DualMOP-RPL: Supporting Multiple Modes of Downward Routing in a Single RPL Network

JEONGGIL KO, JONGSOO JEONG, JONGJUN PARK, and JONG ARM JUN,
Electronics and Telecommunications Research Institute
OMPRAKASH GNAWALI, University of Houston
JEONGYEUP PAEK, Hongik University

RPL is an IPv6 routing protocol for low-power and lossy networks (LLNs) designed to meet the requirements of a wide range of LLN applications including smart grid AMIs, home and building automation, industrial and environmental monitoring, health care, wireless sensor networks, and the Internet of Things (IoT) in general with thousands and millions of nodes interconnected through multihop mesh networks. RPL constructs tree-like routing topology rooted at an LLN border router (LBR) and supports bidirectional IPv6 communication to and from the mesh devices by providing both upward and downward routing over the routing tree. In this article, we focus on the interoperability of downward routing and supporting its two modes of operations (MOPs) defined in the RPL standard (RFC 6550). Specifically, we show that there exists a serious connectivity problem in RPL protocol when two MOPs are mixed within a single network, even for standard-compliant implementations, which may result in network partitions. To address this problem, this article proposes *DualMOP-RPL*, an enhanced version of RPL, which supports nodes with different MOPs for downward routing to communicate gracefully in a single RPL network while preserving the high bidirectional data delivery performance. *DualMOP-RPL* allows multiple overlapping RPL networks in the same geographical regions to cooperate as a single densely connected network even if those networks are using different MOPs. This will not only improve the link qualities and routing performances of the networks but also allow for network migrations and alternate routing in the case of LBR failures. We evaluate *DualMOP-RPL* through extensive simulations and testbed experiments and show that our proposal eliminates all the problems we have identified.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, mesh network, routing protocol, low-power lossy network, downward routing, interoperability, RPL

ACM Reference Format:
JeongGil Ko, Jongsoo Jeong, Jongjun Park, Jong Arm Jun, Omprakash Gnawali, and Jeongyeup Paek. 2015. DualMOP-RPL: Supporting multiple modes of downward routing in a single RPL network. ACM Trans. Sensor Netw. 11, 2, Article 39 (February 2015), 20 pages.
DOI: http://doi.acm.org/10.1145/2700261

For J. Ko, J. Jeong, J. Park, and J. A. Jun, this work was supported by the Korean Ministry of Knowledge and Economy Project #10035570, "Development of Self-Powered Smart Sensor Node Platform for Smart and Green Buildings."
For J. Paek, this work was supported by the Cisco University Research Program Fund.
Omprakash Gnawali was partially supported by Cisco and the National Science Foundation under grant no. IIS-1111507.
J. Ko and J. Jeong are equally contributing co-first authors. J. Paek is the corresponding author.
Authors' addresses: J. Ko, J. Jeong, J. Park, and J. A. Jun, Electronics and Telecommunications Research Institute (ETRI), Korea; emails: {jeonggil.ko, jsjeong, juny, jajun}@etri.re.kr; O. Gnawali, Department of Computer Science, University of Houston, TX; email: gnawali@cs.uh.edu; J. Paek, Department of Computer Information Communication Engineering, Hongik University, Sejong, Korea; email: jpaek@hongik.ac.kr.

DOI: http://doi.acm.org/10.1145/2700261

1. INTRODUCTION

With the capabilities to sense various physical characteristics at a previously unprecedented scale, wireless sensors networks (WSNs) have gained a lot of attention from the research community in the past decade. Research concepts such as *smart dust* made it look like smart electrical grid [Cisco 2015], smart city management [Heo et al. 2014], smart home and building automation, industrial and environmental monitoring [German Federal Ministry of Education and Research 2015; Adler et al. 2005; Paek et al. 2014], health care [Ko et al. 2010], and the *Internet of Things (IoT)* with millions and billions of interconnected devices were in close reach. However, after a decade since the initial kickoff of WSN research, the vision still seems elusive without many real-life industrial-scale deployments. One of the main reasons for this gap between technology advances and real-life applicability was considered to be the lack of global standards to agree on. Realizing this over the past few years, research community and standardization committees together have gathered to design standards that address the unique challenges that WSNs introduce. Specifically, protocols at various layers of the networking stack for WSNs have been designed to connect small-sized low-power embedded devices to the Internet. With the support of various standardization communities such as the IEEE, IETF, and Zigbee, IoT systems are equipped with protocols and application profiles that can initiate large-scale deployments [Ed. et al. 2012; IEEE Std 802.15.4-2003 2003; Montenegro et al. 2007]. However, because these standards themselves are new and are being applied to new and rapidly evolving networks and applications, it is inevitable that flaws in the protocol design are sometimes discovered after standardization. While some flaws lead to minor inefficiencies in network protocols [Clausen et al. 2011], some can be fatal. This article focuses on an interoperability problem that the recently proposed IETF RPL (RFC 6550) [Ed. et al. 2012] protocol possesses.

RPL is an IPv6 routing protocol for low-power and lossy networks (LLNs), designed for resource-constrained embedded devices to meet the requirements of a wide range of LLN applications. RPL constructs tree-like routing topology rooted at an LLN border router (LBR) and supports bidirectional IPv6 communication between network devices by providing routing capabilities in both directions along the tree, *upward routing* from the mesh nodes to the LBR and *downward routing* from the LBR to the mesh nodes. Node-to-node routing between mesh nodes can be achieved by utilizing these two directions. For the *downward routing*, RPL provides two *modes of operation* (MOPs),[1] the *storing mode* and the *nonstoring mode*, for constructing and managing the downward routes (Section 2). And these two MOPs have clear tradeoffs between them. While the *storing mode* is beneficial for reducing networking overhead, the *nonstoring mode* is more suitable for nodes with strict memory limitations where storing a large number of routing states is not an option.

However, while defining these two different downward-routing modes, RPL standard writes the following: "No implementation is expected to support both Storing and Nonstoring modes of operation. Most implementations are expected to support *either* no Downward routes, Nonstoring mode only, or Storing mode only" [Ed. et al. 2012]. Given this statement, this article asks and answers the following question: "How would the network behave if nodes implementing the *storing mode* and nodes implementing the *nonstoring mode* operated in a single network?" We see this as a very likely and also a desirable scenario as we start deploying LLNs at large scales and approach our goal of Internet of Things (IoT). Multiple systems, each deployed for different purposes, can meet at the same geographical area and naturally form a single RPL network to

[1]Another mode of operation exists for RPL based on RFC 6997 [Goyal et al. 2013] (P2P mode of operation for node-to-node communication). However, this is an experimental protocol and is not part of the standard yet.

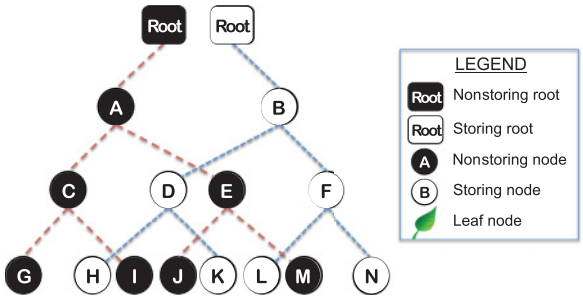

Fig. 1. Two RPL networks with different downward-routing MOPs overlapping in the same geographical region. If the two networks can cooperate, they may form a denser, better-connected, reliable, and robust network and may be capable of handling a root failure in the region.

cooperate with each other. For example, smart grid AMI networks[2] may overlap with smart lighting systems, and the industrial asset tracking network may overlap with environmental sensor networks. Let's take Figure 1 as an illustration of such a network. Allowing multiple overlapping RPL networks in same geographical regions to cooperate as a single densely better-connected network even if those networks are using different MOPs will not only improve the link qualities and routing reliability of the networks but also improve the robustness by allowing for network migrations and alternate routing in the case of LBR failures and power outages. Furthermore, given the diversity in computational capability among various LLN devices and the respective tradeoffs of using the two different downward-routing MOPs, systems can benefit from using both downward-routing schemes simultaneously by allowing resource-rich devices to operate on *storing mode*, while the majority of cheap nodes operate on the *nonstoring mode*. Nevertheless, RPL standard restricts such a use, and the consequences of using a mixed (overlapping or heterogeneous) deployment is yet unknown.

