Reliable Data Delivery in the Presence of Uncontrollable Mobile Sinks in Low Power Wireless Networks

Kevin Andrea
Department of Computer Science
George Mason University
Fairfax, VA, USA
kandrea@gmu.edu

Robert Simon
Department of Computer Science
George Mason University
Fairfax, VA, USA
simon@gmu.edu

***Abstract*—It is increasingly desirable for deeply embedded Low Power and Lossy Network (LLN) systems to be able to in an on demand fashion deliver stored data to mobile sinks whose arrival patterns are entirely unpredictable. These LLNs are prone to potentially severe data loss due to sudden congestion and environmentally caused wireless transmission impairments. This paper presents *ROIST: Reliable network of Observable devices with Itinerant Sinks Transporting Data*. We demonstrate how to use RPL (one of the standard LLN routing protocols) to support simultaneous multiple convergecast tree instances arising from unpredictable mobile sink patterns, introduce a novel congestion control mechanism and automatically adapt to changing and unpredictable wireless transmission conditions. ROIST is designed to be be fully compatible with existing RPL implementations. We have implemented ROIST in the Contiki-NG operating system and evaluated its performance using COOJA. Our results show that ROIST achieves high levels of reliable data delivery to mobile sinks while minimizing congestion and adapting to wireless transmission losses.**

I. Introduction

An important class of Wireless Sensor Networks (WSN) are applications that autonomously perform data collection and upon demand deliver stored data readings to mobile sinks. Examples include precision agriculture [1], disaster management [2] and tactical military environments [3]. One way to characterize these systems is by sink mobility patterns [4]. They range from controllable and predictable, where sink appearances and movements can both be modified and statistically predicted, uncontrollable but predictable, where sink appearances and movements cannot be modified but can be statistically predicted, and uncontrollable and unpredictable, where sink appearances and movements appear random.

This paper presents the design, implementation and evaluation of the *Reliable network of Observable devices with Itinerant Sinks Transporting Data* (ROIST) protocol architecture. WSNs traditionally fall into the class of Low Power and Lossy Networks (LLNs) that are designed to handle high loss rates, instability, relatively low data transfer capabilities and are energy and resource constrained. ROIST targets WSNs in the LLN environment that require delivery of both stored and real-time data to mobile sinks with uncontrollable and unpredictable – random – mobility patterns. It is designed to support multiple sinks that randomly appear and disappear at different topological points.

RPL is the one of the de-facto routing protocols used for data delivery in an LLN [5]. RPL supports on-demand routing topologies that form Directed Acylic Graphs (DAGs) that naturally support the many-to-one data delivery requirements for mobile sinks. Since RPL was not initially designed with mobility in mind there has been much research into incorporating mobility support inside of RPL [6]–[9]. ROIST is designed to be interoperable with any RPL implementation that supports uncontrollable and unpredictable sink movements. We demonstrate how to use RPL to support simultaneous multiple convergecast trees.

ROIST's performance goals are to maximize data delivery for as long as a sink is in range of the WSN. This is accomplished by careful congestion control adaptive retransmission strategies. One of ROIST's operating modes is to have each node transmit all of its data until a limit on outstanding packets has been reached, which will trigger a checkpoint message to the sink to check for any missing blocks. As will be seen our approach works without the need to modify RPL. By controlling both checkpoint timing and node transmission rates the volume of data delivery to the sink can be dramatically improved. ROIST organizes a network of homogenous resource constrained WSN node into a three-tier hierarchy. At the bottom level packet losses will only be incurred by wireless transmission impairments, while at the middle level packet losses are due either to wireless transmission loss or network congestion. The middle nodes detect congestion conditions and piggyback this information using uptree RPL data packets. The sink uses downtree messages for retransmission requests and congestion control.

We evaluated ROISTs performance and overhead by implementing the protocol using Contiki-NG, one of the standard WSN operating systems. We then compared ROISTs performance under a number of topologies, transmission conditions and data management schemes. Our results indicate that ROIST deliveries data delivery levels even for conditions of

high impairments and unpredictable mobility.

II. Background and Related Work

The benefits of having mobile sinks for WSNs has long been recognized. A number of WSN applications such as disaster recovery and tactical military situations require mobile collection points for data collection and management. Such scenarios additionally suffer under congestion from ill-formed networks, or heavy loss in noisy environments. Our work targets resource constrained WSNs in such environments by providing reliability in communications, using the RPL protocol to support convergecast in resource-poor WSNs.

