BRPL: Backpressure RPL for High-throughput and Mobile IoTs

Yad Tahir, Shusen Yang, and Julie McCann

Abstract—RPL, an IPv6 routing protocol for Low power Lossy Networks (LLNs), is considered to be the de facto routing standard for the Internet of Things (IoT). However, more and more experimental results demonstrate that RPL performs poorly when it comes to throughput and adaptability to network dynamics. This significantly limits the application of RPL in many practical IoT scenarios, such as an LLN with high-speed sensor data streams and mobile sensing devices. To address this issue, we develop BRPL, an extension of RPL, providing a practical approach that allows users to smoothly combine any RPL Object Function (OF) with backpressure routing. BRPL uses two novel algorithms, *QuickTheta* and *QuickBeta*, to support time-varying data traffic loads and node mobility respectively. We implement BRPL on Contiki OS, an open-source operating system for the Internet of Things. We conduct an extensive evaluation using both real-world experiments based on the FIT IoT-LAB testbed and large-scale simulations using Cooja over 18 virtual servers on the Cloud. The evaluation results demonstrate that BRPL not only is fully backward compatible with RPL (i.e. devices running RPL and BRPL can work together seamlessly), but also significantly improves network throughput and adaptability to changes in network topologies and data traffic loads. The observed packet loss reduction in mobile networks is, at a minimum, 60% and up to 1000% can be seen in extreme cases.

Index Terms—RPL, Internet of Things, IPv6, Backpressure Routing, Low-power Lossy Networks, Wireless Sensor Networks

◆

1 INTRODUCTION

MANUFACTURERS are adapting a new breed of standards to provide an unprecedented level of transparency between business players, factory operations, and supply chain planning. This results in the fourth industrial revolution, commonly referred as "Industry 4.0". One of the core technological basis for this industrial revolution is the *Internet of Things (IoT)*. Many efforts have been made to incorporate core Internet technology, TCP/IP standards, with emerging IoT standards primarily to ensure interoperability in heterogeneous networks. Herein nodes not only represent smart sensing devices, but also can be actuators, or even traditional Internet endpoints such as computers, tablets or smartphones.

The Internet Engineering Task Force (IETF) has proposed *RPL* [1] as a de-facto IoT routing standard for IPv6-based *Low power and Lossy Networks (LLNs)*. RPL is a generic distance vector routing protocol that allows users to establish logical routing topologies, commonly known as *Directed Acyclic Graphs (DAGs)*, over a shared physical network. DAGs are computed based on *Objective Functions (OFs)* specified by users. Much work in both academia and industry has shown that RPL provides a promising routing solution for a wide range of network types and industrial applications, including Home Automation (RFC 5826) [2], Industrial Control (RFC 5673) [3], Urban Environments (RFC
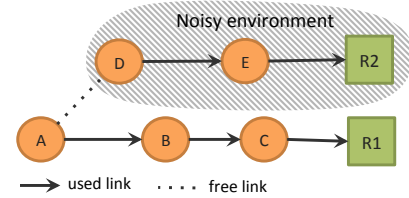

Fig. 1. Used routing paths for a system with RPL and the ETX OF.

5548) [4], Building Automation (RFC 5867) [5], Advanced Metering Infrastructure (AMI) [6], and Smart Grids [7].

1.1 Motivations

To support real-world industrial IoT applications, the underlying networking services for LLNs must meet several requirements, including support for rapidly growing demands for *high-throughput* networks [8], [9], [10], *adaptability* to the data traffic dynamics [9], [11], [12], and in some cases *mobility* of IoT devices [13], [14], [9]. Two of these requirements are formally stated in RFC 5673 [3] and RFC 5867 [5]. However based on the current specifications, RPL can perform poorly in terms of meeting these requirements, which limits its adaption in numerous IoT applications.

1) **Throughput.** As a distance vector routing protocol [15], RPL may suffer from severe congestion and packet loss when the network traffic is heavy (e.g., where multiple IoT applications coexist in a network). This is mainly because the DAGs defined by the OFs may not utilize the full network capacity. Fig. 1 shows an example of this issue, in which node A wants to transfer data to the DAG's roots R1 or R2 (i.e., anycasting). Assume the network uses the standard ETX metric [16] as the OF. In this case, RPL will

- *Y. Tahir and J. McCann are with Department of Computing, Imperial College London.*
  E-mail: {y.tahir11, j.mccann}@imperial.ac.uk.
- *S. Yang is with the School of Mathematics and Statistics, Xi'an Jiaotong University, China.*
  E-mail: shusenyang@mail.xjtu.edu.cn.

use A-B-C-R1 in Fig. 1 to avoid using the noisy path A-D-E-R2 as much as possible. As a result, congestion and packet loss may occur on path A-B-C-R1 when the network is handling a large amount of traffic, which is typical in multi-user IoT systems. This could be mitigated by exploiting the additional capacity of the suboptimal path A-D-E-R2.

2) **Traffic Load Adaptivity.** The highly dynamic and unplanned nature of LLN applications can produce time-varying data traffic patterns (e.g., traffic bursts generated by event-based applications). However, RPL fails to adapt to such traffic dynamics due to its fixed configurations. It is crucial to have an adaptive solution such that it utilizes necessary resources based on traffic demands. When the network experiences low data traffic, the default path specified by OF is sufficient. The suboptimal paths (e.g., path A-D-E-R2 in Fig.1) are needed only when the data traffic volume is high.

3) **Mobility.** Due to the time-varying network topology caused by node mobility, invalid routes and link breakage are likely to exist in DAGs. The lack of adaptivity and mobility-awareness in an OF leads to the slow response to changes in network topologies. This makes RPL highly inefficient and potentially impractical in mobile networks. Network resources must be utilized opportunistically as well as strategically to support low power and lossy IoT systems with mobile nodes.

1.2 Contributions

To address the mentioned limitations, Backpressure RPL (BRPL) is proposed, a new extension of RPL that provides enhanced support for high throughput, adaptivity and mobility without any modification or assumption on RPL OFs. Our contributions are summarized as follows:

1) We developed BRPL, the first work that incorporates the principles of backpressure-based optimization with RPL routing. BRPL is a multi-topology routing protocol that smartly routes traffic based on the gradients of both differential queue backlogs and OFs provided by the users. The basic idea of BRPL is to adaptively and smoothly switch between RPL and the backpressure routing (classic Lyapunov-based routing) according to network conditions. This is achieved by two lightweight control algorithms: *QuickTheta* supporting data traffic dynamics and *QuickBeta* for topology dynamics (i.e., mobility). BRPL has two interesting features: when the traffic loads in the network are high, BRPL can achieve *high throughput* by utilizing all possible resources to handle the overwhelming data traffic. However, when the network experiences light data traffic in all nodes, BRPL is *objective function optimal* routing and becomes identical to RPL routing. As a result, BRPL maintains the advantages of both RPL and backpressure routing, while avoiding their disadvantages.

2) BRPL uses the same control message structures defined in the specifications of RPL. This ensures that BRPL is *interoperable* with RPL, i.e. IoT devices running original RPL or BRPL can operate together seamlessly in a hybrid network, without requiring any source code modification on nodes running RPL. Such software compatibility is very important in practice, as the routing layer of some nodes (such as non-programmable Zigbee chips) is not reprogrammable

or replaceable. To the best of our knowledge, this is the first work that aims to make backpressure-based routing feasible in hybrid networks.

3) BRPL strongly adheres to the "*Application Transparency*" design principle. Improving throughput, adaptivity and mobility of the network in BRPL is achieved without requiring any knowledge or statistical assumptions on OFs and their implementation details. This is vital in IoT as OFs for IoT applications can be widely dissimilar. The current specifications of RPL do not provide any constraint on the types of OFs.

4) We implemented BRPL in Contiki OS [17], an open source operating system for LLNs and IoT. Through both real-word experiments on the FIT IoT-LAB testbed [18] and extensive simulations with Cooja (Contiki's network simulator) over a private Cloud with 18 servers, we demonstrate that BRPL can work seamlessly with RPL, and it has control overhead similar to RPL and backpressure routing. More importantly, the results show that BRPL smartly achieves the advantages of both RPL and backpressure routing and significantly outperforms them in terms of reliability, end-to-end delay, throughput, adaptability to network dynamics, and mobility support.

1.3 Paper Organization

The remainder of the paper is organized as follows. The next section presents system models and more detailed information about RPL. BRPL extension is proposed in Section 3. Section 4 provides a detailed discussion about our proposed solution. Testbed experiments and simulation results are presented in Section 5. Related work is discussed in Section 6. Finally, we conclude the paper in Section 7.

Table 1 summarizes the key symbols used in this work.

TABLE 1
Main symbols used in this paper.

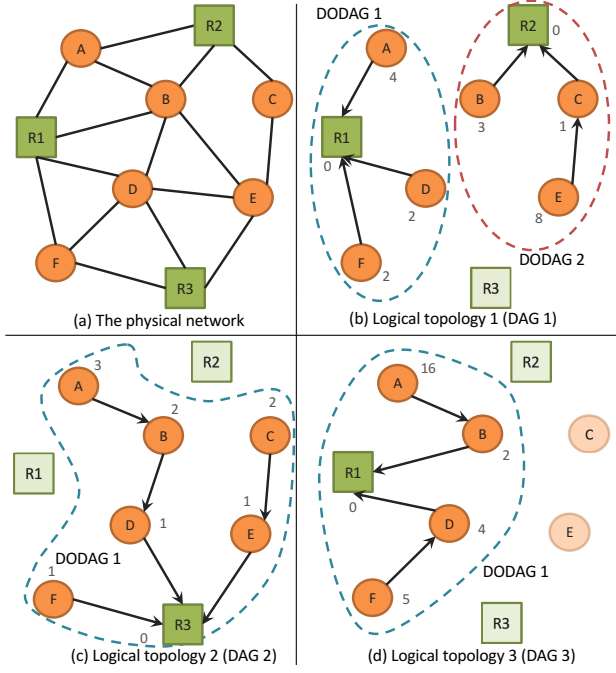|	
$\mathcal{S}, \mathcal{R}$	The sets of all sensor nodes and roots, respectively.
$\mathcal{N}$	The set of all IoT devices $\mathcal{N} = \mathcal{S} \cup \mathcal{R}$.
$\mathcal{L}$	The set of all wireless links.
$\mathcal{N}_x(t)$	The set of x's all neighbors at slot t.
$\mathcal{M}$	The set of all DAGs available in the network.
$\mathcal{R}$	The set of all roots for the network.
$\mathcal{R}_m$	The set of all roots for DAG m.
$c_{x,y}(t)$	Channel capacity of link x, y at slot t.
$f_{x,y}^m(t)$	Data rate for DAG m over link (x, y) at t.
$f_{x,m}^{in}(t)$	Total incoming DAG m data rate of node x at t.
$f_{x,m}^{out}(t)$	Total outgoing DAG m data rate of node x at t.
$RootRank_x^m$	The Rank of root x for DAG m.
$Rank_x^m(t)$	The Rank of node x for DAG m at t.
$p_{x,y}^m(t)$	The penalty over link for DAG m x, y at t.
$Q_x^m(t)$	The queue length of node x for DAG m at t.
$r_x^m(t)$	Data packets generated by node x for DAG m at t.
$Q_y^m(x, t)$	The queue length for the neighbor record y in the neighbor table of node x.
$MaxQ_x^m$	Maximum queue size for DAG m in node x.
$MaxQ_y^m(x)$	Maximum queue size for the neighbor record y in the neighbor table of node x.
$w_{x,y}^m(t)$	The weight over link (x, y) for DAG m at t
$\theta_x^m(t)$	Tradeoff parameter between throughput and OF.
$\triangle Q_{x,y}^m(t)$	the queue length difference between x, y for DAG m.
$\overline{Q}_x^m(t)$	The moving average of x's queue for DAG m at t.
$\beta_x(t)$	The mobility-awareness parameter for node x at t.

