Least Path Interference Beaconing Protocol (LIBP): A Frugal Routing Protocol for The Internet-of-Things

Lutando Ngqakaza[1] and Antoine Bagula[2]

[1] ISAT Laboratory, Department of Computer Science, University of Cape Town, Private Bag X3 Rondebosch, Cape Town, South Africa
[2] ISAT Laboratory, Department of Computer Science, University of Western Cape, Private Bag X17, Bellville 7535, Cape Town, South Africa
ngqlut003@myuct.ac.za, bbagula@uwc.ac.za

Abstract. This paper presents a frugal protocol for sensor readings dissemination in the Internet-of-Things (IoT). The protocol called Least Path Interference Beaconing (LIBP) is based on a lightweight path selection model that builds a routing spanning tree rooted at the sink node based on information disseminated through a periodic beaconing process. LIBPs frugality results from a routing process where the sensor nodes select the least path interfering parents on the routing spanning tree with the expectation of flow balancing the traffic routed from nodes to the sink of a sensor network. The simulation results produced by Cooja under the Contiki operating system are in agreement with previous results obtained under the TinyOS operating system. They reveal that LIBP outperforms different versions of the RPL protocol and the CTP protocol in terms of power consumption, scalability, throughput and recovery from failure as well as its frugality as a routing protocol.

1 Introduction

A new form of modern communication is emerging where sensing, identification and many other types of processing devices are combined with the objective of interacting pervasively with the physical world to provide to different users various services. It is predicted that these devices will be deployed in our daily living environment in thousands of heterogeneous computing elements building multi-technology and multi-protocol platforms that provide access to the information not only *"anytime"* and *"anywhere"*, but also using *"anything"* in a first-mile of the Internet referred to as the *"Internet-of-the- Things" (IoT)* [1]. The next generation IoT infrastructure is expected to include millions of interconnected islands of sensing/identification networks spread around the world to provide services that would not be possible to provide with current generation sensor networks. Such network islands will be using multi-hop routing to avoid the need for the high communication power that might be required from the lightweight IoT devices for communication with each other directly. They will be operating on either an m-to-1 or an m-to-n routing model where where all the nodes will be collecting from their environments sensor readings carrying the information to be sent to either a unique sink node (m-to-1 mono-sink architecture) or multiple sinks (m-to-n multi-sink architecture).

A. Mellouk et al. (Eds.): WWIC 2014, LNCS 8458, pp. 148–161, 2014.

1.1 Routing over the Lightweight IoT Devices

The routing of sensor readings in IoT settings can be formulated as a problem of finding a set of paths for routing the traffic flows carrying these readings from their points of collection to sink nodes which are tasked to deliver these readings to gateways for further processing. When applied to a mono-sink architecture, the traffic packets carrying the sensor readings are routed from nodes to neighbours along the path to the unique sink node following a multi-hop process usually aimed at reducing the energy that each node would spend if it had to send its data traffic directly to the sink. The process can be constrained by spatio-temporal and different other constraints depending on the IoT settings and the application. The solutions to the routing problem above may differ but are usually expected to be self-organized, self-repairing and frugal routing protocols in terms of storage, processing and communication requirements on the lightweight devices that are used in IoT deployments. In a typical mono-sink IoT deployment, the information carried by the sensor readings would typically be aggregated from the nodes towards a unique sink that forms the root node of a tree which is connected to the gateway by the sink with most of the leaf nodes present in the network sending their sensor readings upwards towards the root/sink node for storage, analysis or further processing.

1.2 Contribution and Outline

The LIBP protocol [2, 3] was previously implemented for TinyOS using the Tossim emulator [4]. This paper presents a Contiki [5] implementation of LIBP and evaluates its performance compared to CTP [6] and different versions of the RPL protocol [7] with the objective of assessing the frugality of LIBP and its efficiency compared to these two other routing protocols. While the LIBP implementation presented in this paper has been implemented from scratch following the model proposed in [2], the RPL and CTP implementations considered in this paper are widely available in open-source format on a wide variety of platforms. They did not require any new implementation in the platform of choice for this paper. The remainder of this paper is organized as follows: Section 2 presents the proposed LIB protocol while 3 describes related routing protocols used in IoT settings. The results obtained through comparative simulation study are presented in Section 4, and finally Section 5 draws the conclusions and provide avenues for future work.

