Routing in IPv6 Sensor Networks

Malisa Vucinic

Grenoble INP – ENSIMAG
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Final Report

Laboratoire d'Informatique de Grenoble - team DRAKKAR

Routing in IPv6 Sensor Networks

Mališa VUČINIĆ
3rd year – Master CSE

June 15, 2012

Laboratoire LIG
681 rue de la Passerelle
BP 72
38402 Saint Martin d'Hères

Mentor
Bernard TOURANCHEAU
Academic tutor
James CROWLEY

Contents

1 Introduction

Advances in micro-electro-mechanical systems (MEMS) technology, wireless communications and digital electronics have enabled the development of low-cost, low-power, multi functional sensor nodes that are small in size and communicate untethered in short distances [1]. Majority of sensor applications imply the deployment of a large number of nodes, communicating wirelessly and therefore forming a Wireless Sensor Network (WSN). A direct consequence of such a large number of nodes expected to be deployed is their low-cost requirement. This implies minimal processing and memory capabilities, with a very long life time which requires small energy consumption. WSNs have been under study by the academic community for a number of years, and a number of different solutions has arisen. The application field of home and building automation is particularly interesting as it implies products from a large number of different vendors and therefore emphasizes the importance of interoperability. The IEEE has standardized PHY and MAC layers for Low Power Wireless Personal Area Networks in the IEEE 802.15.4 standard [2]. However, the standard does not specify higher layers allowing the rise of a number of different technologies. [3] summarizes some of the technologies that have already been deployed for applications concerning home and building automation. Focus of the research community has been shifted towards the IP based solutions in order to enable the so-called "Internet of Things", where each sensor or actuator will be accessible from the Internet, due to the well-known advantages of the Internet Protocol. This will allow remote and centralized control of virtually any device present in the household, building, urban environment, etc. and thus will provide means for very efficient energy consumption apart from an improved standard of living. As the expected number of devices in the "Internet of Things" is very large, a natural choice is to use Internet Protocol version 6 (IPv6) due to the huge addressing space provided. However, IP was designed for high-quality links and large frame sizes. Thus, in order to use it for Low power And Lossy Networks (LLNs)[1], which use IEEE 802.15.4 and the corresponding 127 bytes Maximum Transmission Unit (MTU) on Layer 2, it is necessary to do some adaptations. Main challenges of adapting IPv6 to LLNs are given in [4]. Furthermore, [5] discusses the main challenges of implementing the IPv6 stack on memory constrained devices such are sensor nodes. These networks (IPv6 over Low power Wireless Personal Area Networks) are commonly known as 6LoWPAN networks. [6] defines the format and main mechanisms of the adaptation layer used in transmission of IPv6 packets over 802.15.4.

One of the key issues in 6LoWPAN networks is routing. [7, 8, 9, 10] specify routing requirements for building automation, home automation, industrial and urban applications, respectively. Common factor for these applications is the dominance of multipoint-to-point traffic. In essence, this is the primary goal of a sensor network - transfer of the sensed data towards a collector - more commonly named sink. Furthermore, point-to-multipoint traffic is also common, usually representing a query from the sink towards sensors. Point-to-point,

[1] Low power and Lossy Networks (LLNs) are a general case of Wireless Sensor Networks (WSN) that takes into account also wired - PLC (Power Line Communication) links.

i.e. sensor-to-sensor traffic is rare[2].

IETF formed the ROLL working group (Routing over Low power and Lossy Networks) with an aim of designing a routing protocol for LLNs. IPv6 Routing Protocol for Low power and Lossy Networks (RPL) has been proposed in [11], as none of the existing protocols met the initial requirements of LLNs [12]. As of March 2012, RPL has been published as "proposed standard" Request for Comments (RFC) by the IETF.

However, another protocol called "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation" (LOADng) has been proposed in [13]. It uses a completely different approach compared to RPL and some studies show that it performs better in certain scenarios.

The main topic of the training was to perform a thorough study of the two protocols through simulations and tests of different scenarios of interest. For that purpose, an event-driven Operating System called "Contiki" [14], developed specifically for memory constrained devices such are sensor nodes, was used in order to emulate the software executing on individual nodes. The "Cooja" simulator [15], part of the Contiki project, was used as an event generator, in order to evaluate the two protocols in different scenarios and topologies.

The remaining of this report is organized as follows. Section 2 gives a short presentation and current research activities of the DRAKKAR team, which is part of the Laboratoire d'Informatique de Grenoble (LIG), where the training is taking place. Related work on this topic is presented in Section 3. Overview of the RPL protocol, main design ideas and critical points are presented in Section 4. Section 5 summarizes the LOADng routing protocol. Simulation scenario and the observed performance is discussed in Sections 6 and 7, respectively. Main implementation difficulties and challenges encountered during the training are discussed in Section 8. Assessment of the training is given in Section 10. Finally, concluding remarks and observations are discussed in Section 11.

2 Laboratoire d'Informatique de Grenoble

The Grenoble Informatics Laboratory (Laboratoire d'Informatique de Grenoble - LIG) [16] is a laboratory of quite some scale, the academic partners being:

- the CNRS,
- Grenoble INP,
- INRIA Grenoble Rhône-Alpes,
- the Université Joseph Fourier,
- the Université Pierre-Mendès-France,

[2]Point-to-point traffic is present in home automation networks where an actuator may address a switch, to turn off the light for instance.

- the Université Stendhal.

The LIG brings together almost 500 researchers, professors and associate professors, doctoral students, and research support personnel. They belong to different organizations and are located at one of the two LIG sites (the Grenoble university campus or in Montbonnot).

The scientific project of the LIG is "ambient and sustainable IT". The goal is to leverage the complementary nature and recognized quality of the 24 research teams of the LIG to contribute to fundamental aspects of the discipline (modeling, languages, methods, algorithms) and to create a synergy between the conceptual, technological and societal challenges that surround this theme.

This scientific project is a continuation of that of the previous four-year period regarding ambient IT. The challenges to be addressed are indeed numerous and far-reaching. The diversity and dynamism of data, services, interaction methods, and contexts of use require an evolution of systems and software to guarantee essential properties such as reliability, performance, autonomy and adaptability. In rising to these challenges, there is a resonance between the various research areas explored by the LIG:

- infrastructure,
- software,
- interaction,
- knowledge.

For the second four-year period, the LIG will address, with conviction, the issues around the theme of sustainable computing. While ambient computing is a lever, without precedent, for freedom and openness of information with a tremendous potential for individual and social applications in many domains (for example health care, education, environment, transport and intelligent buildings), it also poses the question of its thoughtful and ethical use in the context of new problems confronting society:

- managing energy for ambient computing,
- improving quality of life and the security of goods and people.

The LIG laboratory is focused on the fundamentals and the development of IT science, maintaining a high degree of openness to society to rise to new challenges.

2.1 DRAKKAR team

The Drakkar group at LIG Lab conducts research in many areas of *computer networking and mobile computing* [17]. For several years the group focused on Ambient Networking: how to provide everywhere seamless connectivity to mobile users over different wireless technologies. Moreover, the team investigates how to integrate the information about the

physical world within the current networks and study new ways for supporting collaborative communications. The research objectives come from the observation that wireless communications do not fit well the architecture of the existing Internet: wireless links are very different from the wired counterparts, the addressing scheme and the rigid routing hierarchy do not allow efficient handling of mobile terminals, and the core network infrastructure does not fit the requirements of dynamic and spontaneous operation. Most of the effort goes into providing solutions to all these problems and developing the basis for the future Ubiquitous Wireless Internet.

More specifically, the team considers various aspects of the following themes:

- wireless local area networks (802.11, Bluetooth, advanced MAC protocols),
- mesh networks (advanced MAC protocols, routing protocols, cross-layer interactions),
- seamless mobility (location-identity separation, location service).
- sensor and actuator networks (energy-conserving protocols, 802.15.4-ZigBee, Internet of Things),

The methodology is driven by experimentation: observing and measuring existing technologies and protocols, finding new ideas for improving them, evaluating the ideas either through simulations, or by building operational prototypes that can be measured to get insight about their intrinsic behavior.

The group has obtained highly visible results published in the best conferences in our domain (ACM SIGCOMM, IEEE INFOCOM, ACM CONEXT). The number of citations of the INFOCOM paper on 802.11 performance anomaly has reached 989. The paper on the Idle Sense access method published at ACM SIGCOMM has already obtained 304 citations. Moreover, it has joined the Reading Lists proposed by advanced courses in wireless networking at several leading universities.

The future research of the team will still concern wireless networking, however the group started investigations in two new directions: detecting attacks and traffic anomalies in the Internet, and protocols for future enterprise Intranets.

Future activities perfectly fit the LIG projects on Ambient and Sustainable Informatics. Research focus is on the Intelligent Home Challenge as well as on the Sensor Networks Technological Challenge. Some of the activities are also related to other challenges such as Embedded Systems as well as Systems and Distributed Computing. Results obtained concerning energy efficient protocols in sensor networks will contribute to Green-IT and Sustainable development. Applications of sensor networks in many areas (industry, housing, environment, etc.) will also play a part in Green-by-IT challenges.

3 Related Work

Several papers discussing the performance of different implementations of Routing Protocol for Low Power and Lossy Networks (RPL) [11] have been published recently. The RPL

performance Internet draft [18] evaluates the protocol by considering several routing metrics (path quality, delay bound for P2P routing, routing table size, control packet overhead, loss of connectivity) in real-life deployment scenarios. Stability delays of RPL have been studied in [19] using OMNET++. Multipoint-to-point performance of RPL has been studied in [20] using NS2. Authors in [21] evaluate different optimized broadcast techniques for use in RPL. Implementation of RPL for TinyOS is presented in [22] and authors discuss the packet reception ratio, routing protocol overhead and response ratio, as part of the performance analysis. A framework for low power IPv6 routing simulation, experimentation and evaluation based on Contiki OS and Cooja simulator has been proposed in [23]. This framework will be used in our studies. Authors in [24] use this framework with ContikiRPL implementation in order to evaluate the performance of a PLC network serving as a LLN backbone. Routing overhead and delay induced by ContikiRPL implementation is studied in [25]. Common observation is that RPL performs very well in case of multipoint-to-point traffic (selection of near-optimal routes, bounded delay) but induces a large overhead in scenarios like [8] where point-to-multipoint traffic is non-negligible. Furthermore, the RPL RFC [11] under-specifies some very important mechanisms and thus, leaves as an implementation freedom parameters that are crucial for good performance of the system (see [25, 26]). Finally, [27] gives a thorough study of several critical points of the RPL protocol.