Fig. 2. An illustration for logical routing topologies in RPL. The circles and squares represent nodes and roots, respectively. Ranks are represented as numbers next to the nodes and roots. Both (c) and (d) topologies have one DODAG and one DAG each, whereas the logical topology in (b) has two DODAGs and one DAG.

2 SYSTEM MODELS

We consider an LLN that consists of a set of IoT devices $\mathcal{N} = \mathcal{S} \cup \mathcal{R}$ communicating in a multi-hop fashion as shown in Fig. 2(a); where $\mathcal{S}$ represents the set of all sensor nodes that can generate and relay data packets, and $\mathcal{R}$ is the set of all roots/gateways that collect the data traffic produced by the network. The network operates in discrete time slots $t \in \{1, 2, ...\}$.

2.1 Modeling the Physical LLN

To model the dynamic and lossy wireless transmissions, we define $c^{\max} \geq c_{x,y}(t) \geq 0$ as the logical link-layer channel capacity from node x to node y at time slot t, i.e. $c_{x,y}(t)$ indicates the maximum (integer) number of acknowledged packets that can be *successfully transmitted* from node x to y during time slot t. Here, $c^{\max}$ is the maximum possible $c_{x,y}(t), \forall t$, which is bounded by the data rate of the wireless radio. For instance, experimental studies show that a commonly-used IEEE 802.15.4 transceiver, CC2420 (e.g.[19]), can achieve a data rate of approximate 160 40-bytes packets per second [20] in practice.

The logical channel capacity depends on a wide range of random events such as wireless interference and node mobility. When $c_{x,y}(t) > 0$, it indicates that both node x and y are one-hop away from each other at time slot t. Let $\mathcal{N}_x(t) \subseteq \mathcal{N}$ be the set of all possible one-hop neighbors that node $x \in \mathcal{N}$ can communicate with during slot t:

$$\mathcal{N}_x(t) := \{ y \mid c_{x,y}(t) > 0, c_{y,x}(t) > 0, y \in \mathcal{N} - \{x\} \}$$

The whole network is modeled as a time-varying weighted graph $G(\mathcal{N}, \mathcal{L}, \mathbf{c}(t))$ where $\mathcal{L}$ represents all pos-

sible wireless links for all node pairs in $\mathcal{N}$. $|\mathcal{L}|$–dimensional vector $\mathbf{c}(t)$ holds the channel capacities for all the links at t.

2.2 Network-theoretical Modeling of RPL routing

The IETF Routing over Lossy and Low-power Networks (RoLL) group established the specifications of RPL, published as RFC 6550 [1]. This section presents network theoretic models for the key RPL specifications.

2.2.1 Multi-Commodity Model for Multi-Topology Routing

Topologically, RPL adheres to the concept of *Multi-Topology Routing (MTR)*. RPL allows multiple instances of RPL to run concurrently in a network. Each RPL instance constructs a logical topology called *Directed Acyclic Graph (DAG)* that consists of one or more *Destination Oriented DAGs (DODAGs)* (one DODAG per one root destination). All logical DAGs share the same physical network infrastructure, and simultaneously support different routing optimization objectives. For instance, Fig.2 shows illustrative examples of three DAGs for a common physical LLN, where the DAG in Fig.2 (b) consists of two DODAGs, and the DAGs in Fig.2 (c) and (d) have one DODAG each.

Let $\mathcal{M}$ be the set of all DAGs available in the network. $\mathcal{R}_m$ represents the set of all roots for the DAG $m \in \mathcal{M}$. Hence, the set of all roots for all DAGs can be seen as:

$$\mathcal{R} = \bigcup_{m \in \mathcal{M}} \mathcal{R}_m \qquad (1)$$

We use the multi-commodity model [21] to formalize the RPL protocol in a network. Here, each DAG m in the network can be seen as a commodity. All data traffic in a DAG m should be transmitted to any destination $r \in \mathcal{R}_m$.

Definition 1 [MTR Traffic Feasibility]. *Denote $f_{x,y}^m(t) \geq 0$ as the actual amount of data traffic for DAG m over a wireless link (x,y) at slot t. Let $f_{x,m}^{in}(t) = \sum_{y \in \mathcal{N}_x(t)} f_{y,x}^m(t)$ and $f_{x,m}^{out}(t) = \sum_{y \in \mathcal{N}_x(t)} f_{x,y}^m(t)$ be the total data DAG-m incoming and outgoing traffic of node x at slot t respectively. Then for any feasible multi-topology routing approach (e.g. RPL), the following two conditions should be satisfied:*

$$\sum_{m \in \mathcal{M}} f_{x,y}^m(t) \leq c_{x,y}(t), \forall x, y \in \mathcal{N}, t \geq 1 \qquad (2)$$

$$\overline{r}_x^m + \overline{f}_{x,m}^{in} \leq \overline{f}_{x,m}^{out}, \forall x \in \mathcal{S}, m \in \mathcal{M} \qquad (3)$$

where $\overline{f}_{x,m}^{in}$ and $\overline{f}_{x,m}^{out}$ represent the long-term averages of $f_{x,m}^{in}$ and $f_{x,m}^{out}$ respectively, and $\overline{r}_x^m \geq 0$ is the long-term average of DAG-m data traffic generated by node x.

Condition (2) ensures that the total data traffic for all DAGs over a link should not exceed its channel capacity. Condition (3) states the *flow conservation law*, i.e. the total incoming data traffic for a given DAG m at any IoT device x should not be more than the total outgoing data traffic for DAG m for x.

2.2.2 Objective Function and Routing Gradient

RPL constructs each DAG by using an OF, which defines a specific routing optimization objective, such as minimizing energy consumption or end-to-end delay. All DODAGs within a DAG share the same OF. Each DODAG in a DAG
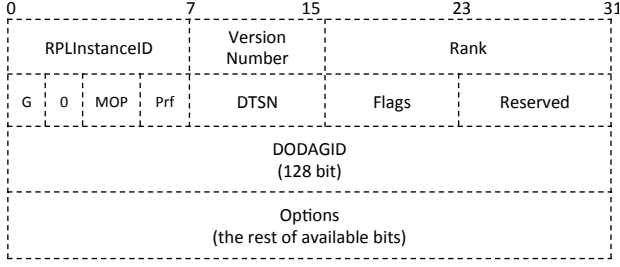
Fig. 3. DIO message structure in RPL.

Fig. 4. Illustration of software architecture of BRPL.

is computed by a scalar variable associated with each node called `Rank`, which is basically the logical distance between a node and the corresponding root of the DODAG. For instance, the Rank in Fig.2(c) represents hop count, and the routing objective is to minimize the number of hops for packet delivery from the nodes to the destination R3. The Rank value of node $x \in \mathcal{N}$ in $m \in \mathcal{M}$ is computed as:

$$Rank_x^m(t) = \begin{cases} \min\limits_{y \in \mathcal{N}_x(t)} (p_{x,y}^m(t) + Rank_y^m(t)) & x \notin \mathcal{R}_m \\ \text{RootRank}_x^m & x \in \mathcal{R}_m \end{cases} \quad (4)$$

where $p_{x,y}^m(t) > 0$ denotes the penalty/cost of using the link (x,y) in DAG m at time slot t, and $\text{RootRank}_x^m \geq 0$ is the smallest Rank value in the DAG m. Hence, when a node x is not a root, its rank is computed by finding a one-hop connection that gives the smallest sum of neighbor rank $Rank_y^m$ and link penalty $p_{x,y}^m$.

2.2.3 DIO Message and DODAG Establishment

The construction of any DODAG [1] is fully distributed, but initially triggered by the root r of the DODAG. r starts by broadcasting a DODAG Information Object (DIO) message to its one-hop neighbors. Fig. 3 that shows the DIO message structure. DIO is an ICMPv6 information message holding important parameters about the DODAG including: RPLInstanceID, Version Number, and the Rank of the sender.

Each neighbor $x \in \mathcal{N}_r(t)$ computes its Rank value based on Eq.(4), updates the Rank parameter, and then broadcasts its DIO message to its one-hop neighbors $y \in \mathcal{N}_x(t'), t' \geq t$. This process repeats itself for all other nodes existing in the network. Based on the computed Rank values, each node x is going to choose its optimal neighbor y_m^* as follows:

$$y_m^*(x,t) = \arg \min_{y \in \mathcal{N}_x(t)} (p_{x,y}^m(t) + Rank_y^m(t)) \quad (5)$$

This neighbor is commonly referred as the *preferred parent* for node x in the DAG m. Whenever a node receives data packets associated with m DAG, it forwards them to its preferred parent. The parent then repeats the process until a root $r \in \mathcal{R}_m$ receives the packets.

