2013 First International Symposium on Computing and Networking

A Depth First Forwarding (DFF) Extension for the LOADng Routing Protocol

Thomas Clausen, Jiazi Yi, Antonin Bas
Laboratoire d'Informatique (LIX) – Ecole Polytechnique, France
Thomas@ThomasClausen.org, jiazi@jiaziyi.com, Antonin.Bas@polytechnique.edu

Ulrich Herberg
Fujitsu Laboratories of America
ulrich@herberg.name

Abstract—**This paper explores the cooperation between the new standards for "Low Power and Lossy Networks" (LLNs): IETF RFC 6971, denoted "Depth-First Forwarding in Unreliable Networks" (DFF) and the ITU-T standardised routing protocol "LOADng" (Lightweight On-demand ad hoc Distance-vector Routing - next generation). DFF is a data-forwarding mechanism for increasing reliability of data delivery in networks with dynamic topology and lossy links, using a mechanism similar to a "depth-first search" for the destination of a packet. LOADng is a reactive on-demand routing protocol used in LLNs. The purpose of this study is to evaluate the benefit of using DFF conjointly with a routing protocol. To this end, the paper compares the performance of LOADng and LOADng+DFF using Ns2 simulations, showing a 20% end-to-end data delivery ratio increase at expense of expected longer path lengths.**

I. Introduction

Low-power and Lossy Networks (LLNs) are composed of devices with strictly limited computational power and storage (1-2MHz CPUs and a couple of KB of memory), communicating over a channel characterised by a high risk of packet losses, (typically) very small frame sizes, and very limited throughput. Transiting data across such a network, especially when multiple hops are present between the source and the destination, is a challenging task: routing protocols finding paths must be frugal in their control traffic and state requirements, as well as in algorithmic complexity – and even once paths have been found, these may be usable only intermittently (*e.g.,* not all packet gets through successfully) or for a very short time due to changes on the channel such as persistent interference (requiring rediscovery of an usable path). Channel failures, resulting in link failures in a routing path can result from a variety of factors such as heterogeneity of sender and receiver hardware, power supply or power control algorithms (leading to different transmission ranges, unidirectional links, or simply that devices are power-cycled asynchronously), the presence of noise or interferences, or even device failure causing a previously selected intermediary router to no longer be available.

The limitations of the devices and the channel capacity in LLNs suggest a routing protocol of extreme simplicity – yet the fragility and transient nature of links suggest the requirement to be able to quickly discover and establish alternative paths when faced with a link failure. These requirements are, seemingly, contradictory. A "standard" proactive routing protocol, such as OSPF (Open Shortest Path First) [1] or OLSR (Optimized Link State Routing) [2], maintaining a network topology graph would remove a "broken" link from its graph and re-run a shortest path algorithm – incurring the requirement of each routing device being able to store (up to) the complete network topology, as well as being able to re-run the shortest path algorithm on a whim. A "standard" on-demand routing protocol would in the similar situation incur route re-discovery, with additional (flooded) control signals being imposed on the network, as well as additional delays on data packet delivery whilst route re-discovery is ongoing, and either buffering of data packets for that duration or retransmission once a path has been re-discovered.

Different proposed and standardised routing protocols for LLN exist, including RPL (Routing Protocol for Low-power and lossy networks) [3] and LOADng (Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation) [4], making different trade-offs and being of different philosophies – yet being united in the fact that when a link, used actively as part of a routing path, fails, then it is up to the routing protocol to recover by discovering alternative paths, with data traffic being either buffered or dropped for the duration of this recovery.

