Poster Abstract: Low-Power Wireless IPv6 Routing with ContikiRPL

Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels
{nvt,joakime,adam}@sics.se
Swedish Institute of Computer Science
Kista, Sweden

ABSTRACT

RPL is the IETF candidate standard for IPv6 routing in low-power wireless sensor networks. We present the first experimental results of RPL which we have obtained with our ContikiRPL implementation. Our results show that Tmote Sky motes running IPv6 with RPL routing have a battery lifetime of years, while delivering 0.6 packets per second to a sink node.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Experimentation, Measurement, Performance

1. INTRODUCTION

IPv6 is emerging as a communication standard for sensor networks and smart objects [4]. In recent years, industry and academia have optimized IPv6 for low-power wireless systems based on the IEEE 802.15.4 standard, and the result is often called 6lowpan [3]. Still, routing is a key part of the IPv6 stack that remains to be specified for such networks.

RPL is on the IETF standards track for routing in low-power and lossy networks [4, 5]. The protocol is tree-oriented in the sense that one or more root nodes in a network may generate a topology that trickles downward to leaf nodes. Much flexibility is given in RPL as to the constraints and metrics that can be used when building a topology. Studies on RPL regarding link churn exist, but they are only on protocol level. Practical experience from implementations in resource-constrained systems has until now been lacking.

We have designed and implemented RPL inside the uIPv6 [1] stack. We show early results on the power efficiency and the implementation complexity of ContikiRPL. Our results show that IPv6 routing with ContikiRPL is both lightweight and power-efficient.

2. IPV6 ROUTING WITH RPL

RPL involves many concepts that make it a flexible protocol, but also rather complex. We give an overview of the main ideas, but refer to the draft specifications released by the ROLL working group for further reading [5].

Topology Formation. A network that manages its routing topologies using RPL may run one or more RPL instances. Each instance defines a topology that is built using a unique metric or constraint within the network. In each RPL instance, multiple Directed Acyclic Graphs (DAGs) may exist, each having a different DAG root. A node may join multiple RPL instances, but must only belong to one DAG within each instance. We illustrate the operation of multiple RPL instances in Figure 1.

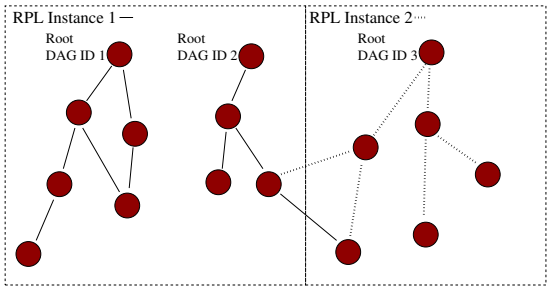

Figure 1: A network with three DAGs in two RPL instances.

When forming the topology, each sink constructs a packet called a DAG Information Object (DIO), and sends it to all children. Any child that decides to join the DAG may pass the DIO further to its own children. The DIO contains a rank that is increased monotonically when the child joins the DAG. The rank prevents routing loops, and helps the nodes to distinguish between parents and siblings. Nodes may store a set of candidate parents and siblings that can be used if the preferred parent is unable to route traffic for the node.

Maintenance. DAGs may need to change if the network restructures because of mobility or link quality variance. RPL ensures that DAGs are adjusted occasionally by having the root send out a new DAG iteration. A Trickle timer regulates when nodes should forward such information to their children, which efficiently suppresses many redundant updates in dense networks [2]. Nodes that detect routing inconsistencies; i.e., the loss of a parent; reset their Trickle timers to their minimum values in order to generate a fast repair.

Point-to-Point Traffic. To let arbitrary nodes communicate with each other, RPL includes a Destination Advertisement Object (DAO) in which children can advertise their own address prefixes to their neighborhood using multicasts, or back to the DAG root using unicasts after joining a DAG.

3. IMPLEMENTATION

We have implemented ContikiRPL in C using the APIs of the Contiki operating system. All parts of the implementation except

IPSN'10, April 12–16, 2010, Stockholm, Sweden.
ACM 978-1-60558-955-8/10/04.

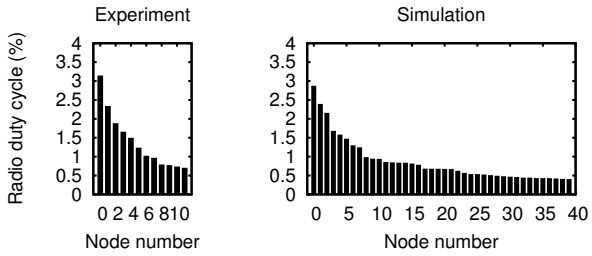

Figure 2: Per-node radio duty cycle. The nodes are sorted by duty cycle.

