Ad Hoc Networks 23 (2014) 121–144

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Proactive versus reactive routing in low power and lossy networks: Performance analysis and scalability improvements

Joydeep Tripathi [a], Jaudelice C. de Oliveira [a,*], J.P. Vasseur [b]

[a] Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104, USA
[b] Cisco Systems, USA

ARTICLE INFO

Article history:
Received 9 August 2013
Received in revised form 16 June 2014
Accepted 30 June 2014
Available online 9 July 2014

Keywords:
Low Power Lossy Networks (LLNs)
Sensor networks
RPL
LOADng

ABSTRACT

In this paper, the classical debate on suitability of proactive versus reactive routing approaches is revisited, however in the context of their application in Low-Power Lossy Networks (LLNs) as opposed to pure Mobile Ad Hoc Networks (MANETs). We argue that LLNs differ from traditional ad hoc networks not only due to node capacity, but also in the nature of traffic generation and routing requirements. This study aims at a fair comparison between two protocols proposed at the IETF, namely RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) – the proactive candidate, and LOADng (LLN On-demand Ad-hoc Distance vector routing protocol – next generation) – the reactive candidate, making use of real traffic scenarios and topology deployments of varied size, as well as random topologies in the particular case of metrics relating to scalability. Several metrics of interest are investigated, including metrics that have not been paid much attention in the existing MANET literature. In the course of this investigation, we also uncovered non-optimal protocol behavior for the case of large networks and proposed new mechanisms that are shown to improve control plane congestion, as a result improving network lifetime for large scale LLNs. We also derive bounds on control overhead for the two protocols, which indicate that RPL incurs an overhead that is lower than $O(N^2)$ and reduces to $\Theta(N\log(N))$ for a balanced tree structure, whereas LOADng has a total control overhead of $\Theta(N^2)$, irrespective of topology.

1. Introduction

Networks connecting smart, power constrained devices, such as sensor nodes, often operate in highly variable link quality conditions, and have been termed as Low Power and Lossy Networks (LLNs). The challenges in these networks include very low device power and memory, highly varying link quality, frequent link outage, amongst others. Requirements of deployments in smart home, building or industrial automation, communication among Advanced Metering Infrastructure (AMI) meters in a smart grid, etc., relate to delay bound, scalability, and strict time bounds on protocol convergence after any change in topology [1]. Link state routing protocols such as OSPF [2], OLSR [3], IS-IS, and OLSRv2 [4] tend to flood the network with link updates. Since links in LLNs suffer from severe temporal variation and frequent outage, these protocols fail to keep a low control cost overhead [5]. Classical distance vector protocols used in the Internet, such as EIGRP [6], and AODV [7], fail to provide quick recovery from link churn, and frequent topology updates.

A suitable protocol for LLNs therefore needs to abide to strict delay constraints, while maintaining a minimum control overhead, and should be capable of providing quick

* Corresponding author.
E-mail addresses: jt369@drexel.edu (J. Tripathi), jau@coe.drexel.edu (J.C. de Oliveira), jpv@cisco.com (J.P. Vasseur).

http://dx.doi.org/10.1016/j.adhoc.2014.06.007
1570-8705/

recovery from frequent link outage. The IETF ROLL (Routing Over Low power and Lossy network) working group was chartered to identify an IP routing solution for such networks under specific deployment scenarios. After surveying existing solutions, the working group concentrated its efforts on designing and recently standardizing RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [8], a distance vector protocol with proactive route creation, which provides mechanisms for multipoint-to-point, point-to-multipoint, as well as point-to-point traffic (for a brief overview of RPL, please refer to Section 3). RPL was designed to meet the requirements spelled out in RFC 5826 [1], which defines home automation routing requirements in LLNs; RFC 5673 [9], which defines routing requirements in the industrial setting; RFC 5548 [10], defining the requirements for urban LLNs, and RFC 5867 [11], defining the requirements in building automation. For instance, home [1] and building [11] automation routing requirements call for an end-to-end delay of less than 500 ms and 120 ms, respectively. Simulation studies of RPL in such environments have provided satisfactory results [12–14]. Ref. [12] has also shown that RPL can perform faster repair to converge in case of link churns, a feature mandated in urban [10] and home [1] LLN routing requirements.

Recently, an alternate protocol called LOADng (LLN On-demand Ad-hoc Distance vector routing protocol – next generation) [15], was proposed at the IETF. LOADng is inspired by the AODV protocol, being reactive to incoming traffic (for a brief overview of LOADng, please refer to Section 3). In [16], the authors of LOADng have compared it to a simplified version of RPL and shown the benefits of the former with respect to control overhead, delay, frame collisions, etc. In their comparison, however, RPL has been naively configured (e.g., time period for DAO emissions) resulting in an unnecessarily cumbersome implementation, which lead the authors to simulate it for a shorter period of time and for a completely different traffic pattern than that of LOADng's, therefore jeopardizing the value of the conclusions reached in the paper. In the following, we attempt to summarize the main issues in the study:

- The study considers that DAO messages in RPL are emitted periodically every 15 s, quite an unrealistic assumption for any RPL deployment. A node unicasts DAO packet to the DAG root (in non-storing mode) or multicasts to parent set (in storing mode) only in two cases: (i) a new DAG sequence number is received and the network is undergoing a global repair, and (ii) the most preferred parent is changed as a result of non-reachability to the previous parent.
- The amount of time RPL and LOADng are run differs and moreover the traffic pattern for each is also different. In RPL, data is sent from the "controller" or collection point every 0.1 s, and for LOADng, that duration between successive data transmission is 30 s. As we show in this study, the performance of LOADng depends hugely on data traffic load. Hence we argue that [16] does not provide a fair ground for comparison between the two.
- Furthermore, the deployed network in [16] does not assume any variation in link quality, link outage, or any of the lossy characteristics that a typical LLN posses, leading to 100% delivery of the packets.

This paper aims at a fair comparison between the two protocols using a realistic simulation study that includes an investigation on appropriate configuration parameters for both protocols. Since many metrics on comparison of proactive and reactive protocols are well studied, we concentrate more on issues and metrics specific to LLNs that have not been offered much attention in existing literature, such as temporal variation of control overhead due to multicast traffic, memory consumption, packet length for source routing, and scaling properties. To accomplish this, the RPL simulator that we developed for the study in [13] was extended to include LOADng's implementation described in [15].

While carrying on a detailed comparison study between RPL and LOADng, we also demonstrated for the first time that RPL, though performing better than LOADng, may not behave optimally in large scale networks in terms of memory consumption. As stated in our earlier work [17], we argue that the rationale behind this observation is mainly the congestion caused by the Destination Advertisement Option (DAO) packets from every node to the DAG root during a global repair (see Section 3), or after the DAG root triggers a network-wide address accumulation using a new DAO Trigger Sequence Number (DTSN). This congestion is lethal as it may lead to increased memory requirement, dropped destination advertisement, more control traffic, and loss or delay in important and time-sensitive data or alert packets. Therefore, in order to control the congestion in large deployments of memory constrained nodes, it is essential to design a suitable approach for DAO message emission. In the same work, we have demonstrated that instead of using a fixed universal timer to control DAO emissions, as recommended in the standard, making use of an adaptive timer at each node allows the network to adjust itself to account for topological changes, and adjust each timer in order to avoid congestion and packet drops, specially near the DAG root. In this paper, we further propose a combination of centralized and distributed approach to control DAO emissions in RPL non-storing mode.

This paper's organization as well as a summary of contributions can be described as follows:

- Section 2 summarizes related work on proactive versus reactive protocols, as well as comments on the suitability of geographic protocols. It also summarizes the literature on performance comparison between RPL and other protocols.
- A brief overview of the protocols, RPL and LOADng, is provided in Section 3.
- The details of the simulation model used and traffic generation are described in Section 4. Gathered link quality trace and topology details from 2 networks – one 86 node outdoor deployment, and a 2442 node AMI network were used to emulate real LLN link behavior and network conditions.

- Sections 5 and 6 show various performance results for metrics of interest for both protocols for typical traffic in smart grid, and for P2P traffic, respectively.
- Section 7 analyzes the protocols' general scaling properties, pointing out RPL's comparatively better, but non-optimal behavior in large scale.
- Section 8 describes the pathology behind RPL's non-optimal behavior in large scale deployment, and proposes a combined distributed-centralized approach, which is shown to outperform the default RPL.
- In Section 9, we summarizing the outcomes observed.
- Appendix A attempts at intuitively predicting the protocols' behavior and providing a simple model for analytical comparison.

2. Proactive, reactive and geographic routing solutions

It may be argued that the debate of reactive versus proactive protocols has been extensively researched in the context of traditional ad hoc networks, and it is a well-known fact that reactive protocols perform better in networks with low traffic volume. We however argue that LLNs are different from traditional ad hoc networks not only due to node capacity, but also in the nature of traffic generation and routing requirements, making it worth to revisit the debate in the context of LLNs.

Some of the recent literature on reactive versus proactive debate include [18–22]. Ref. [18] compares OSPF and AODV in the context of Mobile Ad-hoc NETworks or MANETs using grid networks of sizes 9, 25, 49, and 81 on TOSSIM simulator, and concludes in favor of AODV. Clearly, the network scale considered in this study is much smaller than the size envisioned for several instances of LLNs. The authors, however, realize that as the number of simultaneous flows increases, proactive protocols tend to respond more favorably. Similar result had been concluded analytically by the authors of [23]. Their work compares proactive and reactive approaches from energy consumption aspect, and determines that there exists a cross over point in message duty cycle, beyond which reactive protocols consume more overhead and energy than proactive routing protocols. In [19], the authors considered various QoS metrics qualitatively to judge the merits of several routing protocols in MANET, and concluded that reactive protocols outperform proactive counterparts in terms of scaling, power consumption, control overhead and table size. Ref. [20] compares the protocols AODV, DSR (Dynamic Source Routing [24]) and OLSR in terms of metrics such as throughput, and end-to-end delay, for varying traffic loads, number of flows and network sizes up to 100 nodes. A link state protocol was chosen as a proactive routing candidate and the authors conclude that proactive protocols are better suited for MANETs where the traffic is constant and periodic. Ref. [21] compares a proactive distance vector protocol, DSDV [25], with a reactive protocol, DSR, and a hybrid protocol, ZRP [26], and concludes that DSDV performs poorly when compared to the other protocols with respect to metrics such as delay, number of dropped packets, and routing overhead. In [22], the authors have compared different protocols such as OSPF, DSDV, TORA (Temporally-Ordered Routing Algorithm [27]), DSR, and AODV in terms of throughput, delay and routing load. However, Medium Access Control (MAC) protocol are ignored in the study, thus effects of typical lossiness and congestion of wireless links have been neglected, jeopardizing the results obtained. Almost all of the literature above indicated that proactive protocols have a much lower delay than reactive protocols, but suffer more from high control overhead.

Recently, geographic routing protocols, such as [28], have also gathered attention in solving the data gathering problem in constrained networks. One of the main advantages of geographic routing is its low amount of state required to run the network. Indeed, in [29] the authors have developed a framework to successfully implement geographic routing in an RF mesh network. However, the idea of appending a Global Positioning System (GPS) to every node in some LLNs has detrimental consequences on low power devices, and thus has been abandoned by the IETF ROLL working group [30]. The work in [31] provides performance comparison in terms of delay and hop count between RPL and a geographical routing protocol for a 500 node smart utility network. Geographic routing keeps a state of O(1) regardless of network size, and thus can be effective in large scale urban LLNs. Ref. [31] also shows that geographical routing provides a larger delay and hop count than RPL.

The above mentioned literature fails, however, to reach the network scale of thousands of nodes, which is typical for some LLN deployments. The traffic considered in most cases is CBR, with no multicast traffic analysis. Moreover, previous comparisons did not consider varying link quality in their simulations.

RPL trades off a huge part of control overhead for sub-optimality of path quality. In [32], Wang et al. presented a comparison between RPL and AODV for smart meter AMI networks. Their results indicated that RPL outperformed standard AODV with respect to delay and packet delivery ratio. Our earlier work in [34] is the first to provide a comparison of RPL and LOADng for LLN traffic scenarios, and for networks of different scales. The authors in [33] have used a 25-node home automation network scenario and corresponding traffic, to compare LOADng and RPL using the COOJA simulator. It is worthy to note that the authors in [33] also recently pointed out similar drawbacks with the work carried in [16] and provided similar insights on performance metrics for RPL and LOADng to that observed in [34] for networks of small size.

This article, however, summarizes a much more detailed simulation study on the performance of RPL and LOADng on networks of much larger sizes, including a 2442 node smart grid Advanced Metering Infrastructure (AMI) network, where each node is an AMI meter deployed in an urban area, and encompasses several metrics not investigated in the current literature. This necessitated the development of a new simulator capable of handling large scale networks, and we decided to use OMNET++/Castalia platform for this purpose and extend our existing RPL simulator to include LOADng [13]. In Table 1, we provide a summary of the contributions and differences between the literature here described and this work in term of protocols considered, network scale, traffic pattern,

metrics considered, etc. Note that in this study we have not considered cross layer implementations available for WSNs, such as [35,36], as we aimed at providing the answer to a more generalized question, whether to compute and maintain the route beforehand or not.

3. Overview of RPL and LOADng

In this section we briefly overview the operations of both RPL and LOADng protocols. For more details on RPL, the reader should refer to RFC 6550 [8]. For more details on LOADng, the reader is referred to its IETF draft [15] and a performance evaluation publication [16], which we will be commenting on for the purpose of comparison.

3.1. RPL: IPv6 routing protocol for low-power and lossy networks

RPL is an IPv6 distance vector routing protocol [8], where a proactive routing structure is established by the creation of a Destination Oriented Directed Acyclic Graph (DODAG) using an Objective Function (OF) and a set of metrics/constraints. The DODAG (abbreviated as DAG) minimizes the cost of reaching the LLN Border Router (LBR or DAG root) from any node in the network as per the OF in use, which can be defined to minimize a particular metric, such as hop count or the ETX (Expected Transmission count), or any other from the list of metrics as specified in [37]. A DAG structure helps restrict the size of routing tables for limited storage capacity nodes, as only a subset of destination addresses are stored at any node other than the DAG root.

3.1.1. RPL – Forming and maintaining the DAG

A node that is connected to a non-RPL implementing node or a backbone network, can act as a DAG root or LBR and has a rank of 1. The DAG root initiates the DAG formation by advertising information about the DAG using the DAG Information Option (DIO), which carries several information regarding the DAG, including the issuing node's distance from the LBR. Nodes receiving DIO, calculates its distance from LBR based on cost received in DIO and its own cost to reach the issuing node. Nodes chose a node as their parent which provides the lowest cost to reach the LBR. Fig. 1 illustrates the process of broadcasting DIOs to form the DAG and their propagation downward. The solid lines in the figure represent the parent – child relationship in the DODAG, whereas the dotted lines represent other available links. Each node assumes a rank 1 unit greater than its parent's rank.