"Depth-First Forwarding in Unreliable Networks" (DFF) [5] is an experimental standard which proposes a mechanism for recovery in case of link failure. Colloquially speaking, if a device fails in its attempt to forward a packet to its intended next-hop, then DFF suggests a heuristics for "trying another of that devices neighbours", while keeping track of (and preventing) packet loops. Thus, DFF operates on the "forwarding plane". While DFF can operate independently, *i.e.,* without a routing protocol (which amounts to simply doing a depth-first exploration of the network), it can also be used conjointly with a routing protocol: the routing protocol can provide an "order of priority" of the neighbours of a device, in which data delivery should be attempted – and DFF can also signal to a routing protocol when data delivery to a destination has (possibly repeatedly) failed via a neighbour but (possibly repeatedly) succeeded via another neighbour.

A. Statement of Purpose

This paper explores the cooperation between DFF and the routing protocol LOADng [4], with the purpose of uncovering the benefit of using DFF conjointly with a routing protocol. To this end, the paper compares the performance of LOADng and

978-1-4799-2795-1/13 $31.00 © 2013 IEEE
DOI 10.1109/CANDAR.2013.72

LOADng+DFF. RPL is not further studied in this paper due to the fact that the predominant mode of operation of RPL ("non-storing mode" [6]) employs source routing – which lends itself poorly[1] to on-the-path autonomous routing decisions causing deviations from the established source route and, as such, is incompatible with DFF.

B. Paper Outline

The remainder of this paper is organized as follows: section II briefly describes the routing protocol LOADng, and section III presents an overview of DFF. Section IV outlines how LOADng and DFF are integrated – how the routing protocol provides information for DFF's forwarding decisions, as well as how DFF signals to the routing protocol. It should be noted that there are many different possible ways in which LOADng and DFF (or, indeed, any routing protocol and DFF) can be integrated, and this section outlines only the one studied in this paper. Section V presents a performance evaluation of LOADng with and without DFF. Finally, section VI concludes this paper.

II. LOADng

LOADng [7], [8] is a simplified reactive routing protocol, targeting routing in low-power and lossy networks. It has been standardised by the ITU (International Telecommunication Union) [4] for routing in the "Smart Grid", between electricity meters and other equipment for electricity grid management.

As a reactive protocol, the basic operations of LOADng include generation of Route Requests (RREQs) by a LOADng Router (originator) for when discovering a route to a destination. These RREQs are flooded through the network, each forward of a RREQ installing temporary routing table entries towards the originator of the RREQ. Once an RREQs has has been received the sought the destination LOADng Router, that device will generate a Route Reply (RREPs), which is unicast hop-by-hop towards the originator using the temporary route installed by the received RREQ. This forwarding of an RREP installs routing table entries towards the destination.

If a route is detected broken, *i.e.,* if forwarding of a data packet to the recorded next hop on the route towards the intended destination is detected to fail, a Route Error (RERR) message is returned to the originator of that data packet. The LOADng specification stipulates that when the transmission of a data packet fails, that data packet is dropped and a RERR is sent back to its source – which can, then, trigger a new route discovery.

Extensions to and options for LOADng exist [9], [10] for trying to reduce the impact on the network load of route discovery – but fact remains that in networks where transmission failures are frequent, this behaviour can result in low delivery ratios and possibly high network loads [7].

III. Depth-First Forwarding

DFF [5] is a forwarding mechanism for improving the data delivery success ratio across unreliable multi-hop networks. It operates solely on the forwarding plane, *i.e.,* does not assume any specific routing protocol to be in operation (or, indeed, that any routing protocol is in operation) – but can, as appropriate and as indicated in section I, interact with a routing protocol. DFF relies on an external mechanism providing each router with a list of its neighbours.

