Lightweight On-demand Ad hoc Distance-vector Routing - Next Generation (LOADng): Protocol, Extension, and Applicability

Thomas Clausen[a], Jiazi Yi[a,*], Ulrich Herberg

[a] *Ecole Polytechnique, Route de Saclay, Palaiseau 91128 France*

Abstract

This paper studies the routing protocol "Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation (LOADng)", designed to enable efficient, scalable and secure routing in low power and lossy networks. As a reactive protocol, it does not maintain a routing table for all destinations in the network, but initiates a route discovery to a destination only when there is data to be sent to that destination to reduce routing overhead and memory consumption. Designed with a modular approach, LOADng can be extended with additional components for adapting the protocol to different topologies, traffic, and data-link layer characteristics. This paper studies several such additional components for extending LOADng: support for smart route requests and expanding ring search, an extension permitting maintaining collection trees, a fast rerouting extension. All those extensions are examined from the aspects of specification, interoperability with other mechanisms, security vulnerabilities, performance and applicability. A general framework is also proposed to secure the routing protocol.

Keywords: low power and lossy network, routing protocol, LOADng, reactive protocol

1. Introduction

Low-power and Lossy Networks (LLNs) are composed of Constrained Devices, *i.e.,* devices with strictly limited computational power and storage (1-2 MHz CPUs and a couple of KB of memory), which are communicating over a channel characterised by a high probability 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 must be frugal in their control traffic and state requirements, as well as in algorithmic complexity. Even once paths have been found, these may be usable only intermittently, or for a very short time, due to changes on the channel such as persistent interference (requiring rediscovery of a 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, the presence of noise or interferences, or even device failure causing a previously selected intermediary router along a path 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 (Optimised Link State Routing) [2, 3], 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 having sufficient memory to store (up to) the complete network topology, as well computational power allowing it to frequently re-run a shortest path algorithm. A "standard" on-demand routing protocol would in the same situation incur path re-discovery, with additional control signals being imposed on the network, as well as additional delays on data packet delivery whilst path re-discovery is ongoing, and either buffering of data packets for that duration or retransmission once a path has been re-discovered.

1.1. Background and History

Since the late 90s, the Internet Engineering Task Force (IETF)[1] has embarked upon a path of developing routing protocols for networks with increasingly more fragile and low-capacity links, with less pre-determined connectivity properties, and with increasingly constrained router resources. This, in '97, by chartering the MANET (Mobile Ad hoc Networks) working group, then subsequently

*Corresponding author

Email addresses: Thomas@ThomasClausen.org (Thomas Clausen), jiazi.yi@polytechnique.edu (Jiazi Yi), ulrich@herberg.name (Ulrich Herberg)

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

in 2006 and 2008 by chartering the 6LoWPAN (IPv6 over Low power WPAN) and ROLL (Routing Over Low power and Lossy networks) working groups.

1.1.1. MANET Protocol Developments

The MANET working group converged on the development of two protocol families: reactive protocols, including AODV (Ad hoc On-demand Distance Vector routing [4]) and DSR (Dynamic Source Routing [5]), and proactive protocols, including the OLSR (Optimised Link State Routing [2]) and TBRPF (Topology dissemination based on reverse-path forwarding [6]). Distance vector protocols operate in an *on-demand* fashion, acquiring and maintaining paths only while needed for carrying data, by way of a Route Request/Route Reply exchange. Proactive protocols are based on periodic control messages exchanges, where each router proactively maintains a routing table with entries for all destinations in the network. A sizeable body of work exists, including [7], studying the performance of these protocols in different scenarios, and justifying their complementarity. For the purpose of this paper, it suffices to observe that proactive provides low delays and predictable, constant control overhead – at expense of requiring memory in each router for maintaining complete network topology. Reactive protocols limit the memory required for routing state to that for actively used paths, at the expense of delays for the Route Request/Route Reply exchange to take place, and control overhead dependent on data flows.

