A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN)

Ulrich Herberg, Thomas Heide Clausen

A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-Directional Traffic in Low-power and Lossy Networks (LLN)

Ulrich Herberg
Trusted Systems Innovation Group
Fujitsu Laboratories of America, USA
ulrich@herberg.name

Thomas Clausen
Hipercom@LIX
Ecole Polytechnique, France
Thomas@ThomasClausen.org

ABSTRACT

Routing protocols for sensor networks are often designed with explicit assumptions, serving to simplify design and reduce the necessary energy, processing and communications requirements. Different protocols make different assumptions – and this paper considers those made by the designers of RPL – an IPv6 routing protocol for such networks, developed within the IETF. Specific attention is given to the predominance of bi-directional traffic flows in a large class of sensor networks, and this paper therefore studies the performance of RPL for such flows. As a point of comparison, a different protocol, called LOAD, is also studied. LOAD is derived from AODV and supports more general kinds of traffic flows. The results of this investigation reveal that for scenarios where bi-directional traffic flows are predominant, LOAD provides similar data delivery ratios as RPL, while incurring less overhead and being simultaneously less constrained in the types of topologies supported.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing protocols —*RPL, LOAD, IETF, LLN, evaluation, simulation, comparison, bi-directional traffic, sensor networks*

General Terms

Performance, Standardization

1. INTRODUCTION

Sensor networks differ from more "traditional networks" in that the devices making up a sensor network have connectivity maintenance and data forwarding as auxiliary tasks to their primary raison d'être, such as data acquisition. Ignoring the applications, the network itself can be described by (i) the devices being many thousands in number, (ii) with very limited internal (memory, CPU), external (communications capacity) and energy resources, and that (ii) the communications channel between devices typically has unattractive characteristics: low-bandwidth, high loss-rates and volatile links with limited persistency over time. The term Low-power Lossy Networks (LLN) is therefore commonly used for describing such networks.

Yet, despite these challenges, routing protocols are required for establishing and maintaining multi-hop connectivity in LLNs, for situations where it is unfeasible or impossible to provision a sensor network deployment such that all devices, necessitating communication between each other, are within direct connectivity.

1.1 Background

The Internet Engineering Task Force (IETF) [8] has a long tradition of developing and standardizing routing protocols. Initially, for fixed Internet infrastructures, where the conditions are more lenient than in LLNs: routers generally have abundance of computational capacity and few energy constraints, links are predominantly "good" with few losses – and while Internet routing protocols such as OSPF are able to handle some network topology changes, these are generally rare, and generally occur only as a result of relatively catastrophic events: a cable being cut, for example.

In the late 1990's, the IETF started investigating MANETs [9] – Mobile Ad hoc NETworks. Generally thought of as multi-hop wireless networks of mobile devices, a crop of routing protocols were developed and standardized, notably OLSR [5] and AODV [17][1]. Able to manage more dynamic topologies and the characteristics of wireless network interfaces, this work introduced a new dichotomy in routing protocol classification: OLSRv2 [3] being a classic link-state routing protocol, optimized for MANETs, it maintains paths to all destinations at all times, and this even before such paths are needed – *proactively.* AODV [17] approached the same problem in a different fashion, by discovering and maintaining paths to destinations only as needed by application traffic – *reactively.* For both, however, the assumption was that while the network topology might be dynamic and the wireless connectivity volatile, the devices in the network still had a relative abundance of both computational power and energy.

With the emergence of sensor networks, so did the challenge to the assumption of an abundance of computational power and energy – even transmitting an IPv6 packet with 128-bit long addresses was considered a strain in terms of

[1]And their successors, OLSRv2 [3] and DYMO [1], which are currently being specified by the IETF.

PE-WASUN'11, November 3–4, 2011, Miami, Florida, USA.

energy consumption, and so the IETF started investigating adaptations of IPv6 for LLNs: compressing addresses, removing options considered rarely used, simplifying packet processing etc. Routing protocols, even those developed for MANETs, were considered too heavy and work started around 2005 on IPv6-based routing protocols, adapted for LLNs – formalized by the IETF with the creation of a ROLL ("Routing Over Low-power and Lossy networks") Working Group [10] specifically for managing this development work. This Working Group produced a protocol denoted RPL ("Routing Protocol for Low-power and lossy networks") [19].

1.2 Sacrifices

While the development from fixed Internet infrastructure routing protocols such as OSPF and BGP to MANET routing protocols such as OLSR and AODV was accomplished by way of algorithmic optimizations *e.g.,* on the overhead incurred by sharing link-state information network-wide, MANET protocols remained relatively general routing protocols – in particular, MANET routing protocols provide full IP support and adhere to the "anybody can communicate with anybody" paradigm.

The development towards LLN routing protocols, on the other hand, came with sacrifices in generality: assumptions as to which options in IPv6 headers were to be supported, as well as optimization of routing protocol operation for specific traffic patterns, considering certain such esoteric enough so as to not merit special attention. The ROLL Working Group, in its design of RPL, made a set of such assumptions, notably that sensor-to-controller traffic (*multipoint-to-point*) is predominant, controller-to-sensor traffic (*point-to-multipoint*) is rare and sensor-to-sensor traffic (*point-to-point*) is somewhat esoteric. RPL in its design, therefore, optimizes for *multipoint-to-point* traffic, supports in a less optimized fashion *point-to-multipoint* and provides some basic mechanisms for *point-to-point* traffic – essentially transiting such *point-to-point* traffic between two sensors via the controller.