Schematically, the basic operation of DFF is as follows, when a data packet for a destination arrives at the forwarding plane of a router:

1) The router temporarily creates an ordered Candidate Next Hop list for that packet, which does not contain the neighbour from which the data packet was received (if any), from among the neighbours in the routers' neighbour list.
2) The router attempts to forward the data packet to the first neighbour in the resulting Candidate Next Hop list.
3) There are five possible outcomes from this attempt:
   - The Candidate Next Hop list is empty, in which case the data packet is returned to the neighbour from which it was initially received, and the process for this router stops.
   - Delivery to that neighbour succeeds (*e.g.,* as confirmed by an L2 acknowledgement), and that neighbour is the destination for the data packet. The L2 acknowledgement indicates successful data packet delivery to the destination. The process for this router stops.
   - Delivery to that neighbour fails (*e.g.,* detected by lack of an L2 acknowledgement), in which case that neighbour is removed from the Candidate Next Hop list, and the process resumes at item 2 above.
   - Delivery to that neighbour succeeds (*e.g.,* as confirmed by an L2 acknowledgement), but the data packet is returned from the neighbour as "undeliverable", in which case that neighbour is removed from the Candidate Next Hop list, and the process resumes at step 2 above, with the resulting Candidate Next Hop list.
   - Delivery to that neighbour succeeds (*e.g.,* as confirmed by an L2 acknowledgement), the neighbour is not the destination for the data packet. That neighbour will, now, execute this very same procedure (create its own Candidate Next Hop list, and execute this process, starting at step 1).

The initial Candidate Next Hop list for a data packet, by default, contains all the neighbours of a router, except for the neighbour from which the data packet was received, but may be smaller. The list is ordered, section 11 in [5] suggests several criteria to take into account when ordering that list, including that if a routing protocol is in operation, then the neighbour on the shortest path (as indicated by that routing protocol) must be part of the initial Candidate Next Hop list

[1] Among other things, it would be difficult to use IPSec or similar mechanisms.

– and is recommended to be first in that initial Candidate Next Hop List. Link quality, historical information on "good and bad neighbours as next hop" is suggested to be used for ordering remaining neighbours.

DFF contains mechanisms for detecting looping data packets, encoded as flags and sequence numbers in IPv6 Hop-by-Hop header options, carried in each data packet, and specifies processing here. This incurs a small, but fixed, per-data-packet overhead of 8 octets. This paper does not discuss this signalling and processing in further details.

IV. Integration of LOADng and DFF

DFF requires that a router has a list of all its neighbours available for constructing the Candidate Next Hop list for a data packet. [5] specifies that an external mechanism is to be in place to provide that list, and suggests the use of NHDP (Neighborhood Discovery Protocol) [11] – which is implemented and used for the purpose of the performance studies in this paper.

The routing protocol LOADng provides, at most, one entry in the routing table for each destination, thus the integration of the requirements for ordering the entries in the Candidate Next Hop list for a data packet is met simply by, if a routing table entry for the destination is present, inserting this first in that list. The remainder of the entries in the Candidate Next Hop list are, simply, all the other neighbours discovered by NHDP (and with status SYMMETRIC), excluding of course the neighbour from which the data packet was received.

Additionally, the two following rules govern the integration of LOADng and DFF, for the purpose of the studies in this paper, specifically when the protocol operations for each are activated:

- When a router receives a data packet from another router, for which it does not have a corresponding entry in the routing table:
  - Send data packet according to the DFF forwarding rules, as described in section III
  - Send an RERR message to the originator of that data packet, as described in section II

  An RERR message is sent since while DFF will ensure data delivery, this may be by way of an excessively long path; by sending an RERR message, the routing protocol is instructed to "try to find a better path" whilst DFF concurrently attempts delivery of data in transit (thus reducing delays, retransmissions and/or buffer of data traffic).
- If forwarding of a data packet to the next hop, indicated by LOADng (*i.e.,* the first entry in the Candidate Next Hop list) fails (either by way of the packet being returned by DFF, or by an L2 acknowledgement being absent):
  - Send data packet according to the DFF forwarding rules, as described in section III
  - Send an RERR message to the originator of that data packet, as described in section II

  In this case, an RERR message is sent since, in addition to the reasons listed above, this is indicative of the routing information being inconsistent with the network topology, and therefore needs to be updated.

