Efficient Data Acquisition in Sensor Networks: Introducing (the) LOADng Collection Tree Protocol

Jiazi Yi, Thomas Clausen, Axel Colin de Verdiere
Laboratoire d'Informatique - LIX, Ecole Polytechnique, France
jiazi@jiaziyi.com, thomas@thomasclausen.org, axel@axelcdv.com

***Abstract*—This paper proposes an extension to the "LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation" (LOADng), for efficient construction of a collection tree for data acquisition in sensor networks. The extension uses the mechanisms from LOADng, imposes minimal overhead and complexity, and enables a deployment to efficiently support both "point-to-point" and "multipoint-to-point" traffic, avoiding complications of uni-directional links in the collection tree.**

This paper further compares the performance of proposed protocol extension to that of basic LOADng and to the protocol RPL ("IPv6 Routing Protocol for Low power and Lossy Networks").

I. Introduction

"The Internet of Things" (IoT) assumes objects in our environment to be part of the Internet, communicating with users and with each other – and that these objects have communication as a commodity, rather than as their raison d'être. Communication in "The Internet of Things" is a challenge, subject to resource constraints, fragile and low-capacity links, dynamic and arbitrary topologies. Among the challenges is routing, requiring efficient protocols, able to converge rapidly even in very large networks, while exchanging limited control traffic and requiring limited memory and processing power.

One of the important applications of IoT is for data acquisition in sensor networks: a set of spatially distributed sensors that are used to monitor physical or environmental conditions, etc., and by their own impulse (either periodically or triggered by some event) transmit their data to a data concentrator (sink). These data are transmitted by way of a multi-hop network, and where the intermediary hops (routers) in that network are the sensor devices themselves. The collection of all paths from each sensor to the data concentrator form a *collection tree*. Traffic in such a collection tree is commonly described as being "multipoint-to-point" traffic, or "upwards" traffic, indicating that all traffic flows from the sensors to the data concentrator.

This paper describes a protocol for constructing such a collection tree in multi-hop sensor networks, where the protocol ensures that the resulting collection tree contains bi-directional paths between each sensor and the data concentrator. The protocol is defined as an extension to the LOADng routing protocol [1], which provides point-to-point routes between any two devices in a sensor network. Deploying both in unison permits efficient construction of both point-to-point routes and collection trees, by way of the same, simple, protocol mechanisms.

A. Background and History

Since the late 90s, the IETF[1] has embarked upon a path of developing routing protocols for networks with increasingly more fragile and low-capacity links, with less predetermined connectivity properties and with increasingly constrained router resources. In '97, by chartering the MANET working group, then subsequently in 2006 and 2008 by chartering the 6LowPAN and ROLL working groups.

1) MANET Protocol Developments: The MANET working group has developed of two protocol families: reactive protocols, including AODV [2], and proactive protocols, including Optimized Link State Routing (OLSR) [3]. A distance vector protocol, AODV operates in an *on-demand* fashion, acquiring and maintaining routes only while needed for carrying data, by way of a *Route Request-Route Reply* exchange. A link state protocol, OLSR uses a periodic control messages exchanges, each router proactively maintaining a routing table with entries for all destinations in the network, which provides low delays but constant control overhead. A sizable body of work exists, including [4], studying the performance of these protocols in different scenarios, and justifying their complementarity [5].

2) 6LowPAN, ROLL and related Protocol Developments: The 6LowPAN working group was chartered for adapting IPv6 for operation over IEEE 802.15.4, accommodating characteristics of that MAC layer, and with a careful eye on resource constrained devices (memory, CPU, energy, ...). Part of the original charter for this working group was to develop protocols for routing in multi-hop topologies under such constrained conditions, and over this particular MAC. Two initial philosophies to such routing were explored: *mesh-under* and *route-over*. The former, mesh-under, would, as part of an adaptation layer between 802.15.4 and IP, provide L2.5 multi-hop routing, presenting an underlying mesh-routed multi-hop topology as a single IP link. The latter, route-over, would expose the underlying multi-hop topology to the IP layer, whereupon IP routing would build multi-hop connectivity. Several proposals for routing were presented in 6LowPAN, for each of these philosophies, including LOAD [6]. LOAD was a derivative of AODV, but adapted for L2-addresses and mesh-under routing, and with some simplifications over AODV (*e.g.,* removal of intermediate router replies and sequence numbers). However, 6LowPAN was addressing other issues regarding adapting IPv6 for IEEE 802.15.4, such as IP packet header compression,