After acquiring operational experiences with DSR, AODV, TBRPF, and OLSR, the MANET working group commenced developing successors to these protocols, denoted OLSRv2 and DYMO. Whereas a relatively large and active community around OLSR thus standardised OLSRv2 [3], the momentum behind DYMO (renamed to AODVv2 in 2013) diminished, and development of reactive routing protocols was abandoned by the MANET working group in 2016[2].

1.1.2. 6LoWPAN and ROLL Protocol Developments

The 6LoWPAN working group was chartered for adapting IPv6 for operation over IEEE 802.15.4, accommodating characteristics of that data-link 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 among such constrained devices, and over this particular data-link layer. 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 layer 2.5 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 ("6LoWPAN Ad Hoc On-Demand Distance Vector Routing" [8]). 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 Route Replies and of sequence numbers). However, 6LoWPAN was addressing other issues regarding adapting IPv6 for IEEE 802.15.4, such as IP packet header compression, 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) [9] in 2011.

1.1.3. Finally, Towards LOADng

While LOAD [8] development was suspended by the 6LoWPAN working group, pending the results from ROLL and experiences with RPL, reactive protocol derivatives live on: IEEE 802.11s [10] is based on the principles of Route Request/Route Reply exchanges for Route Discovery, and the ITU-T G3-PLC (Power Line Communication) standard [11], published in 2011, specifies the use of [8] at layer 2 or 2.5, for providing mesh-under routing for utility (electricity) metering networks. Justifications for using a reactive routing protocol in preference to RPL include that such protocols better supports bi-directional data flows such as a request/reply of a meter reading, as well as algorithmic and code complexity [12]. The emergence of LLNs thus triggered a renewed interest in reactive routing protocols for specific scenarios, resulting in work within the IETF [13] for the purpose of standardisation of a successor to LOAD – denoted LOADng (the Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation). LOADng incorporates the experiences from deploying LOAD – including, but not only, in LLNs – and was, included in a subsequent revision of the G3-PLC ITU-T standard for communication in the "smart grid" [14].

1.1.4. Routing – Only Half the Solution?

Different routing protocols for LLNs have been proposed and standardised, including RPL [9] and LOADng [14]. While such protocols make different trade-offs and are of vastly different philosophies, they are united in the fact that when a link that has been actively used as part of a routing path fails, it is up to the routing protocol to recover by discovering alternative paths. Data flows are typically either buffered or dropped during this recovery. Dropping data flows while a routing protocol converges is "harmful", since such traffic will have to be sent again (consuming energy, and creating additional traffic on the links in the network). Unfortunately, so is buffering data flows, as it imposes additional requirements on devices having sufficient memory to hold the buffers. A third alternative is opportunistic forwarding of traffic during route recovery, as proposed in DFF (Depth-First For-

[2] https://www.ietf.org/mail-archive/web/manet/current/msg18997.html

warding in Unreliable Networks" [15]) – as a complement to an LLN routing protocol.

1.2. Statement of Purpose

This paper presents, studies, and evaluates a "complete", yet simple, adaptive, and modular approach to routing in LLNs. Using LOADng as the routing protocol core, this paper explores several extensions for adapting the protocol to different topologies, traffic characteristics, and other conditions: "Smart Route Request" and "Expanding Ring Search" are proposed to improve Route Discovery efficiency ; a "Collection Tree Protocol" is introduced to reduce the routing overhead for building a collection tree ; integration of DFF, and extensions to DFF, are studied to allow a LOADng-routed network rapid recovery from data packet forwarding failures.

Preliminary results have been published in [16, 17, 18, 19], exploring different extensions of LOADng. This paper further extends these results by presenting the protocol components as elements of a modular framework, considering the interoperability and exploring the security vulnerabilities. A generalised security framework for LOADng is also proposed.