Figure 1 gives an example of DFF with LOADng. Node *A* is the data source, and node *D* is the destination. The route originally found by LOADng protocol was *A-B-F-D* (one of the shortest paths). However, when a data packet arrived at node *B*, the link *B-F* was detected broken. By using DFF, a neighbour node from Candidate Next Hop list, node *G*, for example, is chosen as next hop. The data packet is thus forwarded to node *G*, which will handle the packet according its routing table information or DFF, and forward it to the destination node *D*. In the meantime, node *B* will send an RERR message to node *A*, to notify the route failure.

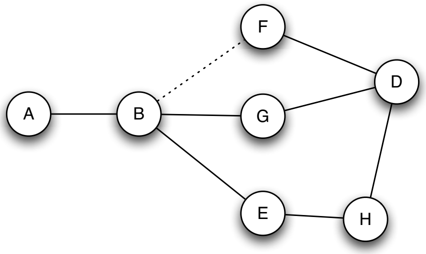

Figure 1. An example of DFF. Node *A* is the source, node *D* is the destination.

V. Evaluation

In order to evaluate and compare the performance of LOADng with and without DFF, network simulations by way of NS2 are employed. While network simulations are, at best, an approximation of real-world performance (particularly due to the fidelity of their lower layers to reality), they do provide a baseline for comparison and, generally, best-case results, *i.e.,* real-world performance is expected to be no better than that which is obtained through simulations. The reason for using network simulations is, that such allow running experiments with different protocols under identical conditions and parameters (MAC layer, distribution, number of nodes, etc.).

Simulations were conducted using the TwoRayGround propagation model and the IEEE 802.11 MAC. Although there are various low-layer technologies more commonly (and, perhaps, more viably) used for LLNs (power line communication, 802.15.4, low-power wifi, bluetooth low energy, etc.), general behaviour of a protocol can be inferred from simulations using 802.11.

To discover bi-directional links in the network, NHDP is used. For NHDP, a HELLO message interval must be chosen. The shorter the HELLO message interval, the more accurate a list of neighbours can be acquired (and so, the better can DFF do their jobs) but at the expense of increased control traffic overhead. For the purpose of these simulations, a HELLO interval of 1s was (arbitrarily) chosen as it represents a "very frequent HELLO message exchange and therefore a good "worst case" example. In a real deployment, the HELLO interval should be selected so as to correspond to the expected local network topology change rate.

A. *Network Topology and Traffic Characteristics*

The general network topology of a scenario is as follows: n (from 63 to 500) devices are placed randomly (while ensuring that the network is still connected) in a square field, so that to maintain a constant device density. A (random) device in the network then creates $n-1$ Constant Bit Rate (CBR) streams, one to each other device and sends one packet of 512 octets every 5 seconds to each of them. As DFF is supposed to be particularly beneficial in lossy networks, the simulations enforce that a packet is lost with a probability of 20%. Simulations were run for 100s each, and for each datapoint 20 different and randomly generated scenarios – all corresponding to the same abstract parameters – were simulated, with the results presented below representing averages from among these.

B. *Simulation Results*

Figure 2 depicts the data delivery ratios obtained for the two protocols. While neither protocol obtains a perfect data delivery ratio, DFF+LOADng introduces a constant ∼20% improvement over LOADng. This improvement comes with an increased control traffic overhead, as shown in figure 3 – this is due to the fact that DFF+LOADng includes operation of NHDP which incurs periodic signalling within each neighbourhood.