We illustrate through several examples and analyses that an interoperating network (network with heterogeneous MOPs), when standard compliant to RFC6550, has serious connectivity problems for *both* upward and downward traffic and may partition the network. Furthermore, we use standard-compliant open-source implementations of IETF RPL to experimentally show that an interoperating network does indeed have serious connectivity problems, and their resulting packet delivery performance is significantly impacted. To address this issue, we propose minimally disruptive enhancements to RPL standard so that a single RPL network can accommodate nodes with both *storing* and *nonstoring mode* mixed together (i.e., to take advantage of their respective strengths) while eliminating the network partition problem and preserving the high bidirectional data delivery performance. We evaluate our proposal through extensive simulations and testbed experiments using open-source RPL implementations and show that our claims are valid.

Specifically, the contributions made in this work can be summarized in three ways:

—We identify the interoperability problems of downward-routing schemes in the IETF RPL protocol and discuss cases where the routes can break down and result in packet delivery failures due to network partitions.
—We propose a novel lightweight solution *DualMOP-RPL* to the aforementioned problem. Our design allows the nodes with different MOPs to interoperate gracefully in

[2]Cisco's Field Area Network for Smart Grid: http://www.cisco.com/web/strategy/energy/field_area_network.html.

a single RPL network and enables the design of RPL networks with heterogeneous downward routing, which opens the potential for novel application developments.
—We extensively evaluate the performance of our proposed enhancements to the RPL routing protocol in both simulation and testbed environments.

The remainder of this article is structured as follows. In Section 2, we present a brief overview of the IETF RPL routing protocol's downward-routing schemes and then discuss their interoperability problems in Section 3. Our proposed design to address these problems is explained in Section 4, and we evaluate our proposal in Section 5. Finally, we conclude the article in Section 6.

2. ROUTING WITH IETF RPL

IETF RPL is an IPv6 routing protocol for LLNs, designed for resource-constrained embedded devices to meet the requirements of a wide range of LLN applications. While data collection is regarded as the dominant traffic pattern in many LLN applications, new emerging applications in the LLN domain has suggested the need for bidirectional routing [Brandt et al. 2010; Ed. et al. 2009; Martocci et al. 2010; Pister et al. 2009; Paek et al. 2010; Paek and Govindan 2010]. For this reason, the RPL protocol provides routing capabilities in two directions, *upward routing* and *downward routing*.

Like many WSN systems, RPL's most common use is to collect data from LLN nodes. This is done using the *upward routing* from an LLN device to an LBR by constructing a tree-like routing topology called Destination-Oriented Directed Acyclic Graph (DODAG) rooted at the LBR. Furthermore, RPL also provides *downward routing* schemes for maintaining routes to individual nodes participating in the network. These two together allow the support for bidirectional IPv6 communication to network devices, which can be used for various network management protocols as well as at the application layer to support control and actuation capabilities (e.g., light or temperature control using individual switches). Upward routing can be supported with little constant size routing state, whereby each node only needs to store the next hop leading to a single destination, the root of the DODAG residing at the LBR. On the other hand, downward routes are constructed using RPL Destination Advertisement Object (DAO) messages, which advertise routing information on how other nodes can reach various destinations and prefixes within an RPL network when traveling down the RPL DODAG.

For the *downward routing*, RPL provides two MOPs, named the *storing mode* and the *nonstoring mode*, for constructing and managing the downward routes, each designed to suit different classes of devices. In *storing mode*, all RPL nodes maintain next hop addresses for the nodes in their subtree (sub-DODAG in RPL terminology) by having each node send unicast RPL DAO messages to its parents to advertise routes. Upon receiving a DAO, a node locally stores in its routing table the address of the sender as the next hop node to reach the advertised targets and also generates a new DAO and transmits it to its parents to ensure that routing information propagates upward in the network. Then, classical hop-by-hop IPv6 routing is used by RPL nodes to reach downward destinations learned from DAOs. On the other hand, in *nonstoring mode*, only the root at the LBR knows the complete Directed Acyclic Graph (DAG) topology of the network. Each RPL node sends unicast DAO messages to the DODAG root, which includes information on the parent set of the sender node, and the DODAG root stores these child–parent relationships in a DAG information table. Then, when the DODAG root needs to send a packet to a node in the network, the root can construct a source route along a path to the destination by recursively looking up the DAO parent information table. The forwarding nodes simply read the source routing header that the root attaches when routing packets to their final destinations.

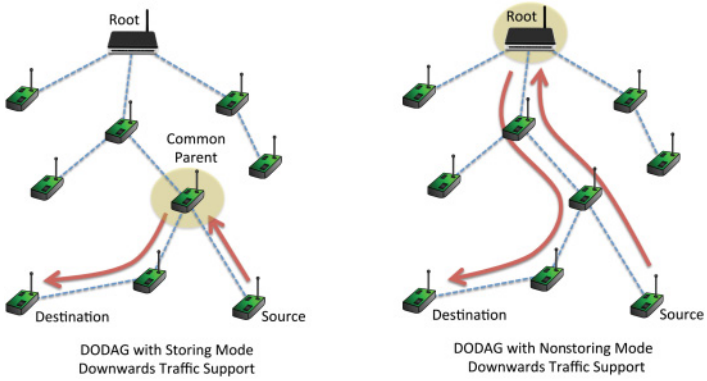

Fig. 2. Node-to-node communication on two different types of downward- routing MOPs supported by RPL. The left illustrates the *storing mode* case, and the right illustrates the *nonstoring mode* case. RFC6550 indicates that either one of the modes should be implemented.

One of the most important differences between the *storing mode* and *nonstoring mode* is how node-to-node communication is done within the RPL network. When a packet is generated by an RPL node and is addressed to another RPL node, it travels up to DODAG until a common ancestor is met, and from here the packet travels back down the downward routes to reach its destination. Since the intermediate nodes do not store routing information for downward routes in the *nonstoring mode*, the only common ancestor in *nonstoring mode* is the DODAG root itself, and thus all packets must travel to the root before reaching the final destination. In the storing mode, the common ancestor may be a node closer to the source and destination. Figure 2 illustrates this.

Thus, there are clear tradeoffs between the *storing mode* and the *nonstoring mode*. The primary advantage of the *nonstoring mode* is that it requires very little memory for storing routing states on the resource-constrained embedded devices with limited processing and storage capabilities. Since the LBRs typically have more compute and memory resources than the LLN devices connected to it, it is a natural choice to have the DODAG root maintain the routing table to all destinations in its network. However, the *nonstoring mode* requires the DODAG root to attach a source routing header to all packets going into the RPL network. Furthermore, it may incur an unnecessarily *long route* for node-to-node communication because it must go up to the root to reach another destination (Figure 2). Source routing headers not only increase the packet size but also make the packet size variable depending on the depth (path length) and decreases the effective MTU of the mesh network. The long-route problem is particularly significant if there are packet drops within the mesh network because end-to-end retransmissions must go through the root again. On the other hand, the advantage of the storing mode is that it follows the classic hop-by-hop forwarding model without the need for a source routing header, and it does not have the *long-route* problem of the *nonstoring mode*. However, each RPL node must store route information to all the destinations in its own subtree, which may be too demanding for the limited memory constraints of small embedded devices. For example, if an entry in the routing table takes 50 bytes of memory, a node with 5,000 nodes in its subtree will require 250KB of memory. Some of these tradeoffs, along with other challenges and critical evaluation of RPL, are also reported in Clausen et al. [2011]. These tradeoffs require the application developer to make a design choice when deploying RPL-based LLNs.