1.3 Motivation

The traffic patterns for which RPL optimizes are, unquestionably, reasonable in some scenarios: data acquisition networks, for example, where sensors monitor an environment and transmit their findings towards a central controller, and where traffic from the controller to a sensor is a rare occurrence. They are, however, not universal. There are scenarios in which sensor-to-sensor traffic is assumed to be the more common pattern, such as [14]. Another example is in utility metering, where a utility company may wish to have a controller inquire household meters as to their consumption – send a request and expect a reply – or even may use the network to change parameters in household meters (expecting a confirmation). An example hereof is for load management on a power-grid, where a controller may wish to instruct individual households to reduce (or increase) their consumption according to the overall load on the grid. Even data acquisition type networks may have this characteristics: a sensor detecting an abnormal condition may signal this – needing to be certain that the signal has been received and thus expecting a confirmation.

Thus, even if assuming that *point-to-point* between sensors inside the network is relatively rare, bi-directional communication between sensors and a controller should be assumed common. In terms of the assumptions on traffic patterns, made in the design of RPL [19], this entails that equal importance should have been given to *point-to-multipoint* and *multipoint-to-point* traffic.

1.4 Statement of Purpose

Thus, one purpose of this paper is to explore the performance of LLN routing protocols for scenarios where bi-directional traffic is prevalent, in particular understanding the behavior of RPL, and the viability of the sacrifices made in generality by the design of that protocol, when exposed to this common traffic pattern. A second purpose is to explore how a more general protocol, LOAD [11], behaves when exposed to the same traffic patterns – both as a point of comparison, and as a way of exploring the viability of that protocol for LLNs. LOAD is a protocol, derived from AODV [17] by simplifying some mechanisms (detailed in section 3) while retaining the generality of supporting all traffic patterns equally and provisioning all paths to be bi-directional.

The rationale for choosing RPL and LOAD for a performance comparison is that RPL is a new protocol just standardized by the IETF as *the* standard routing protocol for LLNs, whereas LOAD is closely derived from AODV. AODV has been standardized by the IETF in 2003, and since then, researchers and engineers have gained a profound experience with the protocol. Comparing RPL with LOAD can therefore help to discover weaknesses of either of the protocols and possibly to improve them in a future work.

1.5 Paper Outline

Section 2 presents a functional overview of RPL [19]. Section 3 similarly provides a functional overview of LOAD [11]. Section 4 performs a comparative study of these. Section 5 concludes this paper.

2. RPL OVERVIEW

The basic construct in RPL is a "Destination Oriented Directed Acyclic Graph" (DODAG), depicted in figure 1. In a converged LLN, each RPL router has identified a stable set of parents, each of which is a potential next-hop on a path towards the "root" (or "controller") of the DODAG, as well as a *preferred parent*. Each router, which is part of a DODAG (*i.e.* has selected parents) will emit *DODAG Information Object* (DIO) messages, using link-local multicast, indicating its respective *rank* in the DODAG (*i.e.* distance to the DODAG root according to some metric(s), in the simplest form hop-count). Upon having received a number of such DIO messages, a router calculates its own rank such that it is greater than the rank of each of its parents, select a preferred parent and then start emitting DIO messages.

The DODAG formation thus starts at the DODAG root (initially, the only router which is part of a DODAG), and spreads gradually to cover the whole LLN as DIOs are received, parents and preferred parents are selected and further routers participate in the DODAG. The DODAG root also includes, in DIO messages, a *DODAG Configuration Object*, describing common configuration attributes for all RPL routers in that network – including their mode of operation, timer characteristics etc. RPL routers in a DODAG include a verbatim copy of the last received DODAG Configuration Object in their DIO messages, permitting also such configuration parameters propagating through the network.

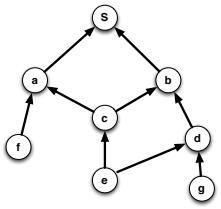

Figure 1: RPL Basic Construct: DODAGs

A Distance Vector protocol, RPL [19] restricts the ability for a router to change rank. A router can freely assume a smaller rank than previously advertised (*i.e.* logically move closer to the root) if it discovers a parent advertising a lower rank, and must then disregard all previous parents of higher ranks. The ability for a router to assume a greater rank (*i.e.* logically move farther from the root) than previously advertised is restricted, to avoid count-to-infinity problems. The root can trigger "global recalculation" of the DODAG by increasing a sequence number, *DODAG version*, in DIO messages.

The DODAG so constructed is used for installing routes: the "preferred parent" of an RPL router can serve as a default route towards the root, or the root can embed in its DIO messages the destination prefixes, included by DIOs generated by RPL routers through the LLN, to which connectivity is provided by the root. Thus, RPL by way of DIO generation provides "upward routes" or "multipoint-to-point routes" from the sensors inside the LLN and towards the root.