LOADng [13] is an adapted version of AODV [28] routing protocol. AODV was designed for MANETs[3] and therefore had to be modified in order to account for the very low memory capabilities of sensor devices. While RPL acts pro-actively in finding routes through the network, AODV and LOAD use a reactive approach. Overview of the LOADng protocol is given in Section 5. Authors in [29] compare various AODV-based routing protocols, including LOAD, and discuss the main design choices. In [30] authors evaluate the route discovery delay as well as the round trip time induced by an implementation of LOAD in TinyOS, in real-life deployment scenarios. A modification of the LOAD protocol called MLOAD is proposed in [31], and it adds a redundant path at the cost of higher memory requirement. [32] discusses advantages and drawbacks of LOAD, DYMO-low and HiLow routing protocols for use in 6LoWPANs. As AODV has been studied thoroughly there has been considerate less work on LOAD, in respect to RPL.

A comparative performance study of RPL and LOADng protocols in case of bidirectional traffic has been performed in [26], with simulations in NS2. The paper shows the significant control overhead of RPL, caused by the maintenance of downward routes, with respect to LOADng. Also, the two protocols have been compared with an ideal routing protocol and it has been shown that RPL provides near-optimal routes while LOADng has a certain gap. As the RPL RFC does not specify the period or mechanism to be used for maintaining downward routes, authors in [26] have assumed an interval of 15 seconds. This choice is questionable and is the main cause of the high control overhead of RPL, presented in the paper. Furthermore, the application-layer scenario used for the comparison is not the same for the two protocols and the question remains how LOADng would perform under the scenario that was used for RPL.

[3]Mobile Ad Hoc Network - generally assuming memory and computationally capable devices

4 Routing Protocol for Low Power and Lossy Networks (RPL)

IETF formed a ROLL working group in 2008 that conducted an analysis of the routing requirements for several applications: home automation [8], building automation [7], urban networks including smart grids [10] and industrial automation [9]. The idea was to design a new or adapt an existing routing protocol to serve most of the routing needs of the future "Internet of Things"[4]. [12] concluded that no existing routing protocol meets all the needs of LLNs. Therefore, a new protocol had to be designed. A very important aspect of the new protocol was that it should make no assumptions on the underlying link layer as it should be run on a variety of link technologies - IEEE 802.15.4, low-power IEEE 802.11, Power Line Communication (PLC) using IEEE 802.15.4 such as IEEE P1901.2, etc [33].

The routing protocol was named RPL and was designed with a proactive approach - the routes are found and maintained without any considerations on the ongoing traffic in the network. In essence, RPL is a Distance Vector protocol that specifies how to construct a Destination Oriented Directed Acyclic Graph (DODAG) with a defined objective function and a set of metrics and constraints. It is important to note that several DODAG instances may be used for the same mesh network which allows for traffic differentiation in classes. For instance, high priority traffic could use the best path through the network in terms of minimal delay while low priority traffic could avoid battery-powered nodes.

A Directed Acyclic Graph (DAG), with sink nodes serving as roots of the graph, is logically formed over a mesh network by specifying how link costs and node attributes have to be combined in order to compute paths costs. RPL splits the DAG into one or more DODAGs (Destination Oriented DAGs), one DODAG per sink [25]. An example of a DAG and an appropriate DODAG is shown in Figure 1.

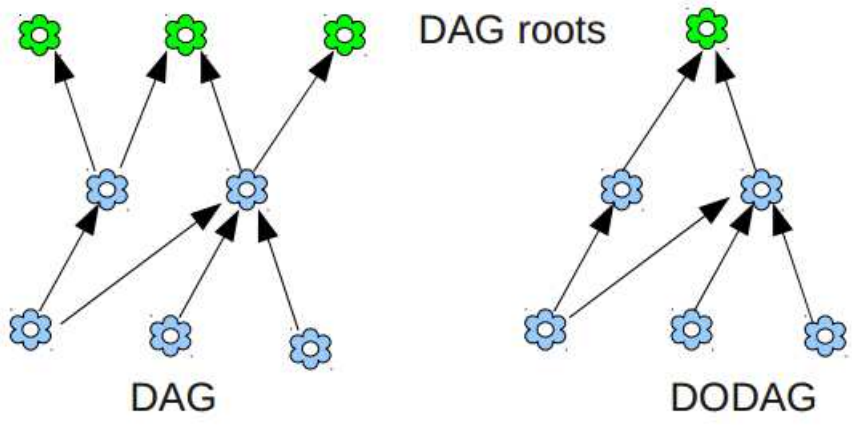

Figure 1: An example of a DAG terminated at sink nodes and a possible DODAG.

[4]"Internet of Things" supposes mostly the mentioned applications.

In order to define and maintain the topology RPL uses four main identifiers:

- *RPLInstanceID*, identifier of a specific RPL instance within the LLN. Each instance can serve different constraints and performance criteria.
- *DODAGID*, specifying one DODAG within a RPLInstance.
- *DODAGVersionNumber*, identifier used within the network in order to monitor changes in the topology. It is incremented each time DODAG is rebuilt.
- *Rank*, identifier of a position of a node in respect to a DODAG root (sink). The rank must monotonically increase as the DODAG is followed towards the DODAG destination. The exact calculation depends on the objective function used.

The RPL routing protocol specifies a set of new ICMPv6 control messages to exchange information related to a DODAG:

- *DODAG Information Solicitation* - ***DIS*** messages are used by a node in order to pro-actively solicit DODAG related information from neighboring nodes. They are usually transmitted when a node joins a LLN.
- *DODAG Information Object* - ***DIO*** control messages are used in order to define and maintain upward[5] routes.
- *DODAG Destination Advertisement Object* - ***DAO*** messages are used to advertise prefix reachability towards the leaf nodes of a DODAG. Therefore, they enable traffic to flow also in downward direction - from a sink towards sensors

A root starts the DODAG building process by transmitting a DIO message. Neighboring nodes will process DIO messages potentially from multiple nodes and make a decision on joining the DODAG based on the objective function and/or local policy. A node has a route towards the DODAG root[6] as soon as it joins the graph. The node computes its own *Rank* in respect to the root and starts advertising DIO messages to its neighbors, with updated information. However, if a node is not configured to act as a router and forward IPv6 packets, it is termed as a *leaf* node. In this case, upon reception of DIO messages, the leaf node selects its preferred parent and updates its routing table accordingly, without further advertising the DODAG. As the process converges, each node in the network will have received one or more DIO messages and will have a preferred parent selected. The preferred parent is therefore used as the next hop on the route towards the sink. The RPL protocol optimizes the upward routes or multipoint-to-point traffic, as described, as it accounts for most of the traffic in a LLN.

In order to support downward routes, RPL uses DAO control messages that are addressed as unicast towards the DODAG root. DAO messages describe prefix information, route lifetime and other information about the distance of the prefix. RPL defines two modes in which a network can operate in respect to managing downward routes:

[5]Common terminology is that a DODAG root - sink is logically placed at the top of the graph

[6]Node selected for a default upward route is termed as a *parent* of the node

- ***Storing mode*** - in which each node keeps track of all 'downlink' prefixes that are accessible through it. As each node joins the DODAG it will send DAO messages to the root via its preferred parent. Node learns the downward prefixes accessible and the next-hop node after processing transiting DAO messages from other nodes. In some cases, maintaining a routing table with downlink entries is not possible due to the memory constraints of sensor devices.

- ***Non-storing mode*** - Intermediate nodes do not process transient DAO messages apart for forwarding purposes. Each node unicasts a DAO message containing one or more of its parents towards the root. The root will eventually receive a DAO message from each node in the network and therefore will be able to construct downward routes towards any sensor in the network. The root can then transmit a message towards a node by using source routing. [27] discusses that this is a critical point of the protocol in case of large networks as the source routing header is limited and far-away nodes may not be accessible. Furthermore, the addition of source routing header increases the probability of fragmentation as MTU of IEEE 802.15.4 is just 127 bytes.

DIO and DAO control messages are used to enable multipoint-to-point (sensors towards sink) and point-to-multipoint (sink towards sensors) communication, respectively. Point-to-point communication (sensor-to-sensor) is enabled as a combination of the two mechanisms. Namely, a packet destined towards a certain node in the LLN in storing mode will travel up to the common ancestor in the DODAG, from where it will be forwarded downwards. In case of non-storing mode, the packet will travel all the way up to the root of the DODAG which will then forward the packet towards its destination. This mechanism is far from optimal but the RPL has been designed with an assumption that these situations are extremely rare. Nevertheless, [8] is one of the examples where such a scenario might be possible.

Emission interval of DIO control messages is regulated by the Trickle algorithm [34]. The idea is to reduce the control overhead of the protocol by emitting DIOs less frequently when there is no change in the topology. On the other hand, when there is an event signalizing that there has been a change in the network, DIOs are emitted more frequently. Complete specifications of the algorithm are given in [35]. On the contrary with DIO messages, emission interval of DAOs is not clearly specified in the RFC [11], and is left as an implementation choice. [26] presents results that are obtained with DAO emission period of 15 seconds which induces significant control overhead. For the remaining part of the training, the focus will be put on a performance study of the RPL when DAO emission interval is more adjusted to actual applications.

5 The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)

As opposed to RPL and its proactive approach, the LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) protocol [13] uses a more conservative - reactive concept. The main idea behind it, is that LLNs are idle most of the time and that a proactive approach in finding a route would only generate an unnecessary overhead. Thus, LOADng establishes a route towards a given destination only on demand - when there is some data to send. As a lot of work has been done in IETF on designing a reactive protocol for MANETs - named AODV [28] - a logical solution was to adjust it for LLNs. LOADng is a simplified version of AODV in order to make it implementable on memory constrained devices.

