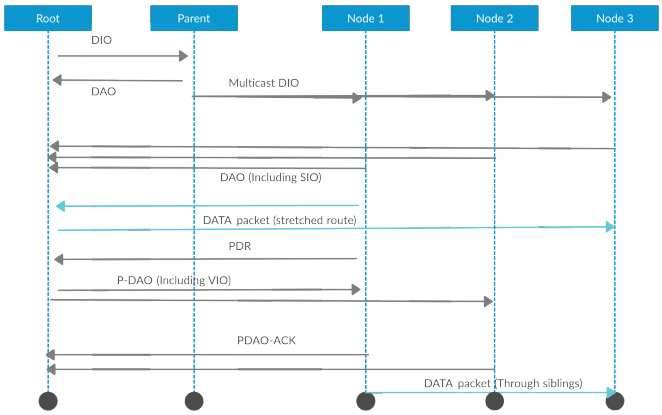

Fig. 3. RPL converges to a set of routes that only include parents. Exchanging the P-DAO allows the in-network nodes to leverage more optimized paths not necessarily limited to parents.

RPL-RP is supposed to provide routes along a track, which is an ordered set of addresses that data packets are supposed to go through. A track is formed to include a single source (track ingress) and destination (the track egress). It is maintained by a local instance of RPL and gets the IP address of the RPL instance as its track ID. A complex track can also be defined to append two or more segments. A track can be installed in the main (Global) instance of RPL to enable routes to the root with different objectives or within other instances of RPL, as for transversal routing.

The new control message types that are introduced are *Projected DAO* (P-DAO), *P-DAO Request* (PDR) and P-DAO-ACK. As the name suggests, PDR is used to ask the root to install the routes towards the track egress for a requested *lifetime*. It is usually sent by the source of the track. The controller responds by sending a sequence of P-DAO messages comprised of *Via Information Option* (VIO) that can be acknowledged using P-DAO-ACK. VIO is a new *Control Message Option* designed to be included in the P-DAO packets, which is a sequence of IPv6 addresses of possible next hops. Figure 2 illustrates the packet format for P-DAO, which is identical to the DAO, except for the VIO option. P-DAO packets carry exactly one VIO option.

Figure 3 presents a sequence diagram of the control packets that enable route projection. The root node initiates the process by sending DIO and consecutively other nodes broadcast their objective function and in turn DAO packets are sent to the root. At this stage, point-to-point routing is performed through the root node. Projected routes can only be installed after initial bootstrapping since they rely on the infrastructure that RPL provides. After exchanging the PDR and Projected DAO, the data packets can be disseminated through siblings.

To achieve low overhead in RPL-RP, routers store the IP addresses of their parents only, as their default route and point of attachment to the root node. This is on the ground that RPL was primarily designed for collecting data. For transversal routing, the traffic is delivered upward (to the root or a common ancestor in non-storing mode or storing mode respectively), and then downward towards the destination.

In RPL non-storing mode for point-to-point routing, data packets are equipped with a source routing header that contains the address for all the intermediate nodes in the path. Although this approach enlarges the routing header, intermediate nodes are not required to maintain the routing information and only the root node keeps the state of parent-child relationships.

However, in RPL storing mode, every single node in the path is stateful and needs to maintain consistency with other nodes and update data packet's next hop using the Hop-By-Hop option. Storing mode is often criticized for higher memory footprint and routing state inconsistency between the nodes. On the plus side, data packets only contain one IPv6 address in the header.

Projected routes also face a similar trade-off whether to choose source routing or hop-by-hop routing header. Currently RPL-RP works with Hop-By-Hop option mode for data packets. The controller needs to send a P-DAO to all the hops in the track once it receives a PDR from track source. For example as illustrated in Figure 4, node 26 starts a flow to node 24 and transmits a PDR to the border router. In response border router establishes a connection to the nodes in the path (except for the track egress), rather than asking the track ingress to forward the P-DAO as in SMRP in [7]. Thereupon, nodes number 26 and 25 receive the P-DAO and install an entry for the projected routes in the routing table with a higher priority than the default upward routes.