3. PROBLEM - INTEROPERABILITY BETWEEN TWO MOPS

While providing two different downward-routing schemes, the RPL standard (RFC6550) indicates that RPL-compliant systems are expected to operate only the *storing mode* or only the *nonstoring mode* but not both. Furthermore, RFC6550 states that when nodes with different *modes of operation* meet in the same physical space and form a single RPL network, the MOP declared by the DODAG root takes dominance and the nodes operating with a different MOP can only join the RPL network as a *leaf node*.[3] While this is a safe design choice, it may restrict the nodes from fully interoperating with one another, thus limiting the benefits and applications that interoperating RPL systems can introduce. As we will demonstrate later in the article, this limitation enforced by RPL can cause two standard-compliant RPL implementations, each developed and deployed with different goals, to not interoperate properly and may even cause network partitions. We emphasize that relaxing this restriction can open up chances for new application development with the RPL routing protocol, and we summarize two major scenarios where such a mixture of different downward-routing schemes can be beneficial.

—Given that *storing* and *nonstoring* modes are each suitable for hardware devices with different capabilities, system developers can design heterogeneous RPL networks with *storing mode* operating on devices with higher memory and processing power, while the majority of the network consists of inexpensive resource constraint devices operating the lighter-weight *nonstoring mode* version of RPL (e.g., hierarchical hybrid MOP-based systems [Schmid et al. 2010]). It would be possible to fully utilize the capabilities of both hardwares if RPL can support both *storing* and *nonstoring* downward- routing modes simultaneously in a single network.
—Standardized RPL-based systems can be designed from different vendors with different application targets. Thus, a system may use either the *storing mode* or the *nonstoring mode* version of RPL. When these systems (with different RPL downward-routing schemes) are colocated in an overlapping geographical region, interconnecting these devices into a denser network can potentially improve the link and routing qualities of the network. Furthermore, it is possible to build a robust network that can handle LBR failures and fail-overs. Therefore, it is important to allow RPL networks with different downward- routing schemes to merge into a single network.

In scenarios such as the ones that we have discussed previously, the recommendation to interoperate as a leaf node in the network when *storing* and *nonstoring* modes are mixed has operability and performance implications for both upward and downward routing. For wide distribution and commercialization of wireless sensor/actuator LLNs, this limitation is critical to resolve. The subsections that follow illustrate in detail how the network operability and performance can suffer when using *storing* and *nonstoring* mode-based devices in a single network.

3.1. Upward Routing

First, a mixture of *storing* and *nonstoring* mode nodes in a single network can partition *upward routing*. When an RPL node attaches itself as a leaf to an existing RPL network with different MOPs, that node cannot act as a router and advertises an RPL rank of infinity (does not allow other nodes to select this node as a parent). Leaf nodes may send their data to their next hop but may *never accept packets for forwarding*. This is not a problem if the node is at the fringe of the network. However, if the node is somewhere in the middle of a forwarding path (if that node has a subtree of nodes that

[3]"Leaf node" advertises rank of infinity and does not allow forwarding of traffic on behalf of other nodes.

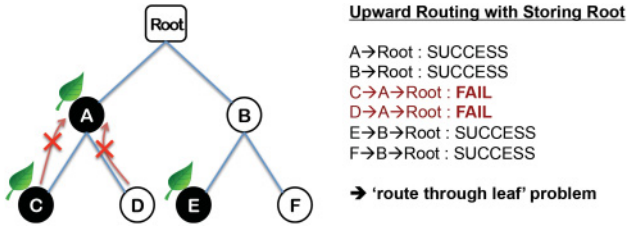

Fig. 3. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *storing mode* root with upward traffic. A *nonstoring mode* node in the middle of a routing path can partition the network when following RFC6550 to act as a leaf node.

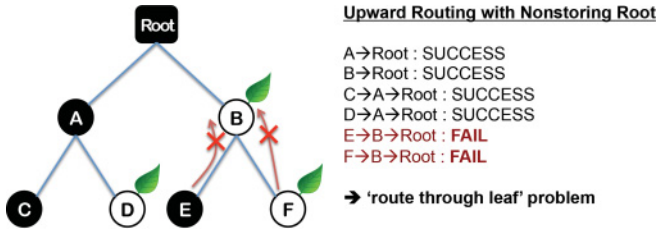

Fig. 4. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *nonstoring mode* root with upward traffic. A *storing mode* node in the middle of a routing path can partition the network when following RFC6550 to act as a leaf node.

needs to connect to itself for routing, and otherwise would be disconnected), then the problem occurs. We call this the *cannot route through leaf* problem or *route through leaf* problem in short.

Imagine a network like Figure 3 with a mixture of *storing* (white circle) and *nonstoring* mode nodes (black circle) and a *storing mode* root (white rectangle). For the purpose of illustration, we assume that the presented links are the only available physical links with reasonable (acceptable) link quality at this time instance. Ideally, all nodes in this network should be able to communicate with the root since the entire network is physically fully connected. However, we face a network partition for subtrees of node C and D because node A in the middle does not have the capability to forward traffic to and from the root.[4] The same problem exists even if the root is in *nonstoring mode* (black rectangle) as shown in Figure 4.

3.2. Downward Routing

Mixture of the two modes also causes network partition for downward routing from the root to the nodes. The main reason is similar to the upward case, but there are two other additional problems. Let's first take Figure 5 as an example: mixed network with *storing mode* root. In this network, packets from the root cannot reach subtrees of C or D not only because node A has attached itself as a leaf node to the network and is not allowed to forward packets, but also because node A does not know of any routes to subtrees of C and D. Since the root is in *storing mode*, it will *not attach any source routing header* to the packets. However, since node A is in *nonstoring mode*, it never received nor processed any DAO messages from the nodes in its subtree, and thus never stored any routing table entries for the nodes in its subtree. For this reason, the only action that node A can take when a packet is received from the root is to either consume

[4]Link local optimization is possible to allow communication between link neighbors, A–C and A–D , but it does not change the fact that any nodes in the subtrees of C and D will be disconnected.

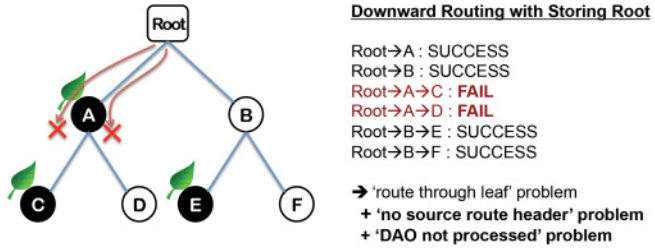

Fig. 5. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *storing mode* root with downward traffic. A *nonstoring mode* node in the middle of a routing path can partition the network since there is no source route header.

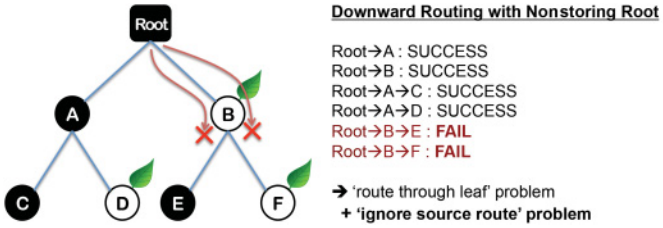

Fig. 6. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *nonstoring mode* root with downward traffic. A *storing mode* node in the middle of a routing path can partition the network since it ignores source route header.

the packet (pass it to upper layer) or forward it back to the default route toward the root, resulting in a routing loop. We call this the *no source route header problem* plus *DAO not processed problem*. To generalize this, any downward packet sent by a *storing mode* node headed to a *nonstoring mode* next hop node will result in forwarding failure. *Storing mode* nodes maintain the next hop for a given destination so that it can simply forward packets to the next hop. When a *nonstoring mode* node receives this packet, it realizes that it does not know how to forward the packet due to the lack of a source routing header and will send the packet upward using the default route, usually back to the node where the packet came from. Without a loop detection scheme, the packet will ping-pong between the two nodes to be eventually dropped when IPv6 TTL expires.