"Downward routes" are enabled by having sensors issue *Destination Advertisement Object* (DAO) messages, propagating as unicast via parents towards the DODAG root. These describe which prefixes belong to, and can be reached via, which RPL router. In a network, all RPL routers must operate in either of storing-mode or non-storing-mode, specified by way of a "Mode of Operation" (MOP) flag in the DODAG Configuration Object from the root. Depending on the MOP, DAO messages are forwarded differently towards the root:

- In *non-storing-mode*, an RPL router originates DAO messages, advertising one or more of its parents, and unicasts it to the DODAG root. Once the root has received DAOs from an RPL router, and from all routers on the path between it and the root, it can use source routing for reaching advertised destinations inside the LLN.
- In *storing-mode*, each RPL router on the path between the originator of a DAO and the root records a route to the prefixes advertised in the DAO, as well as the next-hop towards these (the router from which the DAO was received), then forwards the DAO to its preferred parent.

"Point-to-point routes", for communication between devices inside the LLN and where neither of the communicating devices are the DODAG root, are as default supported by having the source sensor transmit via its default route to the DODAG root (*i.e.,* using the upward routes) which will then, depending on the "Mode of Operation" for the DODAG, either add a source-route to the received data for reaching the destination sensor (downward routes in non-storing-mode) or simply use hop-by-hop routing (downward routes in storing-mode). In the case of storing-mode, if the source and the destination for a point-to-point communication share a common ancestor other than the DODAG root, a downward route may be available (and used) before reaching the DODAG root.

2.1 RPL Message Emission Timing – Trickle

RPL message generation is timer-based, with the root able to configure back-off of message emission intervals using *Trickle* [13], specified in [12]. Trickle, as used in RPL, stipulates that a RPL router transmits a DIO "every so often" – except if receiving a number of DIOs from neighbor routers, enabling the router to determine if its DIO transmission is redundant.

When an RPL router transmits a DIO, there are two possible outcomes: either every neighbor router that hears the message finds that the information contained is consistent with its own state (*i.e.,* the received DODAG version number received corresponds with that which the RPL router has recorded and no better rank is advertised than that which is recorded in the parent set) – or, a recipient RPL router detects that either the sender of the DIO or itself has out-of-date information. If the sender has out-of-date information, then the recipient RPL router schedules transmission of a DIO to update this information. If the recipient RPL router has out-of-date information, then it updates based on the information received in the DIO.

With Trickle, an RPL router will schedule emission of a DIO at some time, t, in the future. When receiving a DIO containing information consistent with its own information, the RPL router will record that "redundant information has been received" by incrementing a redundancy counter, c. At the time t, if c is below some "redundancy threshold", then it transmits its DIO. Otherwise, transmission of a DIO at this time is suppressed, c is reset and a new t is selected to twice as long time in the future – bounded by a pre-configured maximum value for t. If, on the other hand, the RPL router has received an out-of-date DIO from one of its neighbors, t is reset to a pre-configured minimum value and c is set to zero. In both cases, at the expiration of t, the RPL router will verify if c is below the "redundancy threshold" and if so transmit – otherwise, increase t and stay quiet.

3. LOAD OVERVIEW

LOAD [11] is a protocol, derived from AODV [17] and adapted for LLNs. Thus, the basic operation of LOAD is identical to that of AODV: a device with a packet to deliver to a destination, and which does not have a valid entry in its routing table for that destination, will issue a *route-request* (RREQ) message, diffused through the network so as to reach all other devices. When a device forwards this *route-request*, it records an entry in its routing table towards the originator of that *route-request* – a *reverse route* indicating the eventual path from the destination to the originator. If the destination is present in the network, it will eventually receive the *route-request* – and will respond by a *route-reply* (RREP), unicast to the originator of the *route-request* along the previously installed reverse route. As that *route-reply* is being forwarded along this reverse route, the devices forwarding it will instill a *forward route* towards the destination. Once the *route-reply* arrives at the originator of the corresponding *route-request*, a bi-directional path is installed, available for use.

When a link is detected to be broken (typically through a link-layer notification of a data-packet failing to be delivered to a next hop), the detecting router may engage in a *route-repair* operation – essentially a new *route-request*/*route-reply* cycle to discover a path to the destination – and if that fails, issue a *route-error* (RERR) message to inform the source of the failed data-packet of the error.

While this route discovery is performed, any IP-packets to the destination are buffered in the source router. When a route is established, these packets are transmitted – and if no route can be established, they are dropped.

The main differences between AODV and LOAD are:

1. LOAD simplifies the protocol behavior by disallowing that intermediate devices respond with a *route-reply* – even if they have an active route to the intended destination – thereby eliminating the need for destination sequence numbers.

2. Where in AODV, in case a device detects a link breakage, that device will attempt to transmit the *route-error* message to all neighbors which have recently used it as a next-hop on a path to the destination of the undelivered package, LOAD disables that – thereby eliminating the need to a device to maintain a precursor list.

Other, minor, differences include simplification of the packet format, support for compressed IPv6 addresses [15] etc.