On the plus side, sending the P-DAO to multiple nodes, allows the controller to become more resilient against link failures since it gets a separate confirmation from each hop and identifies the failed node in the track by getting P-DAO-ACK containing the error code. Additionally, the data packet's header does not expand with the number of hops. Another advantage is the constrained track ingress nodes are not required to implement the logic for handling the errors and installing the projected routes. Although this design is likely to marginally increase the number of P-DAO packets,

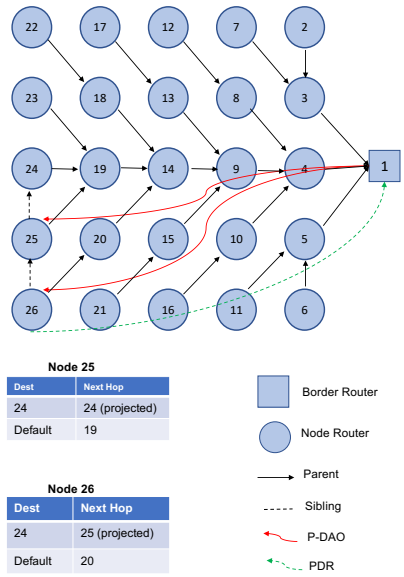

Fig. 4. Simulation topology with 25 routers, where for a traffic from node 24 to node 26, the border router projects the routes after receiving the PDR packet from node 24, assuming that nodes have previously informed the border router of their siblings using SIO option in DAO packets.

our simulation results will show that the additional overhead is negligible.

To collect the extra information at the root node, another option is added to the DAO packets, namely *Sibling Information Option* (SIO). This option contains the IP address of the neighbors and the corresponding link qualities. The number of siblings can be huge, especially in dense networks. In that case, it is not possible to inform the root of all the siblings without an oversized DAO packet, and thus it requires limiting the number of addresses to be included in the SIO. A considerable future work is studying the process of selecting the siblings to include in the SIO option in a dense network or allowing fragmentation of big P-DAO packets. Based on the compression method that is indicated in the flags field for SIO or in the SRH-LoRH field for VIO, the addresses can be of different size from 2 to 16 bytes. Our measurements were based on full form of addresses in VIO, but due to the high number of siblings, we used 8 bytes compressed format in SIO. We defined a maximum number of 3 SIO options to avoid MAC layer fragmentation. The process of selecting the siblings is random. Most probably, choosing more siblings with lower Received Signal Strength Indicator (RSSI) will help in increasing the coverage of the network, though destabilizing the projected routes. It is important to note that devising a smart algorithm for selecting siblings is out of the scope of this paper.

The root node can be co-located with the controller but in this work, these nodes are separate entities, communicating through a JSON-based south-bound API. The root node parses the JSON file and sends P-DAO packets once it gets triggered through a web API or receives a PDR packet. Implementation of the web-based API simplifies further integration with a standard controller such as ONOS. The dashboard is using JavaScript's D3 library [20] to visualize the topology. We make our source code and demo available for the sake of reproducibility[1].

The standard however does not specify any means of learning the capabilities of the nodes or how and which routes should be calculated. It can for example optimize the NP-complete problem of flow maximization or only find the single source or all-pairs shortest path. This is up to the application that is running on top of the controller.

IV. Results and Discussion

This section evaluates the performance of RPL-RP in terms of *end-to-end delay*, *memory footprint* and *communication overhead* incurred by the newly defined primitives. The simulation consists of a border router, a grid of 25 Tmote Sky nodes positioned as in 4 emulated in the Contiki/COOJA environment. The number of active point-to-point UDP flows is also controlled by the controller and increases over time. We gradually increase the number of flows until at least one transversal flow is running between all the nodes. At the MAC layer, we use CSMA and data packets are being sent every two seconds. The controller starts the transversal flows after the initial convergence. We show that with RPL-RP, a negligible overhead in the control traffic and a tolerable memory footprint can be traded for better latency and resiliency in the data plane.

RPL-RP benefits routing in different forms. First and foremost, it reduces the number of hops that data packets go through while lifting the burden of relaying congestion from the nodes that are closer to the root node and balances the load and energy consumption of the nodes. Second, by using the Hop-By-Hop option instead of Source Routing Header in downward routes, it reduces the header size for data packets. Last but not least, in case there is a local source of interference, RPL-RP's controller has the ability to inject convenient routes to bypass the lossy links.