[1] http://www.ietf.org/

978-1-61284-683-5/12/$31.00 ©2012 IEEE

and solving the routing issues was suspended, delegated to a working group ROLL, created in 2008 for this purpose. ROLL produced a routing protocol denoted "Routing Protocol for Low-power lossy networks" (RPL) [7] in 2011 based on the idea of collection tree protocol [8].

RPL as a collection tree protocol has several well known issues with respect to supporting different kinds of traffic patterns, uni-direction link handling, as well as algorithmic and code complexity [9]. On the other hand, AODV and its derivatives have been implemented and used widely, such as IEEE 802.11s [10] is based on AODV, and the G3-PLC standard [11], published in 2011, specifies the use of LOAD [6] at the MAC layer, for providing mesh-under routing for utility (electricity) metering networks. Spurred by these experiences, 2011 saw the emergence of LOADng [1], as a successor to LOAD, and for deployments where the issues in [9] render RPL less applicable.

LOADng, as LOAD is however designed mainly for point-to-point traffic, and not optimized for multipoint-to-point traffic.

B. Statement of Purpose

An extension to LOADng, denoted LOADng Collection Tree Protocol (LOADng-CTP), is proposed in this paper , for building a "collection tree" in environments, constrained in terms of computational power, memory, and in energy. An example of the design target for LOADng-CTP is the ESB (Embedded Sensor Board [12]), with a TI MSP430 low-power micro-controller, an 1MHz CPU, 2kB RAM and 60kB flash ROM. The link layers typically used in LLNs impose strict limitations on packet sizes: in IEEE 802.15.4, the maximum physical layer packet size is 127 bytes, the resulting maximum frame size at the mac-layer is 102 bytes. If link-layer security is used, this may consume up to a further 21 bytes, which leaves just 81 bytes for upper layer protocols.

The LOADng-CTP presented in this paper is thus designed to meet the following requirements:

- Effectively build a route from all sensors to the root, and the route from the root to the sensors if required.
- Uni-directional links being avoided in these routes.
- Low overhead, easy collection tree maintenance.
- Easy extension to LOADng, such that routers using only LOADng (without collection tree extension) can joint the collection tree.

The remainder of this paper is organized as follows: In section II, the LOADng-CTP specification is introduced, including related message format and main operations. The simulation study is performed in section III, in which LOADng, LOADng-CTP and RPL are compared. Section IV concludes this paper .

II. Protocol Specification

LOADng Collection Tree Protocol (LOADng-CTP) is based on the operation and packet format of LOADng. Therefore, the current LOADng implementation can be easily extended to the collection tree protocol. In the following, the basic operation of LOADng is introduced briefly, followed by the single message and protocol processin required for collection tree building and maintenance.

A. LOADng Basic Operation

LOADng contains two main operations: *Route Discovery* and *Route Maintenance*.

1) Route Discovery: During *Route Discovery*, RREQ messages are flooded through the network. In LOADng [1], only the destination of the RREQ will reply by generating and unicasting a RREP to the originator of the RREQ. All RREQ and RREP messages, generated by a LOADng router, carry a monotonically increasing sequence number, permitting both duplicate detection and detecting which of two messages contains the most "fresh" information.

2) Route Maintenance: Route Maintenance is performed when an actively used route fails. Route failure is detected by way of a data packet not being deliverable to the next hop towards the intended destination. In LOADng, the RERR is unicasted to the source of data packet. On receiving the RERR at the source of data packet, a new *Route Discovery* can be performed, in order to discover a new route to the intended destination.