DIOs are also emitted periodically from each node, triggered by a timer (trickle timer) whose duration increases exponentially (doubled after each time it is fired). On any event that causes a change in the DAG structure (such as new parent selection), this timer is reset to the *Imin* value contained in the DIO. By increasing the duration between two DIOs, the protocol eliminates the need of exchanging neighborhood information prematurely as the network may become stable after a few rounds of information exchange. On one hand, for stable networks where variation in link quality is not significant, the protocol gradually decreases the control plane overhead over time. On the other hand, for a more dynamic topology the protocol helps the network to adapt faster than a protocol implementing only conventional periodic updates. The exponential decay in the frequency of trickle timer, thus increase time duration between periodic updates and perfectly suits the needs of LLNs and, in particular, of large smart meter networks. The number of times DIO timer can be doubled before maintaining a constant rate is denoted here by *MaxDoubling*, so maximum period of the timer, $Imax = Imin * 2^{MaxDoubling}$. If a node receives DIOs from multiple potential parents, it may choose to elect a new parent when a better path is found. RPL, however, ignores messages that will

Table 1
Comparison of existing literature and the contributions of this paper.

Works referenced	Protocols considered	Network scale	Traffic pattern	Simulation platform	PHY + MAC model	Link variation	Metrics considered
[18]	AODV, OSPF	9–81	CBR, MP2P	TOSSIM	CSMA/CA	Not used	Overhead, time to recover
[20]	AODV, DSR, OLSR	25–100	CBR, MP2P	OPNET	802.11b, No packet drop within range	Mobile nodes	Throughput, delay, routing load
[21]	DSDV, AODV, DSR, ZRP	10–50	CBR, P2P	NS-2	802.11, No packet drop within range	Mobile nodes	Throughput, average delay, packets dropped, overhead
[22]	SPF, EXBF, DSDV, TORA, DSR, AODV,	30	P2P-Poisson	MaRS (Maryland Routing Simulator)	Dedicated links, no packet drop	Mobile nodes	Packet delivery ratio, delay, routing load
[32]	RPL, AODV	1000	CBR-MP2P and Poisson-P2MP	NS-2	No Interference/ packet drop	Not used	Average delay, packet delivery ratio
[33]	RPL, LOADng	15, 25, 40	P2MP, MP2P, both periodic and random	COOJA/Contiki OS	802.15.4, CC2420	Not used	Delay, hop distance, overhead, table entries
This work	RPL, LOADng	45–2442	MP2P and P2MP, both periodic and random	Castalia/ OMNET++	802.15.4, CC2420	Replays gathered link-traces	Delay, path cost, overhead, buffer occupancy, RAM consumption, packet length

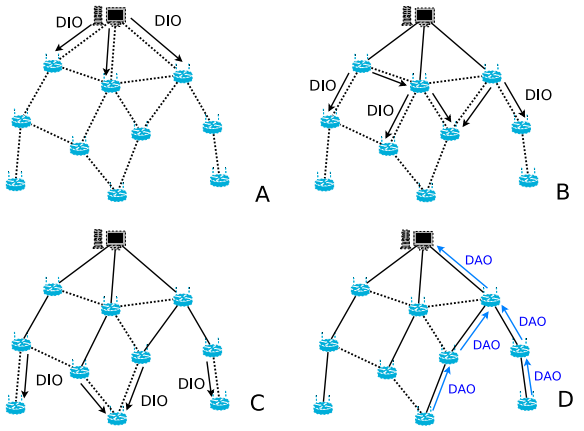

Fig. 1. DIO propagation and DAG formation.

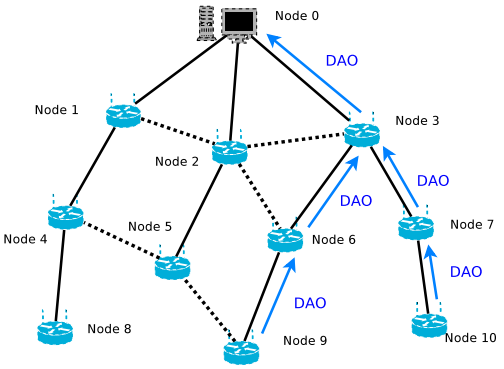

Fig. 2. DAO propagation and route establishment.

incur $x\%$ of change in path cost, where 'x' is a configurable parameter. Hence, a node does not change its parent from the current one, unless the new parent provides a path to the DAG root with a cost less than $(1 - x/100)$ times the current path is found.

A DODAG Information Solicitation (DIS) message may be used by nodes to proactively solicit DAG information (emission of DIO). Nodes who join the DAG also unicast their addresses and reachable prefixes to their parents via Destination Advertisement Option (DAO) messages, which in turn unicast them further up the DAG to advertise destinations reachable through them in support of 'down' traffic. Thus, eventually all DAOs reach the LBR, providing routing information about the whole DAG.

RPL has been standardized to operate on two modes, 'non-storing' mode and 'storing' mode. In storing mode, a node in the LLN is capable of storing routing tables and next hop information for all the nodes in its subtree. Whenever the node forwards a destination prefix available via itself or any of its children nodes (DAO messages), it creates a corresponding routing entry for the particular destination. Therefore 'downward' route is maintained at every node in the DAG. Hence, when a node 'n' advertises a DAO for a specific destination, the receiving nodes store node 'n' as the next hop for that particular destination. The process of DAO propagation is illustrated in Fig. 2, where node 3 acts as next hop to destination nodes 6, 7, 9, 10 to nodes 0, 2, etc.

The non-storing mode is common in large networks, such as a smart grid or city-wide LLN deployment or networks with extremely restricted nodes, where the nodes may not have enough memory to store routing tables for thousands of destinations. In non-storing mode, nodes do not store routes to any destination other than the DAG root. Instead, any node, on receiving a DAO from its children or other nodes in its subtree, forwards it to its parent. Contrary to storing mode, DAOs in non-storing mode are unicasted to the DAG root. Thus, the DAG root or LBR, which is a more capable device than the other LLN nodes, stores all routes to any node in the network. Fig. 2 illustrates an instance of routing a packet in non-storing mode. In the figure, node 3 does not store routes to nodes 6, 7, 9, 10, as seen previously. The DAG root specifies the route through node 3 for destination nodes 6, 7, 9, 10.

The DAG root may also issue a new DAG sequence number periodically to recompute the DAG for major topology changes. When a new sequence number is received, nodes reset their trickle timer, which results in emitting DIOs after the minimum value *Imin*, and issue a DAO to the root after a certain amount of time, which is implementation specific. For local link churn and anomaly, the nodes use a local repair mechanism by poisoning their sub-DAG, and choosing a backup parent.

3.1.2. RPL – Routing through the DAG

MP2P traffic to the LBR, and P2MP traffic from the LBR to nodes follow the parent–child links thus constructed in the DAG. For any traffic destined to the DAG root (LBR), the packet is forwarded to the preferred parent in the DAG, in both storing and non-storing mode. For traffic from DAG root to any other node in LLN, the packet is forwarded to the child which contains the destination prefix in its subtree. In storing mode, a downward route is maintained, and each node 'looks up' the entry for the destination in its subtree and forwards the packet to the next-hop child. This process is explained in Fig. 4.

In non-storing mode, since downward routes are not maintained, the DAG root constructs and inserts into packets a Source Routing Header (SRH), which provides the whole route through the DAG on a hop-by-hop basis. For P2P traffic, where neither source nor destination is the LBR, in storing mode, packets reach up the DAG through node's parents to a common ancestor of both source and destination, and then follow the DAG links down to the destination. In non-storing mode, packets reach the LBR through the parents, and the LBR creates a SRH describing which nodes to traverse downwards the DAG to reach the destination, as illustrated in Fig. 3. Clearly, RPL defines a very sub-optimal route in terms of path length for P2P traffic. However, since the majority of the traffic in LLN is either to or from the DAG root, this DAG based routing can be viewed as a trade-off between path optimality and memory requirements due to storing routing tables in constrained devices.

3.2. LOADng: LLN on-demand ad hoc distance vector routing protocol – next generation

LOADng is an AODV based protocol, adapted to LLN needs. LOADng works by generating Route Requests or RREQs by a source route (originator) while discovering a route to a destination. RREQs are forwarded by the nodes in the network following a controlled flooding or a classical flooding. A node receiving a RREQ packet creates a routing table entry for the generator of the RREQ, and thus by receiving multiple RREQ from the same generator, it learns the best route back to the source. Route Replies (RREPs) are generated upon receipt of a RREQ by the destination. These RREPs are forwarded towards the generator of RREQ via the path learned when the RREQs were received. Every node in the return path of RREP thus learns about the destination node, and installs a forward route to the destination. Also, each RREP is acknowledged by the recipient; if an RREP is not ACKed within RREP_ACK_TIME, a configurable parameter, the RREP is resent to next hop to the originator of the request. Learned routes have an expiration time or R_Valid_Time, after which the nodes have to generate new RREQ if a packet for that destination arrives. If a route is not used within this R_Valid_Time, the route is considered to be expired or obsolete. Fig. 5 illustrates the process of RREQ forwarding. The data follows the reverse path set up by RREP packets, as shown in Fig. 6.

Route maintenance in LOADng is performed on a reactive basis. If a node detects a link down, upon failure of a data packet delivery to the next hop, it may initiate route discovery through the RREQ/RREP cycle again. It may also generate a route error (RERR) packet and forward it back to the source of data traffic, forcing the source to re-initiate the route discovery procedure. Moreover, when a node wishes to send traffic to a destination for which the route is expired, or a next-hop link is broken, it re-initiates the route discovery process.

LOADng and AODV have however some basic differences. Firstly, LOADng accepts number of weak links in a path to be the path cost to adapt to LLN characteristics. When a node receives an RREQ, it increments the path cost by one if the interface has a weak link with the previous hop node, otherwise the cost remains same. If two paths are of equal cost, hop count is used for tie-braking. Thus, the protocol adapts itself to the lossy character of the network. Secondly, no node other than the destination responds w, such as a smart grid or city-wide LLN deploymentith an RREP, even if it has a route to the destination. Furthermore, for RREQ messages, all nodes check the link with previous hop for bi-directionality. If the link is unidirectional, the RREQ is discarded. Lastly, no node maintains a precursor list for the nodes that use itself as destination. Hence, when a link breakage is detected at a node, the node does not send route error (RERR) packets to all such neighbors. When a data packet is received at a node which does not have a route to the destination due to link break-down, it simply sends back an RERR to the source of the data packet, forcing the source to initiate route discovery once more.

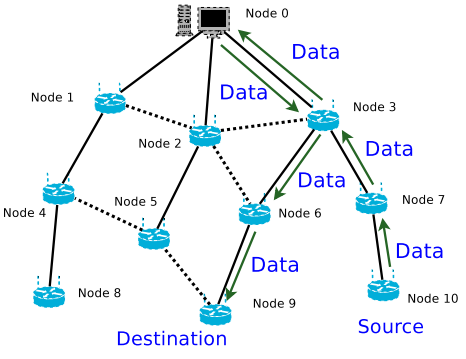

Fig. 3. Data routing in RPL non-storing mode.

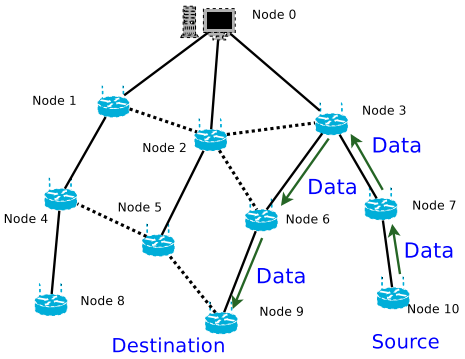

Fig. 4. Data routing in RPL storing mode.

4. Simulation model

A detailed RPL simulator was developed to support the simulation study in [13]. The simulator was developed using OMNET++ [38], a discrete event simulator engine written in C++ and NED. Castalia-2.2 [39] was the wireless sensor network simulator used within OMNET++. The simulator has been extended for this study, to include LOADng as per described in [15]. In the simulator, both RPL and LOADng run over a 802.15.4 MAC/PHY layer on top of a CC2420 radio model for TelosB motes. The simulator was fed with topologies and traffic traces from two real deployment networks (86 nodes outdoor network and a 2442 nodes smart grid AMI network). Random topologies were used only in the scalability study (Section 7).

4.1. Topology and traffic traces

As in [13], real link layer data gathered from the outdoor and smart meter deployments were collected and used to compute the PDR (Packet Delivery Ratio) for each link in the network. A database of time varying link quality data was then created. In the simulation, each link in the topology ‘picks up’ the corresponding link quality model between the same source–destination pair from the database. The link’s PDR in simulation varies according to the gathered traces of the same link, therefore creating a “pseudo-replica” of the real network. The simulation is run for 2 days of simulation for both RPL and LOADng.

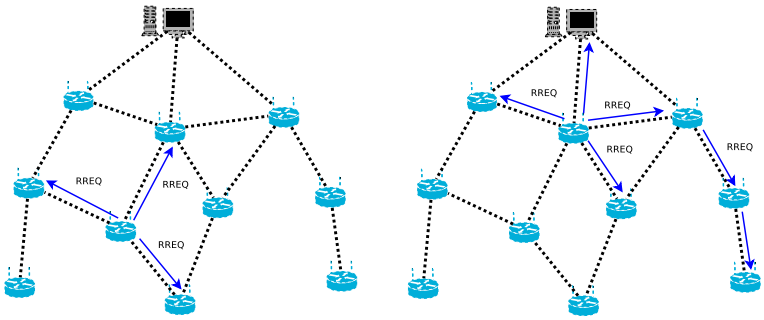

Fig. 5. RREQ forwarding in LOADng.

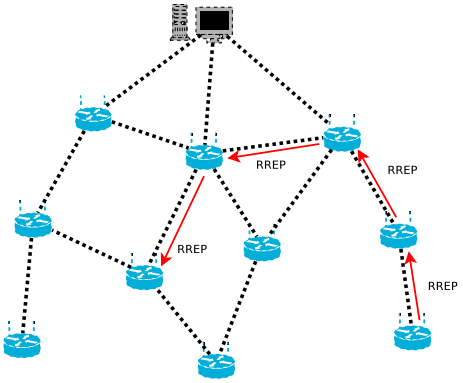

Fig. 6. RREP forwarding in LOADng.

The traffic pattern described below was provided by smart grid AMI meter vendors.

The following types of traffic and patterns were used in the simulation:

- *Polling traffic*: the collection point (LBR) retrieves information (statistic report, status, load curve) from each node in the topology once a day, polling nodes in a round robin fashion.
- *On-demand operation traffic*: reading of index (same traffic pattern as above). Time period between two readings from same node = $F1$. Short volume in both directions (50 bytes). $F1$ is normally kept at a rate of once in 2 h, unless otherwise mentioned.
- *Traffic due to multicast parameter modification (e.g. new tariff)*: 50 bytes sent by the LBR to a node, multicast, with frequency once a day at 2200 s after start of simulation.
- *Traffic due to alarms*: from a meter (node) to the LBR and unsolicited, so unidirectional. 20 bytes of data is sent by each meter to the LBR, once a day.