A device with a packet to send towards a given destination consults a routing table and in case there is no valid entry, it invokes LOADng. The protocol will *multicast* a ***route-request (RREQ)*** message, that will be diffused through the network in order to reach all other nodes. Each node that received a *RREQ* message will check if the destination of the message corresponds to any of the addresses from its address set. In case it does not, it will retransmit the message. The node also learns a reverse path towards the originator of the *RREQ* message and adds this to its routing table. Eventually, the destination node will receive the *RREQ*. It will respond to it by *unicasting* a ***route-reply (RREP)*** message towards the originator of the request. The *RREP* is forwarded by other nodes along previously installed reverse route. In the same time, intermediate nodes learn a forward route towards the destination and add this information in their routing tables. By the time *RREP* reaches the originator of the request, a bidirectional path is installed at intermediate nodes.

In case a link on the path is detected to be broken[7], the detecting node may perform a *route-repair* operation by transmitting a *RREQ* on behalf of the data source node. If *route-repair* fails, the detecting node unicasts a ***route-error (RERR)*** message back to the originator. Intermediate nodes receiving *RERR* message process it and update their routing tables accordingly.

One of the main drawbacks of LOADng, that is due to the reactive approach, is route discovery delay. During the process of discovery, all IP packets that should be transmitted are buffered in the source node, which may cause packet loss in case of memory constrained devices. Another issue is related to collisions of control messages due to the flooding which may lead to unnecessary retransmissions.

There are two major simplifications that have been done in LOADng with respect to AODV:

- Intermediate nodes are not allowed to transmit a *RREP* message even if they have a route towards the destination. This relaxes the need for destination sequence numbers that are used in AODV in order to ensure loop freedom.

[7]LOADng uses Link layer notifications in order to detect link failures

- AODV device keeps a "pre-cursor" list with each entry in the routing table, allowing it to notify all nodes that use a certain route if a link breaks [29]. LOADng avoids this in order to reduce the memory requirements.

In general, most of the performance characteristics of LOADng have been inherited and studied through the work on AODV. Therefore, as discussed in Section 3, there has been considerably less work around LOADng in respect to RPL.

6 Simulation Scenario

Home automation is one of the key applications of future Internet of Things. Nodes that comprise such networks are typical - highly constrained - LLN devices discussed in Section 1. These nodes have a variety of roles in order to perform different automation tasks. Good examples are light dimmers, window shades, motion sensors, typical monitoring sensors, remote control units, etc. Naturally, a controlling unit is also present and it usually communicates with other devices through the same medium.

When it comes to the network architecture, two approaches are generally used [8]:

- *Centralized architecture* - each device in the network communicates with the central controller that makes further decisions on actions to take. For instance, if a motion sensor detects presence in a room, it notifies the controller that may decide to turn on the light.

- *Distributed architecture* - devices within the network may communicate among themselves. In case of the previous example, the motion sensor may directly request the light dimmer to turn on the light in a room where motion was detected.

The majority of existing deployments uses the centralized approach but the distributed solution is also gaining popularity, namely for latency improvements. However, in a futuristic scenario, "smart homes" are expected to be part of a "smart grid" and a "smart city", with dynamical adjustments of their energy consumption and production. Thus, it is envisioned that a central unit, playing the role of a coordinator among the home devices and the external ones, has to exist. For this reason, we have decided to use the centralized architecture for our studies.

6.1 Topology

A very common simplification in the literature is the assumption of a grid-like network topology. The main consequence is a near-uniform distribution of each node's number of neighbors. This, however, does not reflect perfectly the common "smart home" deployment scenarios. Furthermore, it affects the performance of routing protocols. In order to give a fair comparison of RPL and LOADng protocols, a virtual 130 m^2 home plan was used and a realistic topology generator was developed.

We used the home plan as a background for generating topologies of different sized LLNs. Nowadays, networks of around 20 nodes are common. However, this number is expected to grow and to exceed 250 in the near future [8]. Due to the fixed-size area, topologies with larger number of devices in the network are denser, and consequently nodes have a higher number of neighbors.

Topologies used for our studies are generated with an assumption that a device is placed uniformly on the walls of a room with probability of 0.8, while it is distributed uniformly around the room with a probability of 0.2. Number of devices in each room is proportional to its area. For each topology, a controlling unit (sink) is placed in the central room (i.e. living room), according to the given placement criteria. 70 % of devices are assigned the logical role of a sensor, while the remaining 30 % are assigned the role of an actuator. Sensors can be typical monitoring sensors (temperature, humidity, CO_2, etc.) or event sensors (wall switches and motion detectors). The latter accounts for 30 % of all sensors.

An example topology for a network of 25 nodes with the virtual house plan used is given in Figure 2.

Figure 2: An example 25 node topology used for simulation studies with the virtual house plan used for generation.

6.2 Traffic Pattern

The traffic pattern of each node in the network depends on the assigned logical role. Four logical types were used in our studies:

- *Monitoring Sensor* - typical sensing node that periodically pushes one 802.15.4 frame to the controller. In order to avoid the undesired synchronization among nodes, the reporting period is repeatedly and uniformly selected from the [8..12] minutes interval, with 1 second resolution. Thus, on the average, the reporting interval of each sensor is 10 minutes.

- *Event Sensor* - a node that notifies the controller (i.e. pushes a frame) upon certain events. It can model the automated wall switches, motion detectors, etc. As the traffic pattern of this node typically depends on human actions, Poisson arrival process was used to model the number of events. Average number of arrivals per hour for the whole house was set to 10, as a simulation parameter. Thus, mean interarrival time of individual nodes is a function of total number of Event Sensors in the network.

- *Actuator* - a node that typically receives commands from the controller and sends back an application layer acknowledgment. It also periodically reports its state to the controller with the reporting period as used by the Monitoring Sensors.

- *Main Controller Unit* - Apart from sending the application layer acknowledgments for each received frame, this node also sends periodical bursts to different actuator nodes. The burst is sent in two cases: (i) upon reception of a frame from an event sensor (ii) upon expiration of previously extracted random interarrival time. The latter models automated, pre-programmed behavior of the controlling unit (for instance, to turn on the heating if the monitoring sensors reported drop of temperature bellow some threshold). Similarly to the case of Event Sensors, interarrival times between the bursts are exponentially distributed with the mean of 6 minutes (10 arrivals per hour). Size of each burst is 5 frames and different actuators are addressed each time.

It is important to note that the reporting periods and the general traffic pattern have been set as suggested in [8]. Nodes use UDP transport layer protocol and each received message is acknowledged by the application process. Therefore, traffic in the network is evenly distributed between point-to-multipoint and multipoint-to-point. Table 1 summarizes the traffic patterns of each node type.

6.3 Simulation Parameters

In order to have a realistic scenario, the Cooja simulator was used at the OS level - each device in the network under study is running an instance of Contiki OS with the whole protocol stack. The OS was compiled for the Tmote Sky platform that is based on MSP430 microcontroller with 10 KB of RAM and an IEEE 802.15.4 compliant radio [36]. Thus, the simulations performed are taking into account all the hardware constraints of the devices

Table 1: Traffic pattern of different node types present in the network.

Node Type	Traffic Pattern
Monitoring Sensor	Periodic reporting with [8..12] minutes interval
Event Sensor	Poisson process with mean of 10 arrivals (frames) per hour for the whole house
Actuator	Periodic reporting with [8..12] minutes interval and acknowledgment of each received frame
Main Controller Unit	Acknowledgment of each received frame; Send a burst to different actuators as a Poisson process with mean of 10 arrivals per hour and upon reception of a frame from an Event Sensor

under study. The most important settings of the Contiki OS and the Cooja simulator are summarized in Table 2.

Table 2: Contiki OS and Cooja simulator parameter setup.

Settings	Value
Wireless channel model	UDG Model with Distance Loss
Communication range	5 m
Mote type	Tmote Sky
Transport layer	UDP
Network layer	μIPv6 + 6LoWPAN
Max number of queued packets	2
MAC layer	CSMA + ContikiMAC
Radio interface	CC2420 2.4 GHz IEEE 802.15.4
Simulation time	8h

Notice in Table 2 a fairly low communication range at the PHY layer. Indeed, this choice was made with an assumption that nodes will operate with extremely low power (i.e. low power amplifier gain). Furthermore, radio-propagation obstacles (mostly walls, but also other objects) present in a typical home, coupled with 2.4 GHz frequency, limit the communication range of the nodes. With our choice, the whole house is covered within four hops, as suggested in [8]. It is also important to note that a modification to Contiki CSMA was made in order to buffer the multicast packets, and retransmit them in case the channel was found busy at the initial attempt (cf. Section 8). Without this modification, performance of LOADng protocol was highly degraded due to a very large number of dropped RREQ messages.

Another aspect to consider is that ContikiMAC was used as the Radio Duty Cycling (RDC) protocol [37]. This is reflected on the increased packet delays but is very important to take into account due to the nature of LLNs. Furthermore, as the Unit Disk Graph PHY Model (UDGM) does not introduce any losses, the use of ContikiMAC as the RDC

protocol takes into account also possible short-term link failures.[8]

Table 3: Protocol parameters.

Parameters	RPL	LOADng
DIO Min Interval	4s	
DIO Max Interval	1048s	
Mode of Operation	Storing mode	
Net Traversal Time		10s
Route Hold Time		600s, 1800s, 3600s

Table 3 presents RPL and LOADng protocol parameters used. Notice that we vary the Route Hold Time of LOADng in order to give an idea about the protocol performance with respect to the route lifetime. RPL uses the Trickle algorithm [35] to emit DIO messages - and the DIO Max Interval, that is used in steady state, has been set to approximately 17 minutes. Route Hold Time of LOADng and DIO Max Interval of RPL are two very related parameters whose tuning is an engineering challenge that should take into account the lossy behavior of the underlying PHY layer, as well as the probability of node failure. In networks we consider, time-selective channels, on both short and long term, are usually observed, due to the multipath propagation in wireless and the varying impedance in PLC channels [38]. Furthermore, as nodes are often battery-powered, probability of failure is non-negligible. Therefore, the values used to set these parameters need to be finite. Our choices for LOADng cover a broad range of links, while the margin to increase the DIO Max Interval of RPL for more stable networks exists. Nevertheless, the main effect of this parameter for RPL is control plane overhead, while the effect of Route Hold Time for LOADng is discussed in Section 7.