LOAD does not impose any specific roles on any specific devices, notably has no controller or root with specific responsibilities for the network operation. Thus, the default traffic pattern supported by LOAD is bi-directional *point-to-point* traffic. The one sacrifice that LOAD makes with respect to data traffic, in simplifying from AODV, is, that it assumes that a given destination typically is in communication with only a single source at a given time – hence, the suppression of the precursor list.

4. EVALUATION

In order to understand the performance of both RPL and LOAD in LLNs with bidirectional traffic, a simulation study using the Ns2 simulator has been conducted. Standard evaluation metrics, such as data traffic delivery ratio, control traffic overhead, number of collisions, network convergence time etc. are compared between the two protocols. Energy consumption of the two protocols was out of the scope of this paper, as the energy consumption largely depends on the underlying hardware, and can therefore hardly be quantified in network simulations. As in particular receiving and sending messages on the network interfaces drains energy from the device, the energy consumption is related to the quantity of the control traffic overhead of the routing protocol, as well as data traffic. Section 4.1 describes the simulation settings and section 4.2 presents the results of the evaluation.

4.1 Simulation Settings

The specific settings of the scenarios studied are detailed in table 1. For each datapoint, the values have been averaged over 10 runs.

[3]As both RPL and LOAD are agnostic of the underlying link layer, any other link layer, such as IEEE 802.15.4 could be used, with expected similar behavior of the routing protocols. One possible difference, however, could be fragmentation when using smaller MTUs, such as in 802.15.4, as described in [4].

Table 1: Ns2 parameters

Parameter	Value
Mobility scenarios	No mobility, random distribution of routers
Grid size	variable
Router density	50 / km^2
Number of routers	63 / 125 / 250 / 500 / 1000
Communication range	250m
Radio propagation model	Two-ray ground
Simulation time	RPL: 270 secs/LOAD: 1 day
Interface type	802.11b [2] (2.4 GHz)

Both LOAD and RPL have been implemented in Java, using the AgentJ framework [6] to hook the Java code into Ns2. The routers were placed randomly in a square area of variable size, with a density of 50 routers per km^2. Only scenarios were selected where all routers are in the same connected component.

The bidirectional traffic patterns consisted of a "reading scenario", where the controller collects information from all sensors in the network, within 24 hours. Delay of the replies was considered irrelevant, as long as all sensors have replied within the 24 hours. In the Ns2 simulation, the controller sends a single request to each sensor (one datagram, 11 octets payload), to which the sensor replies with a single data packet (100 octets payload).

The settings for RPL are listed in table 2, and for LOAD in table 3.

Table 2: RPL parameters

Parameter	Value
Mode of operation	non-storing
Rank metric	hop count
DIOIntervalMin	2 s
DIOIntervalDoublings	20
DIORedundancyConstant	∞
DAOInterval	15s

Table 3: LOAD parameters

Parameter	Value
RREQ jitter	0 - 0.5 s
Route lifetime	5 s
Address compression	2 octets per IPv6 address

The simulation time was 24 hours for LOAD, and 270 seconds for RPL. The reason for not choosing 24 hours for RPL as well is the required amount of time for conducting the simulation: as DAO messages are sent periodically in the simulation, even if no data traffic occurs in the network, the required amount of time to conduct the simulation would take several days for a single simulation run. Moreover, trace files would grow to hundreds of Gigabytes per simulation. As in LOAD no control traffic is sent when no data traffic is sent, the simulation runs considerably faster,

allowing to simulate the 24 hours interval. Despite this difference of simulation time, the results are comparable, as the transmitted data traffic stays the same. The control traffic overhead of RPL can be extrapolated from 270s to 1 day, as DAO messages are sent periodically.

For RPL, the first traffic request from the controller is sent at 60s after the the start of the simulation, allowing RPL to converge (*i.e.,* to construct the DODAG) before. Once the sensor receives the data request, it replies immediately. The following data request from the controller to the next sensor is sent 0.1 seconds after the previous data request. For simulations with 1000 routers, the last data request from the controller will thus be sent at 160 seconds after the simulation start ($60s + 999 \cdot 0.1s$).

For LOAD, the traffic requests are sent every 30 seconds, starting at 100 seconds after simulation start, *i.e.,* the last request is sent at 30070 seconds or 8.35 hours after the simulation start. In order to reduce the amount of collisions of frames, a random jitter has been added before forwarding a RREQ on a router, as specified in [2]. While this is not specified in the current revision of LOAD, it could be added in a future revision, similar to other current ad hoc routing protocols like OLSRv2 [3].

During the simulation, only a single traffic flow from the controller towards a sensor or vice verse is active, *i.e.,* only a single route has to be maintained on a router running LOAD (assuming that routes expire before the next data packet is transmitted). In the investigated "reading scenario", this does not represent an unrealistic setting, since every sensor is only contacted once in 24 hours by the controller. As a consequence, both RPL in non-storing mode and LOAD require minimal state on the routers.

4.2 Simulation Results

This section describes the results of the simulation.