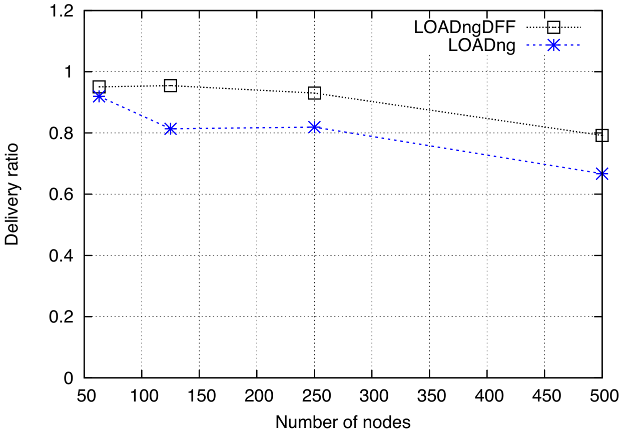

Figure 2. Delivery Ratio

Figure 5 and 4 depict average end-to-end data delivery delays and path lengths for data traffic. While it may appear be intuitive that DFF incurs longer path lengths (after all, a depth-first search for a destination will rarely yield the shortest path), section IV introduced the use of DFF as a way of reducing delays. This, therefore, must be balanced with what happens without DFF: data packets are dropped, and not included in the delay or path length statistics, during the time needed by LOADng to recover.