Results are averaged over 5 simulation runs and are shown with 95 % confidence interval.

7 Performance Evaluation

We focus our study on a 25 node network (cf. Figure 2) and evaluate LOADng and RPL protocols in terms of packet delays, hop counts and routing table size. Furthermore, we study control plane overhead and packet delivery ratio as a function of increasing number of nodes in order to give an idea how these two protocols scale for larger networks, that are expected in the later phase of home automation process [8].

7.1 Packet Delay

Latency studied in this section represents the total delay a packet experiences from the instant it is passed to the UDP layer until it is successfully received at the destination end.

[8]The ContikiMAC protocol transmits a single frame multiple times (termed strobes) until the acknowledgement is received. This accounts for the sleeping receiver as well as for the short-term link failure.

Cumulative Distribution Function (CDF) of packet delays separated by hop counts is given in Figures 3, 4 and 5. As expected, due to the reactive approach, LOADng performs much worse and the packets experience significantly higher delays. Even though typical LLN networks are usually delay insensitive, interactive nature of home automation applications may require a routing protocol with bounded delays. Due to the route discovery procedure of LOADng, delay CDF converges very slowly and 83 % of the packets are bounded within 2 seconds, in case of a 10 minute route hold time. Flooding is less frequent when the route hold time is increased and the delay performance becomes better. However, the delay CDF is still not bounded and approximately 7 % of packets experience a delay greater than 2 seconds (cf. Figure 3 - LOADng 1 hour hold). Pro-active route discovery of RPL results in bounded delay and 99.6 % of packets experience a delay smaller than 1.76 seconds, in case of one hop distance.

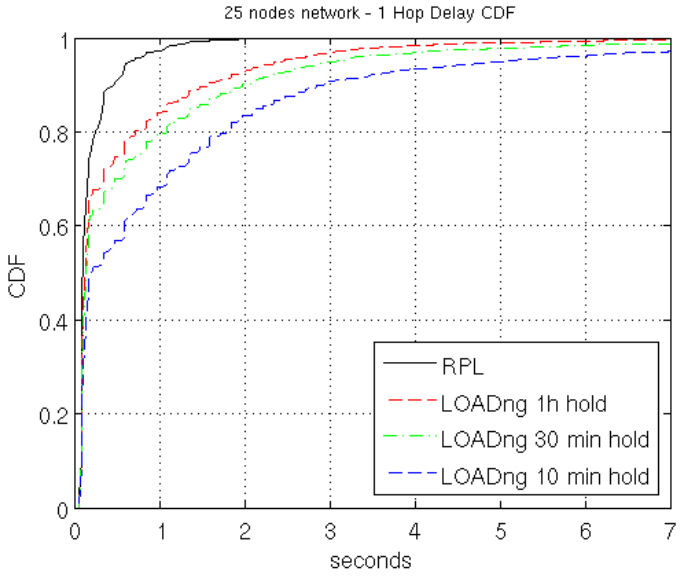

Figure 3: One hop packet delay CDF of a 25 node network.

Packets traversing one intermediate node (two hops away from the sink), experience higher delay, as expected (cf. Figure 4). In case of RPL, a very small percentage of packets (1.2 %) experience a delay greater than 2 seconds. Delay is in this case bounded within 2.4 seconds. With respect to the 1 hop distance delays, delay CDF of RPL converges slightly slower. Naturally, this is only due to the processing time and the channel contention at the intermediary node, as the nodes have a ready route. In case of LOADng and the 1 hour hold time, 15 % of packets experience a delay greater than 2 seconds.

Moving farther away from the sink, packet delays of RPL are barely affected. However, a significant increase in delay has been observed when it comes to the LOADng protocol. More precisely, 25 % of packets experience a delay greater than 2 seconds (cf. Figure 5 - LOADng 1h hold). In case of a lower - 10 minute - route hold time, approximately 40 % of packets undergo significant delay. The reason for the increasing delay with hop

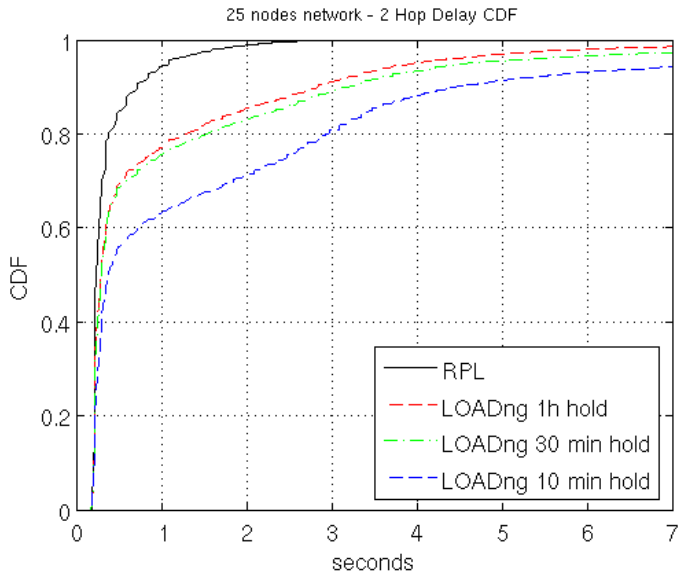

Figure 4: Two hop packet delay CDF of a 25 node network.

distance is the flooding operation of LOADng, as well as the instillation of forward routes. Basically, each node in the network is sending a RREQ with the target address of the sink. The sink responds with a RREP to the first, and each subsequent RREQ with a lower metric (due to the flooding, the sink will receive more than one copy of the same RREQ message). Nodes that form the reverse route and forward the RREPs towards its final destination instill the route towards the sink. The sink, however, does not have a route towards these intermediary nodes and upon reception of a packet from them will have to flood the network with another RREQ in order to send the acknowledgment frame. This will, naturally, result in increased delay, in respect to the packets that are destined to nodes physically closer to the sink.

7.2 Hop Distance

The optimization of constructed routes of the two protocols, for the 25 node network, was studied by path hop and packet hop distances. Path hop distance represents the average number of hops each packet on a given path traverses between a source and a destination. As the traffic in the network is either multipoint-to-point or point-to-multipoint, the sink node is either a source or a destination of each packet. Links in the network are considered unidirectional. It is interesting to note that while RPL has built only bidirectional paths, some path in LOADng networks have been observed as unidirectional.

CDF of path hop distances in the network is presented in Figure 6. Notice that the RPL routes are more optimized than those of LOADng. As mentioned in Section 6.1, the whole house is covered within four hops. We can see from the figure that the average path hop distances of RPL are bounded within four hops, which is not the case with LOADng.

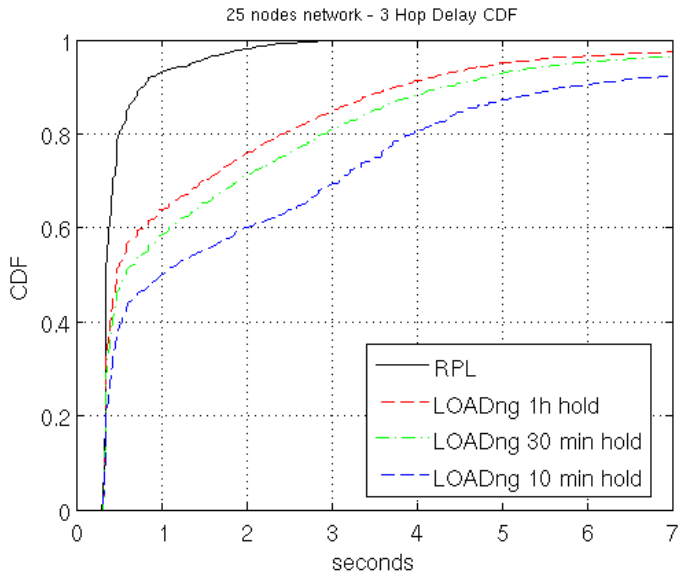

Figure 5: Three hop packet delay CDF of a 25 node network.

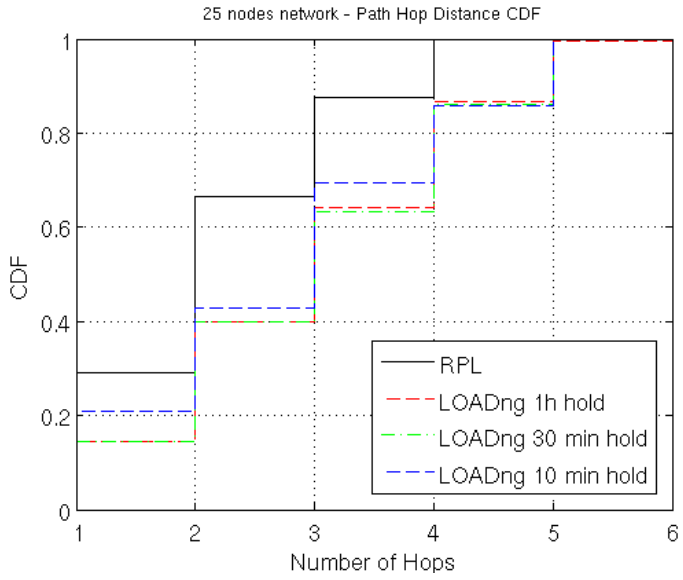

Figure 6: CDF of path hop distances in the network.

Moreover, 29 % of RPL routes are within one hop, while it is around 21 % for the LOADng, in the best case of a 10 minute hold time. This gap is even larger for two hop paths - 67 % of paths in case of RPL, while only 43 % in case of LOADng. Note that the LOADng provides more optimized routes in case of the shorter route hold time. There are two main reasons for this: (i) due to the flooding operation the channel is saturated and some RREQ messages are dropped because they reach the maximal number of attempts of the CSMA protocol; (ii) as the nodes are memory constrained (only able to store two packets

at a time), RREQ messages that should be forwarded are often dropped due to the full buffer. As a consequence, more frequent flooding (shorter route hold time) will supersede the lost RREQs and give a more optimized route as a result. There is, however, another aspect to this. Namely, if a node has a packet to send and the route is not ready, the most desired behavior (and the one implemented in our case) is to buffer the packet. Once the RREP arrives, the node will transmit the packet. Note, however, that the first arriving RREP is not necessarily the optimal one, in terms of the route. Therefore, the packet will traverse a non-optimized route. Indeed, if we take a look at Figure 7, we can notice that a non-negligible number of packets traverse more than four hops. Upon reception of a subsequent RREP, the node will update its routing table and the following packets will follow the updated route. It can be observed that the percentage of packets taking the non-optimized routes is similar for all three variants of LOADng.