On the other hand, if the root is in *nonstoring mode* (Figure 6), there is a different problem. The packet sent by the root has a source routing header, but *storing mode* nodes do not process those source routing headers. Thus, the root cannot reach any nodes that are in the subtrees of E or F due to the fact that node B will ignore the source route on top of being a leaf node itself. We call this the *ignore source route* problem. In general, any downward packet forwarded from a *nonstoring mode* node headed to a *storing mode* next hop node will have a source routing header. When the *storing mode* node receives the packet, it ignores the source routing header because it is not used in *storing mode* operations. In this case, the *storing mode* node is indicated as the destination in the IPv6 header because RPL implements *strict source routing* [Hui et al. 2012]. As a result, the *storing mode* node receives a packet with its own IPv6 address in the destination field and will consume the packet without forwarding it to the actual destination.

3.3. Node-to-Node Routing

Node-to-node routing is basically a combination of upward and downward routing. Thus, all the problems in the upward- and downward-routing cases may occur. When

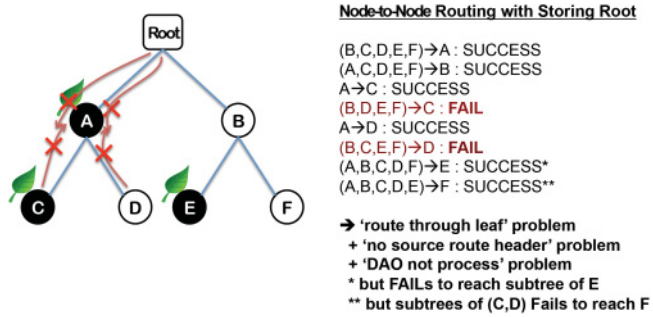

Fig. 7. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *storing mode* root with node-to-node traffic. A *nonstoring mode* node in the middle of a routing path can partition the network since there is no source route header.

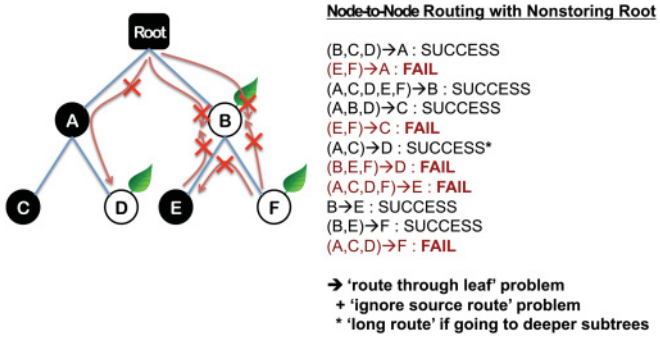

Fig. 8. A sample topology with a mixture of *storing* and *nonstoring mode* nodes and a *nonstoring mode* root with node-to-node traffic. A *storing mode* node in the middle of a routing path can partition the network since it ignores the source route header.

a node sends a packet destined to another node within the RPL network, the packet travels toward the root until it finds a node that knows the route to the destination. In *storing mode*, this can be an intermediate node, whereas in *nonstoring mode*, this node is always the root of the DODAG. However, this routing can fail in several ways in a network with mixed MOPs. First of all, we can have the *route through leaf* problem as before if a node in the middle of the path has an MOP different from the root. If the root is in *storing mode* (Figure 7), all downward packets will not have source routing headers (*no source route header problem*), and intermediate *nonstoring mode* nodes will not have routes to nodes in its subtree (*DAO not processed problem*), thus partitioning any nodes in the subtrees of C and D. On the other hand, if the root is in *nonstoring mode* (Figure 8), the downward packet from the root will have a source routing header, but any intermediate *storing mode* nodes will not process them to route packets to nodes in its subtree (*ignore source route problem*), thus disconnecting any subtrees of D, E, and F.[5]

Thus, a mixture of *storing* and *nonstoring* mode nodes in a single network, an outcome we argue is very likely in the IoT, can result in routing failures in *both upward and downward traffic* and node-to-node traffic. Nevertheless, given the respective benefits and tradeoffs of the two schemes, it is desirable to maintain both modes in the RPL specifications. While keeping both, we want to be able to form interoperable networks

[5]Again, link-local optimization is possible to allow communication between link neighbors, A–D, B–E, and B–F, but it does not change the fact that any nodes in the subtrees of D, E, and F will be disconnected.

regardless of the downward-routing mode being used in individual nodes. Thus, we propose a design and enhancements to the RPL standard with this goal in the following section.

4. *DUALMOP-RPL*: MAKING RPL OPERATE ACROSS STORING AND NONSTORING MODE NODES

When RPL nodes with different MOPs meet in a single network as discussed in Section 3, both upward and downward traffic are prone to routing failures and packet drops. Since data collection is the uppermost consideration in many LLNs, we first introduce a simple scheme to resolve the routing pathology for the upward traffic, and then propose additional modifications to resolve the downward and node-to-node cases.

4.1. Resolving the Problem in "Upward Routing" Case

Enhancement 1: "Leaf as a Router": In this first enhancement, we simply allow all nodes to act as routers *even across different MOPs*. Doing so can resolve the *route through leaf* problem and avoid network partitioning for upward routing as well as contributing to resolve the downward-routing case and the node-to-node routing case. Consider once more the network with mixed MOPs and a *storing* root in Figure 3. With this *leaf as a router* scheme, the *nonstoring mode* node A will no longer use infinity as its rank, and it can now act as a router that can forward packets. Thus, nodes C and D can now select node A as their parent, and the nodes in the subtrees of C and D can send packets through node A to the DODAG root. However, implementing this *leaf as a router* scheme does not fully remove the problems for downward routing (and thus node-to-node routing). Thus, we also propose additional enhancements to resolve RPL's routing pathology in both directions, which will follow next.

4.2. Resolving the Problems in "Downward Routing" Case

Our proposal to provide a bidirectional solution in forming inter-MOP RPL networks involves several small but critical changes to both *storing* and *nonstoring* modes. Especially, given that devices using *storing mode* will be more memory capable than *nonstoring mode* nodes, we pass on most of the additional functionality for supporting interoperability to the *storing mode* nodes while minimizing the changes in the *nonstoring mode*. Our proposed modifications to RPL do not define a new MOP. Each node will still select between *storing mode* and *nonstoring mode*. Our design defines a new enhanced RPL, called *DualMOP-RPL*, which allows interoperability between the two MOPs where nodes are expected to implement either the *storing mode* or the *nonstoring mode* along with our proposed changes that we detail later.