The broadcasting process for RPL relies on the Trickle algorithm specified in RFC 6206. It is not necessary to have a broadcasting process per each DODAG, instead it can be per each DAG. It is important to note that the Trickle

---

1. For brevity, this work only considers RPL nodes operating as both leaf and router. However, it is straightforward to extend our proposed solution to support the leaf-only and router-only modes.
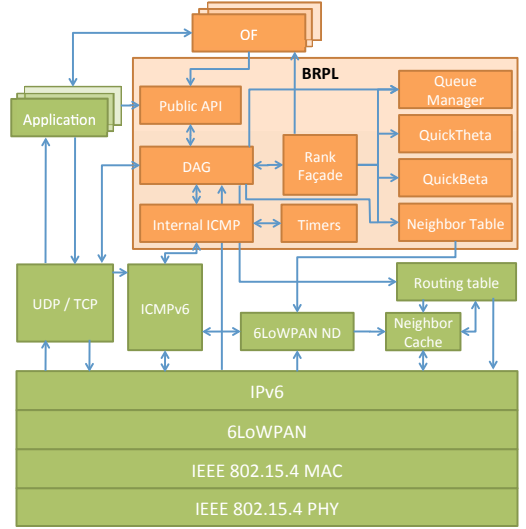
algorithm takes network stability into account. Receiving a DIO message from a sender with a lesser Rank that causes no changes to the recipient's preferred parent or Rank is considered as a 'consistent' DIO message with respect to the Trickle timer. When the network continuously encounters consistent DIO messages (i.e. the network is stable), the algorithm decreases the broadcasting rate, which results in performing less update operations on the neighbor tables. In this case, the tradeoff between energy efficiency and neighbor table consistency is considered to be highly justifiable for LLNs, particularly when nodes have limited resources. However, in time-varying networks, the Trickle algorithm increases the DIO broadcasting rate. Neighbor table maintenance is therefore performed more frequently to ensure DAG consistency and accuracy.

3 BRPL

This section describes our proposed extension to RPL, BRPL, aiming to enhance the performance of RPL in terms of mobility support, high throughput, and adaptivity to the network traffic dynamics. The software architecture of the proposed solution, which is illustrated in Fig. 4, has the following key components:

- `Internal ICMP` communicates with ICMPv6 in order to send/receive DIO, DAO, and DIS control messages. When an ICMPv6 message is received, the internal ICMP component first extracts the payload, and then notifies the DAG component.
- `Timers` holds all timer-related logics including the Trickle algorithm.
- `Public API` provides a clean interface to allow external components such as user applications to adjust or interact with BRPL.
- `QuickBeta` holds the implementation details for the mobility-awareness indicator.
- `QuickTheta` an online algorithm that actively adjusts the parameter settings of BRPL based on current network dynamics.

- `OF` is the objective function provided by the application layer.
- `Neighbor Manager` manages the neighbor table of the node. It communicates with IPv6 Neighbor Discovery Service to synchronize the neighbor table.
- `Queue Manager` manages the data buffer (queue) of each DAG. The queue stores incoming IPv6 data packets.
- `Rank Façade` is responsible for calculating Ranks and link weights for the one-hope neighbors. As the name indicates, this component follows the well-known Façade design pattern [22] in software engineering.
- `DAG` contains all the logic related to DAGs including modifying IPv6 routing table, routing repair, choosing best parent based on `Rank Façade`.

3.1 Multi-topology Queueing System

Similar to RPL, BRPL follows the design principles of MTR. Each DAG is established by an OF. However unlike RPL, BRPL combines network congestion gradients (i.e. differential queue gradients) and OF ranking for handling upward data routing. Each IoT device running BRPL is required to maintain a queue (packet buffer) for each DAG (This queue is maintained by the queue manager in Fig. 4). Let $Q_x^m(t) \geq 0$ be the queue backlog (or queue length) of node $x \in \mathcal{N}$ for DAG $m \in \mathcal{M}$ at slot t. The queue dynamics of x are defined as follows:

$$0 \leq Q_x^m(t) \leq MaxQ_x^m \quad (6)$$
$$Q_x^m(t+1) = |Q_x^m(t) - f_{x,m}^{out}(t)|_+ + r_x^m(t) + f_{x,m}^{in}(t) \quad (7)$$

where $MaxQ_x^m > 0$ is the maximum queue length (i.e. allocated data buffer size) of x for DAG m. $r_x^m(t)$ denotes the amount of data packets produced by node x for DAG m at time slot t. The operator $|a|_+$ means $\max(a, 0)$. The queue length of any root is always equal to zero:

$$Q_r^m(t) = 0, \ \forall r \in \mathcal{R}, \ m \in \mathcal{M}, \ t \geq 1$$

3.2 Neighbor Table Maintenance

When a node receives a DIO message from a one-hop neighbor, it updates a few fields for the sender's record in the neighbor table. This includes *rank*, *queue backlog* and *maximum queue length*.

Because this work considers hybrid networks (i.e. some nodes may use the original specifications of RPL and do not advertise queue backlogs), the 'queue backlog' field for any neighbor record in BRPL is updated as follows:

$$Q_y^m(x,t) = \begin{cases} Q_y^m(t) & y \text{ is a BRPL node} \\ \frac{Rank_y^m(t)}{Rank_x^m(t)} Q_x^m(t) & y \text{ is a RPL node} \end{cases} \quad (8)$$

Where $Q_y^m(x,t)$ denotes the queue backlog (a counter) for the node y in x's neighbor table. Eq. 8 is designed carefully to address hybrid networks. Here we have two cases:

When y runs the original RPL routing protocol, then queue details are going to be missing in y's DIO messages.

In this case, node x updates the $Q_y^m(x,t)$ by scaling its queue length $Q_x^m(t)$ based on the rank of x and y. The use of the $Rank_y^m(t)/Rank_x^m(t)$ ratio avoids x sending data packets to its child nodes, thus less routing loops. Based on Eq. 4, we can observe that nodes choosing x as a parent have ranks equal to or larger than node x's rank.

The second case is when y runs BRPL. Here, $Q_y^m(t)$ is not missing in DIO messages. x updates the $Q_y^m(x,t)$ field such that $Q_y^m(x,t) = Q_y^m(t)$.

Likewise, the maximum queue length field $MaxQ_y^m(x)$ for neighbor y in x's neighbor table is updated according to:

$$MaxQ_y^m(x) = \begin{cases} MaxQ_y^m & y \text{ is a BRPL node} \\ MaxQ_x^m & y \text{ is a RPL node} \end{cases} \quad (9)$$

BRPL also uses the Trickle algorithm to control the broadcasting rate of DIO messages. Similar to RPL, BRPL is a fully distributed routing protocol. Each node only needs to broadcast its DIO messages to its one-hop neighbors. There is no need to relay DIO messages for other nodes. Global repairing operations are generally not required.

3.3 Data Forwarding based on RPL Ranks and Queue Backlogs

At each time slot t, BRPL performs routing and data forwarding operations for upward data traffic as follows:

- **Link Weight Calculation.** Each IoT device x running BRPL computes a weight $w_{x,y}^m(t)$ for each neighbor $y \in \mathcal{N}_x(t)$ by combining queue length information and RPL rank values:

$$w_{x,y}^m(t) = \theta_x^m(t)\tilde{p}_{x,y}^m(t) - (1 - \theta_x^m(t))\triangle Q_{x,y}^m(t)c_{x,y}(t) \quad (10)$$

where

$$\tilde{p}_{x,y}^m(t) = p_{x,y}^m(t) + Rank_y^m(t) \quad (11)$$
$$\triangle Q_{x,y}^m(t) = Q_x^m(t) - Q_y^m(x,t) \quad (12)$$

and $0 \leq \theta_x^m(t) \leq 1$ is the tradeoff parameter between average queue backlogs and minimizing objective function of DAG m, which is adaptively updated in real-time by the QuickTheta algorithm[2].

However in practice, the Eq. (10) may suffer from *scaling* issues between the range values of $\triangle Q_{x,y}^m(t)$ and $\tilde{p}_{x,y}^m(t)$. For example, the $\tilde{p}_{x,y}^m(t)$ can be in [0,255] (e.g. choose hop count as the Rank metric), whereas $\triangle Q_{x,y}^m(t)$ may only between [0,10000]. To solve this problem, we can normalize the left and right operands as:

$$w_{x,y}^m(t) = \theta_x^m(t)\tilde{p}_{x,y}^m(t) - (1 - \theta_x^m(t))\triangle Q_{x,y}^m(t)\frac{c_{x,y}(t)}{c^{max}}$$

$$\tilde{p}_{x,y}^m(t) = \frac{p_{x,y}^m(t) + Rank_y^m(t)}{\text{MaxRank}}$$

$$\triangle Q_{x,y}^m(t) = \frac{Q_x^m(t)}{MaxQ_x^m} - \frac{Q_y^m(x,t)}{MaxQ_y^m(x)}$$

---

2. Please note that the weight $w_{x,y}^m(t)$ for BRPL adopts the typically combination of queue backlog difference $\triangle Q_{x,y}^m(t)$ and penalty $\tilde{p}_{x,y}^m(t)$, which are also used in other backpressure based routing algorithms such as [9], [23].

Where $\mathrm{MaxRank} \leq 2^{16} - 1$ is the maximal rank value, depending on the Rank metric types (e.g. hop count, ETX). Here, $2^{16} - 1$ is the maximum possible value of $\mathrm{MaxRank}$ specified in the RPL specifications. $w_{x,y}^m(t)$ in this case for any two pairs of nodes is always in $[-1, 1]$ range.

- **Routing and Data Forwarding.** With the computed weights $w_{x,y}^m(t)$, $\forall y \in \mathcal{N}_x(t)$, node x will compute its *potential* parent (i.e. next-hop destination) y_m^* for DAG m:

$$y_m^*(x,t) = \arg \min_{y \in \mathcal{N}_x(t)} (w_{x,y}^m(t)) \qquad (13)$$

Data packets for upward data traffic are forwarded to the potential parent y^* if $w_{x,y_m^*}^m(t) > 0 \vee \triangle Q_{x,y_m^*}^m(t) > 0$. The actual data forward should be less than the current channel capacity and the number of data packets in its queue, i.e.

$$f_{x,y_m^*}^m(t) \leq \min(Q_x^m(t), c_{x,y_m^*}^m(t))$$

BRPL has the following two special and interesting cases:

- *Objective Function Optimality*: It is easy to observe that Eq. 10 would be degraded to Eq. 4, when $\theta_x^m(t) = 1$. Therefore, BRPL has identical performance to RPL when $\theta_x^m(t) = 1$ $\forall x, y \in \mathcal{N}$ and both schemes greedily minimizes the routing objective specified by the OF.
- *High Throughput* : When $\theta_x^m(t) = 0$ $\forall x, y \in \mathcal{N}$, BRPL would achieve high throughput, because its behavior (Eq. 10) will be similar to the well-known throughput-optimal *backpressure routing* [24].