1.3. Paper Outline

The remainder of this paper is organised as follows: section 2 presents the LOADng routing protocol, its operations, and other characteristics. Next, a set of extensions to the core protocol are presented. Section 3 studies a way of exploiting existing router state, for unicast route requests – with the goal to reduce the overhead of Route Discovery. Section 4 discusses the application of expanding ring search to the LOADng protocol to improve the Route Discovery efficiency by using neighbourhood routing information. Section 5 explores an extension to allow efficient construction of a collection tree for multipoint-to-point traffic, while introducing minimum routing overhead. Section 6 discusses the use of DFF in conjunction with LOADng in lossy network scenarios. These extensions present performance improvements possible and desirable in different scenarios – and are, also, both interoperable with each other, and with the "core" routing protocol: routers with and without these extensions can coexist in the same network. For each of these extensions, their security characteristics are also evaluated. Section 7 evaluates the performance of different extensions, based on which their applicability is discussed. Section 8 introduces a security framework for LOADng – emphasising the necessary elements for protecting the integrity of the routing infrastructure of a LOADng-routed network. Finally, section 9 concludes this paper.

2. LOADng – Core Protocol

A lightweight reactive distance-vector protocol, LOADng inherits the basic protocol operations of all reactive routing protocols: on-demand generation of Route Requests (RREQs) by a router (originator) for discovering a path to a destination, forwarding of such RREQs until they reach the destination router, generation of Route Replies (RREPs) upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator. If a path is detected broken, *i.e.,* if forwarding of a data packet to the recorded next hop on the path to the destination is detected to fail, local path repair can be attempted, or a Route Error (RERR) message can be returned to the originator of that data packet.

LOADng has been designed with the philosophy of a *minimal core*, containing a small set of protocol operations, and with implementation requirements lending itself to a simple implementation with a small code footprint, as well as small operational state requirements. This minimal core is, at the same time, carefully crafted so as to enable extensions (when needed) to be developed, and deployed, in a fashion remaining interoperable with this minimal core. This paper details both this minimal core, and a certain number of extensions. Thus, distinct from its predecessors, LOADng has the following characteristics:

- **Modular design:** The core specification defines the simple and lightweight core functions of the protocol. LOADng is extensible, by way of a flexible packet format, permitting addition of arbitrary attributes and information via new message types and/or TLV (Type-Length-Value) blocks. The LOADng protocol core is detailed in this section, with subsequent sections illustrating the use of the flexible architecture of LOADng for developing (interoperable and backwards compatible) protocol extensions.
- **Flexible Addressing:** Address lengths from 1-16 octets are supported[3]. The only requirement is that within a given routing domain, all addresses are of the same address length.
- **Metrics:** Support for different metric types, beyond simple hop-count.
- **Destination-Replies:** Intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination. All messages (RREQ or RREPs) generated by a given LOADng Router share a single unique, monotonically increasing sequence number. While [4, 5] both allow intermediate RREPs, the rationale for this simplification in LOADng is reduced complexity of protocol operation and reduced message sizes – which section **??** will show to be without significant influence on performance. Allowing only the destination to reply to an RREQ also simplifies the task of securing the protocol, as discussed in section 3.4.

[3] *i.e.,* IPv6, IPv4, 6LowPAN short addresses, Layer-2 addresses etc. are all supported by LOADng

2.1. LOADng Message Format

LOADng defines four types of protocol messages:

Route Request(RREQ)
Generated by a LOADng Router, when presented with a data packet to a destination, for which it has no valid route, and containing the address of the destination for that data packet. Table 1(a) illustrates the fields in an RREQ message.

Route Reply(RREP)
Generated by a LOADng Router, when it receives and processes an RREQ containing an address for which the LOADng Router is responsible[4] as a response to an RREQ. Table 1(a) illustrates the fields in an RREP message.

Route ReplyAcknowledgement (RREP-ACK)
Generated by a LOADng Router as a response to an RREP, in order to signal to the neighbour that transmitted the RREP that the RREP was successfully received. Table 1(b) illustrates the fields in RREP-ACK message.

Route Error(RERR)
Generated by a LOADng Router when a link on an active path to a destination is detected as broken, by way of inability to forward a data packet towards that destination. Table 1(c) illustrates the fields in an RERR message.