Enhancement 2: "Modified DAO Transmission": Once an RPL DODAG is constructed, all nodes wishing to receive potential downward packets send DAOs to their DODAG parents toward the root. Details on RPL DODAG construction are out of the scope of this article, and we point the readers to our previous work for details [Ko et al. 2011a]. As illustrated in Figure 9, DAOs contain information on how a specific destination can be reached. For *nonstoring mode*, the RPL standard suggests DAOs to be transmitted on an end-to-end basis. Since the only node that stores the information in a DAO is the DODAG root, the IPv6 destination of a DAO is set to the address of the root, allowing the packets to be forwarded without any intermediate processing. However, once *storing* and *nonstoring mode* nodes form a mixed RPL network, intermediate *storing mode* nodes should also be able to maintain routes for destinations in its subtree. For this purpose, we propose that, like storing mode DAOs, *nonstoring mode nodes should also send hop-by-hop DAOs*, providing all *storing mode* nodes on the path toward the root with a chance to store the DAO information. This enhancement will allow all DAO messages, from both *storing mode* nodes and *nonstoring mode* nodes, to

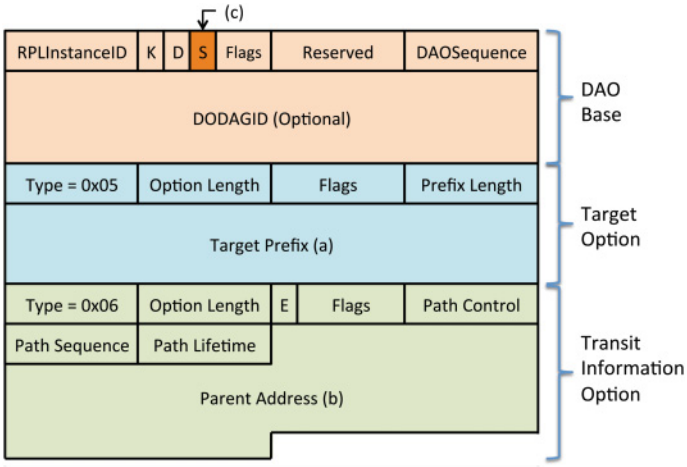

Fig. 9. Packet format of a DAO message including the base, target option, and transit option.

be processed at intermediate node, and this solves the *DAO not processed* problem in Figure 5.

Note that this enhancement does not incur any extra communication overhead in terms of packet transmission or reception; the same message is transmitted from the initiator of the DAO message to the root once anyway. The only cost is in inspecting the DAO message and replacing the destination address in the header at each hop, which is done anyway if standard *storing mode* is used, and we believe this cost is negligible in terms of processing delay.

Enhancement 3: "Modified DAO Format": When a DAO is received at a *storing mode* node, the node records which node initiated the DAO (field **(a)** in Figure 9). Using this information and the IPv6 source address of the DAO, each node can determine the next hop node to use in order to reach a specific destination. On the other hand, in *nonstoring mode*, DAO packets explicitly include information on a target node's parent node in the DODAG (item **(b)** in Figure 9). With such "per hop" routing segments from all the nodes in the network, the root can construct routing paths to each node and include this information in the source routing header when needed. However, the RPL standard states that the address field of the `transit option` (item **(b)** in Figure 9) is optional when operating in *storing mode*. The lack of this information from a subset of nodes makes the downward route construction process difficult. For this reason, we propose that *all DAOs, from both* storing *and* nonstoring mode *nodes, should include the* `transit option` *'s address field* for supporting interoperability. This enhancement, together with others, resolves the *no source route header* problem for packets being sent by *storing mode* nodes.

Note that the additional byte overhead from this enhancement comes only for nodes operating in *storing mode* since in *nonstoring mode*, the transit information option is already mandatory as per RFC 6550. In any case, as we will show in Section 5, the extra overhead incurred by this enhancement is insignificant compared to the total packet size and the benefits we gain from the changes.

Enhancement 4: "Modified Source Routing Header Support": By using the earlier *Modified DAO Format*, each *storing mode* node in an RPL network should have the information required to construct a routing path to reach any destination in its sub-DODAG, just like the *nonstoring* root. In *nonstoring mode*, the root of the DODAG uses this information to construct a source routing header that nonroot nodes can use

to forward packets downward toward the packet's destination, but *storing mode* nodes will ignore the source route header in these packets. To resolve this *ignore source route* problem exemplified in Figure 6, we propose that *all storing mode nodes implement source routing header support* for RPL interoperability. Since RPL requires *strict source routing*, *storing mode* nodes should also understand and follow the routing information included in the source routing header (if present) of the incoming packet [Hui et al. 2012]. By having *storing mode* nodes *attach and understand source routing headers* when needed, downward-routing problems due to *no source route* or *ignore source route* can be resolved. Furthermore, it can also improve the performance of the node-to-node routing. Packets need not travel all the way to the root, but rather a source routing header can be attached at an intermediate *storing mode* node to reduce the stretch. This enhancement requires extra complexity in the *storing mode* nodes, but it is a one-time implementation from which we gain improved network performance.

(Optional) Enhancement 5: "S toring Mode Flag": To further optimize the routing paths that *storing mode* nodes construct, we propose an optional change that requires modifying one of the reserved bits in the DAO base. This flag, **(c)** in Figure 9, is set when the DAO initiating node supports *storing mode* and cleared otherwise. A *storing mode* node receiving this DAO message must store this information (whether *storing mode* flag is set or not) in its forwarding table. This optional flag can assist in constructing a more efficient route toward the packet's final destination since a node can use it to construct downward routes only up to the next *storing mode* node rather than constructing a full end-to-end routing path that can be much longer in length. Furthermore, since this enhancement is using a single bit in one of the unused bits in the DAO base, there is no extra overhead for this enhancement.

4.3. Putting it All Together

When a packet is initially sent from a node, it first travels toward the DODAG root until reaching the first *storing mode* node. In *DualMOP-RPL*, the DODAG root acts in *storing mode* by default but is also capable of constructing a source routing header (like *nonstoring mode*) if needed. Based on the IPv6 destination address, a *storing mode* node determines if the packet should be transmitted using the default route (toward the DODAG root) or downward.[6] Once a *storing mode* node determines that the packet can head downward (the destination is in its sub-DODAG), and if the *storing mode* flag is used, the node checks if the next hop node (based on the forwarding table) uses *storing mode* or not. Whether the next hop node is using *storing mode* or not is known from the *storing mode* flag in the DAO messages that the next hop nodes sent previously. If so, the packet is simply forwarded to the next hop (typical *storing mode* operation). Notice that this is a benefit of using the *storing mode* flag. On the other hand, if the next hop implements *nonstoring mode* or if the storing mode flag is not used (there is no way of knowing whether the next hop node is using *storing mode* or not), then the node searches and computes the route segments in its routing table, copies the resulting path to the source routing header, and then attaches it to the packet before it is transmitted to the next hop (as the *nonstoring mode* root). Other than these operations, all other operations regarding the construction and forwarding of a packet with or without a source routing header follows the specifications of the RPL standard [Ed. et al. 2012].

[6]Nonstoring nodes simply send the packet to the default route if the packet does not have a source routing header.

Table I. Five Test Cases for Evaluation

Test case	Non root Node's MOP	Root's MOP
A	*Storing Mode* Only (Standard compliant)	
B	*Non-storing Mode* Only (Standard compliant)	
C	Mixed MOP (Standard compliant)	*Storing Mode* Root
D	Mixed MOP (Standard compliant)	*Non-Storing Mode* Root
E	Mixed MOP (*DualMOP-RPL*)	*DualMOP-RPL*

5. PERFORMANCE EVALUATIONS

Among many open-source implementations of IETF RPL, we identified two implementations each using different downward-routing schemes: TinyOS implements RPL with the *storing mode* downward routing [Ko et al. 2011a, 2011b], while NanoQplus implements the *nonstoring mode* version of RPL [Jeong et al. 2011]. We experimentally validate the effectiveness of our enhancements to the RPL standard (RFC6550) using these two implementations. For this purpose, we implemented our enhanced RPL on both TinyOS and NanoQplus. One natural (and beneficial) side effect of our work is that we are also testing the standard compliance and interoperability of these two implementations. For the routing metric, we use hop count (e.g., OF0), which is the default routing metric in RPL networks [Thubert 2012]. Also, we use the default CSMA MAC layer in TinyOS and NanoQplus with up to four link-layer retransmissions.