There is a large body of literature focusing on modifying the RPL protocol to support mobility [10]. The work described in [6] supports multiple logical DAGS and backpressure routing. It is specifically designed to support WSNs that are congestion prone and require high throughput. MobiRPL [7] emphasizes routing reliability over energy management and achieves this by careful parent selection. The research presented [8] pushes complexity into the static nodes in order to better support mobile nodes. This work is extended [9] extends this approach to use an by employing Kalman filters, but also requires that mobile nodes have exact topological awareness. ROIST is meant to be interoperable with most versions of RPL.

From a network management perspective hierarchical structures have long been recognized as one of the most powerful ways to organize the network. One of the earliest proposals for a hierarchical structure is presented in [11]. This work uses s a source-node based virtual grid structure. When a sink requires data local flooding is used to query the network. The work in [12] uses a data dissemination strategy that controls the advertisement of the fresh sink's position by establishing a dissemination tree that encapsulates all sensor nodes in the network. Related to this work is TUFT [13]. Here the sink's mobility is modeled after the Gauss-Markov Mobility Model.

Our work assumes unpredictable sink movements. One motivation for supporting these mobility patterns – inexpensive UAS hardware – is that in practice network engineers will encounter the problems of imprecise timing of arrivals, the inability to fly in certain weather conditions, leading to unpredictable arrivals, and the inability to maintain position in certain weather conditions, leading to uncontrollable positioning relative to the forwarding nodes, which on demand form the network with the Mobile Sink. This uncontrollable, unpredictable model precludes much of the recent work in mobility prediction. Finally, an example of a hierarchical architecture for supporting unpredictable sink mobility is HOIST [14]. This work supports multiple sink types and is built using RPL. Due to its native support for RPL, ROIST is based on concepts from HOIST.

With respect to loss in noisy environments, there has been little work involving end-to-end communications patterns; a majority of the research has favored congestion avoidance and control within the convergecast modality. A 2019 survey on the topic identified congestion, resulting from even low-rate traffic, as one of the common issues with this scenario [15]. This follows with the very nature of convergecast routing, where each device forwards towards a singular target sink, which will increase congestion in all forwarding devices, as they will receive traffic from each of their descendant devices in addition to their own. Several approaches [16], [17] exist for alleviating network congestion through multi-path routing in RPL. This approach utilizes packet delivery information to determine if congestion is contributing to undue packet loss. If detected, alternate parent nodes are enabled and transmissions will alternate between parents, to attempt to perform load balancing and reduce the congestion on a particular path.

Such approaches do not address the problem of limiting the data being transmitted itself, so in an application space with untenable levels of traffic, the multi-path tradeoffs may increase congestion on other pathways at the cost of reducing network stability.

For ROIST, the goal was reliability in transmissions, not directly in the avoidance of congestion, however, ROIST does achieve reliable communications with multiple concurrently active DAGs by limiting the scope of communications and controlling for congestion within each DAG. Each DAG is coordinated by the DAG root, which is one of the Bridge devices. Each Bridge may support up to ρ_{max} devices in an RPL system and the Mobile Sink may support up to ρ_{max} Bridges. As such, at the maximum capacity, given a scenario with a single Bridge directly connected to the Mobile Sink, that Bridge will have up to ρ_{max}^2 packets to forward per transmission period of the devices. If the device only has up to σ slots for queueing incoming packets, then the successful forwards will be bounded by σ.

ROIST addresses this proactively by limiting each DAG to a single coordinated data forward at a time. This reduces the total system-wide transmissions at the upper-level back to the number of Bridges, which is bounded by ρ_{max}. The benefit of this is that in an environment with noise and frequent partitioning of the network through the uncontrolled movements of the Mobile Sink, ROIST removes intermediate sources of congestion and facilitates no more traffic than would be present with only the upper-tier of Bridges alone. When congestion does arise on the upper-tier DAG, ROIST uses the end-to-end communications paradigm to send congestion markers back to the originating data collectors, which then autonomically adjust their send rates to reduce the overall traffic until such congestion clears.

III. ROIST Design

ROIST is a hybrid tree-cluster, three-tier hierarchical architecture that coordinates both data relay and data sink collection from multiple, geographically segregated WSN deployments. The objective of the architecture is to insulate the data collection nodes (Collectors) from the arrival and intra-area transit of a mobile sink (Mobile). This is accomplished through intermediary relays (Bridges), which serve as DAG roots for a deployment of Collectors. As the Mobile sink transits the area, as is commonly the case with Unmanned Aerial System devices, only the Bridges need update their routing. Figure 1

shows ROIST in operation, with a Mobile Sink receiving data from the deployed Collectors, via the Bridges.