Figure 2 shows the maximum and average rank of routers in the DODAG, where the number represents the distance of a router to the controller in terms of hops (*i.e.* the maximum rank represents the diameter of the network, the average rank represents the average over all routers). The maximum and average ranks grow logarithmically with the number of routers in the network.

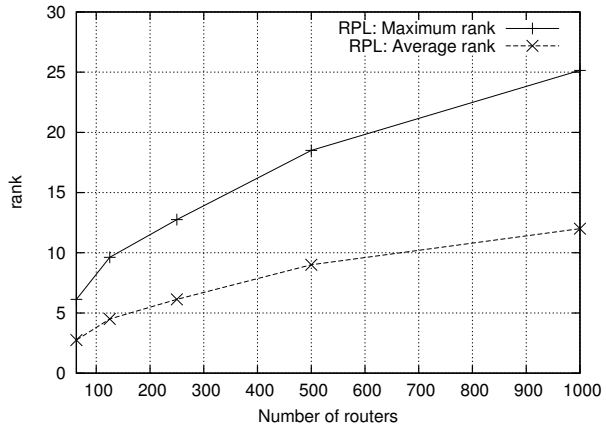

Figure 2: Maximum and average rank of routers

Figure 3 depicts the average number of parents of each router in the DODAG. Keeping the density of the network constant with increasing number of routers, the average number of parents grows logarithmically.

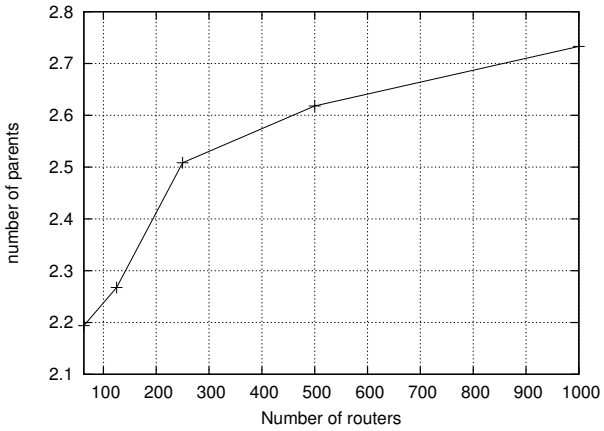

Figure 3: Average number of parents per router

Figure 4 displays the convergence time of the network, *i.e.* the time that is needed for all routers that are in the same connected component as the controller to join the DODAG. Since each router starts sending DIOs two seconds after the last change to its Candidate Neighbor Set, the convergence time is roughly two seconds times the maximum rank of the DODAG. The convergence time grows logarithmically with the number of routers in the network.

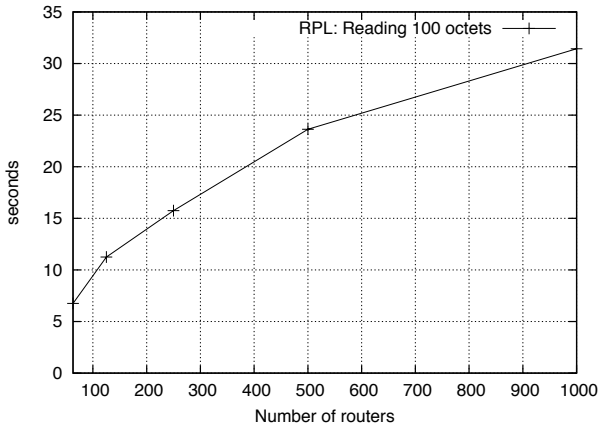

Figure 4: Network convergence time

Figure 5 depicts the cumulative control traffic overhead over the simulation time, *i.e.,* in 270 s for RPL and in 1 day for LOAD. Each forwarding of a broadcast packet is counted separately, *e.g.,* a RREQ in LOAD that is flooded from the controller to n other sensors, counts for $n \cdot sizeof(RREQ)$ bytes. It can be observed that for both protocols, the control traffic overhead increases polynomially with the amount of routers, and that RPL leads to more than twice the amount of the control traffic as LOAD. For RPL, the periodic DAOs account for the majority of the control traffic, whereas the overhead of DIOs is much lower due to the exponentially growing emission intervals of the Trickle timer. For LOAD, the broadcast RREQs lead to a much higher overhead than the unicast RREPs.