For both simulations and testbed experiments, we used five different network configurations to compare five different test cases. These five test cases are summarized in Table I. In the first two test cases, A and B, all nodes ran the same MOP under the same implementation, either *storing mode* only or *nonstoring mode* only, of RPL. The other three test cases C, D, and E, had a mixture of *storing mode* and *nonstoring mode* nonroot nodes where the number of nodes were equally divided between the two MOPs. For the root node, test cases C and D each ran *storing mode* and *nonstoring mode* root respectively, and test case E ran the *DualMOP-RPL* root. The first four test cases, A–D, ran standard compliant RPL (TinyOS and/or NanoQplus implementations, as is), and test case E ran our enhanced version of RPL, the *DualMOP-RPL*. Note that the storing mode flag was enabled in test case E for both simulations and testbed experiments, except for Section 5.3, where we evaluate the benefit of using the storing mode flag.

For these five test cases, we evaluate and compare the average end-to-end packet reception ratio (PRR) for (1) upward data collection traffic, where all nonroot nodes send periodic data packets to the DODAG root, and for (2) downward traffi,c where the root initiates packets to all nodes in the network sequentially, both with an interval of 5 seconds. Another interesting type of traffic to evaluate would be the node-to-node traffic to and from nonroot nodes, but we leave this as part of future work since the results would be highly topology dependent and would be more meaningful when evaluated with a real application scenario. Although the actual consequences of mixing nodes with different MOPs within a single RPL network will be network partitions and link breakages, we are abstracting the whole network behavior into an average PRR for the convenience and clarity of presentation since listing all broken links out of potentially N^2 links would be hard to interpret.

5.1. Simulation

We start our evaluation in a simulation environment. Specifically, using the Contiki Simulation Environment and its Cooja Network Simulator, we generate a random topology for each of the five test cases in Table I and compare the average PRR between them. Figure 10 illustrates an example 25-node simulation topology used for test case C in our evaluations. White diamonds represent a *storing mode* node, black circles

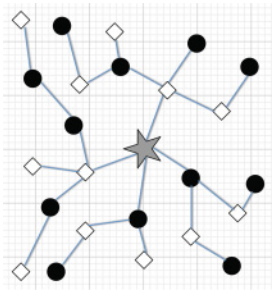

Fig. 10. An example illustration of a simulation topology used for our Cooja simulations. Topologies were randomly generated for all five test cases in Table I. This example is for test case C, where we mix *storing mode* and *nonstoring mode* nodes in the same network with a storing mode RPL root.

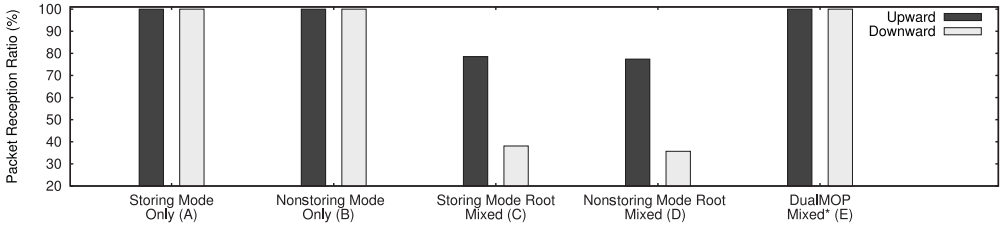

Fig. 11. Simulation result: end-to-end average packet reception ratio (PRR) of upward and downward data traffic observed in the Cooja simulator for all five test cases in Table I.

represent a *nonstoring mode* node, gray start is an RPL root in *storing mode*, and the lines represent the links used by RPL. It randomly mixes together nodes operating in *storing mode* and *nonstoring mode*, and such a topology will result in some of the *nonstoring mode* nodes selecting *storing mode* nodes as RPL parents and vice versa.

Figure 11 shows the simulation result for all five test cases. PRRs for the homogeneous setup (test cases A and B) are very high, almost 100%, and we use these as baseline references for comparison. As predicted, when the network consists of mixed MOP nodes (test cases C and D), the PRRs drop noticeably for both upward and downward traffic due to network partitioning and the selection of nonoptimal paths, and the results are similar regardless of the MOP of the root node (C vs. D). In these cases, the PRR of the downward traffic is significantly worse than the upward case for the four reasons explained Section 3.2. However, as we can see from the result of test case E, the average PRR improves significantly for both traffic types when *DualMOP-RPL* is used, back to almost 100%, similar to the baseline homogeneous cases.

First of all, this result verifies that the RPL standard has a significant interoperability problem when operating in a mixed or heterogeneous MOP configuration. RPL's restrictions regarding downward-routing schemes deteriorate the reliability performance and efficiency of RPL, affecting both upward and downward traffic, when nodes with different MOPs are mixed within a single RPL network. Although the current RPL standard as is can, to some level, provide connectivity to some of the different MOP nodes by allowing them to connect as a leaf node (or with the help of IPv6 link local optimization), because they are not fully interoperable and are not capable of forming a hybrid network, it sacrifices the efficiency and the reliability performance of the network as a whole. Most importantly, this result verifies that our proposed *DualMOP-RPL* solves the connectivity problem and provides near-perfect reliability at the same level as the homogeneous RPL networks.

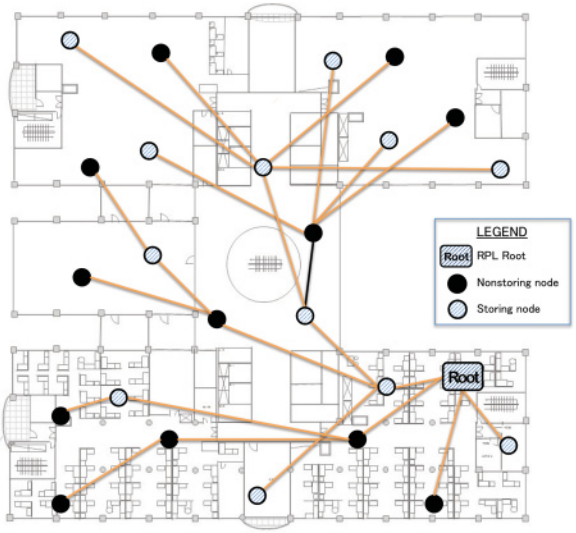

Fig. 12. Topology snapshop of our testbed experiment where we mixed *storing mode* and *nonstoring mode* nodes in the same network with a DualMOP-RPL root.

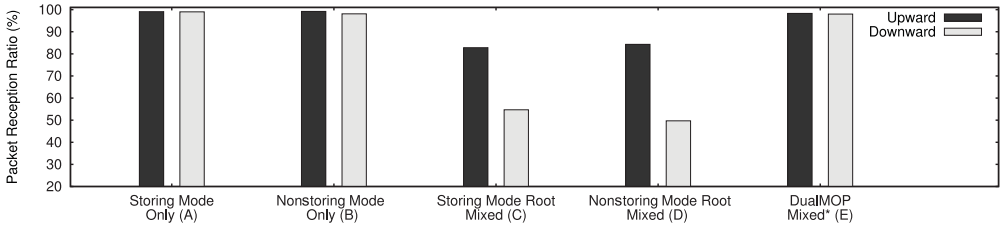

Fig. 13. Testbed experiment result: end-to-end average packet reception ratio (PRR) for upward and downward data traffic on the testbed for all five test cases in Table I.

5.2. Testbed Experiment

With these insights, we moved our evaluation environment to a testbed of 25 TMote Sky nodes in an indoor office environment. Our topology had a maximum of four hops and the experiments were performed in a hallway environment as depicted in Figure 12. We mixed together nodes operating in *storing mode* and *nonstoring mode*, where the black circle represents a *nonstoring mode* node, the white circle (with slight patterns for clarity) represents a *storing mode* node, and the rectangle is the RPL root running in either *storing mode*, *nonstoring mode*, or the *DualMOP-RPL* root, depending on the test cases. We loaded two different types of binaries (*storing mode* implemented in TinyOS and *nonstoring mode* implemented in NanoQplus) on to the motes depending on the five different test cases detailed in Table I. All results reported in this subsection are an average of five independent runs of each test case.