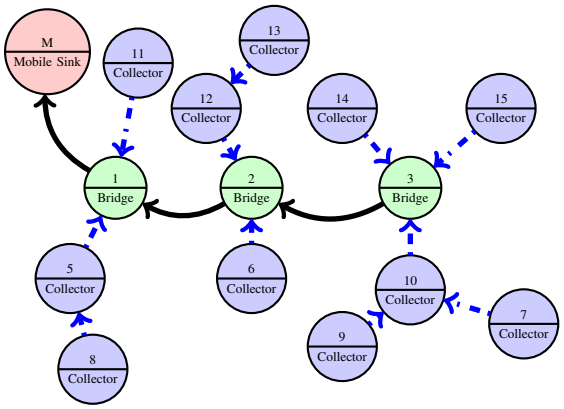


Fig. 1: ROIST Overview with a Mobile Sink Collecting Data

This is accomplished through the use of RPL Instances. An RPL Instance is defined by RFC 6550 [18] as a container for a DAG root. By utilizing this concept, ROIST is able to achieve a goal of utilizing RPL for the routing, without relying upon any particular implementation decision. This will operate using any Objective Function, with any configuration parameters for RPL; it utilizes the extant RPL protocol for all of its routing, in a generally configuration agnostic manner.

Following the arrival of the Mobile Sink ROIST uses a simple handshake to validate connectivity. Each Bridge concurrently begins iterating through its set of descendants and sends a message to begin data transmissions. The Collector, having received and acknowledged this message, suspends data collection and begins transmissions.

The Collector has the current data staged in a circular buffer, with the current data marked for transmission. The device then loads B octets into the payload of the packet and transmits. This process recurs in accordance with the packet send rate, λ, preparing a new block of B octets and transmitting the data. When P_{max} packets are transmitted, the Collector will instead generate a *C*heck message to send to the Mobile Sink. This message is the core of the Checkpoint system (described below) used by ROIST. After P_{max} packets are transmitted, the Collector will request information from the Mobile Sink as to the last contiguously received block of data, and the next it is expecting. On successful transmissions, the Mobile Sink will reply with the number of the next staged block to transmit and the Collector will continue in this manner until all data blocks have been transmitted and confirmed, at which point, the Bridge proceeds to the next Collector while this one resumes data collection.

From a routing perspective sink arrival constitutes the construction and support of a new convergecast tree. Supporting ROIST objectives requires addressing several fundamental questions, including how to support multiple instances of convergecast trees, how to overcome the inherent difficulty of implementing the protocol in a necessarily RPL-based resource constrained environment and how to deal with losses due to either congestion or wireless loss.

A. Supporting multiple convergecast trees in RPL

With the objective of utilizing RPL in a configuration agnostic manner, we sought to limit any modifications to those strictly necessary to fill in the unstated needs of operating in a multiple-instance environment – corresponding to supporting convergecast trees – as pragmatically as possible. There is an immediate dilemma pertaining to the discrimination of which instance to join. RPL makes no provisions under the design of a single instance only, so our architecture first necessitated the creation of a categorical whitelist and blacklist system. As the Instance ID field is defined as one-octet in length, we use the upper nibble as an Instance category and the lower nibble as the Instance identifier itself. This does not affect the transit of ROIST packets across canonical RPL networks, but within a ROIST system, we have Collectors only accepting Bridge category Instances, while Bridges will only accept Mobile category Instances. Without such protections, a Bridge that is configured for two RPL Instances may overhear neighboring Bridges forming their own DAGs for the Collectors and may join all such Instances, leaving no capacity for establishing a network with the Mobile Sink upon its eventual arrival.

Supporting multi-instance extensions requires support for coordinated Instance timeouts. RPL maintains a Route Lifetime system for the route to remain available, measured in Lifetime Units. For routing, this lifetime measurement resets whenever a new packet arrives, indicating that the network is alive and reachable. This is certainly fine for a static deployment, however, with a mobile sink that is only present occasionally, this leads to a false positive indicator that the sink remains, despite being long since departed. The problem occurs when a device receives a periodic DODAG configuration (DIO) packet from a neighbor. These are sent out periodically to both elicit new devices to join the DAG, as well as to provide updated routing and link state information to those within. Upon receipt of this message, the lifetime used will reset, keeping the existence of the sink alive.