2 Least Path Interference Beaconing Protocol (LIBP)

2.1 Protocol Description

LIBP [2, 3] is an implementation of the LIBA algorithm. This routing protocol, like CTP, uses a beaconing process initiated by the source (sink) node. When the process is initiated nodes incident to the sink node will be the first to recognize that a sink node is within one hop distance. This process is then initiated by these nodes to their neighbours and this process is repeated thereafter. This results in a network where each node is aware of its neighbours. The least interference paradigm is integrated into the process

by which nodes select parent nodes which have the smallest number of (supporting) children, which is the parent of least traffic flow interference. This configuration is especially powerful in the situation where sensors are periodically sensing information (which is a very popular sensor use case). LIBP basically aims to provide a way to balance traffic flow in such a way that it results in energy efficiency by having a network where nodes support less traffic. The network building process is highly detailed in the paper by Bagula et al [2].

2.2 LIBP Implementation

RPL and CTP are already implemented in ContikiOS [5], however LIBP is not, this resulted in having LIBP implemented for Contiki. Following the successful methodology of adapting CTP to conform to the LIBP model and ideas [2], this approach was used to preserve the same interfaces that CTP has implemented with the simulation environment (Cooja). At a very high level the link-estimate module for CTP found in Contikis network library was modified to conform to LIBP ideas. This means that the ETX link metric was altered to rather conform to interference represented by the amount of supporting children nodes. Features not required for LIBP were removed (trickle algorithm code for example). It should be noted that since LIBP in its Contiki implementation is forked from CTPs implementation in Contiki, it inherited the same underlying communications stack, Rime [8].

2.3 LIBP Network Building Process

The LIBP network building owes its power to simplicity that builds upon an ad-hoc routing protocol that is also structurally similar to RPL in structure. LIBP uses two control plane messages for network configuration, one being the beacon message, and the other is the acknowledgement (ACK). In the scenario where the network is initialized, the root node will broadcast a beacon at a given interval where the beacon includes important routing information regarding the senders identity and weight. Once the root node advertises the beacon, nodes within the immediate vicinity of the root would have received the beacon. The root node advertises a weight of 0 which prompts the nodes within its vicinity to use that node as a parent. The parent is alerted to the new nodes dependence by the acknowledgement packet. When a node sends an acknowledgement packet to a parent then that parent must increase its weight since that parent is supporting an extra node.

2.4 LIBP Maintenance and Recovery

From the network configuration stage of LIBP (shortly after network epoch) each node keeps a linked list of neighbouring nodes. This list holds an object which characterizes the neighbouring nodes address and its weight (interference) along with its route metric.

Maintenance - Since each node accounts for each of its neighbours in a linked list, it is then possible for nodes to perform rudimentary operations for local network maintenance, and in the event of parent failure, network repair is achievable. The age attribute is there to keep track how long that particular LIBP Neighbour has been in the

list, whenever the LIBPNeighbour linked list is updated then the age attribute is incremented. The route metric attribute describes the precedence in which nodes are tiered by how far they appear to be from the root node, nodes with a low route metric are closer to the sink node. RPL uses a similar metric which can be described as node depth [7].

Recovery - When a node is compromised in such a way that its ability to communicate is impaired then recovery is required. Such a node would have to be removed from the network as a whole. This usually happens when a particular node is unable to acknowledge sent data messages, the main event which alludes to this conclusion is that a node would have retransmitted the same packet for an amount that is equal to the programmed max retransmits. If this happens then the compromised node is removed from the sending nodes LIBP Neighbour list. This in effect removes the parent of the sending node, which requires the sending node to pick a new parent.

3 Related Routing Protocols: RPL and CTP

3.1 Collection Tree Protocol (CTP)

CTP [6] is a routing protocol which extends the Trickle algorithm [9]. It does so because the assumption can be made that data aggregation is one of the primary goals of a WSN. CTP promises to be reliable, efficient, robust, and hardware independent. CTP relies on data packets to validate the routing topology and loop detection. This routing protocol also utilizes adaptive beaconing (an application of Trickle) to dynamically setup and adapt to network changes. Every node implementing CTP maintains an estimate of the cost of its route to a collection point (namely, the sink node). This metric is typically called expected transmissions (ETX).