These LOADng protocol messages are encoded as messages within the "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format" [20]. This format is TLV-based, essentially offering a set of fixed header fields (type, address length, originator address, hop-limit, hop-count and sequence number) followed by a block of "message TLVs". After the block of "message TLVs" follows a block of addresses, with associated "address block TLVs" assigning semantics to each address[5]. The TLV format of [20], furthermore, is "extended" in that each TLV has a type, which specifies the "kind" of information carried in the TLV, and an optional type-extension field, which may specify how the information is to be interpreted. For example, and as used in LOADng, a TLV can be of type "METRIC" and use the value of the type-extension field to specify how the value carried in the "METRIC" TLV is to be interpreted, *e.g.,* delay, bitrate, loss rate, etc. This use of the packet/message format in [20] enables unmodified use of protocol parsers, even when designing an extensible and flexible protocol: extensions can add information to existing messages, without rendering a message unreadable by non-extended protocol implementations. Furthermore, careful design of a protocol and of extensions thereto can permit correct operation of extended and non-extended protocol implementations in the same deployment.

2.2. Protocol Message Extensions and Flags

Several of the protocol extensions, presented in this paper, necessitate adding a piece of information to an existing control message. By way of LOADng utilising the 'Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format" [20], this is easily accomplished by adding TLVs to LOADng control messages. An unextended LOADng implementation will not recognise a TLV for an extension, but will be able to skip over the TLV, and correctly parse the rest of the control message.

Several extensions propose to introduce a binary "flag" in a control message. While not specified in this paper, there are several way in which this can be undertaken. For example, each "flag" can correspond to a TLV type – or, a "TLV type "Flags" with as value a bit-vector, can be introduced.

2.3. LOADng Protocol Operations

LOADng retains the basic reactive protocol operations, including *Route Discovery* and *path maintenance*, albeit in a greatly simplified form, described in this section.

2.3.1. Route Discovery

During Route Discovery, RREQ messages are flooded through the network. In each intermediate LOADng Router (non-destination), the metric in the message is updated, and a path to the RREQ originator is recorded. The message is forwarded until it gets to the destination. As shown in figure 1(a), Router S is the originator, and router D is the sought destination.

In LOADng, only the destination LOADng Router of the RREQ message will respond with an RREP, sent in unicast to the source of the RREQ, shown in figure 1. A path to the destination (LOADng Router D in this example) is thus built.

2.3.2. Path Maintenance

Path maintenance is performed when an actively used path fails. Path failure is detected by way of a data packet not being deliverable to the next hop towards the intended destination[6]. In LOADng, when a path failure is detected, an RERR message is generated, sent as unicast along the path to the source of data packet. On receiving the RERR at the source of data packet, a new path discovery should be performed.

Again, employing end-to-end signalling only eases the task of securing the protocol, as discussed in section 8.

[4] *i.e.,* an address of a destination, local to that LOADng Router

[5] Of passing note, the presence of absence of an address does, in [20], not carry any semantics on its own, but only by the TLV(s) associated to the address. This is to facilitate protocol extensions, and is strictly followed by LOADng.

[6] *e.g.,* by way of absence of a data-link layer acknowledgement.

Field name	Length (bits)	Comment
msg-type	8	Message type, either RREQ or RREP
addr-length	4	length of the address
hop-limit	8	hop limit of the message
hop-count	8	hop count of the message
flag	8	message flags
seq-num	16	sequence number of the message
metric-type	8	metric type
metric	variable	metric value
originator	8-128	message originator
destination	8-128	message destination

(a) RREQ and RREP message fields

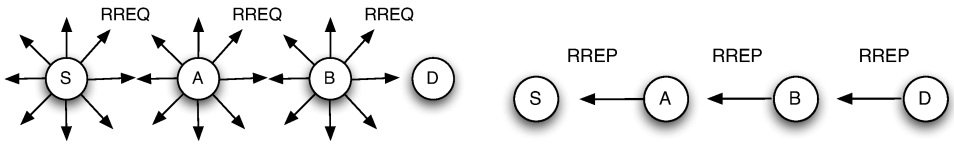
Field name	Length (bits)	Comment
msg-type	8	Message type RREP-ACK
addr-length	4	length of the address
seq-num	16	sequence number of a received RREP
destination	variable	originator of received RREP