ROIST requires that any coordinating mobile sink to arrive and take control of the network, but this is not possible if the DAG is holding information of a prior mobile sink, which would preclude joining the current one. As such, this modification establishes a coordinated routing lifetime that is shared on all outgoing DIO packets. When such a packet is received by another node, it will only receive the current remaining lifetime of the neighbor, and apply $max(lifetime, dio\_lifetime)$ in lieu of performing a full default lifetime reset. In this manner, when a mobile sink leaves, all nodes on the DAG will balance their own internal lifetimes with each other through their normal DIO transmissions and will drop the instances in a coordinated manner, allowing for the next mobile sink to arrive and take control.

As will be seen in Section III-B RPL Storing Mode is needed for a key feature of ROIST: reliable data delivery. This system is built using existing systems within RPL for point-

to-point communications. RPL achieves this by using Storing Mode and DAO messages. When a node changes their parent, a DAO message is generated and travels upwards towards the DAG root. On each intermediate forwarding node, the DAO provides information about the new descendant and the next hop destination to forward all downward traffic through.

In general, RPL provides point-to-point communications through the concept of a common ancestor. If the destination target is unknown by the current node, then it will forward towards the DAG root instead. At some level, to include the DAG root, the destination will be a known descendant and the packet is forwarded towards the registered next hop until it is delivered. RPL implementations are limited in this feature along two dimensions. First, it necessitates additional state in the form of an additional routing table. Second, device limitations on memory constrained devices limit the reachability of devices.

On this second point, if the downward routing table was constrained to ρ_{max} entries, then the maximum number of descendants for any given DAG root would likewise be ρ_{max} nodes. In ROIST, this would provide a total network size of $\rho_{max}+1$ devices per Bridge. In RPL, any excess devices added would be lost in the higher levels, as the routing tables would fill to capacity and prohibit acceptance of any further routing information. Packets generated by the DAG root then would have the lowest probability of containing the complete set of descendants, precluding downward routing to a portion of the deployed devices.

In Convergecast routing, the predominant flow is upwards towards the DAG root. It is under this modality that the modern, "RPL-Lite" implementation was born, removing the state from the downward routing tables to facilitate more memory for application use; this is a common paradigm amongst IoT deployments. ROIST uses, as its core feature, downward routing across multiple RPL Instances from the Mobile Sink to the actual collector of the data being transferred. To facilitate this, each Bridge iterates through its set of descendants and coordinates each one to transmit their data to the Mobile Sink, in full, before moving to the next collector. This removes all intra-DAG sources of congestion and reduces the number of packets being forwarded between Instances by the Bridges to 1 per transmission time unit.

This also solves the problem of end-to-end routing between the Mobile Sink and the originating Collector of the data being transferred. The Mobile Sink only needs an entry for the Bridge forwarding the traffic. On reply, the Mobile Sink uses the source address to send the acknowledgement to the Bridge only. Once there, the Bridge forwards this to the currently coordinated Collector. This enables the Collector to received direct feedback from the Mobile Sink, despite having no common DAG membership.

B. Reliable Data Delivery

The overall retransmission mechanism uses this Checkpointing system, wherein any contiguously received data will be acknowledged by the Mobile Sink back to the Collector. Any gap in the received data will instead be used to update the next expected block to the first that is missing. When the Collector receives this Checkpoint Acknowledgement that block N is expected, it will roll back through the queue and continue retransmissions, beginning with block N, and up to the current P_{max} number of packets before it sends another Check message, as depicted in Figure 2. This is a logical window system that allows a resource-constrained device the simplicity in requesting a retransmission of all packets beginning with the first one that had failed to be delivered. This is a key factor as the Mobile Sink is assumed to be a device with the same resource restrictions as the other nodes within this system.

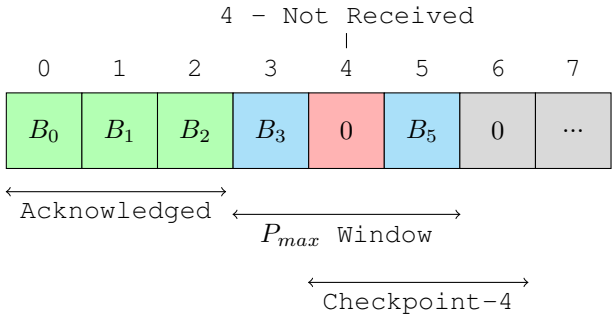


Fig. 2: ROIST Checkpointing

There are two principle sources of data loss that may transpire between the Collector and the Mobile Sink and, as such, the Collector uses this end-to-end flow-control mechanism to attempt to control for the loss to maximize the successfully delivered data while minimizing the duration the Mobile Sink remains on station for the aggregate collection.