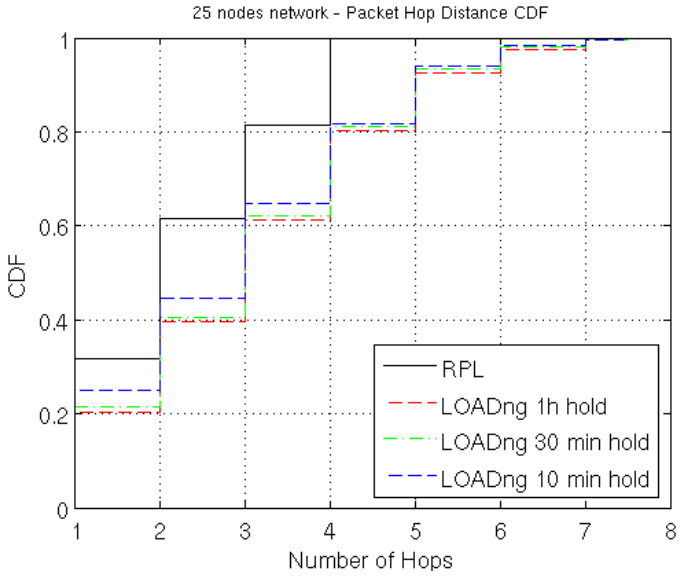

Figure 7: CDF of packet hop distances in the network.

On the other hand, routes that are found with RPL are very optimized, as we can see that not a single packet traverses more than 4 hops.

7.3 Routing Table Size

Due to the memory constraints of LLN devices, size of the routing table is a very important aspect of a routing protocol under consideration. Figure 8 shows the maximal number of routing table entries for each node in the 25 nodes network (cf. Figure 2).

Surprisingly, LOADng requires much higher number of entries with respect to RPL. Again, as a consequence of flooding, each node receiving a RREQ, or on a route of a RREP, is going to instill a route towards the sender. The result is a large number of unnecessary routes. If the implementation care is not taken so that RREP routes are given

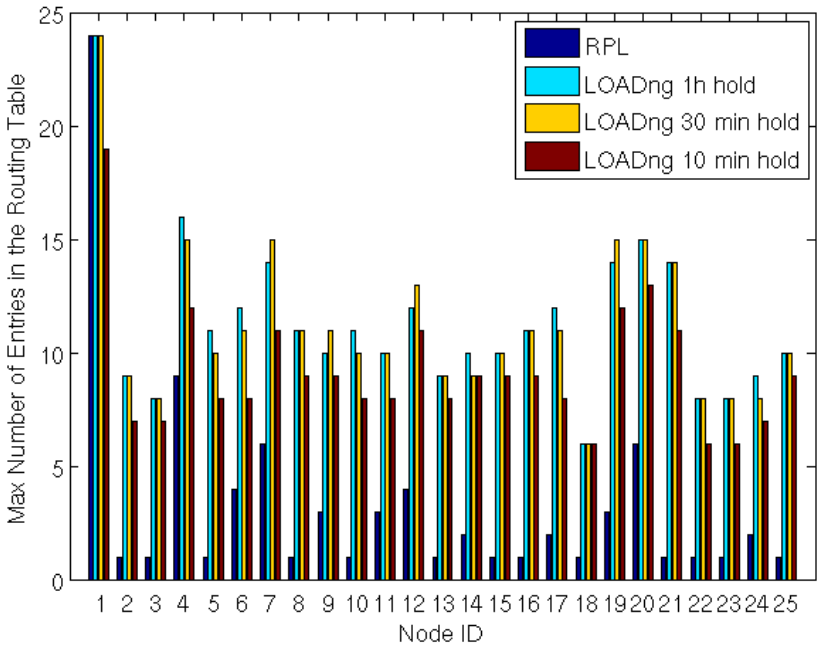

Figure 8: Maximal number of routing table entries at any given point of time during the simulation.

the priority, this fact could endanger the functioning of the protocol in case a node runs out of available memory. On the other hand, most nodes in the RPL network just have a default entry towards the preferred parent. However, depending on their position in the network, some nodes also have a significant number of entries. Namely, as the RPL is operating in "storing mode", intermediary nodes that are selected as preferred parents by others, have to store the downward routes. This is a critical point, because if an intermediary node runs out of memory, a loop may be formed. This, however, depends strongly on the topology of the network.

In order to see how each of the two protocols scale for larger network, we performed simulations studying the average number of route entries in a network, as a function of increasing number of nodes. These results are presented in Figure 9. We can notice that the main consequence of having a long route hold time (i.e. one hour) with LOADng is a sharply increased number of required routing entries. With the shorter lifetime, the protocol scales better as the routes expire faster and the average number of required entries is slowly increasing. With respect to RPL, the average number of routes seems to be slowly increasing. However, for the generated topology of 40 nodes, a significantly higher number of entries in the routing tables has been observed. As previously mentioned, the routing table size of a node in a RPL network strongly depends on its position and the exact topology.

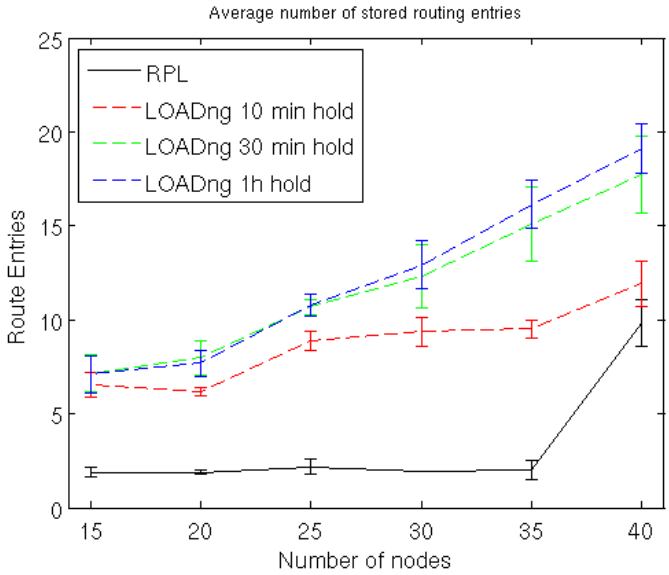

Figure 9: Average number of route entries in a network. Note that the sink node is not taken into account.

7.4 Overhead

The control plane overhead of the two protocols has been evaluated as a function of the increasing number of nodes. Although results in [26] show a significantly higher control plane overhead with RPL, this was not the case in our scenario. Total overhead in bytes over the simulation time for the two protocols can be observed in Figure 10. Note that the presented overhead is evaluated for each control message passed to the lower layers from the routing layer. Thus, it does not take into account the overhead ContikiMAC introduces at the RDC layer.

We can notice that the careful implementation of ContikiRPL results in much lower overhead than what is presented in [26]. As the RPL RFC under-specifies the DAO control message emission, results on overhead strongly depend on the implementation design. Furthermore, it is important to note that our simulation scenario did not trigger any global repairs of the RPL DODAG during the eight hours. In this way, we give an idea about the performance of the two protocols in steady state. Thus, most of the RPL overhead is observed at the beginning of the simulation and is significantly reduced later due to the Trickle algorithm [35]. Overhead of LOADng, on the other hand, strongly depends on the route hold time. The shorter is the route hold time, the more frequent the flooding is going to be. We can notice from the figure that the total amount of LOADng overhead sharply increases with the number of nodes, due to the high density of devices in the network. This could, however, be reduced with an optimized flooding algorithm.

Lifetime of battery-powered nodes is directly related to the usage of the radio transceiver. More precisely, in case of a Tmote Sky platform, power consumption due to the radio usage is three orders of magnitude larger than that due to the CPU processing [36]. With that

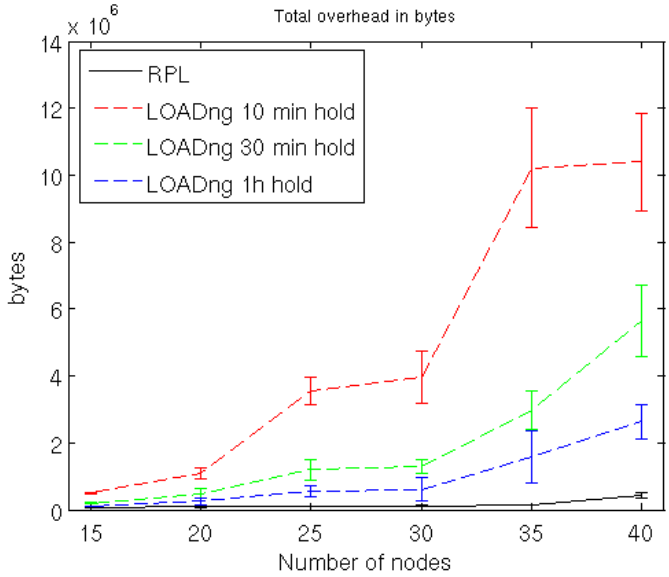

Figure 10: Total control plane overhead in bytes.

in mind, control plane overhead of a protocol is one of the determining factors of node's life expectancy. Our studies show that the control overhead of LOADng does not scale well with dense deployments of "smart home" applications. As the increasing number of devices in home networks is expected over time, this could significantly endanger overall network lifetime.

7.5 Packet Delivery Ratio

Packet delivery ratio as a function of number of nodes in the network is shown in Figure 11. We can notice that LOADng performs better than RPL for small networks. However, the PDR performance of LOADng is strongly related to the amount of flooding in the network. Furthermore, for denser networks the PDR of LOADng is sharply decreasing. This could be limited with an optimized flooding scheme. Performance of RPL, on the other hand, is surprisingly low and this will be the topic of further investigations.