CTP Network Building Process. CTP (and RPL) employ a similar strategy for network construction. CTP extends the use of the trickle algorithm [9] by sending out control messages at a rate which is dependent on how dynamic the network is. In summary when the routing is empty (the network has just been deployed), A set number of nodes in a network advertise themselves as network roots. Thereafter, nodes form a set of routing trees to these roots. In CTP each node selects one parent as a next-hop link and that parent is closer to the root node than the node is.

CTP Maintenance and Recovery. CTPs strength lies in the fact that its network maintenance is implied by its adaptive control messaging implementation.

Maintenance - The adapted trickle algorithm used in CTP also counts for the handling of network inconsistencies. These inconsistencies include node addition, the significant change in link ETX and loop avoidance. The adapted trickle algorithm counts for the ability for CTP to maintain the network. Even if a network is heavily degraded, due to the adapted trickle algorithm, the network should relax to a near-optimum state.

Recovery - CTP employs a simple strategy for detecting node failure. In the case of node failure, all nodes which are dependent on the failed node will find another parent (usually the next best local parent). Node failure is usually recognized when a node cannot unicast a message to its parent, this is when the node uses up all its

retransmissions for a given packet. Once node failure is established then a node will do a lookup in its routing table to find the best replacement if possible.

3.2 Routing Protocol for LLNs (RPL)

RPL [7] is a direct result of The Internet Engineering Task Force (IETF) which recognized the need to form a standardized IPv6-based routing solution for LLNs. The IETF formalized a working group specific for this problem called ROLL (Routing over Low power and Lossy). The direct outcome of this workgroup was RPL.

RPL Network Building Process. RPL is a Distance Vector IPv6 routing protocol for LLNs that specifies how to build a Destination Oriented Directed Acyclic Graph (DODAG) using an objective function and a set of metrics and constraints. RPL basically builds a logical communications graph over a physical network that conforms to satisfying a set of objectives and conforms to a set of constraints which can be set by a network administrator. The graph building process is initiated at the root (or sink) node, multiple roots can exist in the same network. The root(s) start advertising the information about the graph using messages outlined in its RFC and other literature [7].

RPL Objective Functions. An objective function (OF) allows for RPL to optimize, constrain, or scale the routing metric or link metric of a path. It is entirely possible to have multiple objective functions operating on the same node or same network. Objective functions allow network administrators to impose a set of rules which affect the traffic flow of the network. For example, on one subsection of a network one could implement a rule that specifies that paths with the best Expected Transmissions (ETX) must be used and that the paths must be non-encrypted, or that paths with lowest latencies must be used while avoiding battery operated nodes.

Objective Function ETX - The ETX Objective function (OF-ETX) [6] is a widely popular link metric in the field of WSN. It is a link metric that in some way encompasses link congestion and link latency. ETX is simply defined as the expected number of transmissions required to successfully transmit and acknowledge a packet on a wireless link. In practical terms the ETXroot = 0 (the root node is not expected to send data packets) and the ETXnode = ETXparent + ETXlinktoparent. The objective for OF-ETX is to (greedily) choose the route with the lowest ETX. It should be noted that OF-ETX is standardized and thus can be considered as a modular addition to RPL.

Objective Function Zero - The Objective function Zero (OF-0) is a relatively new objective function proposed by the IETF. In comparison to ETX, OF-0 is not highly established since ETX is considered a mature link metric in the field of WSNs. The goal of OF-0 is for a node to select a parent in such a way that it provides or contributes good enough connectivity to a specific set of nodes or to a larger routing infrastructure. OF-0 is described as being an OF which guides nodes in their parent selection using a metric called node rank. The rank computation of OF-0 has a set of constraints and norms which can be seen in its RFC [10].

RPL Maintenance and Recovery. RPL tries to limit the control plane traffic in the network to minimize the impact that control plane traffic has on the network. Some protocols use periodic keep alives (often called beacons) [2]. RPL uses a different paradigm when attempting to maintain and recover the network.