4.2. Parameter configuration and routing stability

In a network built with low power devices operating under lossy links, it is important that the routing structure is kept as stable as possible while the control overhead is also kept low. Both RPL and LOADng have configurable parameters that affect these. In this section we aim to study the trade-offs in configuring these parameters and, in doing so, chose the values to be used in our simulation study. Simulations were run on the large topology (2442 nodes), and data was gathered for 30 runs with the same network. The 30 simulation runs were enough to provide a small enough 95% confidence interval, wherever applicable, since real topology data was used and multiple global repairs were performed. The CDF plots (e.g. Figs. 8 and 10) were obtained using the cumulative results of all simulation runs. For selected metrics, such as RAM or buffer occupancy, the worst case results were plotted (instead of average). The results for average delay with respect to hop distance, however, had more variability, resulting in larger 95% confidence interval, and thus the confidence interval was plotted in Fig. 16, but omitted elsewhere.

As mentioned in Section 3, with RPL, a change in parent–child relationship may change the node's path cost to the root, and hence reset the trickle timer. This will trigger a large number of control information broadcasts in the sub-DAG that would multiply on their way to the leaves of the network. Hence, it is important to set a threshold to control when it should be inevitable to release new DIOs with reset timers. Recall that an RPL implementation ignores DIOs that incur less than a certain percentage of change in path cost, termed as 'tolerance level' for clarity. As we aim to answer the question: "How much change should be ignored?", the following three cases are considered:

- No change in path cost is ignored.
- The implementation ignores DIOs that result in less than 10% change in path cost.
- The implementation ignores DIOs resulting in less than 20% change in path cost.

Fig. 7 shows the effect of these cases on the stability of parent–child relationships, where the average number of times the nodes' preferred parent changed against the depth of the node in the DAG is plotted. As expected, responding to all changes in the metric to reach the DAG root results in more frequent change of parent in the middle of the DAG, which may bring instability of the routing table or instability of the DAG as a whole. This decision does affect the optimum path quality to the root. In Fig. 8, we plot the Cumulative Distribution Function (CDF) of fractional path stretch, defined as the ratio of

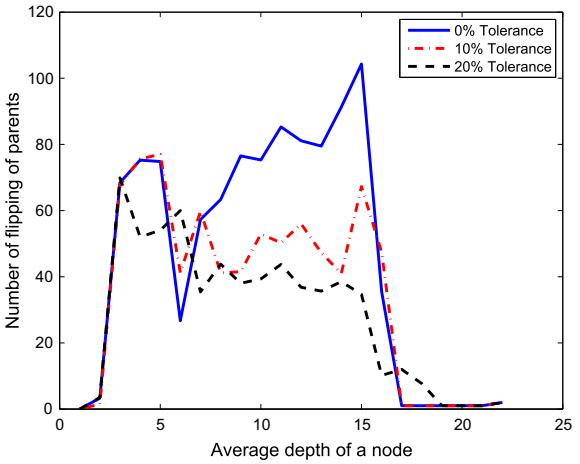

Fig. 7. Number of times parents changed across the DAG.

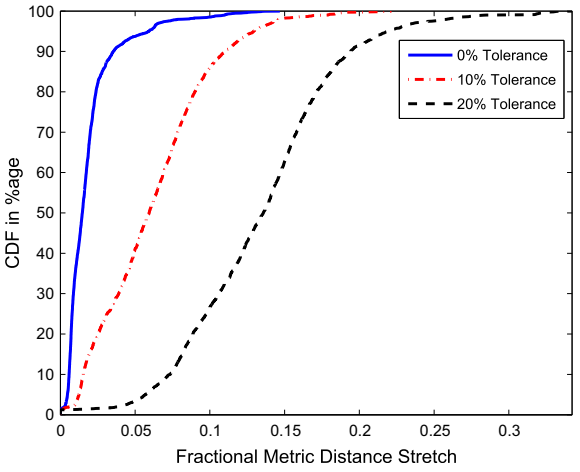

Fig. 8. Fractional path stretch for different tolerance levels.

increase in path cost from a path used by an ideal shortest path algorithm. As observed, for 0% tolerance, 95% of paths used have a fractional path stretch of less than 10%. Similarly, for 20% tolerance level, 95% of paths will have up to 20% fractional stretch. However, the profit gained from the sacrifice in total path cost is increased routing stability and decreased DIO message overhead, as observed in Fig. 9, which shows the number of DIO packets against node depth (X axis) for the three tolerance values. It can be observed that Figs. 7 and 9 are correlated. In a sense, more flipping of parents have contributed to more DIO generation. In the topology considered for this parameter tuning study, the region of DAG corresponding to rank 10–15 appeared to be most unstable, with a spike in number of parent flippings, and consequent DIO emission for 0% tolerance level. We can observe that with a 20% tolerance level, we are able to remove the 'spike' and decrease the amount of DIO transmission. However, flipping of parents or DIO emission cannot be a monotonous function of node rank, for they depend on several factors such as node degree, link variation, and link quality between parent and nodes farther upwards. Also, it must be mentioned that DAO packets contribute more significantly to overall control overhead, a fact also observed in [14], so this variation might seem insignificant while considering total overhead.

It is also important to set an adequate validity time for the routing structure established by the protocols. In RPL, frequent global repair will lead to frequent DAO and DIO message transmissions, increasing the control cost, but keeping the topology up-to-date. However, if the time period between two global repairs (henceforth mentioned as 'DAG repair period') is too large, inconsistencies in the DAG may occur often. To quantify inconsistencies, we measure the time spent by all nodes during which they do not have a parent or path to the DAG root. In Fig. 10, we plot the CDF of this loss of connectivity time for different values of DAG repair period. Thanks to the local repair mechanism, the repair period has less effect on path unavailability. Fig. 11 shows the control overhead against the Node ID for the same repair period values. This plot has been achieved by averaging the control overhead over all runs for each node, and the variation was low enough that a confidence interval plot was omitted.

In LOADng, if the parameter R_Valid_Time (as mentioned in Section 3) is set too high, the network is exposed to high packet drops, whereas a low value results in control cost increase but less packet drops and delay. This trade-off is exhibited in Fig. 12, where the average control overhead is plotted against the number of packet drops.

4.3. RPL and LOADng parameters

Based on our observations in Figs. 10 and 11, a new sequence number is emitted by the DAG root to globally re-construct the DAG every 30 min (DAG repair period) for rest of the study. This value provided a good balance between control overhead, and repair time (protocol convergence). R_valid_time for LOADng is set to the same value as DAG repair period to bring fairness into the comparison, as both these parameters deal with the trade-off between freshness of routes and control overhead. Hence,

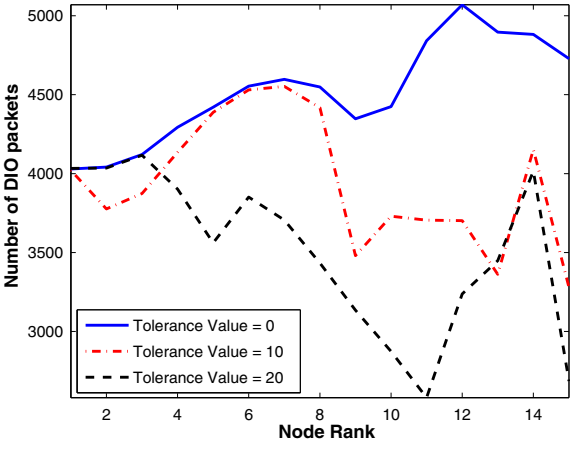

Fig. 9. Control overhead for difference tolerance levels.

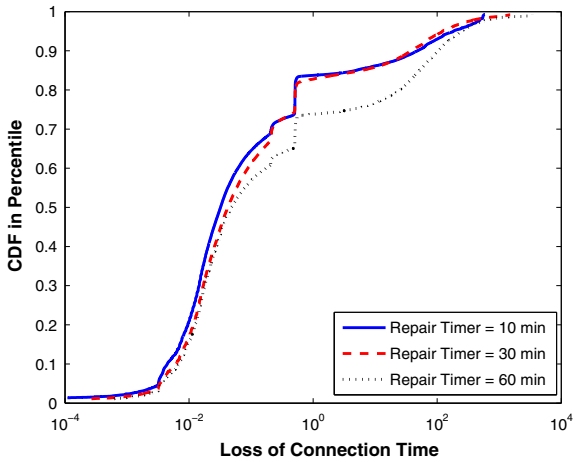

Fig. 10. Unreachability time to DAG root for different DAG repair periods (RPL).

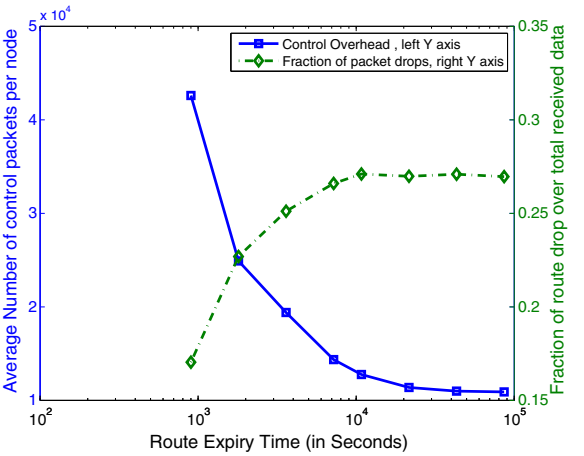

Fig. 12. Control overhead and packet drop against R_Valid_Time (LOADng).

a path in LOADng will be consider invalid 30 min after a route to destination is setup or a successful packet forwarding to the particular destination takes place. Note that a lower value of this parameter may incur more control overhead for LOADng, as mentioned in Appendix A.

In this study, nodes ignore paths advertised via new DIOs if the path cost improvement via the new parent is less than 20%. This value was chosen as it seemed to tone down the amount of parent flipping and resetting DIO trickle timer. The default value of *Imin* is provided to be 8 ms in RFC 6550, however, existing works prefer to set at in the range of 1 s [13,40] or 4 s as in COOJA simulator by Contiki or [14]. In this study, we set $Imin = 1$ s. Every node sends out a DAO message to the DAG root through the preferred parent after a new DAG sequence number is recorded or preferred parent is changed after a time duration proportional to the node's own rank. This is to make sure the reverse path for a DAO-ACK is set when the DAG root receives a DAO. For LOADng, RREP_ACK_TIME is set to 0.2 s, after which nodes transmitting a RREP will check their cache for the respective ACKs. For transmission power settings, please refer to [39]. The simulation details are summarized in following Table 2.

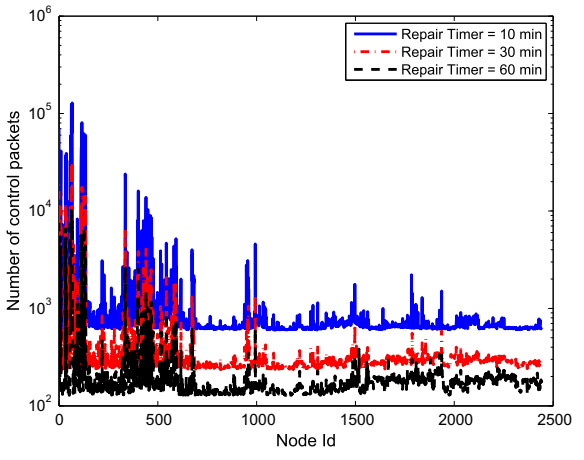

Fig. 11. Control overhead for different DAG repair periods (RPL).

5. Results for smart grid traffic

To evaluate these protocols fairly, we consider both small outdoor deployment and the large smart meter AMI network, and evaluate RPL and LOADng under the same traffic profile described in Section 4.1. In this section, results for end-to-end delay, path quality, control overhead, memory requirements, dependency on application data rate, and packet length are discussed and analyzed. Similar metrics are also studied for a P2P communication scenario in Section 6. It is evident from our study that collecting reading reports more frequently, or adding new application modules will further deteriorate the performance of LOADng, so we consider a lower frequency of reporting operation.

5.1. End-to-end delay

Data in LLNs can be of different types, origins and may require different Quality of Service (QoS) requirements. Some data, such as periodic reports, are delay tolerant up to even few tens of seconds, whereas some are very delay sensitive, for example emergency alarms, fault notification and alert packets. According to RFC5673 [9], in industrial setting, "non-critical closed-loop applications have a latency requirement that can be as low as 100 ms". Clearly, these types of alert packets need a path to the destination as soon as they are generated. This section of the simulation results intends to compare end-to-end delay effectiveness of the protocols under study. For a reactive protocol, it is well established that the delay is larger due to on-demand path computation. This study not only confirms the same, but also points out pathological cases which may further affect the delay bound provided by the protocols. Figs. 13 and 14 show the Cumulative Distribution Function (CDF) of the end-to-end delay for both modes of

Table 2
Simulation settings.

Common parameters	Data rate	240 Kbps
	MAC standard	802.15.4
	Transport protocol	UDP
RPL parameters	Imin	1 s
	MaxDoubling	12
	DAG repair period	30 min
	Path cost tolerance	20%
LOADng parameters	R_Valid_Time	30 min
	RREP_ACK_TIME	0.2 s

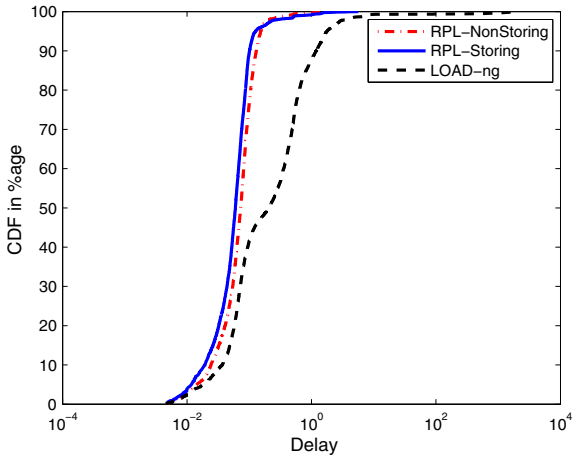

Fig. 13. End-to-end delay for RPL and LOADng.

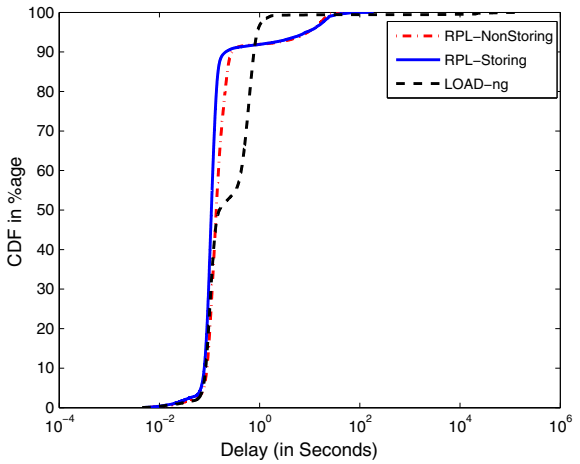

Fig. 14. End-to-end delay for RPL and LOADng – large network.

RPL and LOADng for the small and large deployments respectively, where the *X* axis denotes the delay in seconds (in logarithmic scale) and the *Y* axis demonstrates the corresponding CDF value. For instance, it can be seen in the figure that 90% of the packets have been delivered within 0.1 s or less with RPL non-storing mode, while only 65% of the packets have been delivered within the same timeframe with LOADng.