3.4 QuickTheta

The parameter $\theta_x^m(t)$ in Eq. 10 can be configured directly by the users, but this means that users must have knowledge about the underline routing protocols and the current physical network infrastructure. This is not feasible in practice for dynamic IoT, especially in multi-user IoT systems. To abstract this complexity from the application layer, we propose *QuickTheta*, a lightweight online algorithm that adaptively adjusts parameter $\theta_x^m(t)$, according to network traffic congestion levels, without having any assumption on the deployed applications or their expected traffic levels.

QuickTheta maintains a *smooth queue length* $\bar{Q}_x^m(t)$, which is an *Exponential Weighted Moving Average (EWMA)* of $Q_y^m(x,t)$, i.e.

$$\bar{Q}_y^m(x,t) = \begin{cases} 0 & t = 1 \\ \alpha \bar{Q}_y^m(x,t-1) + (1-\alpha)Q_y^m(x,t) & t > 1 \end{cases}$$

where $0 \leq \alpha \leq 1$ is the smoothing factor. Based on the current smooth queue length value, $\theta_x^m(t)$ is computed as:

$$\theta_x^m(t) = \beta_x(t) \left(1 - \frac{1}{|\mathcal{N}_x(t)| + 1} \sum_{y \in \mathcal{N}_x(t) \cup \{x\}} \frac{\bar{Q}_y^m(x,t)}{MaxQ_y^m(x)} \right) \qquad (14)$$

where the $\beta_x(t)$ parameter is for mobility-awareness, which is discussed in the next subsection. For now, assume

$\beta_x(t) = 1$. The ratio $\bar{Q}_y^m(x,t)/MaxQ_y^m(x)$ is considered as a measurement of y's *local congestion level* for DAG m. This measurement relies on the usage share of the y's queue. The more packets are stored in the queues of the nodes $y \in \mathcal{N}_x(t) \cup \{x\}$, the closer $\theta_x^m(t)$ is to 0. This results in pushing BRPL to increase network throughput based on Eq. 10. The reasons behind the design choice of Eq. 14 are presented in Section 4.2.

3.5 QuickBeta: Mobility Support

QuickBeta computes the $\beta_x(t)$ parameter in Eq. 14 based on the mobility condition of the nodes. This is defined as follows:

$$\beta_x(t) = \frac{1}{\triangle t} \sum_{\tau = t - \triangle t}^{t-1} \frac{|\mathcal{N}_x(\tau) \cap \mathcal{N}_x(\tau + 1)|}{\max(|\mathcal{N}_x(\tau) \cup \mathcal{N}_x(\tau + 1)|, 1)} \qquad (15)$$

which observes the state changes of one-hop neighbor nodes for the node x within the time window $[t - \triangle t, t)$. The more neighbors change their states (i.e. from online to offline or from offline to online), the closer $\beta_x(t)$ to 0 and the more node x is seen as mobile.

For example, let node x has three neighbors $\{A, B, C\}$ at slot τ, and two existing neighbors $\{B, C\}$ leave and a new neighbor D joins at slot $\tau + 1$, i.e. $\mathcal{N}_x(\tau + 1) = \{A, D\}$. We compute the neighbor state changing rate for x from slot τ to $\tau + 1$ as

$$\frac{|\mathcal{N}_x(\tau) \cap \mathcal{N}_x(\tau + 1)|}{|\mathcal{N}_x(\tau) \cup \mathcal{N}_x(\tau + 1)|} = \frac{|\{A\}|}{|\{A, B, C, D\}|} = 0.25$$

From the Eq. 14, it is easy to see that the closer $\beta_x(t)$ is to 0 (i.e. the more node is mobile), the closer $\theta_x^m(t)$ is to 0 too. This causes BRPL to rely more on queue length differential for routing operations based on Eq. 10. In addition, the β parameter in Eq. 14 can be weighted to reduce or increase the sensitivity of mobility awareness in BRPL routing.

The rationale behind the design of QuickBeta algorithm is presented in Section 4.2.

4 DISCUSSION AND ANALYSIS

4.1 Design Principles of BRPL

To provide a solution that is effective in IoT, the following factors have been considered in BRPL:

4.1.1 Low Power and Lossy IoT

Similar to RPL, BRPL is designed mainly for LLNs using IPv6. The characteristics of LLNs are carefully considered in the proposed solution. In particular we focus on satisfying the limited power and processing resources. Control message overheads are intentionally kept to minimum. Only one new 6-byte field is added to RPL DIO control messages. Both QuickTheta and QuickBeta does not require statistical models or a learning/training phase to operate.

4.1.2 Focus on many-to-one routing, yet any-to-any is still supported

BRPL focus mainly on the upward data traffic, which is predominant traffic patterns in LLNs. Here, the traffic is multipoint-to-point. As stated in [25], other traffic patterns like unicast and point-to-multipoint are less frequent in LLNs. For these types of patterns, BRPL uses the two operation modes (*non-storing* and *storing*) that are originally defined by the specifications of RPL.

4.1.3 Different Mobility Schemes

Although BRPL uses the mobility metric defined in Eq. 15, other mobility metrics can be easily integrated with BRPL. This includes, but not limited to, rendezvous-based [26], trajectory-based [27] and social-aware [23] mobility metrics.

4.2 Design Rationale for QuickTheta and QuickBeta

When we designed QuickTheta and QuickBeta, we considered a wide range of factors to ensure having an implementable and practical solution in LLNs. The following list briefly highlights three advantages of our algorithm design:

Single-layer Dependency: Both QuickTheta and QuickBeta are self-contained and relies on one network layer only, the routing layer. This is desirable design property to have. Making QuickTheta and QuickBeta independent from the other network layers not only enables seamless integration with the OSI model, but also ensures usability under diverse communication systems. For example, utilizing a custom MAC layer does not cause any interoperability issue for the two algorithms.

Simplicity: The choice of Eq. 14 and 15 is suitable for nodes with limited amount of resources. Both algorithms are simple to compute, and they do not rely on sophisticated statistical models. There is no need to perform resource-hungry data analysis operations here, nor to keep extensive historical data from multiple network layers. The QuickBeta algorithm only needs to retain the value of $\mathcal{N}_x(t)$ for the time slots between $(t - \triangle t)$ and t. The $\bar{Q}_y^m(x,t)$ for all the nodes in $y \in \mathcal{N}_x(t)$ in the QuickTheta algorithm can be maintained in a vector of $\mathcal{N}_x(t) + 1$ elements.

User Abstraction and Self Parameter Tuning: It is easy to observe that both Eq. 14 and 15 do not incorporate a direct feedback from network users, instead they rely on neighbor table and network traffic congestion levels to adjust their performance. This is intentional and by design because it makes BRPL more suitable for multi-user networks, specially when a network has a DAG serving multiple greedy users. Even when user applications are highly dynamic and unpredictable, users are not required to tune QuickTheta and QuickBeta at runtime. This design also offers an effective solution to overcome user *selfishness* as a user cannot mislead the QuickTheta and QuickBeta algorithms to gain a better position in the network.

4.3 RPL Backward Compatibility Support

To ensure backward compatibility with RPL, BRPL does not introduce any new control message types but rather reuses the control message structures defined in RPL. Node $x \in \mathcal{N}$ broadcasts its queue length Q_x^m to $\mathcal{N}_x(t)$ at time
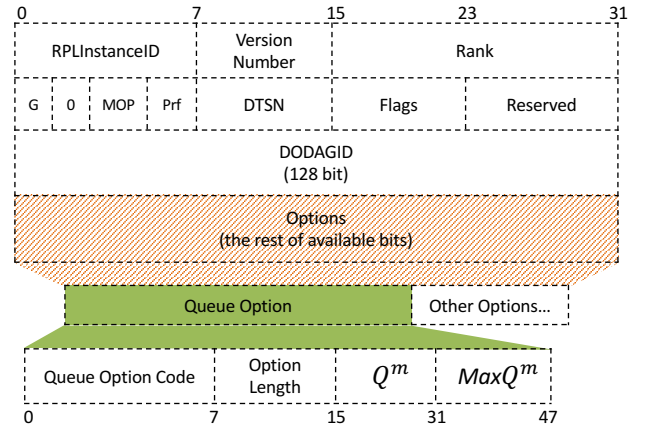
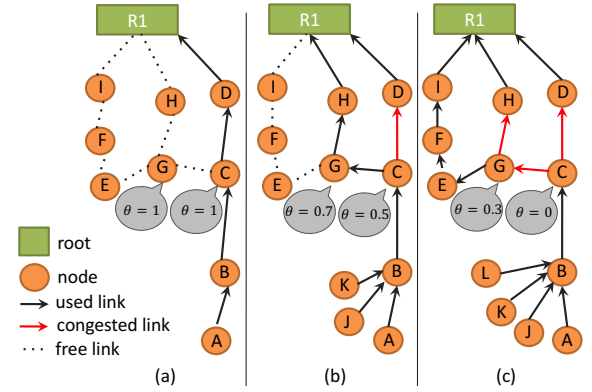

Fig. 5. DIO message structure in BRPL

Fig. 6. An illustration demonstrates how BRPL supports large-scale networks. Assume nodes can concurrently listen to multiple radio channels. In (a), the network is stable and the path B-C-D-R1 handles the data traffic. Both BRPL and RPL produce the same DAG. In (b) and (c), new nodes join the network. BRPL automatically adjusts θ to utilize additional network resources.

slot t via DIO messages. BRPL introduces a new standard ICMP option named `Queue Option` as shown in Fig. 5. The payload of this new option has a length of 4 bytes stored in big endian order.

Maintaining interoperable communications between BRPL and non-BRPL nodes is a key element towards supporting hybrid networks. When a node running RPL processes an incoming DIO message with the Queue Option, it considers the option code to be unknown. The node will then ignore this ICMP option and process the rest of DIO message safely. This ensures a transparent message structure between BRPL and non-BRPL nodes, and allows nodes to seamlessly join hybrid networks without experiencing compatibility-related issues.

4.4 Support for Large-scale Networks

The high adaptability in BRPL also enables deploying large-scale LLNs. When new nodes join the network, they may introduce extra traffic to the network causing BRPL to adjust the θ parameter accordingly. Unlike RPL, when network bottleneck appears in the network, BRPL tries to solve it by allocating more resources for the incoming data traffic. Similarly, when nodes leave the network and traffic level becomes lower, BRPL deallocates resources that are no longer required.