Figure 13 presents the average PRR for all five testbed experiment test cases. We can easily notice that the results are very similar to that of the simulations. The PRRs for the homogeneous setup (test cases A and B) are very high, close to 100%, and the PRR drops noticeably for both upward and downward traffic when the network consists of mixed MOP nodes (test cases C and D). However, similar to the simulation results, the PRR improves significantly for both traffic types when the *DualMOP-RPL* is used (test case E).

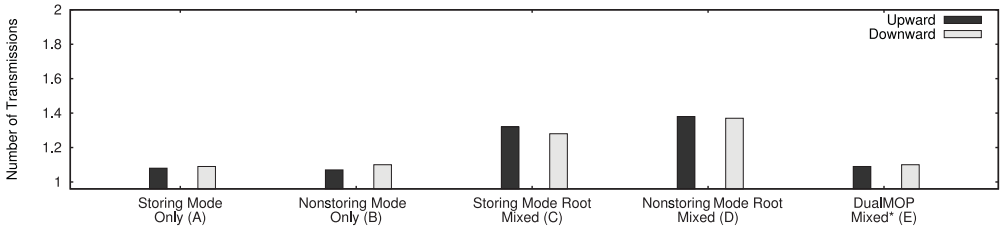

Fig. 14. Average number of per-hop transmissions for each data packet for all nodes in the network.

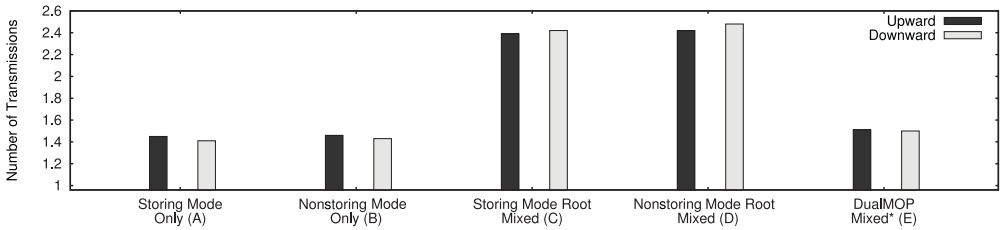

Fig. 15. Average number of end-to-end transmissions needed for each packet received at the final destination for all nodes in the network.

Figure 14 and Figure 15 provides additional explanation behind such results by showing the average number of transmission attempts at each hop (reflecting the quality of each selected link) and the average number of end-to-end transmissions that occurred for each data packet (reflecting the overall quality of the entire route selected), respectively, for all five test cases. We point out that nodes make no end-to-end retransmission attempts upon delivery failure (i.e., no end-to-end ARQ). In test cases C and D, nodes with different MOPs than the root connect to the RPL network only as leaf nodes, forcing the opposite MOP nodes to form a "backbone-like" network for these nodes to connect to. Given that the average link distance increases when the number of usable nodes decreases (e.g., nonstoring mode node positioned between two storing mode nodes cannot be used as a relay node), the average link quality is expected to drop in test cases C and D. This is the reason that the link (and path) quality of the network in test cases C and D are worse than that of test cases A, B, and E. As a result, the network in test cases C and D selects links with poor quality and requires more transmission attempts to deliver a packet to its final destination. On the other hand, test case E uses all the nodes in the network as router nodes. In other words, compared to test case E (and also homogeneous networks in test cases A and B), test cases C and D select links that are less efficient both locally and globally, resulting in a lower PRR in Figure 13.

There are slight differences between the simulation results in Figure 11 and the testbed experiment results in Figure 13. Minor differences in the PRR of 1% to 3% for test cases A, B, and E are simply due to the differences between simulation and reality; there are more link losses in a real wireless environment. There are also small but nonnegligible differences in the PRRs for test cases C and D, where the testbed results are slightly better than those of the simulation. This is due to the fact that the network depth of our testbed experiment topology was shallower, resulting in a lesser number of broken links and network partitions, especially with the help of IPv6 link-local optimizations for one-hop links.

To summarize, testbed experiments confirmed the findings from the simulations. The RPL standard has a significant interoperability problem between its own MOPs, resulting in poor reliability performance for both upward and downward traffic when

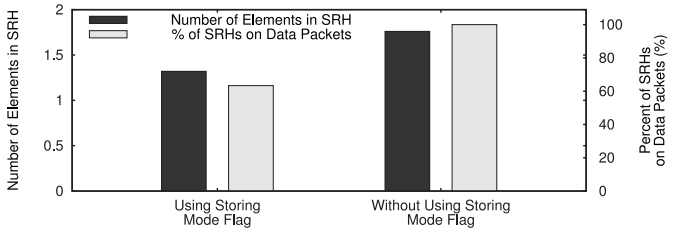

Fig. 16. Average number of routing path elements in source routing headers and the percentage of data packets with source routing headers attached when using (or not using) the proposed storing mode flag in *DualMOP-RPL*.

Table II. Memory Usage Increase When Implementing *DualMOP-RPL* in TinyOS or NanoQplus Implementations of RPL

Operation Mode	ROM	RAM
NanoQplus MOP (Nonstoring Mode)	+0.18KB	+0KB
TinyOS MOP (Storing Mode)	+1.66KB	+0.10KB

nodes with different MOPs are mixed within a single RPL network. It also confirmed that our proposed *DualMOP-RPL* solves all the connectivity problems identified in Section 3, even in a real wireless environment, and provides near-perfect reliability at the same level as the homogeneous RPL networks.

5.3. Effect of the Optional Enhancement "Storing mode flag"

One benefit of allowing intermediate *storing mode* nodes to attach source routing headers to the packets, rather than only allowing the root to do so as in *nonstoring mode*, is the reduction in route stretch compared to packets traveling all the way to the root to attach the source routing header. To further minimize the overhead caused by source routing, we proposed using a storing mode flag in DAO messages (i.e., field **(c)** in Figure 9) as an optional enhancement. This flag helps reduce the number of routing elements contained in each source routing header and also minimize the ratio of data packet transmissions that require these headers. Our results in Figure 16 show that the average length of the routing segments and the percentage of downward data packets with source routing headers decrease when DAO messages utilize the storing mode flag. We expect this trend to hold or improve even further in more complex network topologies as well.

5.4. Memory Usage, Overhead, and Constraints

Our proposed *DualMOP-RPL* is expected to be installed on devices with memory limitations. Furthermore, *nonstoring mode* nodes are expected to have fewer resources than *storing mode* nodes. To determine the practicality of our proposal, we measure the increases in RAM/ROM requirements of the software due to our proposed enhancements. As Table II shows, our modifications increase the ROM requirement by 0.18KB and do not increase the RAM requirement for a TMote Sky binary using the NanoQplus implementation of the *nonstoring mode*. *Storing mode* nodes in TinyOS needed 1.66KB more ROM and 0.1KB more RAM space. Considering that TMote Sky motes have 48KB of ROM and 10KB of RAM space available, this shows that the memory footprint increases are minimal on both types of downward-routing schemes (and especially lower for *nonstoring mode*), suggesting that our scheme is suitable for resource-constrained hardware platforms used in many LLNs.