This behavior stems from the fact that LOADng, and reactive protocols in general, first set up the path before sending the data. Since data communication in LLNs between any two peers is not very frequent (2 h as mentioned in Section 4.1), the established path to a particular destination may be invalid next time data is generated for that particular destination. Also, the next hop might be unreachable, given the fluctuating link conditions in LLNs. Hence, practically every time that peers need to communicate, they need to flood the network with RREQs (preferably in a controlled manner), wait for RREP to be received, and then release the data packet, incurring a larger end-to-end delay. The results for LOADng also show that some data packets may suffer a delay of a few tens of seconds to reach the destination. This is due to the loss of RREQ, RREP and/or RERR packets. Most of these data packets, which take few tens of seconds or more to get delivered, result from multicast traffic that clogs the network with control traffic, as we will observe in Section 5.3.

We then consider only cases where the data packet has been lost, or any RREP/RREQ has been lost due to collision, link conditions, etc. Fig. 15 zooms into such pathological cases. Again, we plot the CDF of delay versus the delay value (in seconds). Since RPL implements backup parents and backup routes (at the DAG root or collection point), the alternate route can be chosen rather quickly upon any delivery failure of a data packet from the lower layer. For LOADng, however, a new route discovery must be initiated. Moreover, on the loss of RREP packets, the data packet will only be transmitted when another RREP for the same destination arrives at the source through another path.

In Fig. 16, we plot the average delay against hop distance for LOADng and RPL non-storing mode, along with their 95% confidence interval. The delay statistics for each hop distance *x* is gathered from the time each packet takes while traversing a distance of *x* hops. For each hop distance *x*, we calculate the average as well as the confidence interval from the delay statistics generated by all such packets. We observe that not only RPL has a lower average delay for any given hop distance between two peers, but also the variance is much lower, yielding a more reliable delay bound. Since both modes of RPL showed similar delay bound in Fig. 13, for the sake of clarity, this plot does not show RPL storing mode.

The unreliability in delay bound for LOADng, statistically shown as a larger 95% confidence interval is also attributed to the reactive nature of the protocol. Since new communication requires a route request broadcast, such communications may result in higher delay due to broadcasting, RREP loss, etc. On the other hand, some communications from the same originator to destination happen right away, as the path is established. Note also that communications in reverse direction between the same peers (as happens in query-based communication) do not require route discovery, and may take place as instantly as the application layer packet reaches the network layer. For these reasons, the end-to-end delay for the same pair

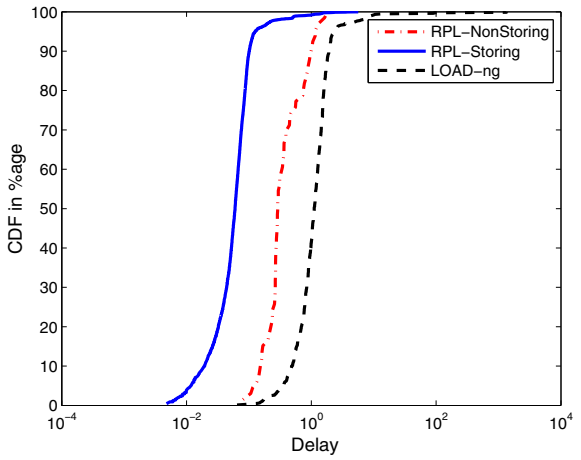

Fig. 15. End-to-end delay for RPL and LOADng, zoomed in.

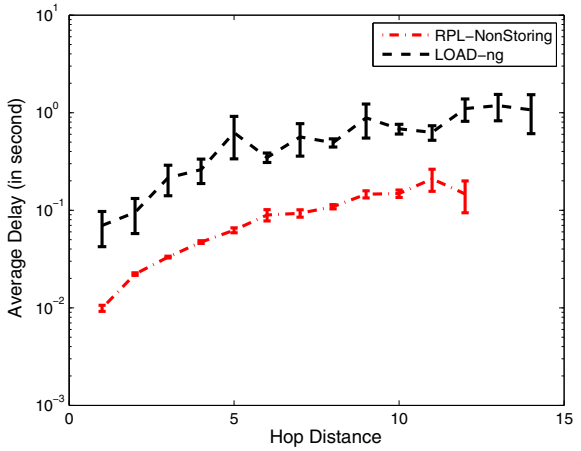

Fig. 16. End-to-end delay versus hop distance for RPL and LOADng.

of peers, or over the same number of hops, may vary widely over simulation time.

5.2. Path quality

Reactive and proactive protocols do not differ much on the path quality. There are several implementations of reactive protocols where a destination node may or may not wait to release a RREP after receiving a RREQ for itself. In LOADng, even if multiple RREPs are received, the data packet is released when the first RREP reaches the originator, which in most of the cases does not result in the best path selection between two nodes. On the other hand, in RPL, P2P paths may often be very non-optimal. RPL may also not select the best path, as it ignores paths that do not yield a certain percentage (20% as mentioned in Section 4.3) of improvement of path cost when it receives a DIO. However, the majority of the data traffic in LLNs flow between the LBR and the meters, the path quality does not reflect the non-optimum path length for RPL protocol in peer-to-peer scenarios. Hop distance or similar path quality metrics are not observed as the most important in LLN deployments. For the sake of fair comparison, in Fig. 17, we compare the CDF of hop count and Fig. 18 shows the CDF of the ETX path cost for all packets in the network for both modes of RPL and LOADng. The CDF is calculated over thousands of packets to and from the collection point, and we can observe all protocols exhibit similar performance in terms of hop count. Hence we can conclude that "path-quality-wise", the three protocols exhibit similar behavior even for a large topology for the specified traffic pattern. For fair comparison, such a P2P type application is studied in Section 6.

5.3. Control overhead and ability to support P2MP or MP2P traffic

Since nodes in LLN have limited capacity in terms of power storage and/or scavenging, as well as face scarce bandwidth, control overhead is one of the most important considerations in choosing a routing protocol. It is well known that in networks with light traffic load and a small topology, a reactive protocol may be better suited than proactive protocols. However, in LLNs, as we investigate, the deployment size and traffic pattern suits RPL better than LOADng. Specific types of applications running over an LLN often require a node to send the same data to multiple recipients even several hops away, therefore requiring an efficient dissemination method or multicast traffic (P2MP) support provided by the routing protocol. P2MP traffic includes, but is not limited to:

- Management information disseminated to nodes in a certain region of deployment, or a certain part of the manufacturing pipeline in an industrial automation setting.
- New tariff notification to a group of users in a smart grid deployment.
- "Large number of unicast flows from the sensing nodes or sensing clusters towards a LBR, or highly directed multicast or anycast flows from the nodes towards multiple LBRs", as specified in RFC5548 [10] for U-LLN deployments.

Though AODV can support multicast IP address in RREQ destination address, no such mechanism is provided by AODV. LOADng does not have provision for supporting multiple route discovery either. A naive solution would be to create a copy of the same message to send, along with creating separate route request (RREQ) messages for each destination and broadcast them. This broadcast event creates a huge control overhead and the protocol does not scale well with the network size. Hence, AODV or reactive protocols in general may become unsuitable to be deployed in a large scale U-LLN where P2MP traffic needs to be supported, even if for 1–2 times a day. This is the reason we included multicast traffic once a day in simulation, and the following simulation results indeed confirm our observation.

Fig. 19 shows the control overhead cost for both LOADng and RPL non-storing mode in the smaller topology,

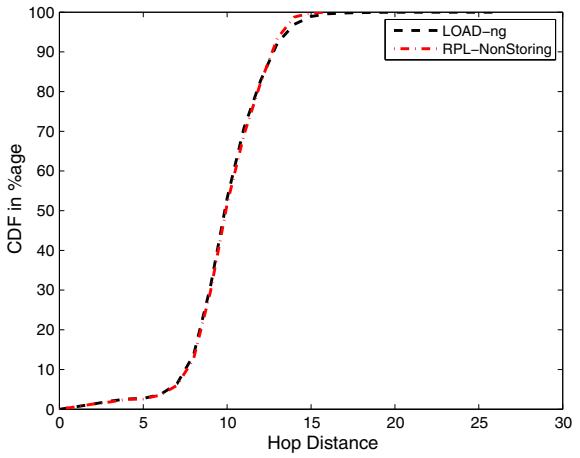

Fig. 17. CDF of the end-to-end hop distance, in a large network.

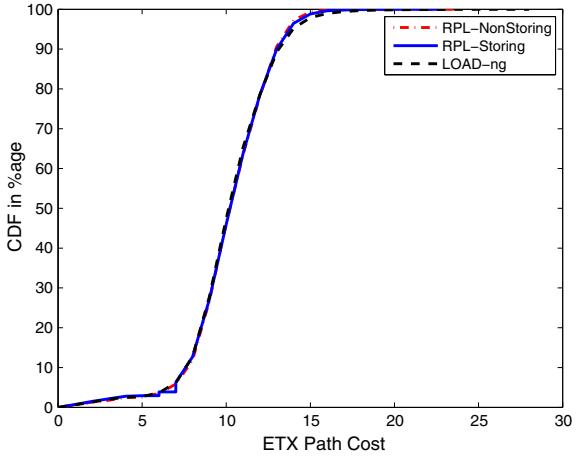

Fig. 18. CDF of the total ETX path cost in a large network.

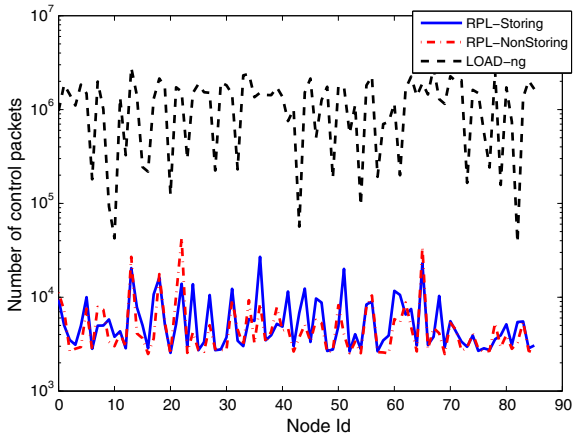

Fig. 19. Control overhead for RPL and LOADng.

whereas in Fig. 20, the same is demonstrated for the larger smart grid deployment. We observe that LOADng yields a control volume a magnitude larger than that of RPL. The

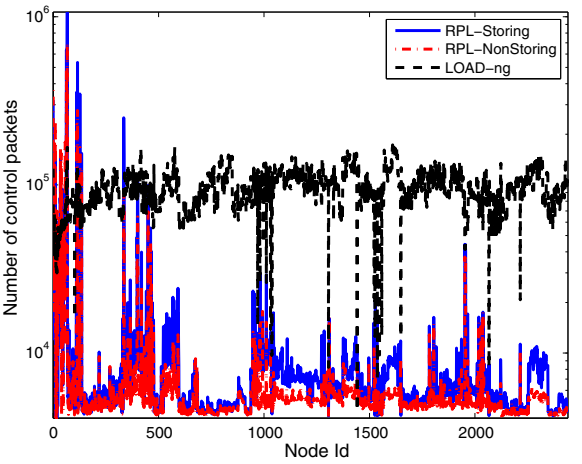

Fig. 20. Control overhead for each node in a large network.

control overhead in RPL depends mainly on two types of advertisements, (i) DIO and (ii) DAO (including DAO-ACK). While the DIO control overhead exhibits an exponential delay over time, the DAO control overhead for a given node depends on the number of nodes in its sub-tree, and thus it inversely varies with the rank of the node. In the figure, it is worth observing that, for RPL, some nodes have higher amount of control message transmission; these nodes (e.g., 22, 66 in the smaller network, and lower ID nodes in the large deployment) are the nodes directly connected to the LBR, and hence are responsible for larger amount of DAO and/or DAO-ACK propagation. Hence, overall, for most nodes (other than a few tens among thousands) LOADng exhibits an order of magnitude more control packet transmissions than either mode of RPL. For LOADng, RREQ and RREP packets travel through nodes closer to the collection point; moreover communication with each destination also incurs RREQ packet forwarding for each node in the network.

Fig. 21 shows the maximum control packets processed by any node over time for RPL non-storing mode and LOADng, with the objective of studying the computational

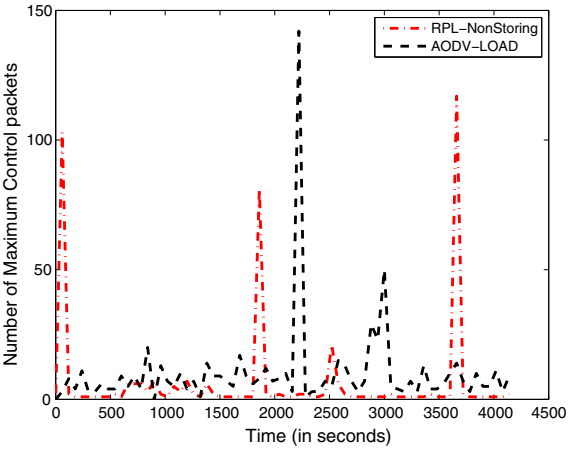

Fig. 21. Maximum control packets processing – RPL versus LOADng.

and processing load exerted on the nodes in the network. The 2 days of simulation time have been broken down to small windows of 1 min each, and we computed the number of control packets processed/transmitted by each node within that window. The maximum over all nodes for the window is plotted against time for both RPL and LOADng. The figure shows that RPL exhibits a peak of control message processing when a new sequence number is emitted for global re-construction of the DAG, such as at 0–60, 1800–1860 and 3600–3660 s. Recall that this implementation considers a DAG repair period of 30 min. LOADng exhibits a peak of maximum control message processing when multicast traffic is generated by the LBR. In subsequent intervals, as the nodes finish broadcasting RREQs and forwarding RREPs, the control volume gradually decreases. We skipped plotting RPL storing mode in this figure and in the next one for the sake of clarity, as we have observed in Fig. 19 that both storing and non-storing mode create almost similar amount of control overhead for this network. Similar behavior is also observed for the large deployment (omitted for brevity).

Though RPL periodically reaches a peak of maximum control overhead processing, this peak is only applicable to nodes directly connected to the LBR, therefore processing a large number of DAOs/DAO-ACKs. For LOADng, however, the control overhead does not depend much on the distance from the LBR. From a network congestion point of view, it is more disadvantageous to have all nodes transmitting large number of control packets, than only a few nodes at lower rank doing so. To look into this scenario more carefully, Fig. 22 shows the total number of control packets processed by all nodes in each 1 min window for the large network. The plot shows the overall higher volume of control traffic when multicast traffic appears. For LOADng, when a multicast traffic appears, every node takes part in broadcasting the RREQ packets, and the number of broadcasts for each node reaches as many as the number of recipients in the multicast. Hence, the total number of control packets in the network becomes significantly larger than that of RPL. The figures also show that even when global repair is initiated in RPL, the total control traffic volume in the network is still lower than that of LOADng. LOADng's high control packet flooding may therefore lead to stall of data communication, and/or network crash.