B. Message for Collection Tree Protocol

LOADng-CTP introduces two flags to RREQ messages:

- RREQ COLLECTION_TREE_TRIGGER: when set, a receiving routers will be triggered to discover with which of its neighbors it has bi-directional links.
- RREQ COLLECTION_TREE_BUILD: when set, a receiving router will build the route to the root.

In addition, a HELLO message is used, in order to permit verification of bidirectionally of links before admitting these to the collection tree.

C. Collection Tree Building

The collection tree is, then, build by way of the following procedure — initiated by the router wishing to be the root of the collection tree:

1) Collection tree triggering (by the root): The root generates an RREQ with *COLLECTION_TREE_TRIGGER* set (henceforth, denoted *RREQ_TRIGGER*). Both the originator and destination of the *RREQ_TRIGGER* are set to the address of the root.

When a RREQ_TRIGGER is generated, an RREQ with *COLLECTION_TREE_BUILD* set (henceforth, denoted *RREQ_BUILD*) is scheduled to be generated in 2×NET_TRAVERSAL_TIME.

2) Bi-directional neighbor discovery: On receiving a *RREQ_TRIGGER*, a router:

- Records the address of the sending router (*i.e.,*the neighbor, from which it received the *RREQ_TRIGGER*) in its *neighbor set*, with the status *HEARD*.
- If no earlier copy of that same *RREQ_TRIGGER* has been previously received:

- The *RREQ_TRIGGER* is retransmitted, subject to a jitter[2] of *RREQ_JITTER*, to reduce the chance of collisions.
- Schedules generation of a HELLO message, subject to a jitter of *HELLO_JITTER* [3]. When the scheduled HELLO message is generated, it lists the addresses of all the neighbors, from which it has received a *RREQ_TRIGGER*.

On receiving a HELLO message, a router:

- If it finds its own address listed in the HELLO message, it records the address of the sending router (*i.e.,*the neighbor, from which it received the *HELLO*) in its *neighbor set*, with the status *SYM* (bi-directional).

Thus, each router will learn with which among its neighbor routers it has a bi-directional (SYM) or uni-directional (HEARD) link.

3) Collection tree building: 2×NET_TRAVERSAL_TIME after the *RREQ_TRIGGER*, the root generates a *RREQ_BUILD*.

On receiving a *RREQ_BUILD*, a router:

- Verifies if the *RREQ_BUILD* was received from a neighbor with which it has an uni-directional link. If so, the *RREQ_BUILD* is silently discarded.
- Otherwise, if no earlier copy of that same *RREQ_BUILD* has been previously received,
  - a new routing entry is inserted into the routing table, with (*next_hop* = previous hop of the *RREQ_BUILD*; *destination* = root)
  - The *RREQ_BUILD* is retransmitted, again subject to a jitter of *RREQ_JITTER*.

Thus, each router will record a route to the root, and this route will contain only bi-directional links; the collection tree is built, enabling upward traffic.

If also routes from the root to other routers (sensors) inside the network is required, each router receiving a *RREQ_BUILD* will unicast a RREP to the root, transmitted and processed according to [1]. Thus, also downward traffic is enabled.

D. Collection Tree Maintenance

During the process described in section II-C, control messages may be lost. Thus, some routers may not be included in the initial collection tree because of transient transmission failure of collection tree building messages. Furthermore, the routing entries may expire because of not updated timely. Both of those result in that route to the root is not available in some of the sensors.

In this case, a router with data traffic to send to the root will initiate route discovery, according to the usual procedures from [1]. To avoid that RREQ being broadcast through the whole network, and take benefits from that "most of other neighbor routers might have an available route to the root", a *Smart Route Request* scheme can be employed: if an intermediate router, receiving the RREQ, does not have an available route to the destination, the RREQ is forwarded as normal. If the intermediate router has a route to the root, that intermediate router will unicast the RREQ to the destination according to the routing table. For the routers that only run LOADng, without collection tree extension (*i.e.,*they can't be verified as bi-directional neighbors because they can't generate HELLO message on receiving *RREQ_TRIGGER*), they can join the collection tree in the same way.