Figure 5 demonstrates the end-to-end delay of data packets measured during the experiment. The end-to-end delay for data packets is proportional to the number of hops in the path. Other parameters such as link quality and number of re-transmissions also matter but RPL-RP accomplishes mostly through reducing the routing stretch. So performance of RPL-RP depends considerably on the topology. For instance, a very deep DODAG can significantly enjoy benefits of projected routes but a shallow network with long east west distance may not take advantage of it so much. In our specific grid topology (same as 4), the routing distance is scaled down from an average of 8 to 2 hops. **In RPL-RP, end-to-end delay is halved in high traffic scenarios compared with the default RPL.** RPL-RP can also tolerate increasing number of flows more smoothly and rate of increasing latency is lower compared to traditional RPL. The confidence intervals indicate that network jitter follows the same trend as mean delay indicating another superiority of RPL-RP.

Besides latency, RPL-RP enhances the resiliency of the routing protocol against different causes of packet loss. In

[1]https://bit.ly/355DZbj

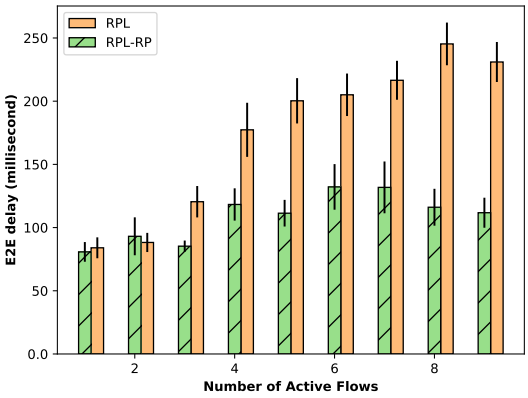

Fig. 5. End-to-end delay of RPL-RP and RPL for different number of flows.

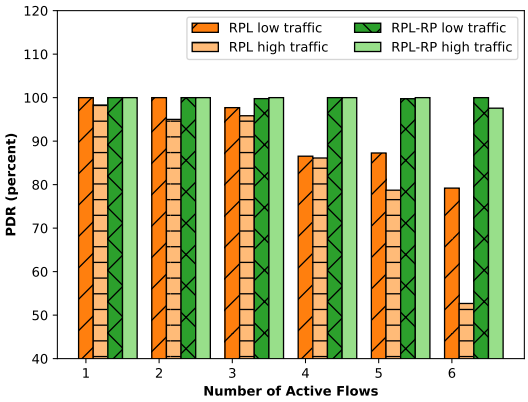

Fig. 6. Packet Delivery Ratio for different number of flows.

LLNs, it is very common to encounter packet losses due to the overflow in the packet queues specially as the closer nodes to the root get congested. In our tested scenario the loss rates of all the links are equal, thus reducing the path length would promote packet delivery ratio. As illustrated in Figure 6, RPL-RP reduces packet losses significantly both with increasing number of flows or with increasing traffic within the same flows. High traffic scenario comprised of 2 packets per second, which increased the loss rate up to 50 percent for RPL with 6 flows. In low traffic scenario nodes transmit 1 packet per second and RPL experienced around 80 percent delivery ratio.

RPL-RP provides nearly 100% packet delivery ratio regardless of network traffic due to bypassing the congested links closer to the root node.

The control traffic is generally governed by two factors: (i) the frequency of the transmissions and (ii) the packet size. To study how RPL-RP expands control traffic, it is important to determine how single packet types evolve within this protocol, then we explore the accumulated overhead. DIO and DODAG Information Solicitation (DIS) packets have the same characteristics for both RPL and RPL-RP, and thus

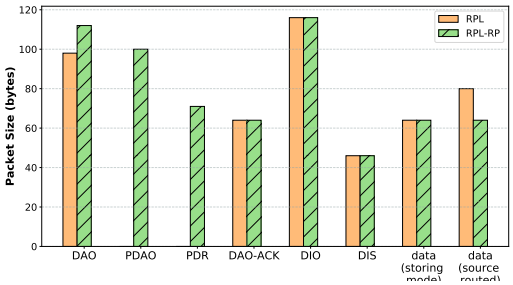

Fig. 7. Comparing size of data and control packets in RPL with RPL-RPL.