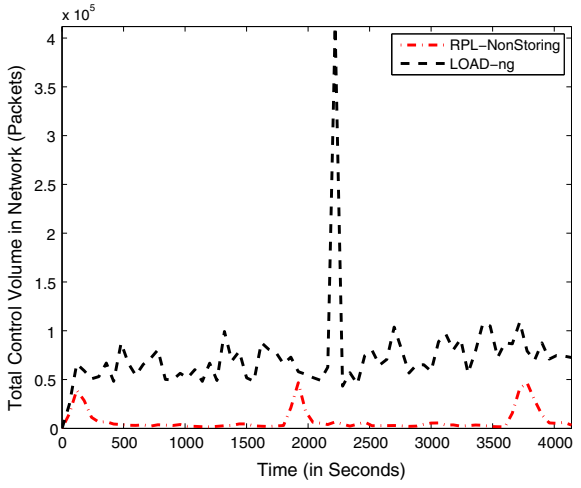

Fig. 22. Total control packets processed in network against time in large network.

5.4. Memory requirements

Memory constrained devices with low power are the pivotal components of LLNs, therefore it is important to study the memory requirements of each protocol. With LOADng, nodes build their routing table based on RREQs and RREP packets received, therefore if any node participates in multiple active flows in the network, the node needs to store next hop and validity information for each source and destination node. Thus, depending on the user traffic, some nodes tend to increase their routing table size proportional to number of flows passing through themselves. However, characteristics of LLNs never guarantee enough storage space in any node for storing routing tables. Destination oriented flooding in LLN, tends to worsen this situation. On the other hand, RPL non-storing mode does require nodes to store any routing information. In this section, we will study the maximum memory requirement for the protocols such that no packet or routing entry is dropped. If a single node in the network in a single scenario needs '*M*' KB of RAM, then all nodes for that network need to be equipped with '*M*' KB of RAM.

Fig. 23 shows the maximum RAM that has been occupied (in Bytes) against the Node ID for RPL non-storing mode and LOADng for the smaller deployment. It should be noted that this RAM occupancy is only a result of storing routing tables (as in LOADng), parent tables (as in RPL), queued packets for transmission, any kind of data structures, etc., and does not include the amount of RAM space the protocol code itself would occupy for implementation. Note also that the collection point has been excluded from this analysis, as the collection point is supposed to be a computationally resourceful device, irrespective of the routing protocol. The maximum RAM requirement for RPL has been found to be 2 KB, and that for LOADng is 6 KB. However, it should be pointed out that LOADng's code would occupy less memory than RPL's; however in

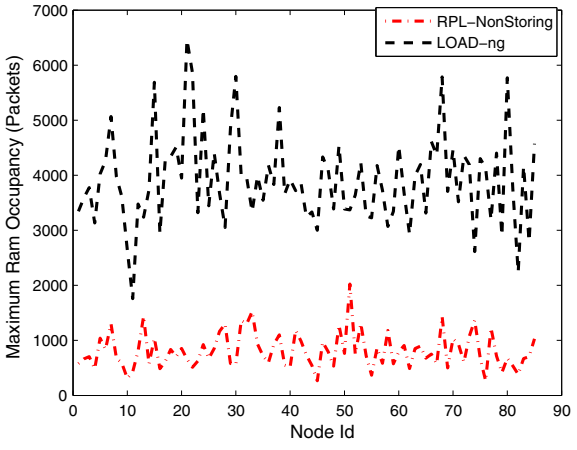

Fig. 23. Maximum RAM occupancy – RPL non-storing versus LOADng.

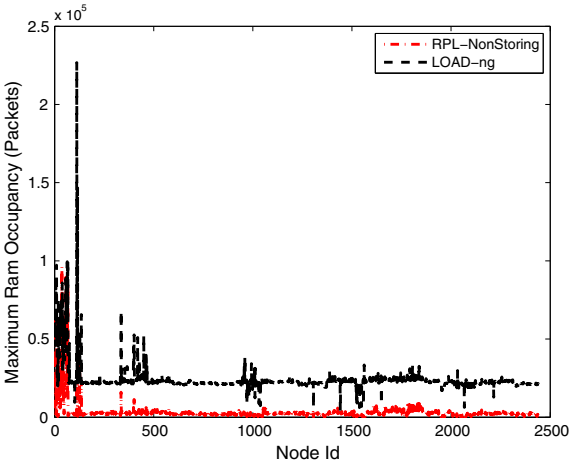

Fig. 24. Maximum RAM occupancy in bytes for each node in a large network.

a large network that advantage may well be lost due to high buffer and routing table requirements for LOADng. Fig. 24 shows the maximum RAM occupancy (in Bytes) for LOADng and RPL non-storing mode for the large smart grid deployment.

5.5. Dependency on application data rate

LLNs are a nascent area, also referred to as IP smart objects networks or the "Internet of Things," constantly growing in importance. Thus, an LLN that is currently provisioned to be used for data gathering purpose only, may include additional application modules in the future. Also, smart grid deployments may need to implement new modules of management traffic from the base stations to AMI meters in addition to what is envisioned at present. Reactive protocols, however, discover route on an on-demand basis, and it is well known that the control overhead of a reactive protocol depends on the application data rate. Hence, if LOADng is configured to meet an overhead criteria based on the current application, future modifications to the application rate may well increase or at least change the total control overhead, invalidating the designed configuration. To illustrate this, Fig. 25 shows the control overhead versus node ID for different polling intervals for LOADng: 30 min, 1 h, 2 h, 6 h and 12 h. One can see that the control overhead increases as the LBR polls the nodes more often.

Fig. 26 shoes the total control packets volume for RPL non-storing mode of operation for different polling intervals. The LBR is set to probe each meter in a round robin fashion with 1 h, 2 h, 6 h, 12 h and 24 h intervals. Clearly, being a proactive protocol, RPL's performance with respect to control overhead does not depend on when and in which fashion application data is being generated, shown by the overlapping results.

6. P2P communication

Clearly, the performance of LOADng depends on the application characteristics and in what manner nodes

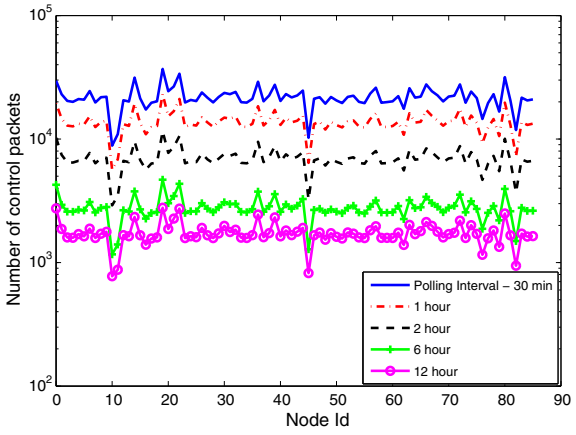

Fig. 25. Control overhead versus node ID for different application rates, LOADng.

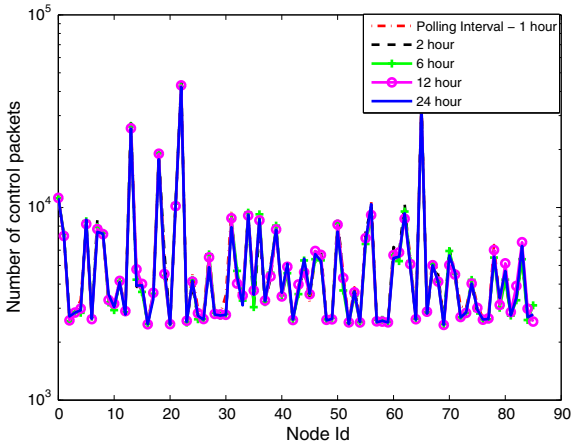

Fig. 26. Control overhead versus node ID for different application rates, RPL non-storing.

communicate with each other. If every node talks to a single meter within a short time duration, a single RREQ broadcast is sufficient for many (however, not all) nodes in the network to gather route information about the destination. At the same time, a single RREQ broadcast provides all nodes in the network with the route to the originator. Thus, while some applications may consume little control overhead, others may create a broadcast storm.

P2P traffic in this section is simulated as follows: every meter communicates with another meter in the network other than the LBR. Each node generates a packet every 60 min, and communicates with a different node in each interval. Therefore, no two nodes communicate with the same node in any given 60 min interval.

6.1. Path quality

Fig. 27 shows the CDF of the path length (in number of hops) for RPL storing mode, RPL non-storing mode and

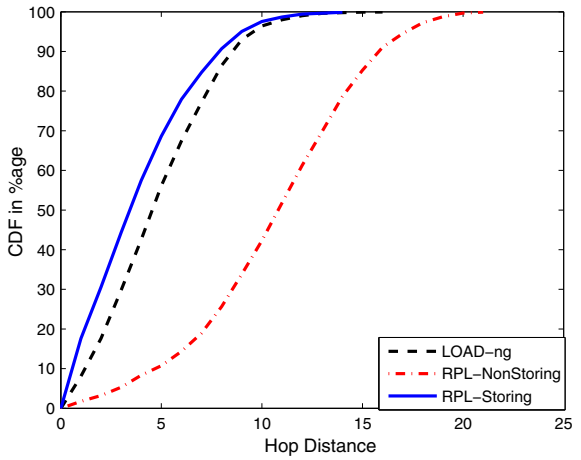

Fig. 27. End-to-end hop distance for RPL and LOADng; P2P application.

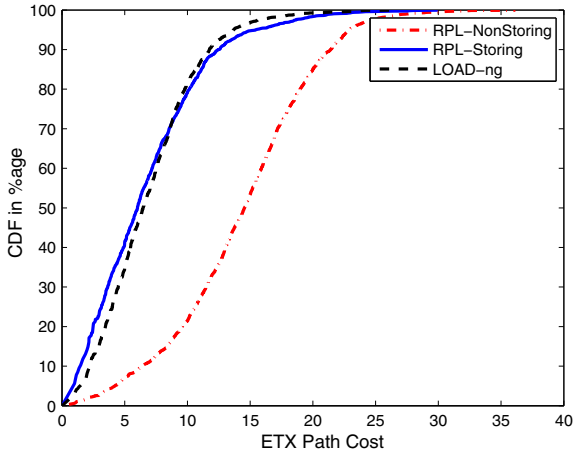

Fig. 28. ETX path cost for RPL and LOADng; P2P application.

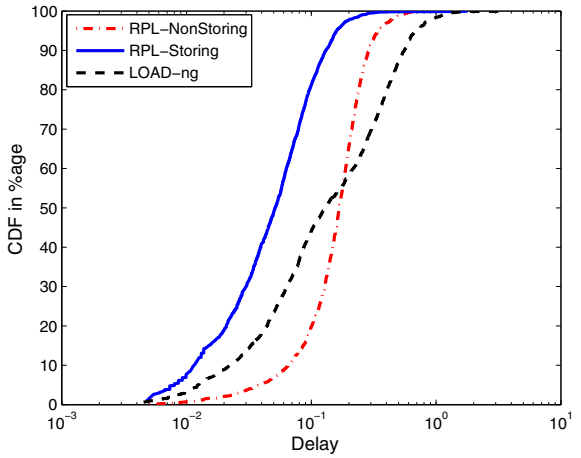

Fig. 29. End-to-end delay comparison: P2P application.

LOADng, with the above P2P traffic profile. It is observed that RPL non-storing mode has a large path length for true P2P application, as all communication is directed via the LBR. LOADng and RPL storing mode result in very close path lengths, even if storing mode does not yield optimum path quality in terms of path length.

Fig. 28 shows the CDF of the ETX path cost (plotted in *Y* axis) against the ETX path cost value (plotted in *X* axis) for both modes of RPL and LOADng. As before, the ETX path cost for RPL storing mode and LOADng are very similar, but LOADng sometimes produces a path with lower cost. RPL non-storing mode is not optimized for P2P applications and therefore provides path with much larger ETX path costs than both storing mode and LOADng.

6.2. End-to-end delay

Fig. 29 shows the CDF of delay in seconds for both modes of RPL and for LOADng for the P2P application scenario. Interestingly, even though RPL non-storing mode incurs a high cost to reach the destination in terms of hop count and ETX path cost, for many paths it provided lower delay than LOADng. This phenomenon can be explained by the reactive protocol's reaction time for a non-existent routing table entry or a route that is no longer valid. But once the route is established, and the path is valid, LOADng may provide lower delay than RPL non-storing mode, as observed in the figure, for up to 55% of the received packets. RPL storing mode, however, outperforms both, providing overall lower delay.

6.3. Memory requirements

We also compare the maximum RAM requirement for RPL non-storing mode and LOADng, shown in Fig. 30. RAM occupancy in RPL storing mode is topology dependent and, for the topology in use, it is very similar to that of LOADng. It is observed that RPL has a maximum RAM occupancy of around 3 KB, and LOADng has a maximum RAM

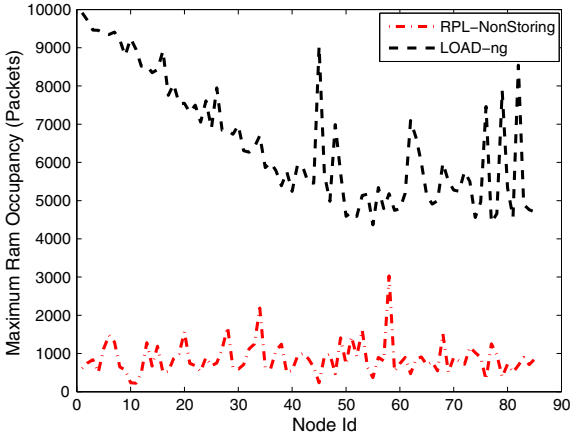

Fig. 30. Maximum RAM occupancy for RPL non-storing and LOADng; P2P application.

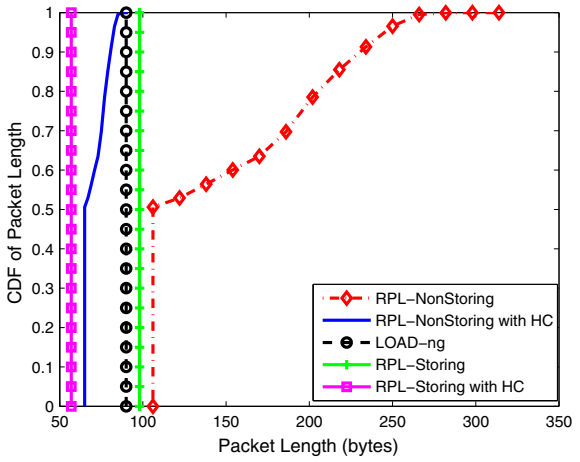

Fig. 31. Packet length for LOADng, RPL storing and non-storing modes.

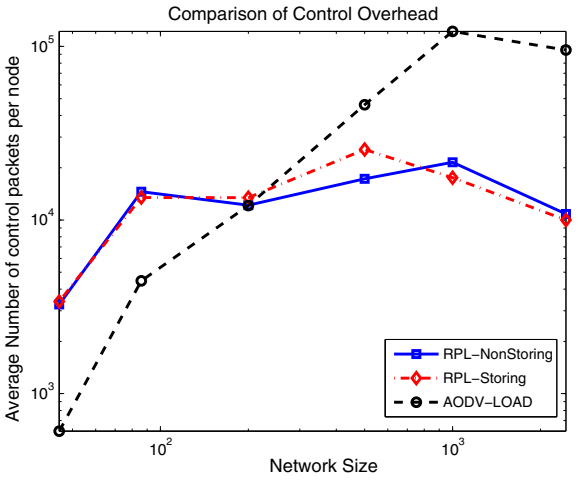

Fig. 32. Average control overhead per node with network size.

occupancy of around 10 KB. However, these results depend on traffic pattern, frequency, etc. As before the LBR or collection point of the network has been excluded form the memory analysis, as it should be a computationally resourceful device. The memory footprint is calculated only for resource constrained LLN nodes. Note that, as mentioned in Section 5.4, the amount of RAM space that would be occupied by the code for implementation of each protocol is not included in this simulation, and it would be smaller for LOADng.