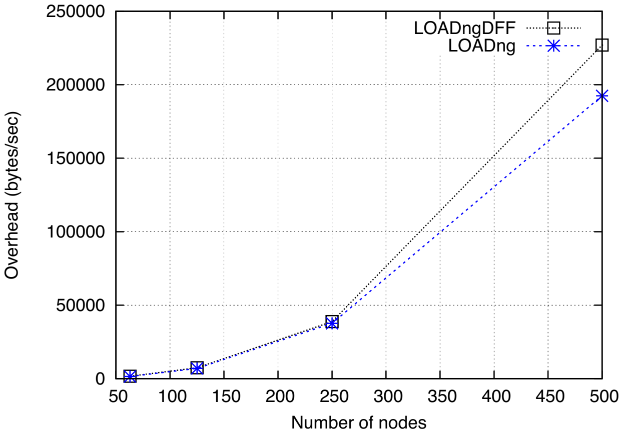

Figure 3. Overhead

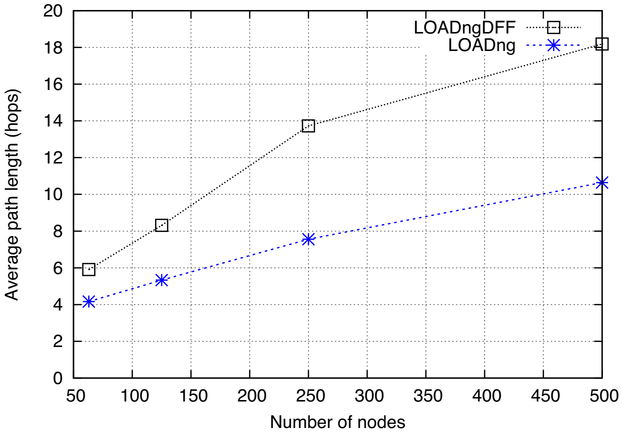

Figure 4. Path lengths

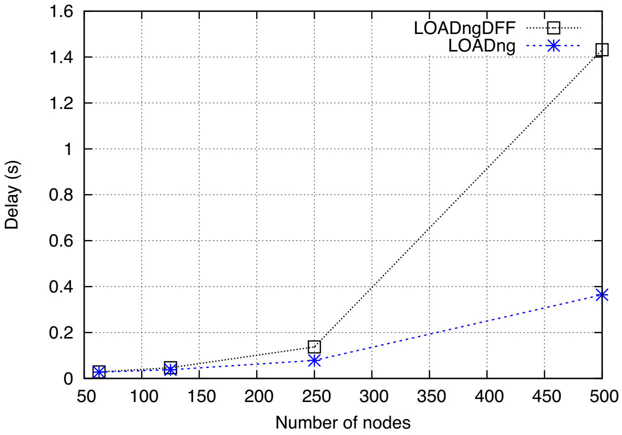

Figure 5. Delays

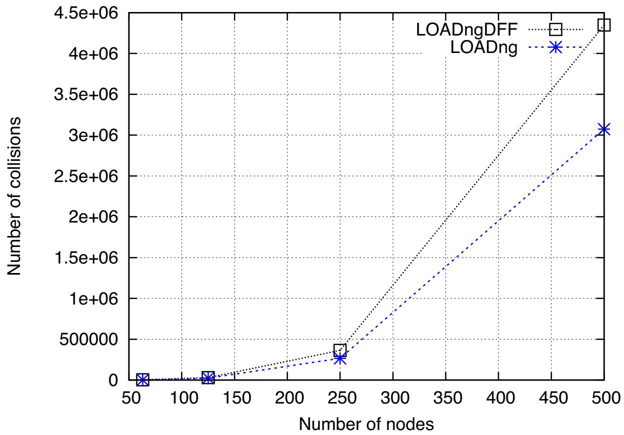

Figure 6. Collisions

Figure 6 depicts the number of collisions incurred (including collisions between control and data traffic). The overhead (see figure 3 of NHDP, necessary for LOADng+DFF, causes an increased number of collisions as compared to LOADng – yet this the data packet loss incurred through this is more than offset through the benefits of using DFF, as depicted in figure 2.

VI. Conclusion

This paper introduces DFF as a forwarding mechanism for LLNs, and its application to the LOADng routing protocol. The implementation of DFF is about 200 lines of additional code, compared to about 5000 lines of code for the LOADng prototype. Because DFF requires the information of bi-directional neighbors, NHDP is employed in this study (an additional 200 lines of code). It is important to note that NHDP is not mandatory if external mechanisms can provide the neighborhood information, such as link layer protocols.

Simulation is performed to study the performance of DFF in harsh lossy networks. Simulation results show that with the DFF extension, up to 20-25% of improvement in data delivery ratio can be achieved compared to LOADng, with little cost in delay and path length. This makes DFF an interesting mechanism for scenarios where the data delay is not critical.

In the implementation tested, a basic (or "naive") mechanism is used – DFF treats every bi-directional neighbor equally, without considering which may be the best candidate "next hop" to the final destination. This, however, is intentional, so as to be able to observe what is the worst case when DFF is employed. With this benchmark, it would be interesting to explore more efficient mechanisms for choosing the "best next hop" in the future.

References

[1] J. Moy, "OSPF Version 2," RFC 2328, IETF, April 1998.

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

[3] 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," RFC 6550, IETF, March 2012.

[4] ITU, "ITU-T G.9903: Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks: Amendment 1," May 2013.

[5] U. Herberg, A. Cardenas, T. Iwao, M. Dow, and S. Cespedes, "Depth-First Forwarding (DFF) in Unreliable Networks," RFC 6971, IETF, June 2013.

[6] T. Clausen, A. C. de Verdiere, J. Yi, U. Herberg, and Y. Igarashi, "Observations of RPL: IPv6 Routing Protocol for Low power and Lossy Networks," Internet Draft, work in progress, draft-clausen-lln-rpl-experiences, IETF, February 2013.

[7] T. Clausen, J. Yi, and A. C. de Verdiere, "LOADng: Towards AODV Version 2," in *VTC Fall*. IEEE, 2012, pp. 1–5.

[8] T. Clausen, A. C. de Verdiere, J. Yi, A. Niktash, Y. Igarashi, H. Satoh, U. Herberg, C. Lavenu, T. Lys, and J. Dean, "The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)," Internet Draft, work in progress, draft-clausen-lln-loadng, IETF, July 2013.

[9] J. Yi, T. Clausen, and A. Bas, "Smart Route Request for On-demand Route Discovery in Constrained Environments." Proceedings of the IEEE International Conference on Wireless Information Technology and Systems, September 2012.

[10] A. Bas, J. Yi, and T. Clausen, "Expanding Ring Search for Route Discovery in LOADng Routing Protocol." Proceedings of The 1st International Workshop on Smart Technologies for Energy, Information and Communication, September 2012.

[11] T. Clausen, C. Dearlove, and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)," RFC 6130, IETF, April 2010.

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