Fig. 7. Node deployment in the FIT IoT-LAB testbed.

TABLE 2
Summary of Evaluation Parameter Settings

Method	Testbed	Simulation	
	IoT-LAB	Factory	QuickTheta
OF	ETX	ETX	Custom
Platform	Contiki/ M3 ARM	Contiki/ Cooja	Contiki/ Cooja
MAC	CSMA/CA+ MiCMAC	CSMA/CA+ MiCMAC	CSMA/CA+ MiCMAC
Mobility	-	Factory Indoor Mobility	-
Transport	UDP/IPv6	UDP/IPv6	UDP/IPv6
Packet Size	160 bytes	160 bytes	160 bytes
Queue Size	150 packets	250 packets	250 packets
Transmission Power	-17 dBm	0 dbm	0 dbm
Transmission Range	-	30 meter	50 meter
Network Scale	100 nodes	130 nodes	100 nodes

Fig. 6 presents an example illustrating how BRPL naturally supports large-scale networks. Let assume that R1, the root of the network, offers multiple communication channels. In (a), the path B-C-D-R1 handles all the data traffic coming from node A. Both RPL and BRPL have the exact same DAG since θ is close 1 when the traffic load is light. In (b), node K and J join the network. Now the path B-C-D-R1 becomes congested and cannot handle all the incoming traffic. Unlike RPL, BRPL observes the traffic congestion in the DAG, and diverts some of the traffic through the path B-C-G-H-R1. In (c), one more node joins the network resulting in further traffic congestion. Again BRPL adjusts the routing and allocates more resources trying to eliminate the traffic congestion. When nodes L, K, J leaves the network, BRPL deallocates the extra resources utilized in (c) and (b) causing the DAG to return back to the DAG showing in (a).

5 EXPERIMENTS

The practical performance of BRPL has been compared to RPL and backpressure routing. This section firstly describes the general configuration settings of our experiments. Then, it shows BRPL performance in static networks using real-life IoT testbed, and mobile networks using a network simulator with a realistic indoor mobility model.

5.1 Implementation and Configuration

BRPL is implemented on top of Contiki OS [17], an open source operating system designed for systems with limited resources including, but not limited to, LLNs and IoT. Out of the box, Contiki provides a C implementation for the IPv6 stack (uIPv6) and RPL. This section summarizes some key modifications that we have done in Contiki before running the experiments.

For the MAC layer, a CSMA/CA driver combining with MiCMAC [28] was used. MiCMAC is mainly employed to perform channel hopping. The maximum number of channels that a node can use is 4 and the maximum channel cycle listening time is 80ms. The neighbor table size is 50 records. The radio duty cycling is disabled during all experiments. The maximum transmission attempts to re-send a packet is set to 5.

In regards to the IPv6 layer, we set the `UIP_CONF_ND6_REACHABLE_TIME` parameter to 5000 and the `UIP_CONF_ND6_MAX_UNICAST_SOLICIT` to 65535

to increase the effectiveness of IPv6 Neighbor Discovery service. The packet reassembly service is disabled and the `UIP_CONF_IGNORE_TTL` is set to zero to ignore the TTL flag in the packet headers. The `HC6` SICSlowpan header compression is used in all our experiments.

For the routing layer, ETX OF has been utilized with the default parameter settings. We also enable the `RPL_MOP_NO_DOWNWARD_ROUTES` option since downward routing is not used in our experiments. Time slot duration is assumed to be one second. `DIO_INTERVAL_MIN` and `DIO_INTERVAL_DOUBLINGS` were adjusted to broadcast routing metadata every 512 to 1024ms period. When the network is highly dynamic (i.e. time-varying traffic and/or topology), the Trickle algorithm pushes the DIO broadcasting rate towards every 512ms. On the other hand, Trickle adjusts the broadcasting rate to 1024ms when weight stability is observed in one-hop neighbors.

The "Queue Manager" component has been implemented, and integrated with the IPv6 stack in Contiki. The Queue Manager uses Last-In-First-Out (LIFO) scheduling. When a node with full data queue receives a data packet, it drops the newly received packet. In addition, to make RPL more fair and reduce its packet loss, data queues has been added to RPL to buffer the incoming packets. However unlike BRPL, data queues in RPL are not considered during routing operations. The queue option code in DIO messages is set as '0xCE', which does not interfere with existing RPL option codes in Contiki.

The rest of BRPL components shown in Fig. 4 has been also implemented successfully in Contiki. In the experiments, BRPL uses the same OF of RPL with the same settings. The neighbor table size of both BRPL and RPL is 50. In addition, we implemented the well-known backpressure routing [24] on top of the IPv6 stack, but without its scheduling policy. All three routing protocols have identical queue settings.

The performance of BRPL, RPL and backpressure routing has been examined in static and mobile networks. The length of each experiment is 4 hours. The application layer generates UDP packets on a regular basis and passes them to the IPv6 layer. The size of a data packet is 160 bytes (8 bytes for payload, the rest is used for IPv6 header). Table 2

(a) packet loss.
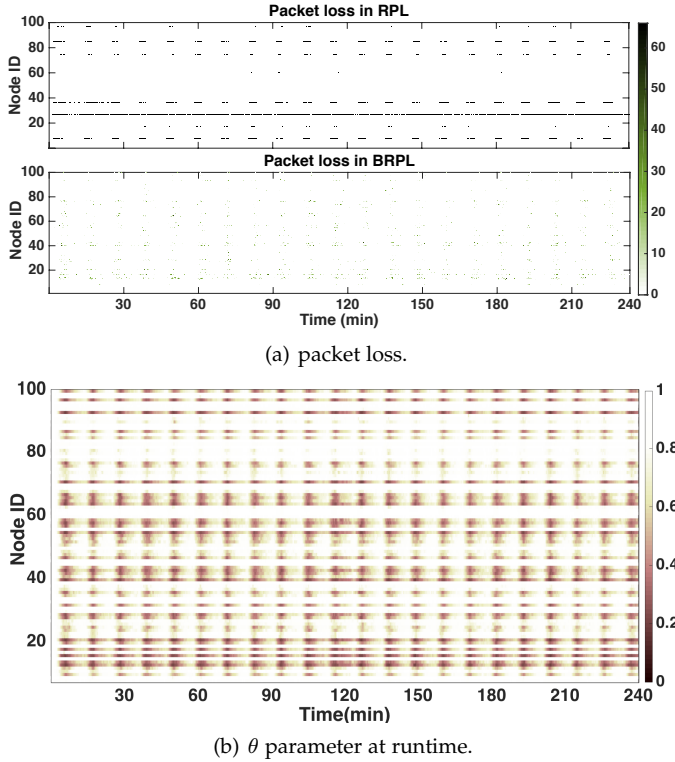

(b) θ parameter at runtime.

Fig. 8. Testbed results when the network has dynamic data traffic. Every 10 minutes, the application layer increases its sensing rate to generate a traffic burst last 3 minutes. (a) shows the packet loss for BRPL and RPL. (b) shows the θ parameter and its adaptivity at runtime.

highlights key settings in our experiments.

5.2 Methodology

Both testbed experiments and simulations are employed to evaluate the performance of BRPL.

5.2.1 Testbed Experiments.

We used 100 nodes (M3 Open Nodes) from the Grenoble site offered by the FIT IoT-LAB testbed [18], shown in Fig. 7. The network had 5 roots and 95 sensor nodes generating UDP packets on predefined time interval. Each M3 open node has a ARM Cortex M3 micro-controller, a 64 kB RAM, a IEEE 802.15.4 radio AT86RF231, several types of sensors, and a rechargeable 3.7 V LiPo battery. To ensure multi-hop mash topology is formed, we set transmission power to -17 dBm. The $MaxQ_x^m$ is set to 150 for all nodes.

5.2.2 Cooja Simulations based on Cloud.

To evaluate how BRPL can extends RPL to support mobile IoT scenarios, we used Cooja - the network simulator of Contiki. We set the $MaxQ_x^m$ to 250 packets. The transmission range is adjusted to 30m to simulate indoor radio limitations. We noticed that Cooja has a poor performance in a 130-node network. To speed up, we run all our simulations on a 18-server cluster provided by the Imperial College London's private Cloud. Each server runs 14.04 Ubuntu LTS with 4 GB of RAM. Git and Puppet, an open-source configuration management tool, are utilized in order to run multiple simulations in parallel with different configuration settings.

5.3 Testbed Experiments

Three sets of testbed experiments were constructed to compare practical performance of BRPL, RPL and backpressure routing based on a 100-node network in the FIT IoT-LAB testbed with settings described in Section 5.1.

5.3.1 Adaptivity to Dynamic Traffic Load

This set of experiments examines the performance of BRPL and RPL in a network with time-varying traffic loads. A simple application has been developed to generate data packets at rate 1 Packet Per Second (PPS). The application has an internal timer that is triggered every 10 minutes to change the packet rate to 4 PPS for three minutes. This simulates a traffic burst condition in the network. Fig. 8(a) shows packet loss for BRPL and RPL as a function of time. RPL drops around 154K packets mainly when the network encounters a traffic burst. On the other hand, BRPL adapts to the traffic dynamics, thanks to the QuickTheta algorithm, and utilizes resources to handle the traffic burst. Fig. 8(b) shows how the QuickTheta algorithm adjusts the θ parameter at runtime according to the traffic congestion levels. This runtime adaptivity results in BRPL having more than 4.5 times less packet loss than RPL.

5.3.2 Throughput Study

In this set of experiments, the data sensing rate is fixed over time. Fig. 9(a) shows the packet loss BRPL, RPL and backpressure routing under different data rates. RPL has the highest packet loss as it tends to always choose the optimal OF path. To enhance the throughput of the network, BRPL utilizes suboptimal paths when the network has high data traffic (when the data rate is 4, 3, or 2 PPS). This results in around 100% reduction in the packet loss which is accommodated with higher end-to-end packet delay and communication overhead[3], as seen in Fig. 9(b) and 9(c) respectively. It is important to note that both backpressure routing and BRPL have similar performance in terms of throughput. However, when the traffic load is light (e.g. 1 PPS), backpressure routing suffers from large high end-to-end packet delays. This is because backpressure routing relies on congestion gradients to route the packets. The data traffic in some settings was not enough to establish stable congestion gradients. This results in network frequently experiencing routing loops. BRPL, on the other hand, relies on the OF to route the packets in this case, as the θ in this case is close to 1. Data is forwarded via the optimal paths defined by the OF. This also explains why the performance of BRPL and RPL is very similar when the data rate is 1 PPS.