(b) RREP-ACK message fields

Field name	Length (bits)	Comment
msg-type	8	Message type RERR
addr-length	4	length of the address
hop-limit	8	hop limit of the message
hop-count	8	hop count of the message
errorcode	8	indicates the error event
unreachableAddress	variable	destination of failed packet
originator	variable	RERR message originator
destination	variable	RERR message destination

(c) RREQ and RREP message fields

Table 1: LOADng message fields



(a) LOADng Route Request (b) LOADng Route Reply

Figure 1: LOADng path discovery without intermediate RREP. S initiates an RREQ for D.

2.3.3. Path Metrics

When receiving an RREQ or RREP, a router updates the metric – the "cost" of the path to the originator of that RREQ or RREP – and uses this updated cost both for internal processing (updating routing tables) and for setting the <metric> field. In its most basic form, this may simply be to "increase the cost by one", corresponding to a hop-count metric.

Because of the heterogeneous nature of links (wireless, PLC, ...) different metrics may be used by devices, such as delay, data rate, packet loss rate, etc. – some of them may even co-exist in the same network. Therefore, LOADng supports different metric types by providing <metric-type> and <metric> fields in the message.

A LOADng Router generating an RREQ or an RREP message specifies which metric type is desired. LOADng Routers receiving the message will process it and update path metric information according to the metric type, if they can. In any event, a "default" hop-count metric is always maintained for all RREQ/RREPs. Thus, a LOADng Router receiving a message with a metric type otherwise unknown to it, can fall back to the default hop-count metric. This enables that multiple metric types can be used, while maintaining basic interoperability.

3. Efficient Route Discovery and Smart Route Request

Reducing the overhead, delay and complexity of the Route Discovery process (RREQ/RREP exchange) is a key to adapte on-demand routing protocols for use in constrained environments. As indicated in section 2, some reactive routing protocols [4, 5] allow a (non-destination) router having a path to the destination sought in an RREQ, to respond by generating an "intermediate RREP" to the originator, and a "gratuitous RREP to the sought destination".

This section discusses the rationale for LOADng not including "intermediate/gratuitous RREPs", and presents an alternative mechanism denoted *Smart Route Requests* (SmartRREQ). The SmartRREQ mechanism attains a performance comparable to that of "intermediate/gratuitous RREPs", while incurring smaller protocol messages, simpler protocol message processing, and offers advantages with respect to securing routing protocol operations. Furthermore, this mechanism remains interoperable with the *minimal core* of LOADng: a network can contain a mixture of routers supporting, and not supporting, SmartRREQ.

3.1. Intermediate Route Replies: to be, or not to be...

During the Route Discovery process of [4, 5] , an intermediate router can generate an intermediate RREP in response to an RREQ if it has a valid path to the destination sought – and must, if so, also generate a gratuitous

RREP and send this to the desired destination in order to establish a complete and bi-directional route. In order to avoid routing loops when permitting intermediate routers to generate intermediate RREPs, an RREQ must carry an RREQ ID, destination sequence number, and originator sequence number in RREQ messages – recorded and maintained by intermediate routers, and used for when processing RREQs and RREPs. This process is illustrated in figure 2: routers A and B already have a valid path to D. When S initiates a Route Discovery for D and broadcasts an RREQ, A will receive it and respond by generating an *intermediate RREP* to S and a *gratuitous RREP* to D, both via unicast.