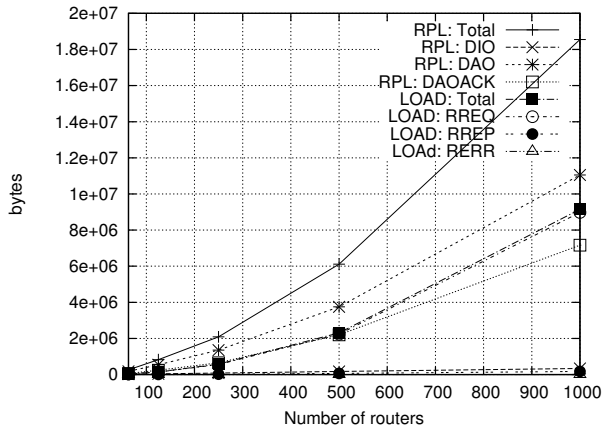

Figure 5: Control traffic overhead

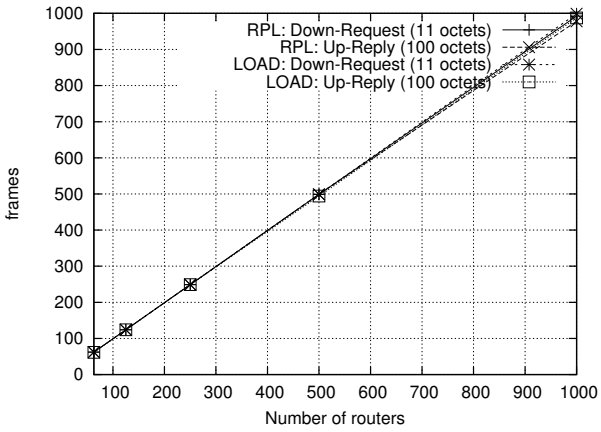

Figure 7: Traffic sent in frames

It has to be noted that the control traffic overhead of LOAD, as for any reactive routing protocol, depends on the data traffic in the network (as well as the validity time of the routes and the dynamicity of the network topology). The control traffic overhead of RPL, as a proactive protocol, does not depend on the data traffic.

Since the simulation time was much lower for RPL, and DAOs are sent periodically, the expected amount of control traffic for a duration of 1 day can be easily extrapolated. Figure 6 depicts the extrapolated control traffic amount of DAOs and RREQs (which account for the majority of the control traffic, as described above). The overhead for the RREQs is the same as in figure 5, as the emission of RREQs only depends on the data traffic.

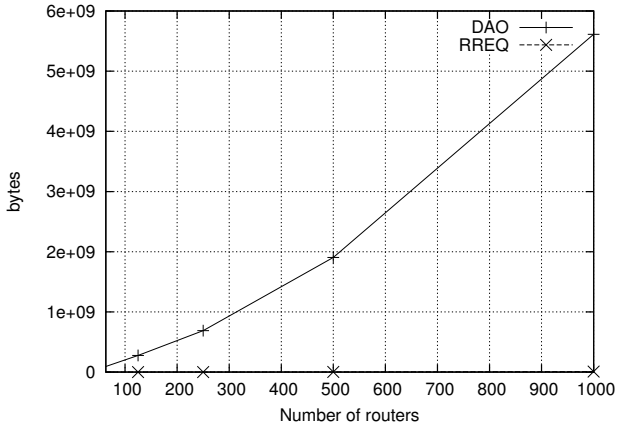

Figure 6: Extrapolated control traffic overhead for 24 hours

The overhead of RPL is two orders of magnitude higher than the overhead of LOAD.

Figure 7 shows the data traffic in number of frames that is sent from the controller to each sensor. Evidently, as one single frame is sent per router, the total overhead consists of $n-1$ frames for n routers.

However, as shown in figure 8, the cumulative amount of data traffic in terms of bytes is different for RPL and LOAD, where each forwarding of a packet is counted. The overhead for the 11-octet request downward from the controller to the sensors differs, as RPL (in non-storing mode) includes source routes for each data packet, which are included in the count of this figure. Therefore, even though a single data request packet is nine times smaller than the data reply, the data requests account for almost half as much overhead as the replies in RPL. The upward data replies in LOAD consume more bandwidth than in RPL, which can be explained with suboptimal choice of paths in the basic version of LOAD, as explained in the following.

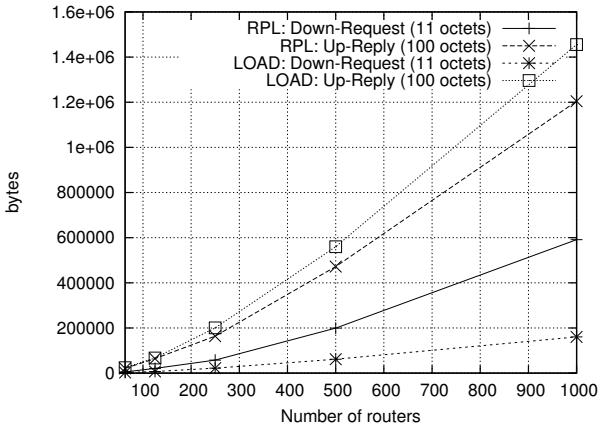

Figure 8: Traffic sent in bytes

Figure 9 shows the path length of the data traffic, for both RPL and LOAD, as well as for a protocol called "God". The God routing protocol uses the "God" object of Ns2, in order to calculate routes to all destinations based on their position and radio range, without any control message exchange and zero convergence time. The expected performance of the GodRP is close to the best possible routing protocol, which helps to understand how well a routing protocol could theoretically perform. It can be observed that LOAD chooses longer paths than RPL, which itself is close to the optimum. Two reasons can explain the longer paths of LOAD:

1. When a router receives a RREQ that is destined to itself, it will reply to the first incoming RREQ and set the reverse route towards the last hop of that RREQ.