6.4. Packet length

RPL non-storing mode does not store any route information. While this approach makes RPL viable in very low memory equipped devices, it has its own disadvantages as the protocol has to perform source routing via the LBR. We have already seen how it affects path length in P2P communication. However, another disadvantage may arise from the source routing header (SRH): the farther away a node is from the LBR, the larger the data packet SRH needs to be. Fig. 31 shows the CDF of data packet lengths for all packets that travel the network. The *X* axis corresponds to data packet length in bytes, and the *Y* axis indicates the corresponding CDF. The application layer data is 50 bytes in size, as indicated in Section 4.1, under traffic pattern. When header compression is not performed, all the packets also bear a 40 bytes IPv6 header. From the figure we can clearly see that RPL non-storing mode has a much larger packet length than LOADng. This can be problematic when RPL operates over low MTU links, such as IEEE 802.15.4 links with an MTU of 127 bytes. RPL storing mode normally yields packet lengths comparable to (though slightly larger than) LOADng. However, when header compression is performed with RPL, in accordance to the compression format for IPv6 datagrams over IEEE 802.15.4-based networks in RFC 6282 [41], all packets are well under the MTU requirement, and thus source routing is possible for LLNs operating with low MTU links.

It should be noted that this result has been obtained from a set of P2P communications over the network. However, the variable packet length only arises due to SRH in non-storing mode from LBR downward the DAG to other nodes. Hence, this result can be obtained if the communication includes any P2MP traffic.

7. Scaling properties

It is hard to estimate the general behavior of a protocol with respect to network size when only two topologies have been simulated. To hint at the results not being completely topology dependent, it is necessary to study a varied range of network sizes. Hence, in this section we study LOADng's and RPL's performance in topologies of different sizes: 45 nodes, 86 nodes, 200 nodes, 500 nodes, 1000 nodes and 2442 nodes.

Network scale has most severe impact on two metrics: control overhead and resource utilization. Fig. 32 shows the average control overhead per node for all three protocols, with varying topology size. The network size is plotted in the *X* axis in logarithmic scale, whereas the average control packet is plotted in the *Y* axis, also in logarithmic scale. One can observe that the average control packet overhead increases more sharply for LOADng as the size of the topology grows. Note that although some os these topologies were created randomly, all link characteristics were gathered from the database created from a real deployment. Some networks have links that are more stable than others and this explains why the largest network is found to be most stable one, with the least link variation, and hence we observe the decrease in average control overhead per node for the largest network to be less than that of the network with 1000 nodes. Fig. 33 shows the total control overhead of all nodes in the network for these three protocols with varying topology size.

As expected, LOADng is more efficient for smaller networks with light traffic load. The average control traffic

volume for these protocols is very similar when the network size is around 200 nodes, for the case where periodic data polling occurs for each node every 6 h. This can also be observed with more frequent statistics reporting for a smaller topology.

This study also recognizes the scarcity of resources while deploying an LLN, and hence, it is impossible to overlook the resource utilization while discussing the scalability of the protocols. With increased number of nodes, LOADng and RPL storing mode will need to store more entries. However, the difference between the two is that for LOADng all nodes in the network store the route to an RREQ advertising node, while the routing table size in RPL storing mode decreases with distance from the collection point. On the other hand, RPL non-storing mode does not store any route in any node other than the DAG root. The DAG root or collection point is a much more capable device and therefore it is not resource constrained, thus we limit the calculation of maximum RAM or packet buffer occupancy to all LLN nodes other than the collection point. Hence, if we consider the maximum RAM utilization, RPL non-storing mode is the least resource consuming. Fig. 34 shows the maximum RAM occupancy in bytes in the *Y* axis, against the network scale in a logarithmic *X* axis.

It might be surprising that despite the fact that RPL non-storing mode does not store route information, the maximum RAM requirement does increase with the network size. The reason for this is that a large portion of RAM is utilized to buffer packets, in either modes of RPL, during each global repair, where DAO packets are propagated upward to the DAG root, and DAO-ACK packets flow downward to the nodes. This leads to a congestion near the collection point: nodes with lower rank. This explains the increasing trend for maximum RAM utilization as observed in Fig. 34. Note that the two RPL parameters, DAO_ACK_TIME and DAO_LATENCY, can be tweaked to achieve less congestion and less buffer requirement. The first parameter describes how long a node should wait to emit a new DAO for each global repair and/or parent change, and the second moderates how long to wait for an acknowledgment of emitted DAO packet before a new one is sent out.

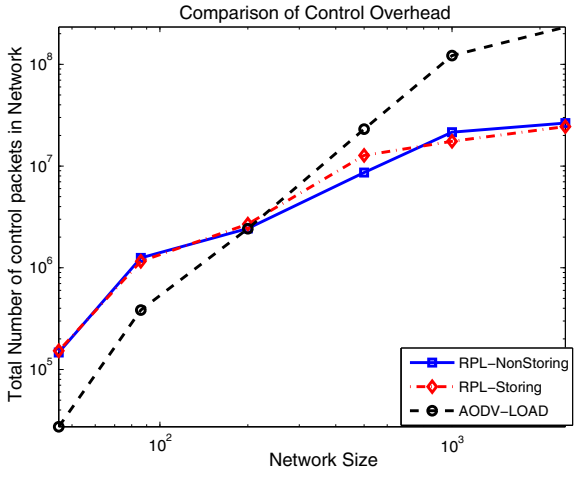

Fig. 33. Total control packets in network with network size.

Motivated by the scaling property study, we set out to improve DAO packet emissions in RPL. Next in Section 8, we propose a combination of distributed and centralized algorithms to control DAO packet emissions by adaptively tweaking the parameter DAO_LATENCY in each node to greatly limit DAO congestion and thus restrict the requirement of larger packet buffers. A preliminary study of our proposed algorithms was presented in [17] and is here extended and improved by considering a combination of the previously proposed algorithms.

8. DelayDAO controller – A combined algorithm proposal to improve RPL's performance

As we have shown in [17], in large networks, when the LBR performs global repair to gather current routes, high congestion and excessive buffer occupancy is a by-product of the default RPL operation. This can be illustrated by a simple random topology of 1000 nodes with disc model for radio propagation. We assume the DAG constructed by RPL is the Breadth First Search (BFS) tree for the topology, and the value of DelayDAO Timer is T_{DD}. In general, the number of DAO packets received by a node at the *i*th T_{DD} interval will be the same as the number of nodes in the *i*th level of it's sub-DAG. In Fig. 35, we plot the number of times a node's radio is busy transmitting or receiving DAO messages against rank and time interval. Note also that this analysis does not consider the DAO-ACK or acknowledgement packets that traverse down the DAG, while some DAOs are being forwarded up the DAG. Clearly, the results presented in this section are optimistic and provide at best a lower bound.

This high buffer requirement stems from the fact that each node needs to store the DAO packets from its sub-DAG for possible aggregation of DAO routes. However, while implementing non-storing mode, route aggregation is not performed. Hence, a node should immediately forward a DAO from its sub-DAG to its parent, contrary to what is pointed out in RPL's RFC [8]. Secondly, in a particular level in the DAG, the generation time for DAO packets should be distributed within a time duration to prevent propagation of a high number of DAO packets at the same time. The time duration should increase exponentially with rank as we have observed that the number of nodes tend to increase in this manner. This is illustrated in Algorithm 1, which determines the DelayDAO Timer duration. It assumes that each node generates a random number amongst an interval that increases exponentially with the rank of the node. The base of the exponent, or the parameter '*Base*,' is used for coarse tuning of the timer value, whereas the linear parameter '*K*' is used for fine tuning. In the next sections, we will present mechanisms that will be used to estimate values of '*K*' and '*Base*'. Each node initialize the value of these parameters from the value provided by the LBR in DIO packets, and updates them based on the algorithms described in this section.

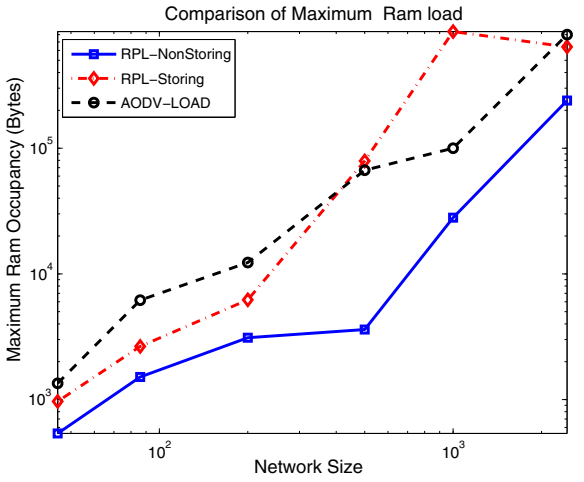

Fig. 34. Maximum RAM occupancy in bytes against network size.

Algorithm 1. Generation of DelayDAO timer duration

```
seed = ⟨EUI_64⟩/MAC address or node ID
T_DD ← random(K * Base^(Rank−1), K * Base^Rank);
DAOStartTime := CURRENT_TIME;
Arm DelayDAO Timer with value = T_DD
Issue a DAO when DelayDAO Timer fires.
```

In [17], we proposed two methods to control the value of this timer. We observed that while a distributed algorithm provides less RAM consumption and lower DAO round trip time, a centralized algorithm helps the LBR gather the topology information faster. In this paper, we propose a joint mechanism, so that we can retain the benefit from both approaches in terms of memory consumption and in faster building the whole DAG. We propose that border routers or collection points run a centralized algorithm to estimate the parameters '*K*' and '*Base*' from which DelayDAO timer should be computed at each node. The LBR stays updated on the current network topology by receiving DAO packets from nodes in the network. The parameters are broadcast before every global repair of the network, with the help of DIO messages with a new DAG sequence number from the LBR. Each node, on receiving the estimated parameters, and based on the round trip time of DAO packets and DAO acknowledgements, makes fine adjustment to these parameters, to further reduce the DAO congestion, thus limiting the buffer requirement.

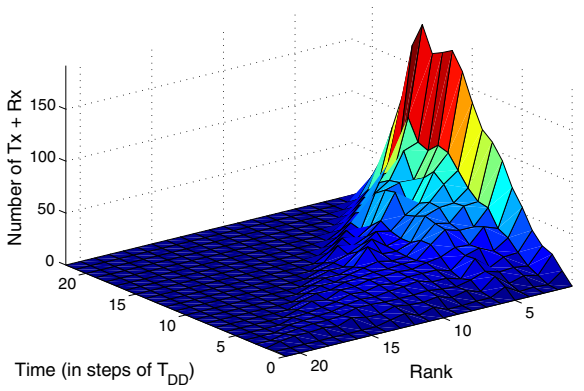

Fig. 35. Total DAO Tx + Rx versus time and rank for the 1000-node grid topology.

8.1. Routine followed at LBR or collection point

Since all nodes send their DAO message to the DAG root, the root has an overall view of the whole network. Hence it would be advantageous to outsource the computations related to the network to the DAG root. In this section, we will present an algorithm that computes *Base* and *K* upon receiving DAOs from the network. After computation, these values are distributed in the network during the next global repair or increased DTSN via a new DIO packet. The values of *Base* and *K* can be added as objects in the DIO packet.

We define the node rank set *L*, which contains for each rank *R*, $(1 \leqslant R \leqslant H+1)$, the nodes that are at rank *R*. Also, the DAG root maintains a parent list *P* that contains the parent for each node *n* in *N*. We define the function Find_Rank to intake a node ID, and returns the rank of the node, as described in Algorithm 2. As illustrated in the flowchart in Fig. 36, upon receiving each DAO, the data structures are updated as in Algorithm 3. Before each global repair or DTSN increase, the DAG root estimates the parameters to determine the value of DelayDAO Timer for all nodes in the network, as shown in Algorithm 4.

Algorithm 2. Rank Finder

```
int Find_Rank (node n)
{  int Rank = 0;
   P := Parent (node n);
if P == DAG_Root then
     return (1);
end if
   Rank := Rank + Find_Rank (node P);
   return (Rank);};
```

Algorithm 3. Construction of Node Rank Set

```
n ← Source of DAO message;
R_n ← Find_Rank (n);
L[R_n] ← L[R_n] \ {n};
Update Route_Table with new parent for n;
R̂_n ← Find_Rank (n);
L[R̂_n] ← L[R̂_n] ∪ {n};
```

Algorithm 4. Centralized Parameter Calculator

$W \leftarrow \max(\{L[i]\}),\ 1 \leqslant i \leqslant H;$

$R_W \leftarrow R$ s.t. $\{L[R] = \max(\{L[i]\}),\ 1 \leqslant i \leqslant H+1;\}$

$Base \leftarrow \exp\left(\frac{\ln W}{R_W}\right);$

$K \leftarrow \max\left(2 * T_{Tx} \times \frac{i * \ln i}{Base^i}\right),\ 1 \leqslant i \leqslant H+1;$

8.2. Routine followed at nodes other than LBR

Each node, on their first global repair, sets the parameters with the values of the received DIO packets. As shown in the flowchart in Fig. 37, nodes use adaptive filtering to correctly estimate these parameters as they continue to receive acknowledgements from the LBR. On reception of DAO-ACKs, nodes use the round trip time to decrease congestion, as shown in Algorithm 5. When global repair is performed, nodes do not give away their learned parameters, but adjust according to the received parameters in the DIO, as shown in Algorithm 6.

Algorithm 5. Distributed Parameter Estimator – On DAO-ACKs

$RoundTripTime \leftarrow Current\_Time - DAOStartTime$;
if $RoundTripTime > T_U \times Rank$ **then**
  $K \leftarrow K \times (1 + \delta_K)$;
end if
if $RoundTripTime < T_L \times Rank$ **then**
  $DAO\_ACK\_TIME \leftarrow \alpha_D \times DAO\_ACK\_TIME + (1 - \alpha_D) \times RoundTripTime$
  $K \leftarrow K \times (1 - \delta_K)$;
end if
if $RoundTripTime > DAO\_ACK\_TIME$ **then**
  $Base \leftarrow Base \times (1 + \delta_B)$;
end if
if $RoundTripTime < T_B \times Rank$ **then**
  $DAO\_ACK\_TIME \leftarrow \alpha_D \times DAO\_ACK\_TIME + (1 - \alpha_D) \times RoundTripTime$
  $Base \leftarrow Base \times (1 - \delta_B)$;
end if

The constants T_U, T_L and T_B depend on the transmission time T_{Tx} of the DAO packets. If the DAO packets have a length of L_{DAO}, and the data rate of the radio is given by B_R, we chose these constants as:

$$T_{Tx} = \frac{L_{DAO}}{B_R}, \quad T_U = 8 * T_{Tx}, \quad T_L = 4 * T_{Tx}, \quad T_B = 2 * T_{Tx}.$$

Algorithm 6. Distributed Parameter Estimator – On DIOs

$K_{New} \leftarrow K$ value received in DIO TLV.
$K \leftarrow \alpha_K \cdot K + (1 - \alpha_K) \times K_{New}$
$Base \leftarrow \alpha_B \cdot Base + (1 - \alpha_B) \times Base_{New}$

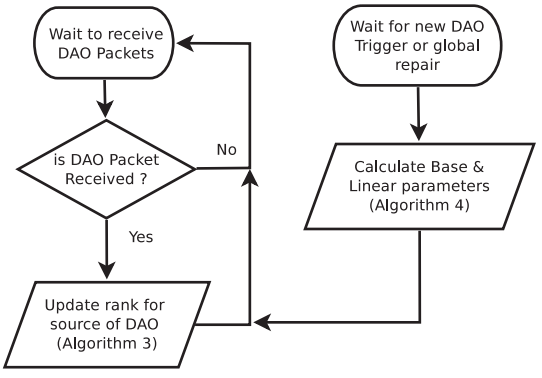

Fig. 36. Routine at LBR – centralized parameter estimation.