5.3.3 Support for Hybrid Networks.

a multi-hop network where are all the nodes running RPL has been constructed. The data rate in this experiment is set to 4 PPS to simulate that the network is facing relatively high traffic load. Then we randomly replaced 20 nodes of RPL with BRPL nodes. The results were gathered, and then the same process was repeated until all the nodes run BRPL.

---

3. Please note that communication overhead reflects not only network congestion levels, but also the energy consumption of nodes. This is because it is well recognized that packet transmissions are the major energy consumer of routing protocols [29], [30].

(a) packet loss.   (b) end-to-end delay.   (c) communication overhead (in 10^6 packets).
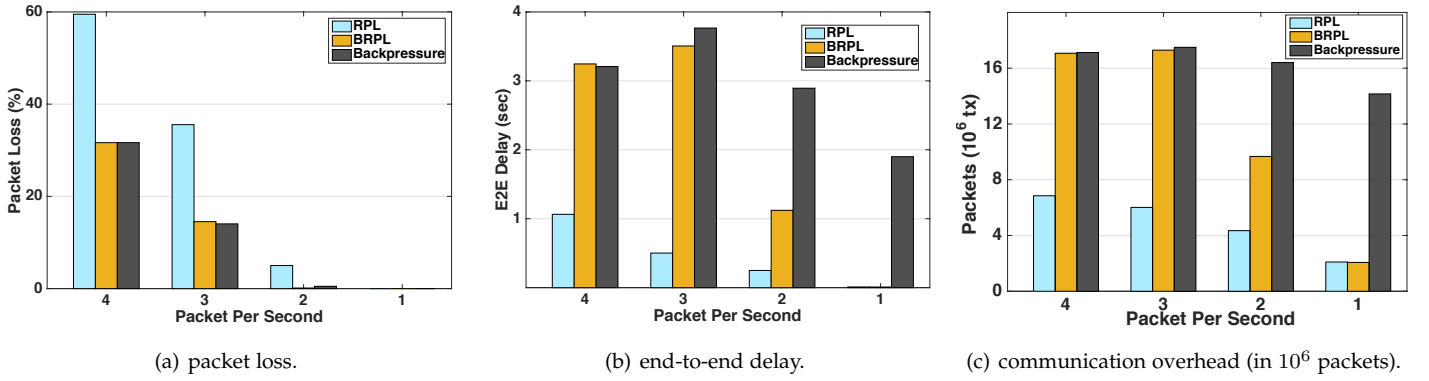
Fig. 9. The performance of BRPL, RPL and backpressure under different traffic loads in the testbed. RPL has the highest packet loss. BRPL and backpressure routing uses suboptimal paths when the traffic load is high (i.e when data rate is 4, 3 or 2 PPS) which results in a significant reduction in packet loss with an increase in end-to-end packet delay and communication overhead. When the traffic load is light (e.g. 1 PPS), both BRPL and RPL have similar performance, whereas backpressure routing still suffers from large delay and communication overhead.

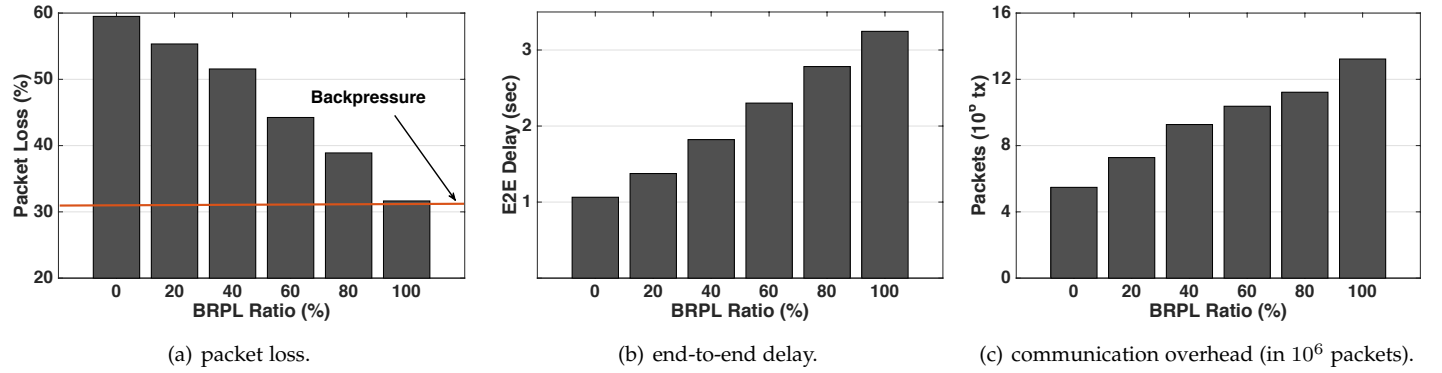(a) packet loss.   (b) end-to-end delay.   (c) communication overhead (in 10^6 packets).

Fig. 10. The performance of a hybrid network with various BRPL node deployments. The data rate is fixed to 4 PPS. As seen, the more BRPL nodes deploy in the network, the lower packet loss is expected. This is accommodated with higher delay and communication overhead as BRPL nodes may use suboptimal OF paths to improve the throughput.

From these experiments, the following conclusion can be realize: first, BRPL is backward compatible with RPL. Nodes running RPL or BRPL can operate together in the same hybrid network. Interoperability for data and control messages between RPL and BRPL nodes is successfully verified. Second, the more BRPL nodes deployed in the network, the less packet loss is observed as shown in Fig. 10(a). Because the traffic level of the network in the experiments is considerably high, BRPL nodes tends to utilizes all the possible network capacity to increase the throughput and reduce the packet loss as much as possible. This is accommodated with an increase in average end-to-end delay and communication overhead as shown in Fig. 10(b) and Fig. 10(c) respectively.

5.4 Cooja-based Simulations

Cooja network simulator is used to provide more insights about the performance of BRPL under other network conditions. These simulations firstly show BRPL performance for a network following an indoor mobility model. Then, we examine how QuickTheta algorithm achieves the practical balance between throughput and minimizing OF.

5.4.1 Study of Mobile Networks

The performance of BRPL has been evaluated in networks where mobile devices exist. We managed to obtain a 500mx250m layout for a poultry processing factory from [31], shown in Fig. 11. The transmission range is adjusted

to 30m to simulate indoor radio limitation. 130 nodes has been semi-randomly placed, generating UDP packets which are required to be collected by the mobile roots. From the layout, an *indoor mobility graph* has been created as shown in Fig. 12. The graph defines all possible destinations and paths that mobile roots can take while traveling within the factory. The doors are assumed to be always open during the simulations. A custom Cooja plugin has been developed to import the mobility graph to Cooja. In the simulations, each root has 9 preferred destinations (randomly selected during the bootstrap phase). A root travels to one of its preferred destinations 90% of the time. For the other 10%, it randomly chooses a destination from the indoor mobility graph. When a root reaches its destination, it waits for one second, selects a new destination, and the process repeats itself. The travel speed of all roots is set to 1.4 m/s with 0.2 variance to simulate average man walking speed.

Fig. 13(a) shows the packets loss for BRPL, RPL, and backpressure routing under various network and traffic conditions. As seen, the packet loss of BRPL is between 1.2 and 20 times less than RPL. RPL relies on the ETX OF alone, commonly known to be unsuitable for mobile networks. Outdated DAGs and link breakage were the main reasons for losing 70% to 80% of the data packets in RPL. BRPL and backpressure routing have relatively close performance (usually within $\leq 2\%$ differences), and thanks to adaptive routing operations they utilizes mobile resources more effec-
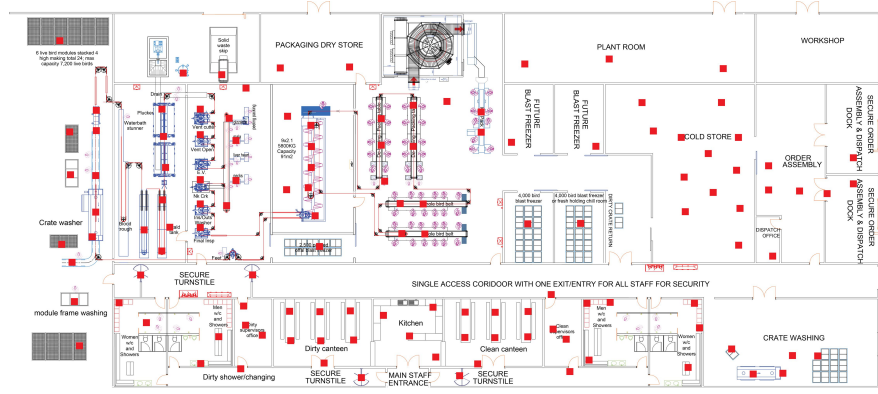
Fig. 11. A poultry processing factory layout obtained from [31]. The red squares denote the locations of the deployed sensor nodes.
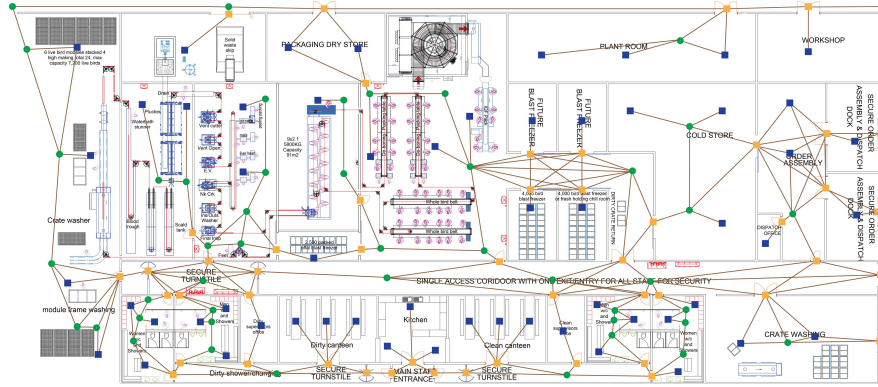

Fig. 12. The *indoor mobility graph* constructed from the factory layout in Fig. 11. The dark blue squares are *possible end destinations*. The yellow circles are *doors*. The green squares are *path points*, which define the mobility paths roots can take.

tively than RPL. As observed in Fig. 13(a), the more mobile roots are available, the less packet loss is expected from the network running BRPL or backpressure routing.