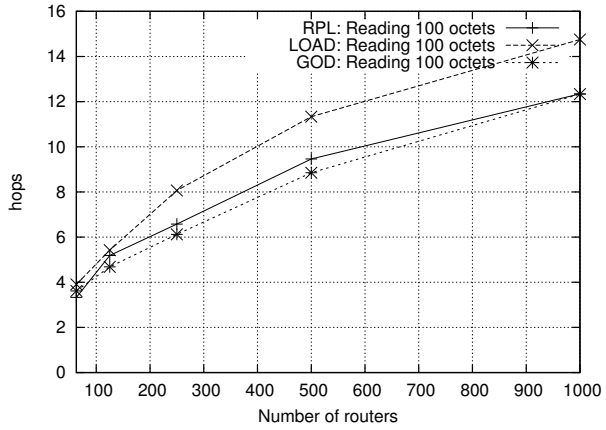

Figure 9: Path length

However, it may happen that shorter paths are available, but that the RREQ for that shorter path arrives only later (and since only a single data packet is sent, the better path will not be used). If the router had waited longer, before replying with a RREQ, it could have chosen a shorter path, at the expense of a higher delay.

2. As LOAD in its basic form applies "Classic Flooding" for distributing the RREQs throughout the network, longer paths occur than *e.g.,* when using MPR flooding, which leads to optimal paths ([18]).

An observation concerning the path length can be made for RPL in non-storing mode, which uses source routes for the downward traffic: the maximum length of the source routing header [7] is limited to 136 octets, including an 8 octet long header. As each IPv6 address has a length of 16 octets, not more than 8 hops from the source to the destination are possible for "raw IPv6". Using address compression [15], the maximum path length may not exceed 64 hops. This excludes scenarios with long "chain-like" topologies. The size of the header with increasing length of the source route is depicted in figure 10.

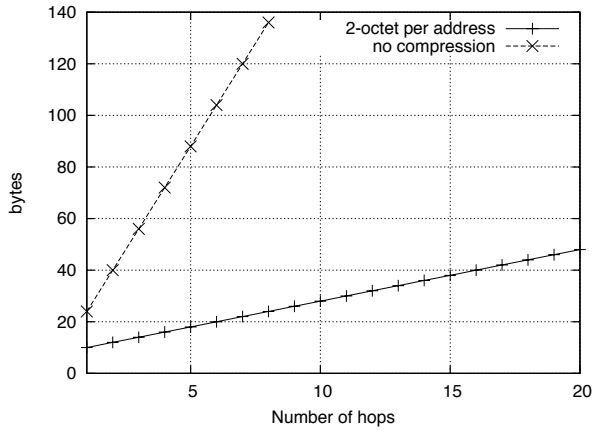

Figure 10: Source route overhead

Figure 11 depicts the delivery ratio of the data traffic. For both protocols, it is close to 100%, which is due to the two-ray ground model applied in the simulations. Unless there are collisions on the MAC layer, no packets are lost. This does, of course, not reflect experience from real wireless communication, and has to be considered when interpreting the simulation results. Simulating realistic channel behavior and propagation models is non-trivial, and better observed in real testbeds than in network simulations. Still, the simulation results allow to validate whether the routing protocol choose correct paths.

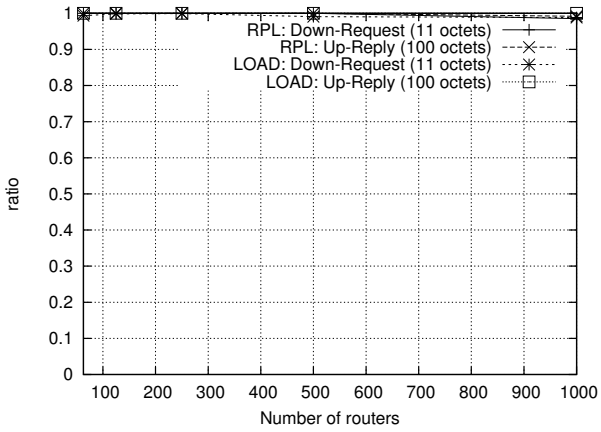

Figure 11: Delivery ratio

Figure 12 shows the end-to-end delay of a data traffic transmission. Data traffic is buffered in LOAD when no route is available, until a RREP from the destination has been received (or is dropped after a timeout if no RREP is received). The waiting time depends on the number of hops the RREQs and RREPs have to traverse as well as the time required for relaying a packet at a router, and therefore increases with the number of routers in the network. As no buffering is applied in RPL, the delay is much lower. However, in scenarios where delay is negligible – such as the evaluated "reading" scenario within 24 hours – the higher delay may be acceptable.

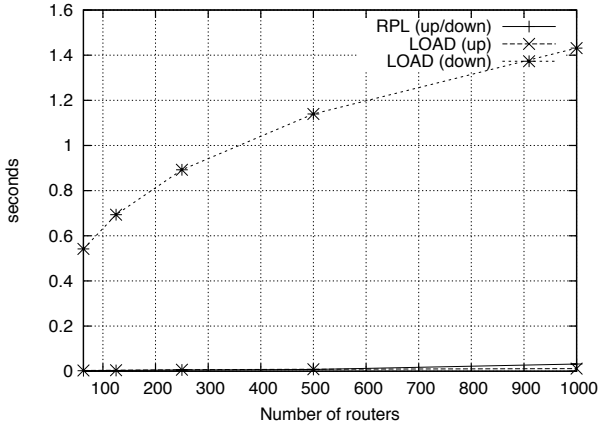

Figure 12: Delay

Finally, figure 13 depicts the number of collisions of frames for both RPL and LOAD. As LOAD applies Classic Flooding

for the RREQ, the well-known broadcast storm [16] leads to a higher collision rate in LOAD. A more efficient flooding mechanism could reduce the amount of collisions (as well as the control traffic overhead).