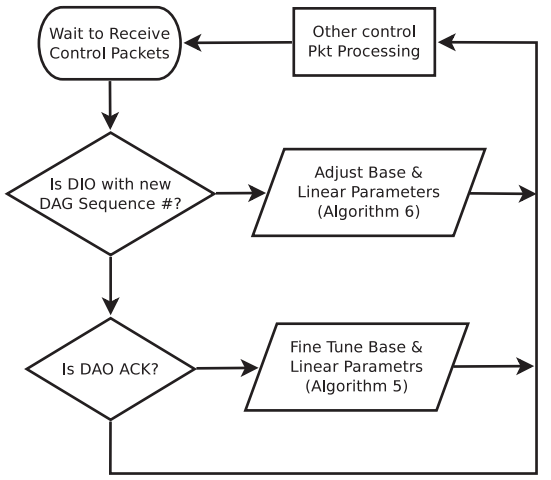

Fig. 37. Routine followed at nodes – distributed parameter tuning.

8.3. Evaluation of proposed approach

We simulate both the default specification of RPL, with a constant timer value, and our proposed method. We mainly concentrate on aspects of congestion, such as RAM occupancy, and data packet delivery delay, Fig. 38 shows the CDF of data delivery delay and we see that the proposed mechanism drastically improves the latency with which the data packets are delivered. This is an obvious outcome of mitigated congestion, as congestion causes both data and control packet to be buffered.

In Fig. 39, we plot the maximum number of packets buffered with network size and see that the propose mechanism achieves a significant reduction, with a gain in buffer size as high as 15× for the largest network. The gain increases as network size increases, a trend which is also observed for RAM consumption, as shown in Fig. 40. Less use of buffered packet leads to less use of precious memory, and the proposed mechanism consumes only 40 KBytes of memory as opposed to the default mechanism, which consumes ∼700 KB of RAM.

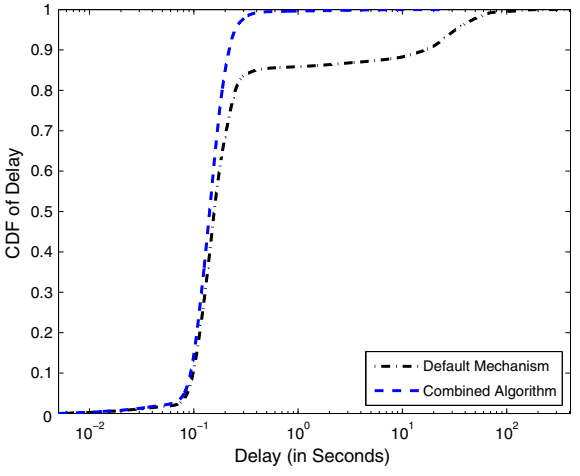

Fig. 38. CDF of data delivery delay.

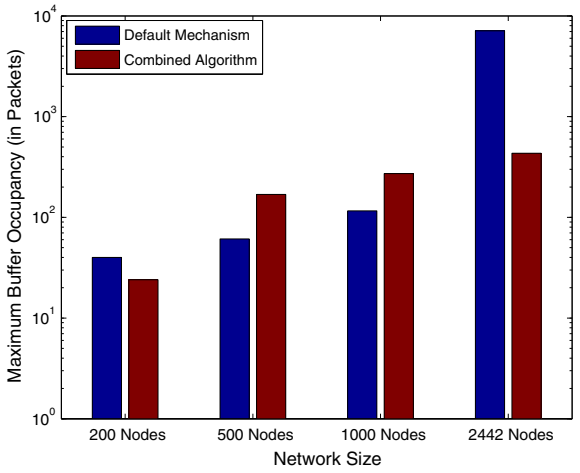

Fig. 39. Maximum buffer occupancy against network size.

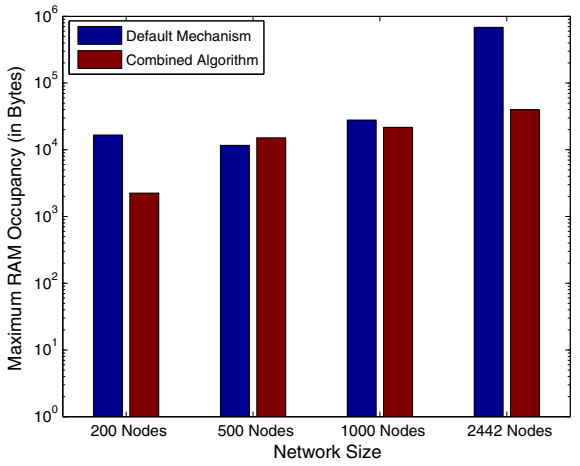

Fig. 40. Maximum RAM occupancy against network size.

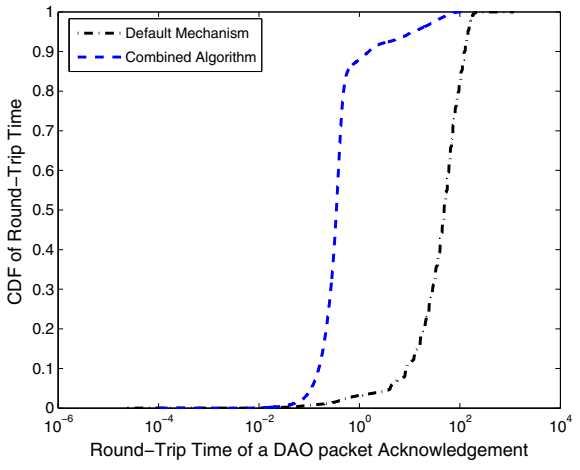

Fig. 41. CDF of DAO round trip time.

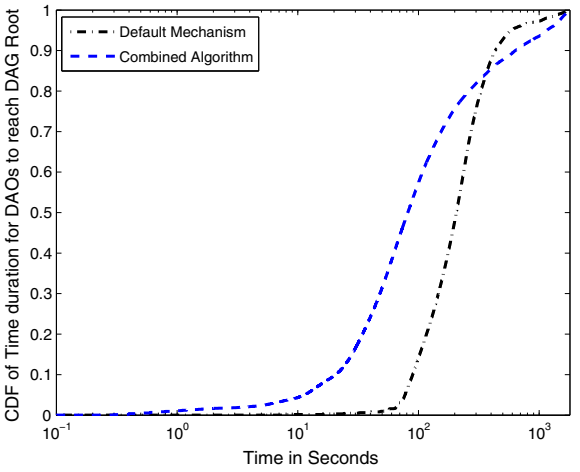

Fig. 42. CDF of time taken by DAOs to reach the DAG root.

In Fig. 41, we plot the CDF of the round trip time required by the DAO packets. Achieving a low round trip time, on one hand, shows signs of congestion free network. At the same time, a low round trip time also prevents unnecessary duplicate DAOs to be resent to the DAG root, thus saving on important control plane bandwidth. We again observe how the propose mechanism improves the round trip time when compared to the default RPL mechanism.

As it can easily be perceived, the Destination Advertisements (DAO) packets are scheduled in such a way that a congestion is avoided. By avoiding this congestion, we achieve less delay in order for data packets to reach the root, and less round trip time of DAO packets, which helps with the unnecessary issuance of destination advertisements. However, the price we pay by deploying this mechanism is high discovery time for some nodes, mostly the ones that are farthest from the root node. In Fig. 42, we show the CDF of time taken by each DAO packet to reach the LBR.

As we observe, for around 85% nodes, DAO packets in the proposed method reach the LBR earlier than the proposed default method. However, the LBR learns about 15% of nodes in the network at a later point of time than with the default mechanism. These 15% of nodes are the ones that have large ranks. In other words, the proposed method helps the LBR learn about nodes near itself faster, but for some nodes with large ranks it consumes more time, which is the price we pay for congestion avoidance. However, as DAO includes information on a node's parents only when it is generated, the information learnt is not outdated.

9. Conclusions

This paper presented a detailed performance comparison study between RPL (storing and non-storing node) and LOADng, for several topologies of interest. In particular, results were collected for a small deployment topology with 86 nodes and a large smart meter deployment with 2442 nodes. Other topology sizes were also considered in the scalability study. In the course of this investigation, we also uncovered non-optimal protocol behavior for the

case of large networks and proposed new mechanisms that are shown to improve control plane congestion, as a result improving network lifetime for large scale LLNs. Some of the important observations drawn include:

- In terms of control overhead, RPL scales well with the network size. In particular, for the large deployment of smart grid AMI network with 2442 nodes, RPL provided connectivity with the border router with much less control cost.
- Control overhead is a function of application data rate for LOADng, where it is independent for RPL.
- Path quality in terms of hop distance and total ETX path cost is very close for P2MP and MP2P traffic. However, for P2P traffic, RPL non-storing mode yields a much longer and more costly (in terms of total ETX path cost) path.
- End-to-end delay is comparable between the two protocols for the topologies studied. However, in some cases, LOADng may result in a high end-to-end delay between nodes. This is explained by the reactiveness of the protocol as well as control plane floods.
- Due to control packet flooding and buffering of data packets, LOADng has higher buffer size requirement.
- RPL tends to consume more buffer space with increase in network size, but proper scheduling of destination advertisements make RPL scale much better with increasing topology size.
- The use of our proposed DelayDAO controller – a combination of routines to generate new DAO message from nodes – drastically improves RPL's performance in terms of memory occupancy, delivery of data packets and time taken by DAO packets to reach LBR once they are generated. The performance improvement comes at a small price: 15% of nodes sitting farther away from the LBR will generate their DAO messages later than in the default RPL configuration. However, 85% of nodes enjoy a quicker delivery of destination advertisements to the LBR.

In general, RPL outperformed LOADng for several critical metrics, taking into account the traffic profile that is typical in LLNs, and only in a few cases, both protocols had similar performance. It is true that LOADng's complexity is lower and if run in a topology with very light traffic load, the protocol will thrive. Such a scenario is however becoming further and further away from the reality of current LLNs. While we highlighted some inefficiencies in RPL in large scale topologies, we also proposed a mechanism to fix this issue and showed that it outperformed the default RPL configuration.

Appendix A. Theoretical comparison: A balanced aggregation/dissemination tree model

Being a reactive protocol, LOADng has its own boon and bane. While it does not proactively disseminates control packets, on demand route discovery needs the packet to be buffered while RREQs are multicasted and RREP is received, leading to higher delay bound than a proactive protocol as RPL, as well as more buffer space. Also, control packet volume scales in proportion to the number of flows in the network. In Smart Grid AMI Meter networks, or in a building/home/industrial automation system, the LBR sends periodic data to every sensor in the system. For a network of size N, this operation needs N different RREQs to be generated by the LBR alone over the period. As each node forwards a particular RREQ for a particular destination at least once, the control overhead scales with $\Omega(N^2)$ for a network of size N. In this section we will show that RPL, by creating a DAG and forwarding DAOs only through the parent–child links in the DAG, results in an overall control overhead that is lower than $O(N^2)$, and reduces to $O(N\log(N))$ for a balanced tree structure.

Recall that RPL creates a Directed Acyclic Graph (DAG), in which routing takes place in an LLN. Since the created graph is acyclic by definition, links between preferred parents and children create a spanning tree of the network through which data aggregation (MP2P), or data dissemination (P2MP) takes place. In this section, we attempt to provide a lower and upper bound on control packets for data aggregation/dissemination traffic for both protocols. For the sake of simplicity, we first assume that the tree created by constructing a DAG is a balanced tree, and show how the control overhead for RPL and LOADng compares to one another.

A.1. Total control overhead for RPL with balanced tree

To determine the theoretical bounds, we first consider that the data aggregation tree, as constructed from the DAG structure in RPL, is a balanced tree, with B children under each parent. An example of a balanced tree with 3 children under each node is shown in Fig. A.43. The tree has a height of H and hence, the number of nodes in the network is given by $N = (B^{H+1} - 1)/(B - 1)$. Therefore, for any node at a rank R, $1 \leqslant R \leqslant H + 1$, the number of nodes in its subtree including itself is given by

$$N_R = 1 + B + B^2 + \cdots + B^{H-R+1} = \sum_{i=0}^{H-R+1} B^i \tag{A.1}$$

Note that the root itself has a rank equal to 1. Now, if the time period to emit a new DAG sequence number, as mentioned in Section 3, is T, then a new DAO is emitted by each node every T seconds. For simplicity of calculation, we assume that there is no link breakdown or topology change within this T time. For RPL, since a node forwards

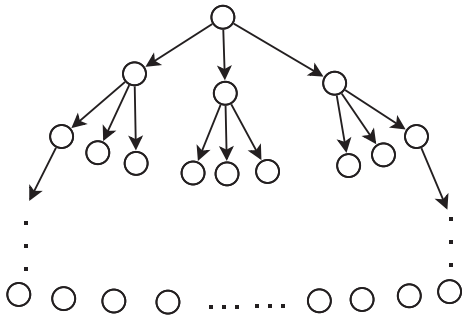

Fig. A.43. A balanced tree with $B = 3$.

all DAOs from all nodes in its sub-tree to the DAG root or LBR, the node X would handle N_R DAO messages in a period T. We assume the same topology stability during T for LOADng, and therefore the route validity time (R_Valid_Time) for any route in LOADng protocol is kept as T. Also, in LOADng, all N_R nodes under a node X at rank R would route their data to the sink/LBR through the node X and vice versa. Hence, the node X would be responsible to forward N_R RREPs to the LBR for data dissemination. Also, for both RPL and LOADng, the first respective DIO/RREQ received by a node, arrives via the best path and, hence, no further DIO/RREQ updates the parent entry/routing entry that is generated. So, for LOADng, effectively one RREQ is forwarded per destination. This assumption is necessary, because it is not known when the RREQ/DIO corresponding to the best path would arrive in a randomized situation, so we assume that they arrive through the best path first.