Fig. 13(b) presents the average end-to-end delays for the delivered data packets. RPL has usually worst performance than the other two. The outdated DAGs causes nodes to send packets to a destination where the roots no longer exist and the ETX OF does not cope well with utilizing opportunistic resources. BRPL has better performance than backpressure routing as it tends to, especially under light traffic loads, to utilizing OF reduces routing loops.

The communication overhead for the three routing schemes are similar as seen in Fig. 13(c). The Trickle algorithm and the IPv6 Neighbor Discovery Service causes the mobile roots to continuously transmit DIO and IPv6 ND control messages. These messages are the main reasons for the large communication overhead.

5.4.2 Study of Adaptive Balancing in QuickTheta

The purpose of this set of simulations is to show QuickTheta performance at runtime. We want to demonstrate how QuickTheta can effectively find the best possible balance between maximizing throughput and minimizing RPL OF. To do so, the performance of QuickTheta is compared to the drift-plus-penalty routing technique [32] under various V settings.

In order to easily visualize the dynamics of the θ parameter, a simple grid topology is used. The topology has 100 nodes with spacing of 30 meters between adjacent nodes. Radio transmission range is set to 50 meters. The network

has 98 sensor nodes and two roots (root1 and root2). Root1 is set to be plugged in a power line with unlimited amount of energy. Root2, on the other hand, runs on battery with a finite amount of energy. Root1 and root2 are respectively deployed in the top left corner (grid cell (0,0)) and the bottom right (grid cell (10,10)). The network uses a custom OF such that it is designed to reduce maintenance cost, and therefore always prefer to use root1 for the data traffic. Based on the implementation of the OF, root2 will be used only when root1 is not available.

Fig. 14(a) shows the θ parameter for the network under two different traffic loads. Depending on the traffic congestion levels, the QuickTheta algorithm tends to converge the θ parameter into a relatively fixed range. For example, in Fig. 14(a) when the data rate is 2 packet per second, θ for the most nodes tends to be in the range [0.25,0.35]. However, the nodes, which are closer to the roots, will have a higher θ (closer to 1) since the traffic levels are less in these spots than the rest of the network.

The drift-plus-penalty technique proposed in [32] has been implemented to provide another approach to combine RPL OF with backpressure routing. However as going to be discussed in Section 6, the key challenge in utilizing this technique is that it requires manual tuning for the V parameter, which is the tradeoff parameter between throughput and achieving the optimal solution for the penalty function (i.e. RPL OF in our case). Finding the optimal value for the V parameter is difficult, especially in multi-user IoT systems. V tightly depends on the expected traffic levels from the application layer. The traffic levels in IoT can be highly

(a) packet loss.

(b) end-to-end delay.
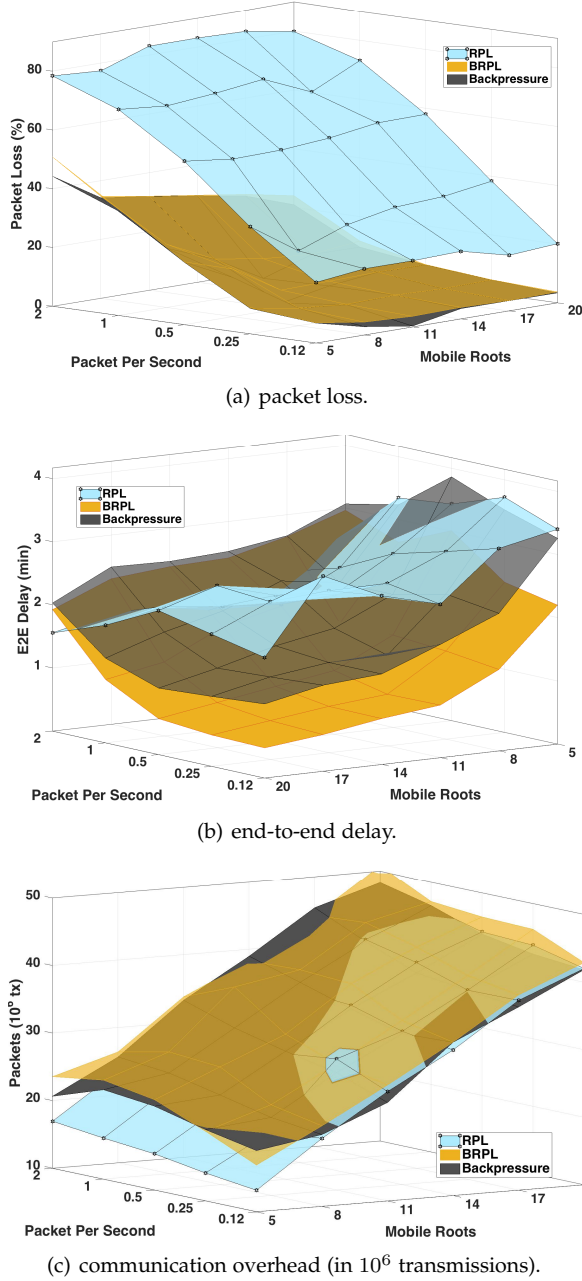

(c) communication overhead (in 10^6 transmissions).

Fig. 13. The performance of BRPL, RPL and backpressure routing in a mobile network simulated based on the layouts in Fig. 11 and 12. BRPL outperforms RPL in terms of packet loss and average end-to-end delay.

dynamic and time-varying. For this set of simulations, we manually tested the network under various V values.

Fig. 14(b) shows the performance BRPL, RPL, backpressure and drift-plus-penalty technique with different V settings. Here the data rate is fixed to 1 packet per second. Backpressure routing provides a costly allocation as 41% of data packets are forwarded to root2 (i.e., 41% of the data is gathered using the stored energy from root2's batteries). This is expensive in terms of maintenance costs. RPL, on the other, always tends to forward all the data packets to root1 using OF optimal paths, which results in 45% packet loss. BRPL first tries to greedily forward data to root1. However as the θ parameter gradually increases because of network traffic congestion, it utilizes root2 and other suboptimal paths to upload the rest of the packets. The

result is root1 gets 78.6% of the data packets and root2 gets 19% of the data packets. From Fig. 14(b), it is clear that BRPL performance is very similar to the drift-plus-penalty technique when V is close to 5. If V is below 5, the allocation is not as optimal since more data packets are forwarded via energy stored in root2's batteries. If V is above 5, a higher packet loss then will be observed. A similar story can be seen when the data rate is 2 PPS. As shown in Fig. 14(c), BRPL sends 42% and 49% of data packets to root1 and root2 respectively. A similar performance can be found when V equalizes to a certain point in the range (3,4). Increasing V beyond that point will causes packet loss as the routing scheme aggressively minimizes the OF. Decreasing V from this range will cause more data forwarding towards root2, which is not optimal either. Therefore, QuickTheta in both cases managed to find the best practically possible tradeoff between throughput and minimizing the OF.

6 RELATED WORK

6.1 RPL

The proposed extension in [25] addresses the data reliability problem in RPL. Similarly, ORPL [33] incorporates opportunistic routing with RPL to achieve low latency, robustness, and good scalability. On the other hand, [34] provides a modified MAC layer for supporting multipath forwarding. Kalman positioning and the Corona mechanism have been used in KP-RPL[35] and Co-RPL [36], respectively, to address the issue of mobility support in RPL. BRPL relies on the smart and smooth switching between RPL and backpressure routing to enhance mobility support. This results in a lightweight solution that does not require any assumption or knowledge about the underline mobility pattern of the network. In addition, this paper presents a modular framework. The implementation of QuickBeta can easily be adjusted to incorporate other mobility metrics, including Kalman-based and Corona-based metrics. Enhancing RPL in terms of mobility, throughput and traffic adaptability all together has not yet been examined by the current literature.

We consider QU-RPL, recently proposed in [37], to be the closest to our work. The authors here combine queue information with the OF0 to improve the *load-balancing* of RPL routing in heavy-traffic networks. BRPL also exploits queue backlogs during routing decision making. However, BRPL does not rely on the *propagation* of queue metadata in multi-hop paths. Instead, routing decisions in BRPL are performed based on *pure hop-by-hop* queue information, similar to backpressure routing. Clearly, this makes BRPL a more dynamic routing solution, applicable not only for heavy-traffic networks, but also for networks with highly time-varying traffic loads and topologies, such as networks with node mobility which are sadly ignored in QU-RPL. In addition, BRPL eliminates the need for having a naive *propagation reduction factor*, λ in QU-RPL. BRPL also offers an attractive solution to adaptively tune the sensitivity of queues. Manually setting these parameters as suggested in QU-RPL is challenging in dynamic, large-scale IoT systems. Furthermore unlike QU-RPL, BRPL is not limited to OF0. In fact, any RPL OF can be used with BRPL since data queues are completely decoupled from OFs while both are still strongly normalized. This is vital to truly improving the performance of MTR deployments.
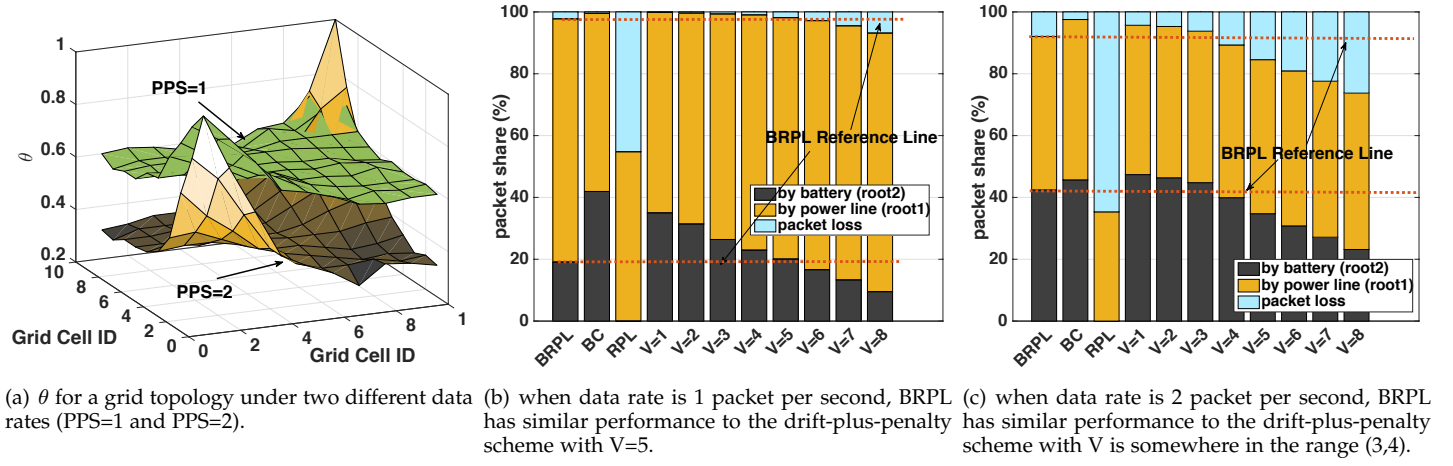
(a) θ for a grid topology under two different data rates (PPS=1 and PPS=2).