they are excluded from the overhead calculation. Frequency of DAO packet transmission is also unaffected but since we append the new SIO option (with maximum 3 siblings), this new option falls into the overhead category. As Figure 7 illustrates, in RPL-RP, we see a slight increase in the size of DAO packet. This raise depends on the number of siblings that are being included in the packet (8 bytes for each sibling). The most significant difference lies in the case of P-DAO, PDR and P-DAO-ACK (optional), which are solely defined for DAO projection. In Figure 7, packet types are associated with the number of bytes each control packet consists of. Fortunately, none of packet types crosses the threshold of 127 bytes, which is IEEE 802.15.4's MTU and there is no need for packet fragmentation. Another important observation is the shrinkage in the size of downward data packets in the source routing header (8 bytes for each hop removed). RPL uses source routed data packets in non-storing mode and as expected we do not see any distinction for Hop-By-Hop mode of addressing the packets.

Now to evaluate the accumulated overhead, it is worth mentioning that the transmission frequency of the P-DAO is defined by the controller. For frequency of P-DAO packets, it will suffice to send P-DAO only when controller gets informed about a topological change not as frequent as basic primitives like DIO. P-DAO ACK is obviously following the same trend as it is sent to acknowledge reception or malfunctioning routes. In our scenario, the controller initialises the data flows which does not necessarily hold true for all the applications, but PDR is also not so frequent since it is only required when asking for a P-DAO. On the other hand, DIO packets are the most frequent RPL packets and are ruled by the *Trickle* algorithm. Therefore, after initialization of DODAG, there is still a fair amount of DIO packets in the network. In contrast, DIS and DAO packets are rarely seen after RPL's initial convergence. DIS packets are meant to ask for a DIO from neighbors if the node does not have any route to the root node. DAO packets notify the root of the routing state DAO and DIS are usually exchanged more frequently in the early minutes. To better visualize the results, Figure 8 presents a logarithmic scale of control traffic accumulated per minute. DIO packets are the most dominating element followed by DAO and DIS packets. P-DAO and PDR packets appear only after four minutes from the beginning of simulation when the transversal flows start.

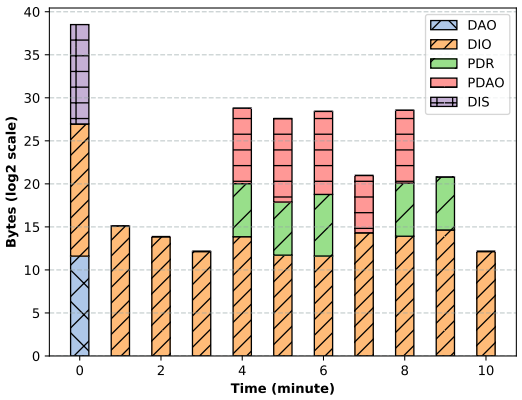

Fig. 8. Accumulated overhead of control packets in RPL-RP.

TABLE I
MEMORY FOOTPRINT OF THE ROUTING PROTOCOLS.

	RPL-RP		RPL	
	ROM	RAM	ROM	RAM
UDP client	44 KB	7340 KB	43 KB	7460 KB
Border Router	170 KB	70 KB	163 KB	70 KB

At this point, according to the logarithmic scale DIO packets generated almost 4 orders of magnitude higher amount of traffic compared to P-DAO and PDR. **Collectively, P-DAO and PDR packets sum up to a 3% of the total control traffic in RPL-RP**.

Furthermore, due to the memory constraints common to low-power devices, it is important to keep track of the memory footprint of RPL-RP. As Table I shows, there is only about 1 KB increase both in volatile and nonvolatile memory of the in-network nodes. The border router can handle more overhead since it is usually deployed on capable devices. Overall, the memory footprint of supporting route projection is inconsequential.