8 Challenges Encountered and Implementation Difficulties

Throughout the duration of the training, a number of implementation and conceptual challenges have arisen which needed to be solved. A non-exhaustive list, in chronological order of appearance, with brief summary is given bellow.

- *Incompatibility of the LOADng implementation with the latest version of Contiki* - Due to the constant updates of Contiki OS, more precisely changes in the build

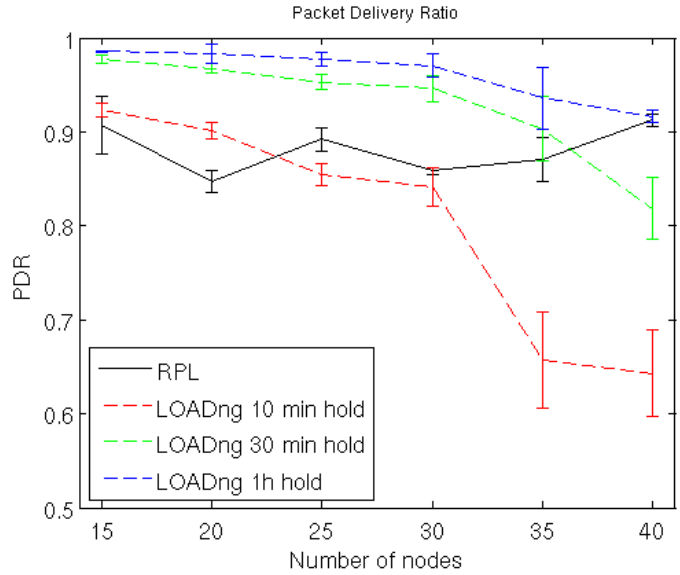

Figure 11: Packet delivery ratio.

process and the parameter setup, the prototype of LOADng did not function properly with the official release of Contiki 2.5. The problem was in the misconfiguration of the IPv6 Neighbor Discovery (ND) protocol.

- *Incompatibility of the LOADng implementation with certain application architectures* - The prototype did not function properly for applications that would manually set the IPv6 address. This was a consequence of the fact that the LOADng implementation only took into account nodes' IPv6 addresses that are present at boot time. The application processes, where manual IPv6 addresses are added, are normally polled after the system processes at boot time. Thus, the node receiving a RREQ message with the manually added target IPv6 address would not recognize itself as the destination.

- *The LOADng implementation did not buffer packets awaiting for a route* - Although the LOADng draft does not specify what should a node do with such packets, simply dropping the packet would significantly reduce the Packet Delivery Ratio of the protocol. Therefore, it is usually desired that a node buffers the packet until reception of a RREP. This modification has been introduced to the prototype so that the packets are buffered for 2 * NET_TRAVERSAL_TIME seconds, accounting for the maximal delay that may be experienced.

- *An issue with Contiki CSMA* - Default implementation of CSMA protocol in Contiki does not retransmit the multicast frames in case the channel was found busy at the initial attempt. This highly degraded the performance of LOADng protocol as a large part of RREQ control messages were simply dropped during the flooding operation. The usual behavior of this protocol for multicast frames is that a frame should be

transmitted once. The CSMA implementation in Contiki has, thus, been modified in order to retransmit multicast frames.

- *Design and implementation of a smart home scenario* - In order to set a realistic test for the two protocols, traffic patterns of the devices in the network have been designed as suggested in [8]. They are briefed in Section 6.2. The nodes in the simulations are running an instance of Contiki OS with a separate application process for each node type. As the network topologies vary with deployments, we have decided to use a virtual home plan that was considered for generation of various topologies. The idea is that the "home automation" networks are going to have an increasing number of devices over time. Therefore, our topology generator uses the same virtual home plan to generate denser topologies for the increasing number of nodes. The generation process is briefed in Section 6.1.

- *Implementation of node types in Contiki* - The main difficulties regarding the implementation of the node types in Contiki were due to the absence of a floating-point unit on the Tmote Sky platform. Therefore, float data types are not inherently supported. In order to generate exponential random variables the use of the logarithm function is needed. Thankfully, a lightweight implementation of the floating point arithmetics library for Contiki was available internally for the Drakkar team. Another issue experienced is related to the memory constraints of the nodes (only 10 KB of RAM). Sensor and actuator nodes do not perform much processing and their memory requirements are low. However, in order to keep track of the losses, packets had to be stamped with sequence numbers. Sensor and actuator nodes only need to store a single counter as all the packets they transmit are addressed to the sink. The sink, however, needs to keep track of the sequence numbers for each node in the network. The problem arises with larger networks as the sink node runs out of memory. Furthermore, in some instances when the node was operating "on the limit", the memory map of the OS would get corrupted.

- *Loops found in RPL topologies* - During the simulation studies phase, a small number of paths in the RPL networks were found to be unavailable. Indeed, loops were observed in the topology and packets would eventually get dropped due to the zero time-to-live field in the IPv6 header. This was a consequence of an RPL implementation bug in official release of Contiki 2.5. The loop would form in case a No-path DAO message, notifying the preferred parent that a new preferred parent has been selected, would get lost. A patch solving this issue was posted with Contiki release 552 in April 2012.

9 Implementation - Formal Background and Specifics

This section briefs the implementation specifics of the Topology Generator and the theoretical background of various choices taken during the training.

9.1 The Topology Generator

A realistic generator of topologies for "smart home" applications has to take into account the spatial plan of the object. Thus, first step in the generation procedure is to import the specifics of a home plan. This is done by specifying the origin of each room in Cartesian coordinates. Additional object-dependent parameters are also specified, such as wall thickness and ceiling height. Moreover, topology-related parameters are defined - ratio of sensors/actuators in the room, probability that a device is placed on walls and the room where the central controller will be placed. The main input to the generator is the desired number of nodes in the house.

The first step in the generation procedure is to define the exact number of devices in each room. The following relation has been used:

$$n = round(\frac{roomArea * densityCoefficient}{totalArea} * (N - 1)) \quad (1)$$

Where n denotes number of devices in a room; $roomArea$ denotes area of the room; $totalArea$ denotes total area of the house, $densityCoefficient$ is a special coefficient used in order to differentiate between rooms with higher and lower device density; N denotes total number of devices in the house. Note that in topologies used for our studies, $densityCoefficient$ for each room has been set to 1. Thus, the number of devices in each room solely depends on the size of the room. Furthermore, $N - 1$ is used instead of N in order to account for the deterministically placed controller node[9]. Clearly, due to the rounding, the sum of n over each room does not necessarily correspond to the total number of devices in the network, initially defined. Thus, further processing is done in order to ensure that the total number of devices in the house is equal to the value initially set.

The second step, after the number of devices in each room has been defined, is to determine the device type. This is done by simply using the defined percentages of each device type. Similarly to the previous step, due to the rounding, further processing needs to be done in order to ensure the consistent number of devices.

The third step corresponds to the placement of devices. A list of all rooms is traversed, and devices are placed according to the defined probabilities. For each device, a uniformly distributed random number x is extracted from the [0..1] interval. The device is placed on the walls of the room in case:

$$x \leq wallProbability \quad (2)$$

Otherwise, it is placed around the room. The exact position of the device is then determined. This is done by extracting additional uniformly distributed random numbers in the intervals [0..*roomWidth*], [0..*roomLength*], [0..*ceilingHeight*], for each of the three dimensions. Clearly, in case the device has been selected for a wall placement, one of the five possibilities is similarly selected (four walls and the ceiling).

[9]The controller node is placed deterministically in a predefined room but at a random position

The final step consists of generating the XML output file that will be fed to Cooja. At this time, different Contiki application processes are linked with the appropriate node types (monitoring sensors, event sensors, actuators, controller). An example topology for 175 nodes is shown in Figure 12.

Figure 12: An example topology with 175 nodes. The green node (ID 1) is the controller, monitoring sensors are represented with orange, actuators with purple, and event sensors with yellow color.

9.2 Random Number Generation

Generation of random numbers has been used for the Topology Generator as well as for Contiki applications. The Topology Generator assumes uniformly distributed random variables and a typical Congruential generator of pseudo-random sequences has been used:

$$X_{n+1} = a * X_n \ mod \ m \tag{3}$$

In Eq. 3, integer X_n is the input to the routine and X_{n+1} is the output. In case of the Topology Generator that executes on 32 or 64 bit platforms, following values have been used, as suggested in [39]:

$$a = 16807 \tag{4}$$
$$m = 2147483647 \tag{5}$$

However, the Tmote Sky platform has a 16 bit MCU so the parameters had to be adjusted [39]:

$$a = 219 \tag{6}$$
$$m = 32749 \tag{7}$$

Consistent generation of uniformly distributed pseudo-random numbers is a very important aspect of the implementation process. Moreover, almost all methods for generation of instances of variously distributed random variables, in some form, take as input an instance of a uniformly distributed random variable.

Indeed, as discussed in Section 6.2, event sensors and the controller use a Poisson process to model the number of events (arrivals). The implementation exploits the basic property of a Poisson process that interarrival times are exponentially distributed. Thus, in order to generate instances of exponential random variables, the inverse transformation method has been used.

Formally, an instance of a random variable X, whose CDF is denoted by $F_X(x)$ can be obtained from:

$$x = F_X^{-1}(u) \tag{8}$$

where x denotes an instance of the desired random variable, and u denotes an instance of a random variable uniformly distributed in [0..1]. Thus, as the CDF of exponentially distributed random variable Y is given by:

$$F_Y(y) = 1 - e^{-\lambda y} \tag{9}$$

By exploiting Eq. 8:

$$x = -\frac{1}{\lambda} \ln (1 - u) \tag{10}$$

where $\frac{1}{\lambda}$ is the mean of the exponential random variable. Therefore, in order to generate an instance of exponential random variable, an instance u generated by Eq 3, is substituted in Eq. 10. Implementation difficulties of Eq. 10 on the Tmote Sky platform are discussed in Section 8.