Maintenance - Instead of using a periodic keep alive for network node maintenance, RPL uses an adaptive timer mechanism called the trickle timer. This algorithm dictates the sending rate of control messages. In essence the trickle timer treats the network as a distributed system that suffers from a consistency problem. A set of events confirms graph inconsistency, for example if a node detects a loop then the network is considered inconsistent, or when a node joins a network, or when a node leaves a network. The more inconsistencies that are detected the more control messages that are sent in the network. The more consistent the network is then the less control messages that are sent.

Recovery - RPL employs two techniques in order to recover the network from node and link failure. In essence RPL uses both local and global repair to initiate graph recovery. When a link or parent node failure is detected, the child node will quickly find an alternative route that conforms to the rules of the OF upon it. This is local repair, given enough local repairs, the graph may diverge from optimum setup. At this point it may be necessary for the graph to be rebuilt using global repair. Global repair is the rebuilding of the graph as if the network was newly deployed as outlined in the RPL Network Building Process section of this paper. Thus global repair is costly as that imposes a high flow of control traffic in the network.

4 Performance Evaluation

Testing Environment - These experiments will be conducted on the Contiki [5] platform. The mote that will be emulated in Cooja for this experiment will be the Tmote sky mote. In the case that emulation is not required; Cooja motes will be used for simulation. The experiment will be conducted in a simulation environment in which UDGM (Distance Loss) will be the radio medium of choice. RPL and CTP are already implemented in Contiki. LIBP was implemented by forking the CTP code found in Contiki and modifying it in order to meet the LIBP requirements.

Data Collection - Metrics in the experiment were collected by implementing the energest [11](Energy Estimation) module in Contiki, energest is used for obtaining per-component power consumption. This module gives metrics which are related to the amount of power required by certain modes of operation. The metrics that can be obtained from energest is the count of power utilized for radio RX and TX, Low Powered Mode(LPM), and Normal Powered Mode (NPM) also known as awake mode. By using the Tmote sky datasheet. The power utilized is described below.

To calculate the power we need an intermediary function which helps us calculate the power utilized.

$$f(x, y) = ((x \times 64) + (y \times 64)/1000) \tag{1}$$

And to calculate the power utilized given the energest RX TX LPM and APM values we calculate the power.

$$P = 3 \times \frac{APM \times f(1,800) + LPM \times f(0,545) + TX \times f(17,700) + RX \times f(20,0)}{64 \times (APM + LPM) \div 1000} \quad (2)$$

Cooja also has an online data collection application called the shell collect view. The shell collect view gives a comprehensive breakdown of node specific status variables and meta-data. Cooja has another nice feature which comes in a Cooja application called PowerTracker. PowerTracker is an online real-time radio duty cycle monitoring tool. PowerTracker can be used to deduce the amount of time that a node spends in a particular state with regards to its radio.

Testing Variables - RPL will be run as two experiment instances since RPL can be run with various objective functions (OF). As a result RPL will be run with OF-0 and OF-ETX and thus for the rest of the paper RPL will be referred to either RPL-0 or RPL-ETX to refer to RPL coupled with their objective functions respectively. RPL itself cannot be tested as a routing protocol rather RPL and an objective function needs to be tested against CTP and LIBP respectively. Since there are implementations for OF-0 and OF-ETX on Contiki already, the experiment variables will be CTP, LIBP, RPL-0, and RPL-ETX.

Table 1. Simulation Setup

Test Attributes	Test Value
Topology	175mx175m grid of 30 randomly placed nodes (density 30m2/node)*
Beacon Interval	30 seconds (LIBP), Adaptive (CTP, RPL)
Messaging Interval	30 seconds
Message Contents	Hello from node
Simulation Runtime	10 minutes (2 minutes for network self organization)*
(LIBP)	1
TX/INT Range	50m/100m