Since the number of nodes at a given rank R is given by B^{R-1}, the total number of DAOs to be forwarded by all nodes at a rank R is

$$DAO_R = B^{R-1} * \sum_{i=0}^{H-R+1} B^i \tag{A.2}$$

Since for all nodes in the network excluding the root node, R can vary from 2 to $H+1$, the total DAO packets in a network operating RPL is given by

$$DAO_{Total} = \sum_{R=2}^{H+1} \left(B^{R-1} * \sum_{i=0}^{H-R+1} B^i \right) \tag{A.3}$$

Since the inner sum of Eq. (A.3) can be written as

$$\sum_{i=0}^{H-R+1} B^i = \frac{B^{H-R+2}-1}{B-1} \tag{A.4}$$

Hence,

$$\begin{aligned} DAO_{Total} &= \sum_{R=2}^{H+1} \left(\frac{B^{H+1}-B^{R-1}}{B-1} \right) \\ &= \sum_{R=2}^{H+1} \left(\frac{B^{H+1}}{B-1} \right) - \sum_{R=2}^{H+1} \left(\frac{B^{R-1}}{B-1} \right) \end{aligned} \tag{A.5}$$

As mentioned in Section 3, every T seconds, DAO_{Total} control overhead is generated in the network operating RPL due to DAO propagation. Each node issues a DIO every time the trickle timer for DIO fires, and the timer period is doubled after DIO emission. Hence, each node, on average emits $\log_2 T$ DIO messages within T seconds. Assuming no delivery error, no DIS message in RPL, and that DAO_ACK messages are to be acknowledged by the LBR (same in number as DAO messages), the total control overhead in RPL for a time period S is given by

$$C_{T,RPL} = \frac{S}{T} \left[N \log_2 T + 2 \sum_{R=2}^{H+1} \frac{B^{H+1}-B^{R-1}}{B-1} \right] \tag{A.6}$$

Clearly, the assumptions here are that S is large enough to ignore the remainder of S divided by T in comparison to S, or S is divisible by T. Normally, in a practical implementation, T would be kept around half an hour, and total operation time S will be in order of days or even years maybe. So this assumption is reasonable. Some simple mathematical manipulation of Eq. (A.6), considering $N = (B^{H+1}-1)/(B-1)$ and $H = \log_B N$, can yield

$$\begin{aligned} C_{T,RPL} &= \frac{S}{T} \left[N \log_2 T + 2 \sum_{R=2}^{H+1} \frac{B^{H+1}-B^{R-1}}{B-1} \right] \\ &= \frac{S}{T} \left[N \log_2 T + 2H * \left(N + \frac{1}{B-1} \right) - \frac{2B}{B-1} * \sum_{i=0}^{H-1} B^i \right] \\ &= \frac{S}{T} \left[N \log_2 T + 2H * \left(N + \frac{1}{B-1} \right) - \frac{2B}{B-1} * \frac{B-1}{B} \left(N + \frac{1}{B-1} \right) \right] \end{aligned}$$

Since $1/(B-1)$ is a constant, and a fraction, when compared to N, we disregard it from the complexity analysis. We assume $(N + 1/(B-1)) \simeq N$, which yields, from Eq. (A.7),

$$C_{T,RPL} = \frac{S}{T} [N(\log_2 T - 2) + 2N \log_B N] \tag{A.7}$$

Clearly, when RPL creates a balanced tree through the DODAG creation, the total control overhead scales with $\Theta(N \log N)$, where N is the network size.

A.2. Total control overhead for LOADng in similar topology

Since LOADng is a reactive protocol and sets up the routing path when data transfer is needed, the control packet overhead will depend on the traffic pattern. Lets assume a simple traffic pattern of the LBR sending data to each node in a round-robin fashion, with a period F. In an LLN, F may range from 30 min to few hours. So, the LBR communicates with all N nodes once in a period F. The cache period of considering a route valid is assumed to be T, as mentioned before. Now, if $F \geqslant T$, each time the LBR wants to send some data to a particular node, the route entry for that node would be invalid. Hence, the LBR needs to set up the routing path by issuing RREQ for each node within F time. Since we assume that one RREQ is broadcasted by each node for each destination, the total RREQ transmission within F time is equal to N^2. If we assume the same topology as in the RPL analysis before, all RREPs would follow the path taken by DAOs in case of RPL. Hence, the number of RREPs in F time is same as the number of DAOs transmitted with RPL in T time, as we assume that RREQ received first corresponds to best path, and each route request generates exactly one RREP. So, similar to Eq. (A.4), the total number of RREPs in F time is given by

$$RREP_{Total} = \sum_{R=1}^{H} \left(B^R * \sum_{i=0}^{H-R} B^i \right) \tag{A.8}$$

In addition, there will be same number of RREP_ACKs as RREPs. So, the total control overhead (excluding RERR, given our assumption of no topology change within T or R_Valid_Time) for a run-time of S is given by

$$C_{T,LOAD} = \frac{S}{F} \left[N^2 + 2 \sum_{R=2}^{H+1} \left(B^{R-1} * \sum_{i=0}^{H-R+1} B^i \right) \right] \tag{A.9}$$

By simple manipulations, the total control overhead is

$$C_{T,LOAD} = \frac{S}{F}\left[N^2 + 2N(\log_B N - 1)\right] \quad (A.10)$$

Eqs. (A.7) and (A.10) provide the total control overhead for RPL and LOADng during S runtime, assuming the DAG created by RPL creates a balanced tree for the topology, and LOADng uses a similar path for the same topology. It is also assumed that both DIO and RREQ reach every node via the best path first, and the link quality does not change for time T, where T is the period at which global repair occurs for RPL, and routes are valid in LOADng. However, topologies do not tend to create a balanced tree at the LBR, and when RPL yields a completely unbalanced tree at DAG root, such as for cases in a chain-like topology, the total control overhead scales with $\Theta(N^2)$. To understand why it is so, consider a chain-like topology, where N nodes create a DAG of height equal to N. The leaf node will be responsible for 1 DAO (1 RREP for LOADng), the node above the leaf will forward 2 DAOs (2 RREP for LOADng), and so on. So, with RPL, the number of DAOs in T time will equal to

$$DAO_{Total} = 1 + 2 + \cdots + (N-1) = \frac{N(N-1)}{2} \quad (A.11)$$

The case for RREP messages for LOADng in F time will be similar. Hence both protocols will have a total control overhead complexity of $\Theta(N^2)$ for a chain-like topology. For any other practical topology, the case will be between the two extremes of a balanced tree and a chain-like topology, and in most cases RPL is expected to provide less control overhead depending on the value of T and F, as we observed in Section 5.

References

[1] A. Brandt, J. Buron, G. Porcu, Home Automation Routing Requirements in Low power and Lossy Networks, RFC 5826, April 2010. <http://tools.ietf.org/html/rfc5826>.

[2] J. Moy, OSPF Version 2, RFC 2328, April 1998. <http://tools.ietf.org/html/rfc2328>.

[3] T. Clausen, P. Jacquet, The Optimized Link State Routing Protocol (OLSR), RFC 3626, October 2003. <http://tools.ietf.org/html/rfc3626>.

[4] T. Clausen, C. Dearlove, P. Jacquet, U. Herberg, The Optimized Link State Routing Protocol version 2, DRAFT, October 2012. <http://tools.ietf.org/html/draft-ietf-manet-olsrv2-17>.

[5] P. Levis, A. Tavakoli, S. Dawson-Haggerty, Overview of Existing Routing Protocols for Low Power and Lossy Networks, DRAFT, April 2009. <http://tools.ietf.org/html/draft-ietf-roll-protocols-survey-07>.

[6] R. Albrightson, J. Garcia-Luna-Aceves, J. Boyle, EIGRP-a fast routing protocol based on distance vectors, in: Proc. Networld/Interop, vol. 94, 1994, pp. 136–147.

[7] C. Perkins, E. Belding-Royer, S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561, July 2003. <http://tools.ietf.org/html/rfc3561>.

[8] T. Winter et al., RPL: Routing Protocol for Low Power and Lossy Networks, RFC 6550, March 2012. <http://tools.ietf.org/html/rfc6550>.

[9] K. Pister, P. Thubert, S. Dwars, T. Phinney, Industrial Routing Requirements in Low-Power and Lossy Networks, RFC 5673, October 2009. <http://tools.ietf.org/html/rfc5673>.

[10] M. Dohler, T. Watteyne, T. Winter, D. Barthel, Routing Requirements for Urban Low-power and Lossy Networks, RFC 5548, May 2009. <http://tools.ietf.org/html/rfc5548>.

[11] J. Martocci, P.D. Mil, N. Riou, W. Vermeylen, Building Automation Routing Requirements in Low-power and Lossy Networks, RFC 5867, June 2010. <http://tools.ietf.org/html/rfc5867>.

[12] J. Tripathi, J. de Oliveira, J. Vasseur, Applicability study of RPL with local repair in smart grid substation networks, in: 2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010.

[13] J. Tripathi, J. de Oliveira, J. Vasseur, Performance Evaluation of Routing Protocol for Low power and Lossy Networks (RPL), RFC 6687, July 2012. <http://tools.ietf.org/html/rfc6687>.

[14] N. Accettura, L. Grieco, G. Boggia, P. Camarda, Performance analysis of the RPL routing protocol, in: 2011 IEEE International Conference on Mechatronics (ICM), 2011, pp. 767–772.

[15] T. Clausen et al., The LLN On-demand Ad-hoc Distance-Vector Routing Protocol-Next Generation (LOADng), DRAFT, July 2012. <http://tools.ietf.org/html/draft-clausen-lln-loadng-05>.

[16] U. Herberg, T. Clausen, A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN), in: Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks, PE-WASUN '11, ACM, New York, NY, USA, 2011, pp. 73–80.

[17] J. Tripathi, J. de Oliveira, On adaptive timers for improved RPL operation in low-power and lossy sensor networks, in: Fifth International Conference on Communication Systems and Networks (COMSNETS), 2013.

[18] A. Koliousis, J. Sventek, Proactive vs Reactive Routing for Wireless Sensor Networks, Tech. Rep., University of Glasgow, Department of Computing Science, 2007.

[19] S. Mohseni, R. Hassan, A. Patel, R. Razali, Comparative review study of reactive and proactive routing protocols in MANETs, in: 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), 2010, pp. 304–309.

[20] C. Mbarushimana, W. Vanderbauwhede, Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks, in: 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW '07, vol. 2, 2007, pp. 679–684.

[21] K. Pandey, A. Swaroop, A comprehensive performance analysis of proactive, reactive and hybrid MANETs routing protocols, International Journal of Computer Science Issues 8.

[22] S. Das, R. Castaneda, J. Yan, R. Sengupta, Comparative performance evaluation of routing protocols for mobile, ad hoc networks, in: Proceedings. 7th International Conference on Computer Communications and Networks, 1998, pp. 153–161.

[23] Q. Zhao, L. Tong, Energy efficiency of large-scale wireless networks: proactive versus reactive networking, IEEE J. Sel. Areas Commun. 23 (5) (2005) 1100–1112.

[24] D. Johnson, Y. Hu, D. Maltz, The Dynamic Source Routing Protocol (DSR) for Mobile Ad hoc Networks for IPv4, RFC 4728, February 2007. <http://tools.ietf.org/html/rfc4728>.

[25] C.E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in: Proceedings of the Conference on Communications Architectures, Protocols and Applications, SIGCOMM '94, ACM, New York, NY, USA, 1994, pp. 234–244.

[26] Z. Haas, M. Pearlman, P. Samar, The Zone Routing Protocol (ZRP) for Ad Hoc Networks, DRAFT, July 2002. <http://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04>.

[27] V. Park, S. Corson, Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification, DRAFT, July 2001. <http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04>.

[28] C. Petrioli, M. Nati, P. Casari, M. Zorzi, S. Basagni, ALBA-R: load-balancing geographic routing around connectivity holes in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 529–539.

[29] B. Lichtensteiger, B. Bjelajac, C. Muller, C. Wietfeld, RF mesh systems for smart metering: system architecture and performance, in: First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010, pp. 379–384.

[30] IETF Routing Over Low-Power Lossy Networks Working Group. <http://tools.ietf.org/wg/roll>.

[31] G. Iyer, P. Agrawal, E. Monnerie, R. Cardozo, Performance analysis of wireless mesh routing protocols for smart utility networks, in: 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 114–119.

[32] D. Wang, Z. Tao, J. Zhang, A. Abouzeid, RPL based routing for advanced metering infrastructure in smart grid, in: IEEE International Conference on Communications Workshops (ICC), 2010.

[33] M. Vucinic, B. Tourancheau, A. Duda, Performance comparison of the RPL and LOADng routing protocols in a home automation scenario, in: IEEE Wireless Communications and Networking Conference (WCNC), 2013, pp. 1974–1979.
[34] J. Tripathi, J. de Oliveira, Proactive versus reactive revisited: IPv6 routing for low power lossy networks, in: 47th Annual Conference on Information Sciences and Systems (CISS), 2013.
[35] A. Camillò, M. Nati, C. Petrioli, M. Rossi, M. Zorzi, IRIS: integrated data gathering and interest dissemination system for wireless sensor networks, Ad Hoc Netw. 11 (2) (2013) 654–671.
[36] A. Camillo, C. Petrioli, Hands on IRIS: Lessons learned from implementing a cross layer protocol stack for WSNs, in: Global Communications Conference (GLOBECOM), 2012 IEEE, 2012, pp. 157–163.
[37] J. Vasseur, M. Kim, K. Pister, H. Chong, Routing Metrics used for Path Calculation in Low power and Lossy Networks, RFC 6551, March 2012. <http://tools.ietf.org/html/rfc6282>.
[38] A. Varga, The OMNeT++ Discrete Event Simulation Systems, in: European Simulation Multiconference (ESM'2001), 2001.
[39] A. Boulis, Castalia: revealing pitfalls in designing distributed algorithms in WSN, in: 5th International Conference on Embedded Networked Sensor Systems (SenSys'07), 2007.
[40] O. Gaddour, A. Koubaa, S. Chaudhry, M. Tezeghdanti, R. Chaari, M. Abid, Simulation and performance evaluation of DAG construction with RPL, in: Third International Conference on Communications and Networking (ComNet), 2012.
[41] J. Hui, P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, RFC 6282, September 2011. <http://tools.ietf.org/html/rfc6282>.

Joydeep Tripathi received his B.E. degree from Jadavpur University, Kolkata, India in Electronics & Telecommunication Eng. in 2006, and his M.S. degree in Computer Engineering from Drexel University, Philadelphia, USA in 2010. He is currently a PhD candidate in Drexel University under Dr. Jaudelice C. de Oliveira. His research interests include working in routing protocols for Low Power and Lossy networks, performance evaluation and realistic simulator design, software defined networking and traffic engineering in Low-power and Lossy Networks, etc.

Jaudelice C. de Oliveira is an Associate Professor in the in Department of Electrical and Computer Engineering at Drexel University. Her research interests include the development of new protocols that improve the performance of ad hoc, sensor and computer networks.

J.P. Vasseur is a Cisco Fellow and Co-chair of IETF PCE and IETF ROLL working group. His research expertise includes IP, Traffic Engineering, Network Recovery, Ad-hoc networks, QoS, Multicast, MPLS, Routing and the "Internet of Things". He has also been Co-founder and Chair of the Technology Advisory Board of the IPSO alliance from 2008 to 2011.