Another interesting aspect has appeared during the implementation of event sensor nodes. Namely, the traffic pattern in the network assumes mean number of events for the whole house, and not for individual nodes. In order to implement this, a property of a Poisson process has been used that states that a sum of independent Poisson processes is a Poisson process. More precisely, let $N(t)$ denote a Poisson process defined by:

$$N(t) = \sum_{i=1}^{M} N_i(t) \tag{11}$$

where M denotes the total number of event sensors in the house, and $N_i(t)$ is the Poisson process with mean interarrival time $\frac{1}{M\lambda}$. Then, the mean interarrival time of $N(t)$ will be:

$$\sum_{i=1}^{M} \frac{1}{M\lambda} = \frac{1}{\lambda} \tag{12}$$

Thus, if the reporting period of individual event sensor nodes is modeled with a Poisson process with mean interarrival time $\frac{1}{M\lambda}$, the controller will on average receive a frame each $\frac{1}{\lambda}$ time units.

9.3 Confidence Intervals

As each metric measured during a simulation run represents only one instance of a random variable, statistical processing of simulation outputs is necessary. Thus, each point presented in Section 7 is averaged over five simulation runs. However, estimation of the average over finite number of samples introduces an error. Confidence interval is an estimation of this error and tells us the interval in which the real average is found with a given certainty (usually, 90 %, 95 % or 99 %). In order to calculate confidence intervals, three assumptions are made:

1. The observed process is stationary.
2. Estimated random variable X has a mean m and a variance σ^2.
3. The n observed instances of $X(x_1, x_2, ...x_n)$ are independent-identically distributed (iid) variables.

By the central limit theorem, for large n, the estimate is normally distributed. However, as we are only estimating the mean over five simulation runs[10], assuming normal

[10]Simulations are averaged over 5 simulation runs for practical reasons - i.e. one run of a 25 node simulation lasts approximately 5 hours, while the simulation of a 40 node network takes 17 hours. As the results were presented for 6 network sizes, and a separate simulation run is needed for each protocol and its variant, averaging over larger number of samples would be unfeasible during the duration of the training period.

distribution would introduce a significant error. Therefore, Student's T-distribution with four degrees of freedom has been used. Confidence interval is then given by:

$$I = [\bar{x} - \frac{\sigma t_{n-1,\alpha/2}}{\sqrt{n}}, \bar{x} + \frac{\sigma t_{n-1,\alpha/2}}{\sqrt{n}}] \tag{13}$$

In Eq. 13, $t_{n-1,\alpha/2}$ is a Student's T-distribution critical value, that depends on the number of degrees of freedom, as well as on the desired certainty (95 % in our case); $\bar{x}$ is a mean estimator:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{14}$$

and σ is an estimator of the standard deviation:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{15}$$

10 Personal Assessment and the Work Plan

Work plan of the training has been organized in four main phases:

1. Bibliography readings and introduction to Contiki and Cooja.
2. Modifications of the existing implementation of RPL and the prototype of LOADng.
3. Implementation of the topology generator, Contiki applications for each node type and simulation automation scripts.
4. Simulation studies

Apart from the usual introduction to the subject by a review of wide background material summarized at the end of this document, Phase 1 also dealt with an acquaintance to Contiki OS and Cooja simulator for LLNs. The latter was done by writing basic programs in C for Contiki and going through introduction tutorials [40, 41].

Contiki OS provides an open-source implementation of RPL called ContikiRPL [42] that was used for our simulations. A number of papers has been published, and summarized in Section 3, that use this implementation and framework in [23] for performance measurements of RPL. Only minimal modifications of parameter settings were needed in order to adjust it for the scenario that was used in our study. An existing prototype implementation of LOADng protocol developed internally at the DRAKKAR team was used for performance analysis. However, couple of aspects regarding this prototype had to be modified and are summarized in Section 8.

This training has given the author a valuable insight into the research area of sensor networks. Moreover, due to the open-source nature of the Contiki project, the author had an opportunity to experiment with software modifications of each layer of the operating

system - from the application to radio duty cycling drivers. Indeed, a significant issue was found in the implementation of Contiki CSMA, and is briefed in Section 8, that highly degraded the performance of the LOADng protocol. The resolution of this problem had to take into account implementation specifics of the surrounding layers - 6LoWPAN and ContikiMAC RDC. This has provided the author with a valuable experience appart of the routing layer which was the main topic of the training. Furthermore, during the Phase 3, the author has worked with a variety of tools, programming and scripting languages in order to automate the simulation process and the extraction of results. Aside of the C programming language that was used in order to implement the topology generator and the applications in Contiki, JAVA was used for small modifications of the Cooja simulator. Furthermore, Cooja simulation scripts were coded in JavaScript. Parsing of the results and the generation of plot files was done in Perl, while the sequential execution of commands with logical flow of the simulations was done in the Bash scripting language. Coding in, both, Perl and JavaScript, was new to the author.

Apart from the implementation difficulties, whose resolution has provided a significant engineering experience, results of the simulation studies are contributing to the research community around 6LoWPANs. Namely, there has been a lot of discussion recently regarding the applicability scenarios of each of the two protocols. The "home automation" scenario consists an important part of the future "Internet of Things", and the comparison of LOADng and RPL protocols in such a case is surely valuable. Furthermore, aspects of LOADng that could improve its performance have been identified - LLN applicable optimized flooding algorithm, jitter-ed RREQ messages and a more "intelligent" route storing procedure, that would lower the memory requirements.

The initial work plan of the training (cf. Figure 13) presented in the Preliminary Report has been partly met. The plan assumed simulation studies of four scenarios - home, building, urban and industrial automation - but the experience showed that a single thorough simulation analysis requires much more time than what was initially thought. Therefore, in order to provide consistent and high-quality results for a realistic scenario, a decision has been made to devote all the available time to the "home automation" scenario, even though scenarios for other three applications have been designed. Furthermore, a considerable amount of time (around 4 weeks) was spent on implementing the designed "home automation" scenario - the topology generator, Contiki applications for each node type, Cooja scripts and result parsing scripts. It is important to note, however, that most of these tools with slight modifications can be reused for other scenarios.

Summa summarum, the training has met all the initial goals and expectations, and has prepared the author to continue his research career in the area of Sensor Networks.

11 Conclusion

Routing is one of the key issues for LLNs and has been a hot topic in the research community around sensor networks for several years. RPL and LOADng are the two major protocol candidates for standardization in 6LoWPANs.

Authors in [26] presented a comparative performance study of the two protocols and showed that RPL introduces a significant control overhead in case of bidirectional traffic, in respect to LOADng. The study demonstrated, however, that RPL performs much better than LOADng in terms of finding a more optimized route. Nevertheless, authors used a different application scenario for the two protocols which leads to an unfair comparison. Furthermore, the parameter choice of DAOInterval taken in the simulation in case of RPL is arguable and is the direct cause of such a high control overhead.

We confirmed the previous findings that RPL provides more optimized routes in respect to LOADng but also showed that the introduced overhead of RPL strongly depends on the implementation choices taken. This is due to the under-specification of the RFC 6550 in respect to the DAO control message emission interval. Thus, in case of a widely-accepted ContikiRPL implementation, control plane overhead of RPL was found to be significantly lower than that of LOADng. The margin between the two increases with the increasing number of nodes in the network, due to the LOADng flooding which does not scale well for larger networks. It is important to note, however, that our scenario assumed typical "smart home" topologies where the density of devices in the network increases.

We presented a delay study for the two protocols and showed that LOADng is not suitable for interactive applications of "smart home" networks, due to the unbounded delay CDF. Furthermore, a significant increase in delay has been observed for packets traversing multiple hops. On the other hand, RPL constructed a more optimized routes tree that kept the small depth in a dense network.

Memory usage of the two protocols has been evaluated by comparison of maximal number of entries in the routing table. LOADng was found to require much larger routing tables, due to the instillation of unnecessary routes. Most nodes in the RPL network have a very low number of entries in the table. However, depending on the formed DODAG, some nodes close to the root have been found to require a large number of entries in the table, as well. These nodes are selected as preferred parents by others and therefore have to store the downward routes, as the network was operating in the "storing mode". This could be a critical point of the protocol in case such a node runs out of memory, as the loops would be formed. Additional aspect that should also be taken into account is the implementation complexity and the RAM usage by the protocol operation. RPL specification is much more complex, which leaves less memory for application processes on constrained devices.

Finally, packet delivery ratio has been evaluated as a function of increasing number of nodes in the network. LOADng performed very well in case of small networks but the number of dropped packets sharply increases with the increased density of the network. RPL performance was surprisingly low and this will be the topic of further investigations.

Everything being said, at the current stage of specifications, LOADng could serve well sparse network deployments with low-priority traffic where route hold time could be set to a very large value or even infinity. However, for applications regarding "home automation", where the interactive aspect is playing a significant role, this is not possible, and our results suggest that LOADng may not be the best candidate. RPL, on the other hand, has shown better results but at the price of significant complexity. In fact, reducing the specification complexity of RPL, at the minimal performance price, is an interesting research challenge

that will be addressed in the future.