4.1 Methodology

The table above outlines the experiment runtime. In short, unless otherwise specified, the networks are each given a 2 minute period to allow for the network to settle; thereafter the network is run for 8 minutes to give a total simulation runtime of 10 minutes. Each node will periodically send a packet containing the string Hello from node as its packet data. Since each node is given 8 minutes to send the data at a period of 30 seconds, the nodes will each send 16 packets data to be collected by the sink. For the various experiments, all of Coojas existing profiling tools were used as experimentation tools. Simulation timers and node real-time timers were used as experimentation tools for time sensitive experiments. For discerning between control plane traffic and data plane traffic the packets were flagged accordingly, the packets would then trigger a counter which would hold a value that shows how many times a packet of that particular classification occurred as traffic during simulation runtime. Routing protocols have to be tested in terms of scalability. 10 random topologies were generated ranging from a topology sizes of 10 to 100 (in increments of 10). Each topology had the same node density. The benefit of having all these network topologies is so that metrics related to the routing protocols can be observed while the topology size increases.

4.2 Results and Evaluation

In this section, CTP, and RPL (alongside its OFs) is evaluated against the new implementation of LIBP on Contiki.

Energy Profile. The average power consumption in Figure 1 (a) was taken by averaging the power consumption amounts of each node. RPL seems to be significantly more power hungry on average when compared to CTP and LIBP. This could be put down to the fact that the radio in the sink node in RPL is always on, in addition to that, RPL is built on top of a slightly more capable but heavyweight communications protocol.

The graph of Figure 1 (b) shows the average radio duty cycle. It should be noted that the duty cycles represents the percentage of time that the radio was in a particular stage during the 10 minute simulation runtime. The TX and RX power draw are roughly the same on many motes.

RPL makes the assumption that the sink nodes are typically well powered. This is shown in the graph Figure 1 (c), the radios in the sink nodes for RPL are always turned on, which results in a very power hungry sink. The sink nodes in RPL would consume in the region of 60 mW, whereas the sink nodes in CTP and LIBP would consume power in the region of 4mW. In effect the sink nodes in RPL consume more than 1 order of magnitude more energy than the CTP and LIBP sink nodes. This can mostly be put down to the always on radio.

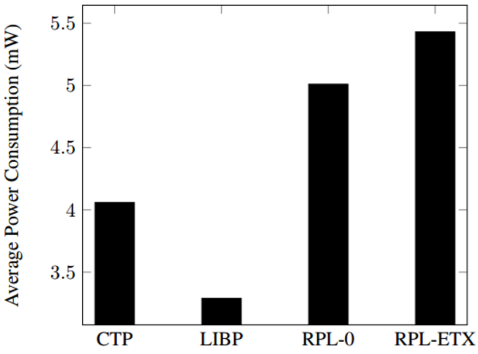
Table 2. Power Distribution

Routing Protocol	Mean (mW)	Standard Deviation (mW)
CTP	4.06	0.474
LIBP	3.24	0.278
RPL-0	5.01	10.81
RPL-ETX	5.43	10.73

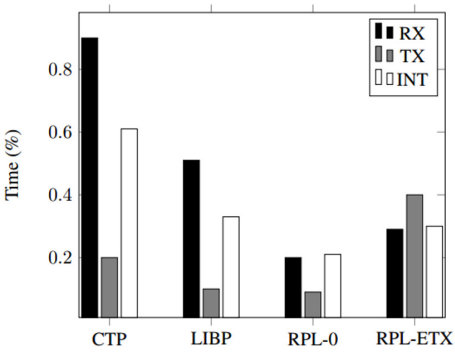
The standard deviation of the power consumption presented in Table 4.2for the routing protocol can describe how well distributed the energy consumption will be in the topology. This is a very important metric in figuring out the amount of time that a network can be deployed before requiring a battery change. Having a energy usage low mean and a low energy usage standard deviation shows that the protocol is energy efficient in its distribution and energy efficient in its implementation.

Routing Profile. The routing metrics of each routing protocol include the amount of supporting children per node, the average path length and the agility of the protocol.