When a link on an active route to a destination is detected as broken (by way of inability to forward a data packet towards that destination), an RERR (route error) message is unicast to the source of the undeliverable data packet. Both this intermediate router and the source router need to initiate a new route discovery procedure.

III. Simulation and Performance Analysis

In order to understand the performance impact of the collection tree extension to LOADng, this section presents a set of ns2 simulations, comparing LOADng, LOADng-CTP and RPL, with the parameters of the trickle timer in RPL is set according to [7] Simulations were made with varying numbers of routers from 63 to 500, equipped with IEEE 802.11 wireless interfaces, and placed statically randomly in a square field while maintaining a consistent network density.

The network is subject to multipoint-to-point (MP2P) traffic with all routers generating an 80 seconds burst of 512-octet data packets every 5 seconds, towards a single destination (sink).

A. Simulation Results

Figure 1 depicts the delivery ratio of three protocols. Both LOADng-CTP and RPL obtain delivery ratios close to 100%, regardless of number of nodes. LOADng, initiating route discovery for every router (network-wide broadcast), incurs a high number of collisions on the media, and thus a lower data delivery ratio, especially in larger scenarios. The data delivery delay is depicted in 2, showing that while LOADng-CTP and RPL have routes a-priori available, the route discovery process of LOADng causes longer delays. From these figures, LOADng-CTP and RPL exhibit identical performances. Figures 3 and 4 depict the control traffic overhead in total number of packets sent, and number of bytes/s , respectively, required for every router to have a route to the root.

The overhead of LOADng-CTP and RPL grow linearly with RPL sending twice as many packets as LOADng-CTP, and RPL sending 10 times more bytes/s as compared to LOADng-CTP, due to the RPL control packets (mainly, the DIOs) being bigger [9]: a DIO packet[4] takes up to 40 octets in these scenarios, whereas a LOADng-CTP RREQ and RREP packet typically is 10 octets. The overhead of LOADng grows exponentially as the number of nodes increases, up to 700,000 packets for scenarios of 500 nodes (not drawn in the figure). The peer-to-peer based basic LOADng mechanism is not optimized for MP2P traffic.

[2]According to the recommendations in [13].

[3]Where *HELLO_JITTER* > *RREQ_JITTER*

[4]Base header of 24 octets, plus other options and addresses.