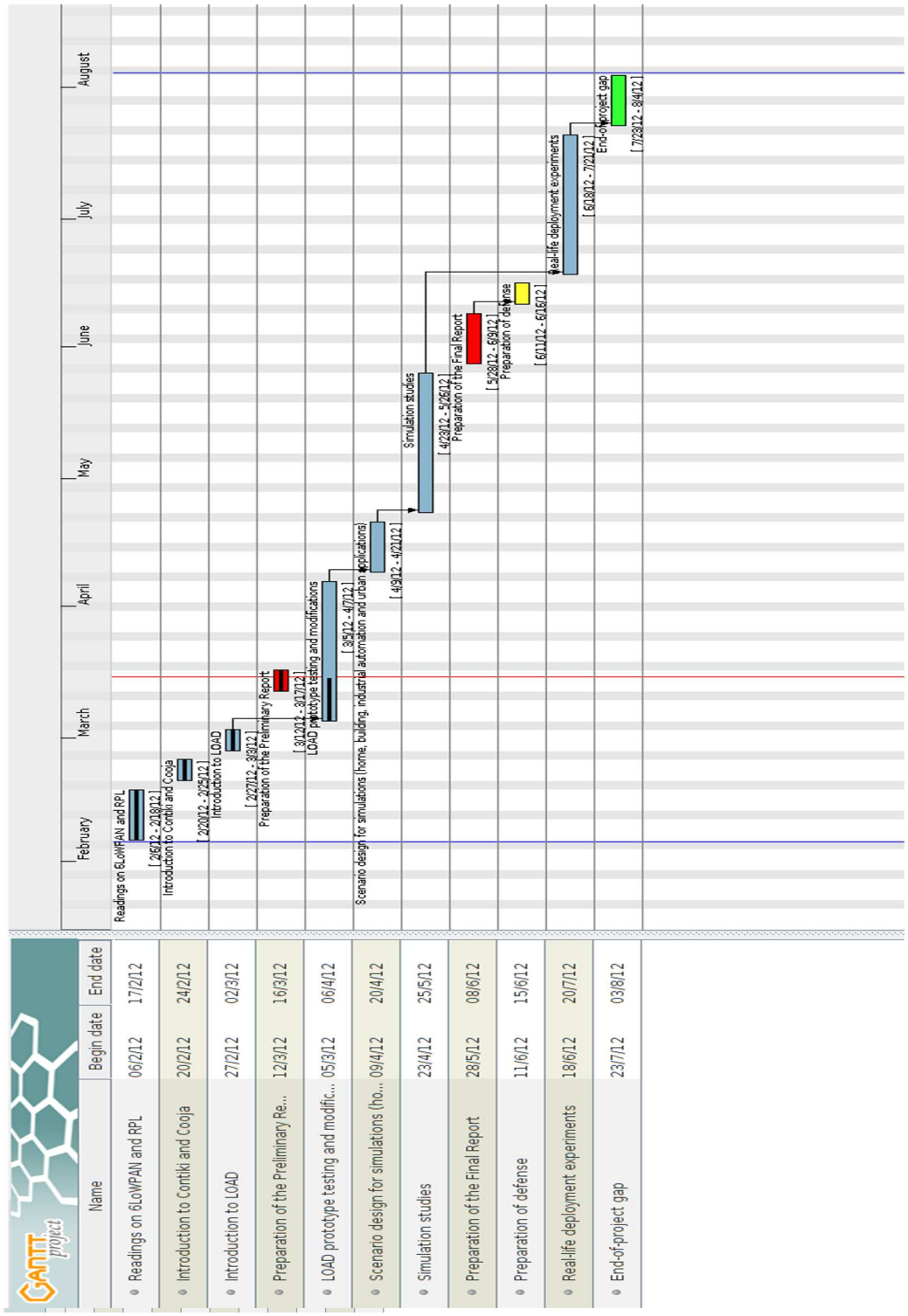

Figure 13: Gantt diagram of the training project.

References

[1] I.F. Akyildiz, T. Melodia, and K.R. Chowdury. Wireless multimedia sensor networks: A survey. *Wireless Communications, IEEE*, 14(6):32 –39, december 2007.

[2] IEEE Standard 802.15.4-2003.

[3] C. Gomez and J. Paradells. Wireless home automation networks: A survey of architectures and technologies. *Communications Magazine, IEEE*, 48(6):92 –101, june 2010.

[4] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement and goals. RFC 4919, IETF, 2007.

[5] Julien Abeille, Mathilder Durvy, Jonathan Hui, and Stephen Dawson-Haggerty. Lightweight IPv6 stacks for smart objects: the experience of three independent and interoperable implementations. Whitepaper, Internet Protocol for Smart Objects (IPSO) Alliance, 2008.

[6] G. Montenegro, N. Kushalnagar, and J. Hui. Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944, IETF, 2007.

[7] J. Martocci. Building Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5867, IETF, June 2010.

[8] A. Brandt, J. Buron, and G. Porcu. Home Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5826, IETF, Apr. 2010.

[9] K. Pister and P. Thubert. Industrial Routing Requirements in Low-Power and Lossy Networks. RFC 5673, IETF, Oct. 2009.

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

[11] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and JP Vasseur. RPL: IPv6 routing protocol for low power and lossy networks. RFC 6550, IETF, April 2012.

[12] P. Levis, A. Tavakolli, and S. Dawson-Haggerty. Overview of existing routing protocols for low power and lossy networks. Internet draft, IETF, 2009.

[13] T. Clausen, A. Colin de Verdiere, J. Yi, A. Niktash, Y. Igarashi, and U. Herberg. The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng). Internet draft, IETF, April 2012.

[14] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455 – 462, nov. 2004.

[15] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641 –648, nov. 2006.

[16] Martin Hervé. LIG - Laboratoire d'Informatique de Grenoble: A word from the director. http://www.liglab.fr/spip.php?article107. Accessed March 15th 2012.

[17] Andrzej Duda. Team DRAKKAR. Internal Report, 2009.

[18] A. Tripathi, J. de Oliveira, and JP. Vasseur. Performance evaluation of routing protocol for low power and lossy networks (RPL). Internet draft, IETF, 2012.

[19] M. Nuvolone. Stability analysis of the delays of the routing protocol over low power and lossy networks. Master's thesis, KTH Royal Institute of Technology, 2010.

[20] Thomas Clausen and Ulrich Herberg. Multipoint-to-point and broadcast in RPL. Technical Report No. 7244, INRIA, 2010.

[21] T. Clausen and U. Herberg. Comparative study of RPL-enabled optimized broadcast in wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2010 Sixth International Conference on*, pages 7 –12, dec. 2010.

[22] JeongGil Ko, Stephen Dawson-Haggerty, Omprakash Gnawali, David Culler, and Andreas Terzis. Evaluating the performance of RPL and 6LoWPAN in TinyOS. In *IPSN'11*, 2011.

[23] Nicolas Tsiftes, Joakim Eriksson, Niclas Finne, Fredrik Österlind, Joel Höglund, and Adam Dunkels. A framework for low-power IPv6 routing simulation, experimentation, and evaluation. *SIGCOMM Comput. Commun. Rev.*, 40(4):479–480, August 2010.

[24] Leila Ben Saad, Cedric Chauvenet, and Bernard Tourancheau. Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies. In *International Conference on Sensor Technologies and Applications SENSORCOMM 2011*, Nice, France, August 2011. IARIA.

[25] N. Accettura, L.A. Grieco, G. Boggia, and P. Camarda. Performance analysis of the RPL routing protocol. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 767 –772, april 2011.

[26] Ulrich Herberg and Thomas Clausen. A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN). In *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '11, pages 73–80, New York, NY, USA, 2011. ACM.

[27] T. Clausen, J. Yi, and U. Herberg. Experiences with RPL: IPv6 routing protocol for low power and lossy networks. Internet draft, IETF, 2012.

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

[29] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells. Adapting AODV for IEEE 802.15.4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment. In *World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a*, pages 9 pp. –170, 0-0 2006.

[30] Vladimir Iliev and J. Schoenwaelder. Mesh routing for low-power mobile ad-hoc wireless sensor networks using LOAD. Guided research report, Jacobs University Bremen, 2007.

[31] Jian-Ming Chang, Ting-Yun Chi, Hsin-Yun Yang, and Han-Chieh Chao. The 6LoWPAN ad-hoc on demand distance vector routing with multi-path scheme. In *Frontier Computing. Theory, Technologies and Applications, 2010 IET International Conference on*, pages 204 –209, aug. 2010.

[32] Gee Keng Ee, Chee Kyun Ng, Nor Kamariah Noordin, and Borhanuddin Ali Mohd. A review of 6LoWPAN routing protocols. In *Proceedings of the Asia Pacific Advanced Network*, 2010.

[33] JP Vasseur, Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertrand, and Cedric Chauvenet. RPL: The IP routing protocol designed for low power and lossy networks. Technical report, Internet Protocol for Smart Objects (IPSO) Alliance, 2011.

[34] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI*, pages 15–28, 2004.

[35] P. Levis, T. Clausen, J. Hui, O. Gnawali, and Ko J. The Trickle Algorithm. Standards Track RFC 6206, IETF, 2011.

[36] Moteiv. Tmote Sky - ultra low power IEEE 802.15.4 compliant wireless sensor module. Datasheet.

[37] Adam Dunkels. The ContikiMAC radio duty cycling protocol. Technical report, Swedish Institute of Computer Science, SICS, 2011.

[38] Sabih Güzelgöz, Hüseyin Arslan, Arif Islam, and Alexander Domijan. A review of wireless and PLC propagation channel characteristics for smart grid environments. *Journal of Electrical and Computer Engineering*, vol. 2011(ID 154040):12 pages, 2011.

[39] Pierre L'Ecuyer. Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation*, 68(225):249–260, 1999.

[40] SICS. ContikiWiki. http://www.sics.se/contiki/wiki/index.php/Main_Page, 2010.

[41] Zolertia. Contiki Lesson 0. http://zolertia.sourceforge.net/wiki/index.php/Mainpage:Contiki_Lesson_0, 2010.

[42] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. Low-power wireless IPv6 routing with ContikiRPL. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 406–407, New York, NY, USA, 2010. ACM.

[43] Aminul Haque Chowdhury, Muhammad Ikram, Hyon-Soo Cha, Hassen Redwan, S. M. Saif Shams, Ki-Hyung Kim, and Seung-Wha Yoo. Route-over vs mesh-under routing in 6LoWPAN. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 1208–1212, New York, NY, USA, 2009. ACM.

[44] Adam Dunkels and JP Vasseur. IP for smart objects. Whitepaper, Internet Protocol for Smart Objects (IPSO) Alliance, 2009.

[45] Jonathan Hui, David Culler, and Samita Chakrabarti. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture. Technical report, Internet Protocol for Smart Objects (IPSO) Alliance, 2009.

[46] Alessandro Ludovici, Anna Calveras, and Jordi Casademont. Forwarding techniques for IP fragmented packets in a real 6LoWPAN network. *Sensors*, Vol. 11:992–1008, 2011.

[47] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364 – 369, april 2005.

[48] Thomas Clausen and Ulrich Herberg. A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN). Technical Report No. 7637, INRIA, June 2011.

[49] Thomas Clausen, Ulrich Herberg, and Matthias Philipp. A critical evaluation of the "IPv6 routing protocol for low power and lossy networks" (RPL). Technical Report No. 7633, INRIA, May 2011.