V. CONCLUSION

We presented RPL-RP, an extension to RPL that supports injecting point-to-point routes on-demand by a centralised entity. The new system defines new control packet types and options that collect extra sibling information to be visualized in a dashboard. This enables the administrator to define routing states in the in-network nodes. Overall, evaluation of RPL-RP showed the improvements incurred by the projected routes can surpass its overheads. Although its performance highly depends on the quality of the projected routes, we showed that with a reasonable overhead in control traffic and memory, RPL-RP achieves an almost perfect packet delivery for transversal routes with routes that optimized the latency for hundreds of milliseconds.

Similar to most of the solutions in the related work, RPL-RP only supports installing routes with highest priority and single address destinations (not a range of addresses) which satisfies the requirements of most IoT networks. For the future work, it is worth considering scenarios in which it is necessary or at least useful to install not only high priority routes but also backup routes for fast fail-over as it is supported by OpenFlow.

REFERENCES

[1] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. K. Alexander *et al.*, "Rpl: Ipv6 routing protocol for low-power and lossy networks." *rfc*, vol. 6550, pp. 1–157, 2012.

[2] S. Duquennoy, J. Eriksson, and T. Voigt, "Five-nines reliable downward routing in rpl," *arXiv preprint arXiv:1710.02324*, 2017.

[3] H. Fotouhi, D. Moreira, and M. Alves, "mrpl: Boosting mobility in the internet of things," *Ad Hoc Networks*, vol. 26, pp. 17–35, 2015.

[4] H. Fotouhi, D. Moreira, M. Alves, and P. M. Yomsi, "mrpl+: A mobility management framework in rpl/6lowpan," *Computer Communications*, vol. 104, pp. 34–54, 2017.

[5] B. Safaei, A. Mohammadsalehi, K. T. Khoosani, S. Zarbaf, A. M. H. Monazzah, F. Samie, L. Bauer, J. Henkel, and A. Ejlali, "Impacts of mobility models on rpl-based mobile iot infrastructures: An evaluative comparison and survey," *IEEE access*, vol. 8, pp. 167 779–167 829, 2020.

[6] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The trickle algorithm," *Internet Engineering Task Force, RFC6206*, 2011.

[7] P. Thubert, R. Jadhav, and M. Gillmore, "Root initiated routing state in RPL," Internet Requests for Comments, RFC Editor, RFC-draft 1654, July 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-roll-dao-projection-15

[8] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *IEEE Communications letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[9] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 513–521.

[10] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1–6.

[11] L. L. Bello, A. Lombardo, S. Milardo, G. Patti, and M. Reno, "Experimental assessments and analysis of an sdn framework to integrate mobility management in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5586–5595, 2020.

[12] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving sdn for low-power iot networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 71–79.

[13] S. Dawson-Haggerty, A. Tavakoli, and D. Culler, "Hydro: A hybrid routing protocol for low-power and lossy networks," in *2010 First IEEE International Conference on Smart Grid Communications*. IEEE, 2010, pp. 268–273.

[14] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: Scalable opportunistic routing with orpl," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, 2013, pp. 1–14.

[15] E. Baccelli, M. Philipp, and M. Goyal, "The p2p-rpl routing protocol for ipv6 sensor networks: Testbed experiments," in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*. IEEE, 2011, pp. 1–6.

[16] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci, "Reactive discovery of point-to-point routes in low power and lossy networks," *IETF Request For Comments RFC*, vol. 6997, 2013.

[17] H. Fotouhi, M. Vahabi, I. Rabet, M. Björkman, and M. Alves, "Mobifog: Mobility management framework for fog-assisted iot networks," in *2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 2019, pp. 1–8.

[18] I. Rabet, S. P. Selvaraju, H. Fotouhi, M. Vahabi, and M. Björkman, "Poster: Particle filter for handoff prediction in sdn-based iot networks." in *EWSN*, 2020, pp. 172–173.

[19] P. Thubert, M. R. Palattella, and T. Engel, "6tisch centralized scheduling: When sdn meet iot," in *2015 IEEE conference on standards for communications and networking (CSCN)*. IEEE, 2015, pp. 42–47.

[20] N. Q. Zhu, *Data visualization with D3. js cookbook*. Packt Publishing Ltd, 2013.