The first source is congestion. While ROIST removes the issue of congestion within each Bridge-rooted DAG, there is still the source of congestion at the upper-tier of the network, in the DAG formed by the Bridges and rooted at the Mobile Sink. Each of those Bridges does track the state of their σ slots in the incoming packet queue. If the number of queued packets approaches σ, then a congestion-flag is piggybacked on all forwarded traffic from that Bridge to the Mobile Sink. This form of ECN is achieved by setting the Payload Length (the L/E Octet as depicted in Figure 3) to a 1, as none of the traffic with ECN information contains any payload.

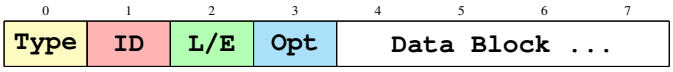


Fig. 3: ROIST Header

The Mobile Sink aggregates and tracks the congestion flags for all of its Bridges and, if any have flagged for congestion, then the Mobile Sink will send a Congestion indicator on all acknowledgements back to the Collectors. Once a Collector receives this acknowledgement with a Congestion indicator present, they will autonomically adjust λ by reducing the number of packets per unit time being transmitted back to a

baseline level; this is the mechanism for reducing system-wide congestion.

When a Collector receives an acknowledgement without a congestion indicator, then it will increment the packets per unit time to transmit. With sufficient acknowledgements without congestion, this will again cap out at the λ_{max} representing the highest send rate possible.

The second primary source of loss is environmental. Whether due to external noise, environmental affects, or simply poor positioning by the Mobile Sink that leads to higher data loss, this factor indicates that a higher degree of retransmissions will be needed overall. To reduce the number of unnecessary retransmissions of large data packets, ROIST will autonomically reduce P_{max} on any acknowledgement that requires a rollback. Instead of continuing to send multiple packets to the Mobile Sink at a time, which in turn may request many of those packets to be retransmitted in the presence of these environmental factors, ROIST instead reduces the number of full data packets that can be sent at a time and will therefore sent small Check messages more frequently.

As data is lost, then P_{max} is reduced to force more frequent checking with the Mobile Sink, reducing the amount of data needed to be retransmitted. As data is acknowledged properly, then P_{max} is increased to allow more data transmissions between checks.

Algorithm 1 Autonomic Flow Control Adjustments in ROIST

$N_{checkpoint} \leftarrow ROIST.header.opt$
$N_{expected} \leftarrow dataset.nextBlock$
if $N_{checkpoint} \neq N_{expected}$ **then**
  $P_{max} \leftarrow max(P_{max} - 1, 1)$
else
  $P_{max} \leftarrow min(P_{max} + 1, MAX\_PACKETS)$
end if
$Congestion \leftarrow ROIST.header.ecn$ ▷ L/E Field
$TU \leftarrow$ Time Units ▷ Seconds per Send Interval
$SI \leftarrow$ Send Interval ▷ TUs between Sends
if $Congestion = 1$ **then** ▷ Double Time between Sends
  $\lambda \leftarrow max(1/((SI * TU) * 2), 1)$
else ▷ Linearly Reduce Time between Sends
  $\lambda \leftarrow min(1/((SI + 1) * TU), MAX\_SENDRATE)$
end if

These two autonomic adjustments, shown in Algorithm 1 are the core of ROIST and utilize the end-to-end flow-control mechanisms in this three-tier hierarchical IoT architecture to ensure timely data delivery with the minimal linger time for the uncontrollable Mobile Sink. This is an AIMD approach to the send rate adjustments as the congestion will have been ongoing and pervasive by the first communication back to the Collectors to reduce their send rates.

C. Retransmission Mechanism

IV. Evaluation

This section describes our implementation and experimental evaluation procedures.

A. Implementation

Contiki-NG is the latest version of a leading embedded IoT-focused Operating System. Contiki provided one of the first implementations of RPL in 2010, presently referred to as "RPL-Classic", providing baseline support for the new routing system. The creation of their next-generation operating system, Contiki-NG, a new implementation was created, "RPL-Lite", with the decision to eschew features to minimize the state within each node [19]. One feature partially implemented by "RPL-Classic", allowing multiple RPL Instances, was removed entirely from this newly refactored implementation.

We selected Contiki-NG for this work as a popular and well-maintained OS with its core implementation for RPL, however, many modifications were needed, even with "RPL-Classic", in order to actualize support for ROIST. One of the many updates needed was in the full support of RPL Instances themselves. While allowing for the configuration of multiple Instances, the implementation provided only references to a single "default_instance". This is also an area where the RFC was expounded upon, as RPL does not specify any design for operating multiple instances concurrently; only providing design guidance for how a single instance would operate.