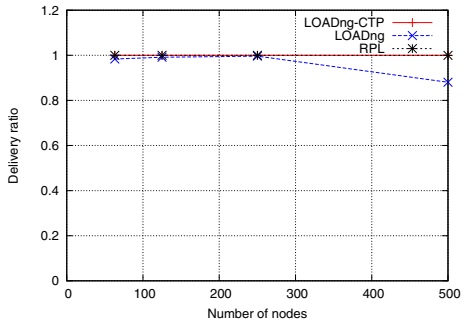

Figure 1. Packet delivery ratio

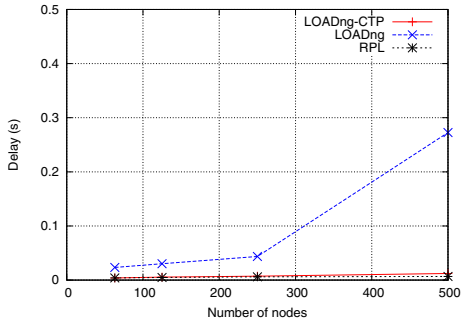

Figure 2. Average end-to-end delay

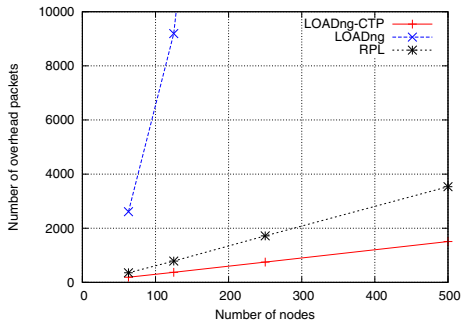

Figure 3. Number of overhead packets

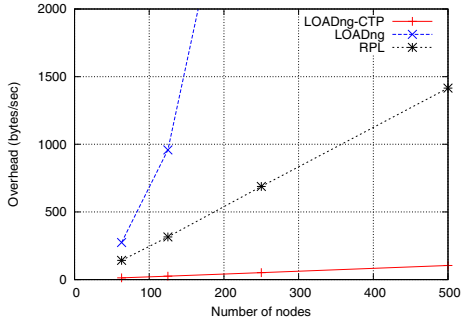

Figure 4. Overhead bytes per second

IV. Conclusion

This paper has presented a simple extension, LOADng-CTP, to the reactive LOADng routing protocol, permitting efficient and on-demand construction of collection trees for supporting MP2P traffic types. The performance of this extension has been studied, revealing delays and data delivery rations, comparable with RPL, are obtained while at the same time yielding considerably lower control traffic overheads. Compared to basic LOADng, the performance of the LOADng-CTP extension yields better performance: lower overhead, higher data delivery ratios, and lower delays.

It is worth noting that all the links used in LOADng-CTP routes are validated as bi-directional, whereas RPL, links in the collection tree are validated only to be uni-directional from root to the sensors (but not validated as bi-directional). While the simulations in this paper do not reveal any ill effect (in part, because most links turned out to be bi-directional), it is expected - and a subject for further study - that in scenarios with a higher number of uni-directional links, LOADng-CTP will outperform RPL.

Another key aspect of LOADng-CTP is, that any router can at any time determine that it needs to act as a data sink for MP2P traffic, and spawn a collection tree construction; this, without requiring that said router be specifically provisioned for this purpose (no extra state, processing power, required).

References

[1] T. Clausen, A. C. de Verdiere, J. Yi, A. Niktash, Y. Igarashi, H. Satoh, and U. Herberg, "The lln on-demand ad hoc distance-vector routing protocol - next generation," The Internet Engineering Task Force, October 2011, internet Draft, work in progress, draft-clausen-lln-loadng.

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

[3] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," Experimental RFC 3626, October 2003.

[4] L. V. T. Clausen, P. Jacquet, "Comparative study of routing protocols for mobile ad-hoc networks." Proceedings of the IFIP MedHocNet, September, Sardinia, Italy, 2002.

[5] T.Clausen, P.Jacquet, and L.Viennot, "Analyzing control traffic overhead versus mobility and data traffic activity in mobile ad hoc network protocols," *ACM Journal on Wireless Networks, vol. 10 no. 4*, 2004.

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

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

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "The Collection Tree Protocol (CTP)," TinyOS, Tech. Rep., 2009. [Online]. Available: http://sing.stanford.edu/pubs/sing-09-01.pdf

[9] T. Clausen, A. C. de Verdiere, J. Yi, and U. Herberg, "Experiences with RPL: IPv6 Routing Protocol for Low power and Lossy Networks," The Internet Engineering Task Force, January 2012, internet Draft, work in progress, draft-clausen-lln-rpl-experiences.

[10] G. Hiertz, S. Max, R. Zhao, D. Denteneer, and L. Berlemann, "Principles of ieee 802.11s," in *Proceedings of WiMAN in conjunction with the 16th ICCCN*, Honolulu, Hawaii, USA, Aug 2007, p. 6.

[11] "ITU-T G.9956: Narrow-Band OFDM power line communication transceivers - Data link layer specification," November 2011.

[12] "The ESB Embedded Sensor Board," http://www.sics.se/ adam/contiki/contiki-2.0-doc/a00435.html, 2011.

[13] T. Clausen, C. Dearlove, and B. Adamson, "Jitter Considerations in MANETs," IETF Inf. RFC 5148, February 2008.