In LOADng, as illustrated in figure 1, even if LOADng Routers A and B already have available and valid paths to D, intermediate RREPs are prohibited, so as to reduce the control message size and, still, guarantee loop freedom[7]. A LOADng Router, receiving an RREQ, is either the ultimate destination – and, if so, must respond by an RREP – or, it is an intermediate LOADng Router and, if so, has to rebroadcast the RREQ, even if it otherwise has a valid path to the destination. [19] shows that this simplification of LOADng renders the protocol more adapted to constrained environments, attaining lower routing overhead and fewer collisions.

While allowing only the destination to reply to an RREQ does reduce the size of RREQ/RREPs, this may conversely result in more RREQ (re-)transmissions in certain scenarios. Consider the obvious case where a set of LOADng Routers in the same part of the network topology, for example, all seek a path to a gateway: not using the topology information in intermediate LOADng Routers will cause all RREQs to have to transverse the network, and RREPs to be sent back.

Figure 3(a) shows five LOADng Routers from an N-router network (where $N > 5$). S initiates an RREQ for D. The neighbours of S: *A, B, C* already have valid routes to D. With LOADng, all LOADng Routers other than the destination have to retransmit the RREQ, *i.e.,* there are at least $N - 1$ RREQ transmissions. In contrast, with intermediate RREP, Route Discovery will remain local to the neighbourhood of S: *A, B, C* would generate intermediate RREPs to S. Although in this example, those RREPs would be discarded as providing longer paths, using intermediate RREPs would avoid RREQs being disseminated blindly through the whole network.

In some network types, such as sensor networks, it is common to have sensor-to-root (multipoint-to-point – or MP2P) traffic as illustrated in figure 3(a) with D being the root. While eliminating intermediate RREP can reduce the size of control message and simplify the protocol process, the effect of blindly flooding RREQ cannot be ignored in this kind of scenarios.

[7] The sequence numbers from [4, 5] guarding against loops are removed from LOADng to better adapt to links with tiny MTUs

3.2. Smart Route Request

To avoid blind flooding of RREQ in scenarios where MP2P traffic prevails, *SmartRREQs* are proposed. Retaining the lightweight nature of LOADng, and incurring no additional signalling[8], SmartRREQ permits benefitting from existing routing information in intermediate routers during a Route Discovery.

When SmartRREQ is used, a LOADng Router initiates a Route Discovery by broadcasting an RREQ message with a *smart-rreq* flag set (henceforth, a *RREQ_SMART*).

On receiving an *RREQ_SMART*, an intermediate[9] LOADng Router performs the following procedure:

1. If the intermediate LOADng Router has a valid path to the destination, AND the <next-hop> field of the corresponding routing tuple is not equal to the previous hop address of the RREQ, then the *RREQ_SMART* is unicast to the <next-hop>.
2. Otherwise the *RREQ_SMART* is broadcast, as usual, to all its neighbours.

This is illustrated in figure 3(b): S solicits a path to D. A and B already have paths to D, and upon receiving the *RREQ_SMART* initiated by S will unicast the RREQ, according to their routing table. When the *RREQ_SMART* arrives the destination, the RREP is unicast as in figure 1(b).

With this, in the example in figure 3(a), an *RREQ_SMART* message will stay local, and rather than being flooded to the whole network will be unicast to the destination only.

If an intermediate LOADng Router detects a broken link when trying to send a unicast *RREQ_SMART*, then it should broadcast the *RREQ_SMART* instead.

3.3. Interoperability Considerations

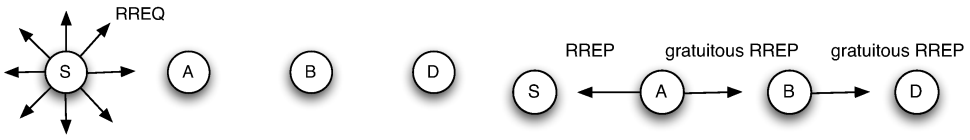
SmartRREQ is an extension that is fully interoperable with unextended LOADng: an unextended LOADng can correctly parse the *RREQ_SMART* message, and will handle it as normal RREQ message (*i.e.,* will always broadcast). Conversely, a LOADng router with the SmartRREQ extension is able to process and forward all RREQ messages as unicast or broadcast.