We modified Contiki-NG to incorporate sufficient resources and code-hooks to allow for multiple, concurrent RPL Instances, and to support a device being a DAG root of one, while a member of a second. This modification facilitated the three-tier hierarchical support needed for ROIST. Furthermore, the functions involved within the networking stack were modified to support the passing of state from the incoming packet, allowing proper forwarding of traffic along RPL in accordance with the RPL Instance it was associated with. This enabled multi-Instance intercommunication without routing interference, in accordance with the modifications we had made to the RPL protocol.

Current limitations with our Contiki-NG implementation still remain with regards to the routing table size and with the Storing Mode implementation that does not react well to the change of parent events, persisting old descendants in their downward routing tables. This occurs primarily in noisy environments wherein the updating packet, the DAO, is not properly received as multiple devices may change their own parents, precluding the routing through all prior ancestors, or by simple loss of the DAO.

B. Experimental Evaluation Setup

The experimental evaluation was performed on the Cooja simulator. Cooja uses an MSP430 emulator to run the code on each of the Zolertia Z1 devices used in the simulation. For this work with ROIST, we elected one Mobile Sink, which would arrive after 5 minutes of simulation time, to allow the initial formation of the devices within RPL, and to allow for sufficient, albeit greatly accelerated, data collection to necessitate a full data buffer to transfer.

To provide a source of noise, we used the Logistic Loss Radio Medium [20] to present a realistic loss model. Under this model, the devices were procedurally deployed for each

test using $\{range|4 \leq range \leq 6\}$m maximum distance to the neighboring device. This range was selected in conjunction with the source of noise for the experiments.

To test under different levels of noise, we evaluated the α parameter of the LogisticLoss model, ranging from 2.0 (higher noise) to 4.0 (light noise) levels. For this validation, we set $\alpha = 3.0$, which corresponded to an average RSSI in the range $[-78.98, -84.29]$.

ROIST was evaluated under two Topologies for this validation: Linear and Mesh. Under each model, two Bridge roots were placed within $range$ of the location where a Mobile Sink would arrive. Under the Linear topology, Collector devices were split between the provided Bridge devices and their positions were generated within $range$ of the prior generated Collector for that Bridge. These deployments were all generated along the same direction from the Bridge. Under the Mesh topology, all Collector devices were deployed to a random position within $range$ of any existing device, placed at a random heading from that device.

The objective of these validations was to assess ROIST in a high-throughput environment. We set the data patterns therefore to use 10 octet Block sizes, with one Block per data packet. This produces smaller 6LoWPAN packets (55 octets) than the MTU can support, but it also provided an increase in the scenario traffic when compared against the 128-octet MTU possible. The initial Send Rate (λ) was set for 16 packet transmissions per second. This is also much higher than would be used in a deployment, but provides higher congestion incidences throughout the network for the validation.

All of the tests in the validation use the same node deployment positions, which are procedurally generated at the beginning of the set of tests, once per test topology. Each test used a different, randomly generated seed. A sample of the deployment positions is shown in Figure 4, with 2 Bridges (yellow), 7 Collectors (red), and one Mobile Sink (green) in the Cooja Test Environment.

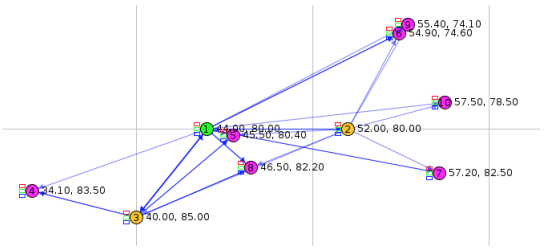


Fig. 4: ROIST Evaluation Running in the Cooja Network Simulator

V. Experimental Results

Each of the results are depicted in Figure 5. In these graphs, **`No-CP`** indicates no Checkpointing, **`CP/NA`** indicates that Checkpointing was Enabled, but the Autonomic Controller was Disabled, while **`CP/A`** indicates both were activated.

The first validation set to establish a baseline for looking at the overall Packet Delivery Ratio (PDR) and the Block Delivery Ratio (BDR), which assesses the ultimate efficacy in delivering the data to the Mobile sink. For this validation, we assessed a baseline without any Checkpointing or autonomic adjustments; this was a measure of the average packet delivery success solely through the RPL routing. The results, shown in Figure 5a showed an average PDR of 59.5% without any end-to-end reliability mechanisms of ROIST for the 91 total transmitted packets.