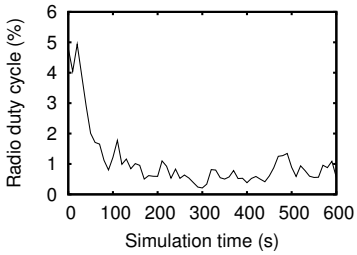

Figure 3: Power consumption is higher in the network-setup phase (0-100 s) than during steady-state operation.

for the uIP-specific ICMP message processing are portable to other operating systems. The routing protocol uses Contiki's modular IPv6 routing interface. This interface has three functions: *activate*, *deactivate*, and *lookup*.

The *activate* function initializes the DAG construction by sending a DAG Information Solicitation (DIS). Neighbors that belong to a DAG will reset their Trickle timers, and shortly thereafter the node will receive at least one DIO. The *deactivate* function deallocates internal structures and sends a no-DAO to the node's neighbors. After deactivation, the module stops responding to route-lookup requests, but may be reactivated later. If uIPv6 detects that the destination for a packet is not an immediate neighbor, it asks RPL for the route using the *lookup* function.

4. PRELIMINARY EVALUATION

We evaluate ContikiRPL in terms of power efficiency and implementation complexity.

4.1 Power Efficiency

To evaluate the power efficiency of ContikiRPL, we run it in a 41-node simulation and in a small-scale 13-node Tmote Sky deployment in an office environment. We run ContikiRPL on top of the ContikiMAC duty-cycling MAC protocol and measure the power consumption of the system by using Contiki's power profiling mechanism. Since the radio transceiver is the most power-consuming component of a typical mote, our power metric is the radio duty cycle, i.e., the percentage of time that the radio transceiver is on.

For the simulations we use Contiki's COOJA/MSPsim. By having a cycle-accurate emulation of a network of Tmote Sky motes, and a bit-accurate emulation of the radio transceiver, the simulator makes it possible to execute the exact same binary code in simulation and on real hardware.

We run the experiments for one night and the simulation for 10000 simulated seconds. The nodes generate a total of 40 best-effort UDP packets per minute. After the network setup phase, the simulation delivered 100% of all packets to the sink. In the experiment, we lost ten packets.

Figure 2 shows the per-node radio duty cycle of the experiment and the simulation, whereas Figure 3 shows the average duty cycle over time. Leaf nodes have a duty cycle of 0.5-0.8% and routing nodes have a duty cycle of 1-3%, depending on the amount of traffic they forward, giving a lifetime of several years, for both leaf nodes and routing nodes, on standard 3000 mAh AA-size batteries.

4.2 Implementation Complexity

Table 4.2 shows the implementation complexity for ContikiRPL. The total ROM size is 3224 bytes, which is approximately 6.5% of the Tmote Sky's ROM. The RAM size is dependent on the amount of candidate neighbors that should be stored; ContikiRPL reserves space for 10 neighbors. This setting results in a RAM footprint of 800 bytes, which is approximately 8% of the available space.

Table 1: Implementation complexity

Module	RAM (bytes)	ROM (bytes)
Generic IPv6 routing	420	484
RPL packet generation and parsing	2	1316
RPL protocol logic	378	1074
RPL timer handling	0	350
ContikiRPL Total	800	3224

5. CONCLUSIONS

We present ContikiRPL, an implementation of the RPL routing protocol for low-power and lossy networks. We conduct experiments both in a low-power wireless network and in simulation. Our results show several years of network lifetime with IPv6 routing on Tmote Sky motes.

Acknowledgments

This work was partly financed by SSF, the Swedish Foundation for Strategic Research, through the Promos project; by the European Commission under the contract FP7-ICT-224282 (GINSENG); and by the Swedish Energy Agency.

6. REFERENCES

[1] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pages 421–422, Raleigh, North Carolina, USA, Nov. 2008.

[2] 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 USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Mar. 2004.

[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet proposed standard RFC 4944, Sept. 2007.

[4] J. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[5] T. Winter (Ed.), P. Thubert (Ed.), and the ROLL Team. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. Internet Draft draft-ietf-roll-rpl-06, work in progress.