(b) when data rate is 1 packet per second, BRPL has similar performance to the drift-plus-penalty scheme with V=5.

(c) when data rate is 2 packet per second, BRPL has similar performance to the drift-plus-penalty scheme with V is somewhere in the range (3,4).

Fig. 14. The performance of for BRPL, backpressure routing, RPL, and drift-plus-penalty scheme with various V settings in a grid topology. The network has two roots deployed at the corners. (a) shows the average θ values for the nodes under two different data rates. (b) and (c) present the observed packet loss and the amount of data packets forwarded to each root.

6.2 Backpressure Routing

The proposed solution in this work is heavily inspired by the elegant *backpressure* routing [24]. Introducing the drift-plus-penalty technique [32] and combining it with [38], [39], [40] contributions in utility optimal networking provided a theoretical framework for backpressure-based stochastic optimization. This framework has been used in a wide range of applications including power control [41], [42], selfish data relays [23], sensor networks [9], and mobile networks [43], [44]. However, utilizing this framework directly in RPL for dynamic IoT systems is not practical. The framework has parameter tuning issues. The V parameter sets the tradeoff between queue backlogs and *penalty/objective* function optimization. Users are required to set the V parameter based on the expected traffic level from the application layer. This is challenging because the routing layer must have certain assumptions about OFs and expected data traffic, which is technically difficult, or even impossible, in LLNs with event-driven applications. A network may need to serve multiple concurrent users with heterogeneous time-varying bandwidth demands. This work presents QuickTheta as a practical solution to this problem.

Many efforts have been made to make backpressure practical. Various techniques have been proposed to improve the performance of backpressure in terms of the reduction of memory overhead [45], [46] and delay [47], [9]. However, to our knowledge this is the first work that utilizes backpressure routing in hybrid networks where some nodes use backpressure routing while others do not.

7 CONCLUSION

This work addresses three key limitations of RPL: low throughput, poor adaptability to time-varying data traffic loads and lack of support for node mobility. To this end, we develop BRPL, a backward-compatible extension for RPL, which can adaptively and smoothly switch between RPL and backpressure routing depending on network conditions. We present *QuickBeta* and *QuickTheta*, two adaptive online algorithms for BRPL, to respectively support node mobility and balance the tradeoff between network throughput and RPL OF minimization. Through extensive experiments driven by real-world testbed and cloud-based simulations, we show that BRPL works seamlessly with RPL and achieves significant performance improvements in terms of network throughput with 60% packet loss reduction at a minimum in mobile networks. An interesting future direction is to study resource fairness issues among multiple coexisting DAGs in RPL and BRPL.
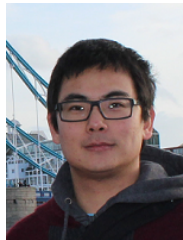
ACKNOWLEDGMENT

REFERENCES

[1] T. Winter and et al, "RPL: IPv6 Routing protocol for Low-Power and Lossy Networks," *IETF RFC 6550*, 2012.

[2] A. Brandt and J. Buron, "Home Automation Routing Requirements in Low-Power and Lossy Networks," *IETF RFC 5826*, 2010.

[3] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," *IETF RFC 5673*, 2009.

[4] M. Dohler, D. Barthel, T. Watteyne, and T. Winter, "Routing Requirements for Urban Low-Power and Lossy Networks," *IETF RFC 5548*, 2009.

[5] J. Martocci, P. Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks," *IETF RFC 5867*, 2010.

[6] G. Iyer, P. Agrawal, E. Monnerie, and R. S. Cardozo, "Performance analysis of wireless mesh routing protocols for smart utility networks," in *IEEE SmartGridComm*, 2011, pp. 114–119.

[7] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart Grid technologies: communication technologies and standards," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 529–539, 2011.

[8] P. T. A. Quang and D.-S. Kim, "Throughput-aware routing for industrial sensor networks: Application to ISA100. 11a," *IEEE Trans. Ind. Inform.*, vol. 10, no. 1, pp. 351–363, 2014.

[9] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proc. Int. Conf. Inf. Process. Sens. Netw. (IPSN)*, 2010, pp. 279–290.

[10] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[11] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: a survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1369–1390, 2014.

[12] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, 2013.

[13] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Distributed mobile sink routing for wireless sensor networks: a survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 877–897, 2014.

[14] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, no. 1, p. 7, 2011.

[15] D. Medhi and K. Ramasamy, *Network routing: algorithms, protocols, and architectures*. Morgan Kaufmann, 2010.

[16] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[17] [Online]. Available: http://www.contiki-os.org

[18] [Online]. Available: https://www.iot-lab.info

[19] http://www.openautomation.net/uploadsproductos /micaz_datasheet.pdf.

[20] A. Sridharan and B. Krishnamachari, "Explicit and precise rate control for wireless sensor networks," in *Proc. ACM SenSys*, 2009, pp. 29–42.

[21] T. C. Hu, "Multi-commodity network flows," *Operations research*, vol. 11, no. 3, pp. 344–360, 1963.

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[23] S. Yang, U. Adeel, and J. A. McCann, "Selfish mules: Social profit maximization in sparse sensornets using rationally-selfish human relays," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 6, pp. 1124–1134, 2013.

[24] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[25] E. Ancillotti, R. Bruno, and M. Conti, "Reliable data delivery with the ietf routing protocol for low-power and lossy networks," *IEEE Trans. Ind. Inform.*, vol. 10, no. 3, pp. 1864–1877, 2014.

[26] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *Proc. ACM MobiHoc*, 2008, pp. 231–240.

[27] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *Proc. ACM SenSys*, 2011, pp. 82–95.

[28] "MiCMAC, a Channel-Hopping extension of ContikiMAC," *Available at http://bit.ly/1OfP7jn*.

[29] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Commun. Survey Tuts.*, vol. 15, no. 2, pp. 551–591, 2013.

[30] S. Yang, Y. Tahir, P.-y. Chen, A. Marshall, and J. McCann, "Distributed optimization in energy harvesting sensor networks with dynamic in-network data processing," in *Pro. IEEE Infocom*, 2016, pp. 1–9.

[31] "Poultry processing equipment.com," *Available at http://www.poultryprocessingequipment.com*.

[32] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[33] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: Scalable opportunistic routing with orpl," in *Proc. ACM Conf. Embedded Netw. Sens. Syst.*, 2013, pp. 1–14.

[34] B. Pavković, F. Theoleyre, and A. Duda, "Multipath opportunistic rpl routing over IEEE 802.15.4," in *Proc. ACM MSWiM*. ACM, 2011, pp. 179–186.

[35] M. Barcelo, A. Correa, J. Vicario, A. Morell, and X. Vilajosana, "Addressing mobility in rpl with position assisted metrics," *IEEE Sensors J.*

[36] O. Gaddour, A. Koubâa, R. Rangarajan, O. Cheikhrouhou, E. Tovar, and M. Abid, "Co-rpl: Rpl routing for mobile low power wireless sensor networks using corona mechanism," in *IEEE SIES*. IEEE, 2014, pp. 200–209.

[37] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load balancing under heavy traffic in rpl routing protocol for low power and lossy networks," *IEEE Trans. Mobile Comput.*, 2017.

[38] M. J. Neely, "Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks," *IEEE Trans. Autom. Control*, vol. 54, no. 3, pp. 565–579, 2009.

[39] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, 2008.

[40] M. J. Neely, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 5, pp. 1188–1199.

[41] S. Yang, X. Yang, J. A. McCann, T. Zhang, G. Liu, and Z. Liu, "Distributed networking in autonomic solar powered wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 750–761, 2013.

[42] C.-p. Li and M. J. Neely, "Energy-optimal scheduling with dynamic channel acquisition in wireless downlinks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 4, pp. 527–539, 2010.

[43] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, 2005.

[44] S. Yang, U. Adeel, and J. A. McCann, "Backpressure meets taxes: Faithful data collection in stochastic mobile phone sensing systems," in *Proc. IEEE INFOCOM*, 2015.

[45] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. IEEE INFOCOM*, 2009, pp. 2936–2940.

[46] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 244–257, 2013.

[47] B. Ji, C. Joo, and N. B. Shroff, "Throughput-optimal scheduling in multihop wireless networks without per-flow information," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 634–647, 2013.

Yad Tahir received the B.Sc. degree with Honors in Computer Science from University of Sulaimani, Iraq, in 2008, and the M.Sc in Software Engineering with Distinction from Heriot-Watt University, UK, in 2010. He has recently finished his Ph.D in Computing at Imperial College London, UK. His research interests include resource management, network control and optimizations, Internet of things, software and system engineering.

Shusen Yang received his PhD in Computing from Imperial College London in 2014. He is currently a professor in the Institute of Information and System Science at Xi'an Jiaotong University (XJTU). Before joining XJTU, Shusen worked as a Lecturer (Assistant Professor) at University of Liverpool from 2015 to 2016, and a Research Associate at Intel Collaborative Research Institute ICRI from 2013 to 2014. His research interests include mobile networks, networks with human in the loop, and data-driven networked systems. Shusen achieves "1000 Young Talents Program" award, and holds an honorary research fellow at Imperial College London. Shusen is a senior member of IEEE and a member of ACM.

Julie A. McCann is a Professor in Computer Systems at Imperial College. Her research centers on highly decentralized and scalable algorithms for spatial computing systems e.g. sensor networks. She leads both the Adaptive Embedded Systems Engineering Research Group and the Intel Collaborative Research Institute for Sustainable Cities, and is working with NEC and others on substantive smart city projects. She has received significant funding though bodies such as the UK's EPSRC, TSB and NERC as well as various international funds, and is an elected peer for the EPSRC. She has actively served on, and chaired, many conference committees and is currently Associative Editor for the ACM Transactions on Autonomous and Adaptive Systems. She is a Fellow of the BCS.