With the Checkpoint configuration enabled, albeit without any autonomic adjustments to the parameters, the BDR increased to 100% for each topological model, which is as expected under the system of checkpointing and controlled retransmissions to ensure data delivery. The PDR increased to an average of 76.1% under a Linear topology and to 77.5% under a Mesh topology.

Adding the autonomic adjustments to the Checkpointing, the PDRs increased further, with an average of 81.6% for Linear topology and 81.7% for the Mesh topology, as each required fewer transmissions overall to achieve the 100% BDR. This achieved the marked purpose of ROIST, which was reliability in data delivery to the uncontrollable and unpredictable Mobile sink under RPL.

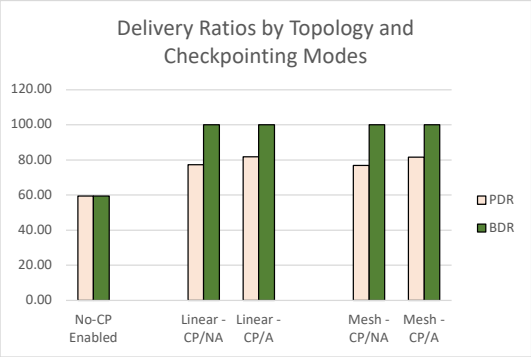
The second validation studied how the various Checkpointing modes addressed the total transmissions needed in order to achieve the 100% BDR objective. These results, shown in Figure 5b, demonstrate that under Checkpointing without the autonomic control we required the transmission of an average of 133.7 and 133.8 packets under Linear and Mesh topologies respectively. With the addition of autonomic adjustments these each fell precipitously to an average of 118.6 and 119.0 packets under the same respective topologies.

The reduction in transmitted traffic is a key figure for this validation as it shows the autonomic adjustment system was able to adjust in response to the interference, noise, and congestion in such a way as to increase the reliability of each packet delivery. While this validation was short in duration, only accounting for one set of block deliveries to the Mobile sink under a limited deployment area, each of the devices would nevertheless maintain their last settings, allowing for a better starting point under the next data retrieval event.

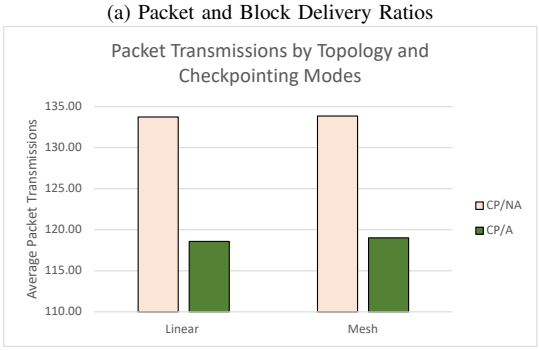
The final assessment was on the Linger time of the Mobile sink. This is a measure of the duration required from first transmission to the last block acknowledged by the last Collector. The average Linger time without autonomic adjustments, as shown in Figure 5c was 214.78 sec for the Linear topology, and 273.59 sec for the Mesh topology.

When autonomic adjustments were enabled, the Linger time did increase slightly to 258.72 sec under the Linear topology, and to 311.62 sec under the Mesh topology. While the total number of packets was reduced, there were nevertheless delays resulting from the additional end-to-end communications to make these flow-control adjustments, increasing the Linger time under such a small deployment validation.

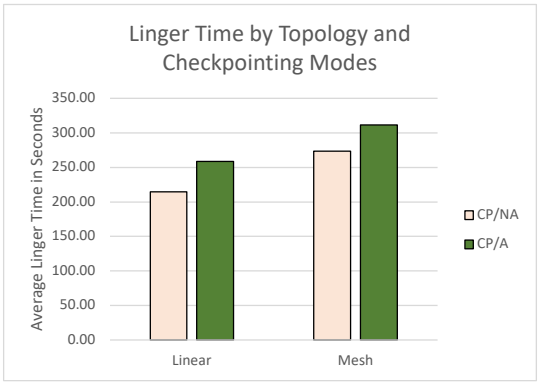
These validation experiments could not be compared against a baseline RPL implementation. For the reasons specified in III, the formation of such a hierarchical network is not directly possible.



(a) Packet and Block Delivery Ratios



(b) Total Packet Transmissions



(c) Linger Time

Fig. 5: ROIST Validation Results

VI. Conclusion

ROIST provides reliability in data transmissions for LLN based IoT deployments through the use of end-to-end communications and Checkpointing. This is facilitated through reasonable extensions to the standard RPL routing protocol, which were limited to providing definitions for the underspecified operations of multiple concurrent RPL Instances.