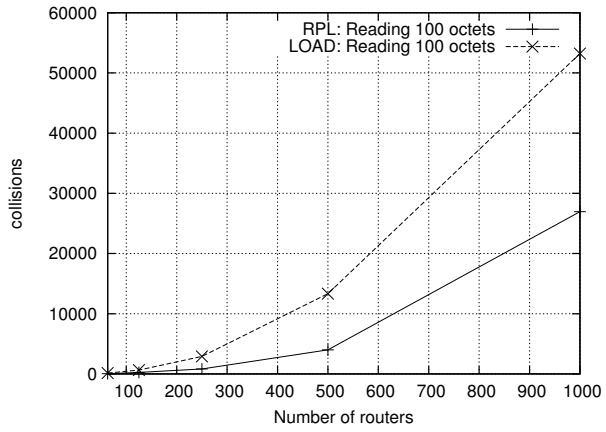

Figure 13: Collisions

5. CONCLUSION

RPL and LOAD represent two different philosophies for routing protocols in LLNs. RPL is optimized for specific topologies and traffic patterns – a central controller with specific responsibilities for topology formation and maintenance, and towards which the majority of traffic flows. Thus, the strength of RPL is proactive construction of a collection tree for forwarding such traffic. LOAD represents a perhaps less optimized protocol, however one wherein the philosophy is an entirely distributed mode of operation, where paths are discovered on demand and so as to be bi-directional.

Observing that a large set of deployment scenarios for sensor networks imply the need of bi-directional traffic flows – utility metering, building automation and data acquisition networks where possible alarm signals need be acknowledged – this paper studies the performance of these two protocols in such scenarios. While both protocols are able to provide reasonably high and definitely comparable data delivery ratios, LOAD yields a consistently lower control traffic overhead than does RPL – all the while being less constrained both in terms of traffic types and topologies supported.

While generalizing from simulation studies always is to be done with utmost care – the adage of "show me a protocol, and I will construct a scenario wherein it performs badly", the results presented in this paper nonetheless suggest that for deployments in which bi-directional traffic flows predominate, LOAD is a more evident candidate routing protocol for LLNs than is RPL.

6. REFERENCES

[1] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet Draft, work in progress, draft-ietf-manet-dymo-21.txt, Jul. 2010.

[2] T. Clausen, C. Dearlove, and B. Adamson. Jitter Considerations in Mobile Ad Hoc Networks (MANETs), Feb. 2008. RFC 5148.

[3] T. Clausen, C. Dearlove, and P. Jacquet. The Optimized Link-state Routing Protocol version 2. Internet Draft, work in progress, draft-ietf-manet-olsrv2-12.txt, Jul. 2011.

[4] T. Clausen, U. Herberg, and M. Philipp. A Critical Evaluation of the "IPv6 Routing Protocol for Low Power and Lossy Networks" (RPL). In *Proceedings of the International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct. 2011.

[5] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR), Oct. 2003. RFC 3626.

[6] U. Herberg and I. Taylor. Development Framework for Supporting Java NS2 Routing Protocols. In *Proceedings of the International Workshop on Future Engineering, Applications and Services*, May 2010.

[7] J. Hui, J. Vasseur, D. Culler, and V. Manral. An IPv6 Routing Header for Source Routes with RPL, Mar. 2011. Internet Draft, work in progress, draft-ietf-6man-rpl-routing-header-03.

[8] IETF. Web site. http://www.ietf.org.

[9] IETF MANET working group. Charter. http://www.ietf.org/html.charters/manet-charter.html.

[10] IETF ROLL working group. Charter. http://www.ietf.org/html.charters/roll-charter.html.

[11] K. Kim, S. D. Park, G. Montenegro, S. Yoo, and N. Kushalnagar. 6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD), Jun. 2007. Internet Draft, work in progress, draft-daniel-6lowpan-load-adhoc-routing-03.

[12] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm, Mar. 2011. RFC 6206.

[13] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the Symposium on Networked Systems Design and Implementation*, 2004.

[14] J. Martocci, P. D. Mi, N. Riou, and W. Vermeylen. Building Automation Routing Requirements in Low Power and Lossy Networks, Jun. 2010. RFC 5867.

[15] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks, Sep. 2007. RFC 4944.

[16] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the International Conference on Mobile computing and networking*, 1999.

[17] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing, Jul. 2003. RFC 3561.

[18] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *Proceedings of the Hawaii International Conference on System Sciences*, 2001.

[19] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks, Mar. 2011. Internet Draft, draft-ietf-roll-rpl-19.