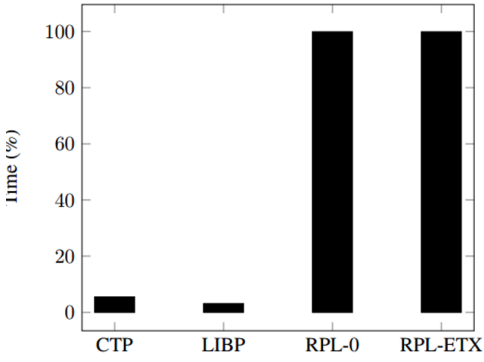
Figure 3 (a) shows the path interference (contention) in terms of average amount of children that a node would support. This value was obtained by counting the amount of times each node referenced a parent and then averaging those values. Having a smaller number of average children is a desirable metric because it can help with energy distribution in the network which helps with leaving all the nodes at more or less the same battery life. Having a high contention undesirable since it may also introduce a higher rate of packet loss or interference into the network. LIBP being the protocol which tries



(a) Average Power Consumption (mW)



(b) Radio Duty Cycle



(c) Radio Duty Cycle For Sink nodes

Fig. 1. Power consumption: average consumption and duty cycles

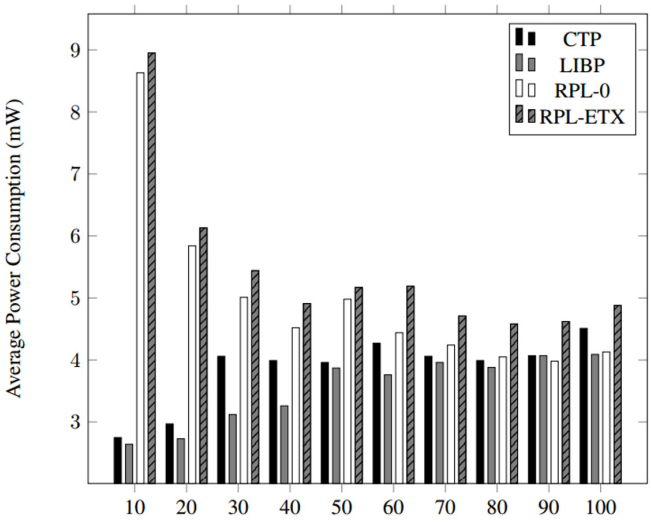


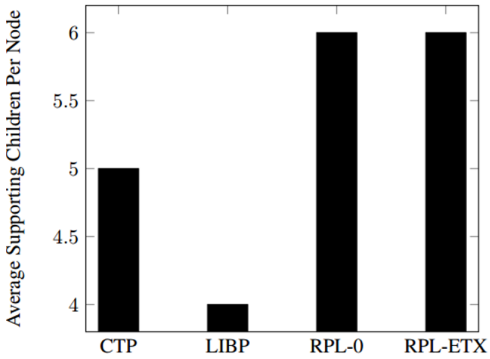
Fig. 2. Scalability for Average Power Consumption

to minimize the average amount of children in the pursuit for better energy distribution does better in this experiment.

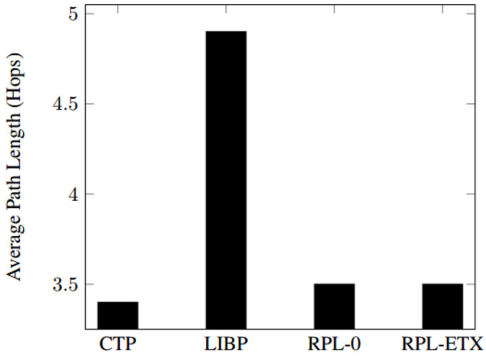
The average path length depicted in Figure 3 (b) was obtained by computing the TTL like attributes in the protocol control plane packets. LIBP and RPL use TTL (time to live) however CTP uses time has lived (which is TTL_MAX TTL). Once the number of hops was obtained they were averaged to give an average path length metric for each protocol respectively. Depending on the application, A high average path length is desirable for better for energy distribution but a lower average path length can result in a lower latency between the leaf nodes and sink nodes.

Recovery from Failure. The graph in Figure 4 (a) shows how quickly the protocols can come up with contingency routes if a node with high contention fails. To simulate this event, a node with a high degree of children (4 children) was chosen and deleted at the 10 minute mark. The times represented in the graph above shows the amount of time required for all 4 of the children to find alternate parents/routes. The data above shows how agile the routing protocols are in terms of how they deal with catastrophic failure.