Regarding the overhead of mandating the optional transit address (parent address) in all DAO messages (enhancement 2), the additional byte overhead comes only when

operating in *storing mode* since in *nonstoring mode*, the transit information option is already mandatory as per RFC 6550. While RFC 6550 does not specify when DAOs should be transmitted, in our implementations (both NanoQplus and TinyOS), DAOs are generated based on a Trickle Timer and the timer is reset when a new preferred parent is set due to changes in uplink conditions. In our 25-node testbed experiments, for a test duration of 24 hours, an average of 1.87 DAO packets were transmitted per node per hour. Considering that 50% of the nodes were storing mode nodes and each DAO transmission requires an additional 16 bytes of transit information option for these storing mode nodes, this totals ∼14.96 bytes per node per hour overhead in the network. Considering an IEEE 802.15.4 250kbps link, this results in an additional ∼0.34msec (∼6.48ms →∼6.82ms) of packet transmission times for DAO per node per hour. Therefore, we believe that the overhead in terms of transmission time and energy expenditure is minimal. As a result, we believe that the extra overhead incurred by this enhancement is insignificant compared to the total packet sizes and the benefits we gain from them.

Finally, we acknowledge the fact that 25 nodes are not sufficient enough to showcase the full benefits of maintaining a heterogeneous network, nor to fully demonstrate the effectiveness of our proposed scheme at a large scale. Nevertheless, our evaluation was constrained by the limited memory bounds of the hardware (TelosB/Tmote Sky)[7] and the software platforms (TinyOS) used for our evaluation.[8] In an RPL network of n nodes, if at least one *storing mode* node is included in the network, this *storing mode* node should in the worst case be able to maintain a routing table for all other $n - 1$ nodes. However, given the memory limitation, 25 was the maximum number of nodes that a *storing mode* node could support in our implementations. In the future, when our proposal is adopted to a real commercial system, we envision that the small number of nodes that operate the *storing mode* of RPL will utilize higher-power MCUs such as ARM Cortex-M3-based platforms with more memory, while the majority of the network consists of resource-constrained nodes operating in *nonstoring mode*. Nevertheless, our results using the 25-node network is still enough to prove the superior performance of our proposed scheme compared to the cases where the current standard (RFC6550)-based RPL is used in a mixed or heterogeneous network by resolving RPL's fundamental interoperability problems. Furthermore, we believe that the performance benefits of *DualMOP-RPL* would have been even greater if we had tested it at a larger scale due to the fact that the standard-compliant implementations would most likely have more network partitions if the network was larger or deeper.

6. CONCLUSION

With the introduction of the IETF RPL protocol and other IPv6-related standards, industrial kick-off of low-power wireless sensing systems is finally in close reach. It is now the role of both academia and the industry to validate the performance of these protocols in as many realistic scenarios as possible. In this article, we have discussed the significant interoperability problems in the downward-routing schemes of the RPL protocol. We have identified the cases where the routes can break down and result in packet delivery failures due to network partitions and also verified our analysis and findings through simulations and testbed experiments. Since each has its own strengths and tradeoffs on devices with different capabilities, it is important to preserve the two types of downward-routing MOPs in RPL. For this reason, we have proposed a novel lightweight solution, *DualMOP-RPL*, that supports nodes with

[7]TelosB and Tmote Sky motes have RAM of 10KB and ROM of 48KB.

[8]This is implementation dependent, and it is possible to run larger-sized experiments with newer-generation hardware or smaller code-size software implementations.

different MOPs to interoperate gracefully in a single RPL network. This enables the design of RPL networks with heterogeneous downward-routing schemes, which opens the potential for new application developments. We have extensively evaluated the performance of our proposed enhancements to the RPL protocol in both simulations and testbed environments and proved that our proposal eliminates all the problems we have identified and achieves packet delivery performance identical to the homogeneous cases.

REFERENCES

R. Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Lakshman Krishnamurthy, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. 2005. Design and deployment of industrial sensor networks: Experiences from the north sea and a semiconductor plant. In *Proceedings of ACM Sensys 2005*.

A. Brandt, J. Buron, and G. Porcu. 2010. Home automation routing requirements in low-power and lossy networks. *RFC 5826* (April 2010).

Cisco. 2015. Smart Grid - Field Area Network. Retrieved from http://www.cisco.com/web/strategy/energy/field_area_network.html.

T. Clausen, U. Herberg, and M. Philipp. 2011. A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL). In *Proceedings of the 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'11)*. 365–372.

T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander. 2009. Routing requirements for urban low-power and lossy networks. *RFC 5548* (May 2009).

T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander. 2012. RPL: IPv6 routing protocol for low-power and lossy networks. *RFC 6550* (March 2012).

German Federal Ministry of Education and Research. 2015. Project of the Future: Industry 4.0. Retrieved from http://www.bmbf.de/en/19955.php.

T. Heo, K. Kim, H. Kim, C. Lee, J. Ryu, Y. Leem, J. Jun, C. Pyo, S-M. Yoo, and J. Ko. 2014. Escaping from ancient Rome! Applications and challenges for designing smart cities. *Transactions on Emerging Telecommunications Technologies* 25, 1 (2014), 109–119. DOI: http://dx.doi.org/10.1002/ett.2787

J. Hui, J. P. Vasseur, D. Culler, and V. Manral. 2012. An IPv6 routing header for source routes with the routing protocol for low-power and lossy networks (RPL). *RFC 6554* (March 2012).

IEEE Std. 802.15.4-2003 2003. IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks. Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). Retrieved from http://www.ieee802.org/15/pub/TG4.html.

J. Martocci, P. De Mil, N. Riou, and W. Vermeylen. 2010. Building automation routing requirements in low-power and lossy networks. *RFC 5867* (June 2010).

J. Jeong, J. Kim, and P. Mah. 2011. Design and implementation of low power wireless IPv6 routing for NanoQplus. In *Proceedings of the International Conference on Advanced Communication Technology (ICACT'11)*.

K. Pister, Ed. P. Thubert, S. Dwars, and T. Phinney. 2009. Industrial routing requirements in low-power and lossy networks. *RFC 5673* (Oct. 2009).

J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. 2011a. Evaluating the performance of RPL and 6LoWPAN in TinyOS. In *Proceedings of the Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN'11)*.

J. Ko, J. Lim, Y. Chen, R. Musaloiu, A. Terzis, G. Masson, T. Gao, W. Destler, L. Selavo, and R. Dutton. 2010. MEDiSN: Medical emergency detection in sensor networks. *ACM Transactions on Embedded Computing Systems (TECS), Special Issue on Wireless Health Systems* (2010).

J. Ko, A. Terzis, S. Dawson-Haggerty, D. E. Culler, J. W. Hui, and P. Levis. 2011b. Connecting low-power and lossy networks to the internet. *IEEE Communications Magazine* 49, 4 (April 2011), 96–101.

M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci. 2013. Reactive discovery of point-to-point routes in low-power and lossy networks. *RFC 6997* (Aug. 2013).

G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. 2007. Transmission of IPv6 packets over IEEE 802.15.4 networks. *RFC 4944* (Sept. 2007).

J. Paek and R. Govindan. 2010. RCRT: Rate-controlled reliable transport protocol for wireless sensor networks. *ACM Transactions on Sensor Networks* 7, 3 (Oct. 2010), Article 20, 45 pages.

J. Paek, B. Greenstein, O. Gnawali, K-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler. 2010. The tenet architecture for tiered sensor networks. *ACM Transactions on Sensor Networks* 6, 4 (Juy 2010), Article 34, 44 pages.

J. Paek, J. Hicks, S. Coe, and R. Govindan. 2014. Image-based environmental monitoring sensor application using an embedded wireless sensor network. *Sensors* 14, 9 (2014), 15981–16002. DOI:http://dx.doi.org/10.3390/s140915981

T. Schmid, R. Shea, M. Srivastava, and P. Dutta. 2010. Disentangling wireless sensing from mesh networking. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensors (HotEmNets'10)*.

P. Thubert. 2012. Objective function zero for the routing protocol for low-power and lossy networks (RPL). *RFC 6552* (March 2012).

Received March 2014; revised July 2014; accepted September 2014