Using these necessary extensions, ROIST supports simultaneous multiple convergecast tree instances arising from unpredictable mobile sink patterns, introduces a novel congestion control mechanism and automatically adapt to changing and unpredictable wireless transmission conditions, as demonstrated through our Contiki-NG implementation and validated experimentally using the COOJA simulator. Our results show that ROIST achieves high levels of reliable data delivery to mobile sinks while minimizing congestion and adapting to wireless transmission losses.

References

[1] Y.-D. Yao, X. Li, Y.-P. Cui, J.-J. Wang, and C. Wang, "Energy-efficient routing protocol based on multi-threshold segmentation in wireless sensors networks for precision agriculture," *IEEE Sensors Journal*, vol. 22, no. 7, pp. 6216–6231, 2022.

[2] S. Singh, A. S. Nandan, A. Malik, N. Kumar, and A. Barnawi, "An energy-efficient modified metaheuristic inspired algorithm for disaster management system using wsns," *IEEE Sensors Journal*, vol. 21, no. 13, pp. 15 398–15 408, 2021.

[3] K. Ghosh, S. Neogy, P. K. Das, and M. Mehta, "Intrusion detection at international borders and large military barracks with multi-sink wireless sensor networks: An energy efficient solution," *Wireless Personal Communications*, vol. 98, no. 1, pp. 1083–1101, 2018.

[4] A. Oliveira and T. Vazao, "Low-power and lossy networks under mobility: A survey," *Computer networks*, vol. 107, pp. 339–352, 2016.

[5] H. Lamaazi and N. Benamar, "A comprehensive survey on enhancements and limitations of the rpl protocol: A focus on the objective function," *Ad Hoc Networks*, vol. 96, p. 102001, 2020.

[6] Y. Tahir, S. Yang, and J. McCann, "Brpl: Backpressure rpl for high-throughput and mobile iots," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 29–43, 2018.

[7] H. Kim, H.-S. Kim, and S. Bahk, "Mobirpl: Adaptive, robust, and rssi-based mobile routing in low power and lossy networks," *Journal of Communications and Networks*, 2022.

[8] M. Bouaziz, A. Rachedi, A. Belghith, M. Berbineau, and S. Al-Ahmadi, "Ema-rpl: Energy and mobility aware routing for the internet of mobile things," *Future Generation Computer Systems*, vol. 97, pp. 247–258, 2019.

[9] M. Bouaziz, A. Rachedi, and A. Belghith, "Ekf-mrpl: Advanced mobility support routing protocol for internet of mobile things: Movement prediction approach," *Future Generation Computer Systems*, vol. 93, pp. 822–832, 2019.

[10] 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.

[11] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "Ttdd: Two-tier data dissemination in large-scale sensor networks," *ACM MONET (Mobile Networks and Applications)*, pp. 85–89, 2003.

[12] A. Hawbani, X. Wang, H. Kuhlani, S. Karmoshi, R. Ghoul, Y. Sharabi, and E. Torbosh, "Sink-oriented tree based data dissemination protocol for mobile sinks wireless sensor networks," *Wireless Networks*, vol. 24, no. 7, pp. 2723–2734, 2018.

[13] O. Busaileh, A. Hawbani, X. Wang, P. Liu, L. Zhao, and A. Y. Al-Dubai, "Tuft: Tree based heuristic data dissemination for mobile sink wireless sensor networks," *IEEE Transactions on Mobile Computing*, 2020.

[14] K. Andrea and R. Simon, "Design and evaluation of an RPL-based multi-sink routing protocol for low-power and lossy networks," in *MSWiM 2015 - Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2015.

[15] C. Lim, "A Survey on Congestion Control for RPL-Based Wireless Sensor Networks," *Sensors*, vol. 19, no. 11, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2567

[16] O. Iova, F. Theoleyre, and T. Noel, "Exploiting multiple parents in rpl to improve both the network lifetime and its stability," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 610–616.

[17] M. A. Lodhi, A. Rehman, M. M. Khan, and F. B. Hussain, "Multiple path rpl for low power lossy networks," in *2015 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, 2015, pp. 279–284.

[18] A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Tech. Rep., mar 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6550

[19] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[20] A. Elsts. (2022) Logistic loss radio medium. [Online]. Available: https://github.com/atiselsts/cooja/wiki/LogisticLoss-radio-medium