The graph of Figure 4 (b) demonstrates how agile the routing protocols are in the ad-hoc sense. The experiment was set up by running a normal collect experiment of 30 nodes, except 1 node would be out of reach from the network (thus not part of the network). At the 10 minute mark from the start of the simulation the secluded node would be introduced to the network. The times in the above graph represent the time it took for that node to have acknowledged a parent (to become part of the network).



(a) Path interference



(b) Average path length

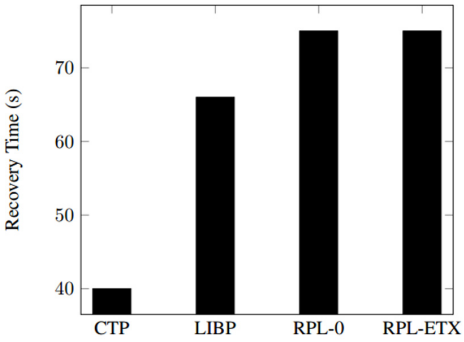
Fig. 3. Routing profile: Interference and path length

Traffic Profile. It is a worthwhile effort to see how much energy is spent on the control plane as opposed to the data plane. It should be noted that in the case of CTP, since beacon information piggybacks on data transmission, it counts as a beacon sent.

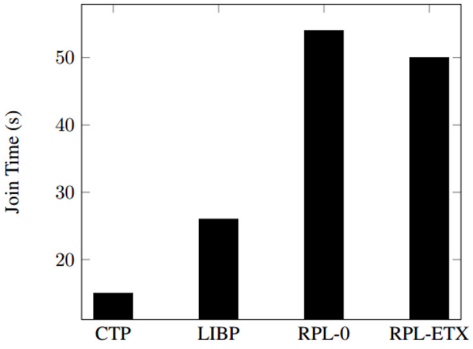
Table 3. Successful Transmission Rate

Protocol	Success Rate
CTP	99.7%
LIBP	99.7%
RPL-0	100%
RPL-ETX	100%

The table above describes the percentage of data packets that were collected by the sink node. Most packets were successfully collected by CTP and LIBP by achieving a higher than 99% transmission to the sink node. RPL achieved a 100% transmission rate. This astounding transmission rate could be attributed to how complete the communications protocol that RPL is built on top of is. Whereas CTP and LIBP are built on top of Rime [8].



(a) Network recovery



(b) Routing agility

Fig. 4. Network recovery and routing agility

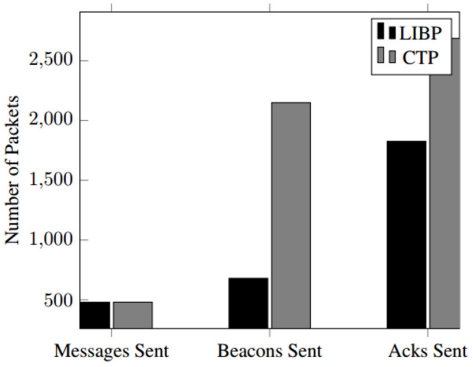


Fig. 5. Control Packets Sent

5 Conclusion and Future Work

This paper presents a comparison between routing protocols. Experiments were conducted between CTP, LIBP and RPL. The simulations revealed that CTP and LIBP are relatively light in their implementation and goals but lack a few features that RPL has like mote to mote communication. RPL also can utilize the full stack of security mechanisms present in IPv6. The inherent heaviness of RPL can be attributed to its underlying protocol and how the underlying protocol uses larger more feature rich packets. CTP and LIBP are very similar in their performance metrics, this could be attributed to the fact that they both use the same underlying communications stack. CTPs strength lies in its very agile nature. CTPs trickle timer allows it to react to adverse changes in the network very quickly. One of LIBPs main goals was to have a routing protocol that was more efficient in its global energy consumption. This resulted in a protocol that is very efficient in how each node in the network consumes a similar amount of energy. LIBP is a very compelling routing protocol to use for battery powered deployed sensor networks since this routing protocol evidently seems to be better adapted for that scenario. Future work may include testing the topology lifespan to see how long these topologies last before requiring a re-charge or battery replacement. Examining the security mechanisms of RPL has also been reserved for future work.

IoT networks and connected-oriented share the same routing paradigm of setting traffic flows from source to destination. Building upon the least path interference principle previously implemented for connection-oriented networks in [12–14, 16], the LIBP protocol presented in this paper reveals that traffic engineering techniques previously designed for traditional networks can be revamped to be applied to the emerging IoT. Similarly, there is room for revamping some of the techniques previously intended for gateway design [15] and quality of service (QoS) support [17] and long distance IoT deployment [18] for deployment in 6LoWPAN settings to enable flexibility and heterogeneity in IoT settings.

References

1. Vasseur, J., Dunkels, A.: Interconnecting Smart Objects with IP, The Next Internet. Morgan Kaufmann (July 2010) ISBN: 9780123751652
2. Bagula, A., Djenouri, D., Karbab, E.B.: Ubiquitous sensor network management: The least interference beaconing model. In: Proceedings of the IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), September 8-11, pp. 2352–2356 (2013) ISSN 2166-9570
3. Bagula, A.B., Djenouri, D., Karbab, E.: On the Relevance of Using Interference and Service Differentiation Routing in the Internet-of-Things. In: Balandin, S., Andreev, S., Koucheryavy, Y. (eds.) NEW2AN 2013 and ruSMART 2013. LNCS, vol. 8121, pp. 25–35. Springer, Heidelberg (2013)
4. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Simulating large wireless sensor networks of tinyos motes. In: Proc. of ACM SenSys 2003, Los Angeles, CA, pp. 126–137 (November 2003)
5. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462 (2004)

6. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. In: Proc. of ACM SenSys 2009, Berkeley, CA/USA, November 4-6 (2009)
7. Winter, T., et al.: RPL: IPv6 Routing protocol for Low-Power and Lossy Networks, RFC 6550 (March 2012)
8. Dunkels, A.: Poster Abstract: Rime: A Lightweight Layered Communication Stack for Sensor Networks. In: European Conference on Wireless Sensor Networks (EWSN), Delft, The Netherlands (January 2007)
9. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self regulating algorithm for code maintenance and propagation in wireless sensor networks. In: Proc. of the USENIX NSDI Conf., San Francisco, CA (March 2004)
10. Thubert, P.: Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). Internet Engineering Task Force (IETF), Request for Comments 6552, 1–14 (2012)
11. Dunkels, A., Osterlind, F., Tsiftes, N., He, Z.: Software-based Online Sensor Node Energy Estimation. In: ACM Proceeding of the 4th Workshop on Embedded Networked Sensors (EmNets 2007), pp. 28–32 (2007)
12. Bagula, A.: On Achieving Bandwidth-aware LSP/LambdaSP Multiplexing/Separation in Multi-layer Networks. IEEE Journal on Selected Areas in Communications (JSAC): Special issue on Traffic Engineering for Multi-Layer Networks 25(5) (June 2007)
13. Bagula, A.: Hybrid traffic engineering: the least path interference algorithm. In: Proc. of ACM Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, pp. 89–96 (2004)
14. Bagula, A.B.: Hybrid routing in next generation IP networks. Elsevier Computer Communications 29(7), 879–892 (2006)
15. Zennaro, M., Bagula, A.: Design of a flexible and robust gateway to collect sensor data in intermittent power environments. International Journal of Sensor Networks 8(3/4) (2010)
16. Bagula, A., Krzesinski, A.E.: Traffic engineering label switched paths in IP networks using a pre-planned flow optimization model. In: Proceedings of the Ninth International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2001), pp. 70–77 (August 2001)
17. Bagula, A.: Modelling and Implementation of QoS in Wireless Sensor Networks: A Multi-constrained Traffic Engineering Model. Eurasip Journal on Wireless Communications and Networking 2010, Article ID 468737, doi:10.1155/2010/468737
18. Bagula, A., Zennaro, M., Inggs, G., Scott, S., Gascon, D.: Ubiquitous Sensor Networking for Development (USN4D): An Application to Pollution Monitoring. MDPI Sensors 12(1), 391–414 (2012)