3.4. Security Considerations

In addition to attaining smaller control message and reduced processing complexity, an important reason for eliminating intermediate/gratuitous RREP is security: with intermediate RREPs, any router with an available path to the destination is able to respond to an RREQ by generating an RREP. This is, however, based on the assumption that all the intermediate routers are "honest". In a malicious environment, an attacker can, simply, spoof a route

[8] Neither in form of additional message types nor additional content in existing message types.

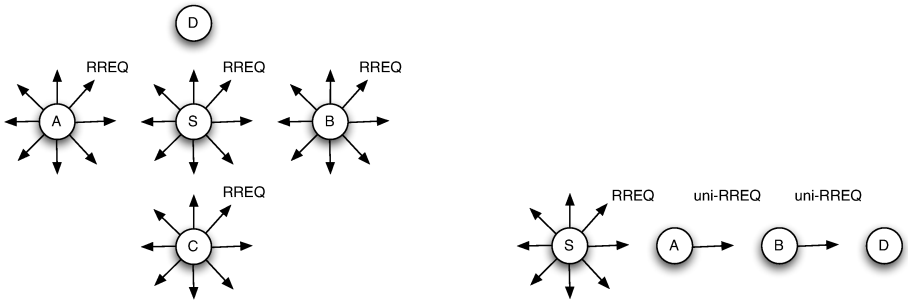
[9] *I.e.,* which is neither the source nor destination.



(a) Route request

(b) Intermediate/Gratuitous route reply

Figure 2: Route Discoverywith intermediate RREP. S initiates an RREQ for D. A and B has already an available path to D



(a) LOADng Route Discovery. S initiates an RREQ for D. A, B and C have already routes to D.

(b) LOADng Route Discovery with smart RREQ. S initiates an RREQ for D. A and B already have an available path to D.

Figure 3: LOADng Route Discovery options.

by sending an RREP to the originator of an RREQ[10]. In this case, the recipient of an RREP cannot validate if the path advertised really exists – even when using a digital signature or timestamp mechanism on the RREPs. Thus, intermediate/gratuitous RREPs render a network vulnerable to man-in-the-middle attacks.

When using LOADng, with or without SmartRREQ, only the destination router is allowed to generate RREPs. Thus, the destination router can include Integrity Check Values (ICVs), signatures, timestamps, etc., making it is possible for a recipient of the RREP to verify the integrity of the message. With SmartRREQ, this even while retaining the main advantage of intermediate RREP (*i.e.,* reduced overhead).

4. Expanding Ring for LOADng

Expanding Ring flooding is a technique aiming to limit the need for network-wide dissemination of RREQs. A router will at first send an RREQ with a reduced TTL (Time-To-Live) – causing the RREQ to not be flooded through the entire network, but only up to a limited distance. If the destination sought receives the RREQ, an RREP is generated and a network-wide flooding is avoided. For protocols allowing intermediate/gratuitous RREPs, if an intermediate router has a path to the sought destination, an intermediate/gratuitous RREP is generated, and a network-wide flooding is avoided. If LOADng is used with the SmartRREQ-extension, if an intermediate router has a path to the sought destination, a SmartRREQ is generated, and a network-wide flooding is avoided.

If no RREP is received by the originator in expected delay, another RREQ message is, after a brief delay, generated with increased TTL to eventually cover the entire network.

Note that while this may be an advantage in some cases, this mechanism can also be a double-edged sword, and cause increased rather than decreased control traffic: if no router closer to the originator of an RREQ than the final destination has a path to the destination, much more control traffic is generated by such repeated Expanding Ring floods. With this caveat, this section explores an expanding ring extension for LOADng.

4.1. Expanding Ring flooding for LOADng

The Expanding Ring flooding extension defines a new TLV for the RREQ message type, called MNB (Maximum Number of Broadcasts), to limit the number of hops allowed for RREQ broadcasting. The value of that TLV is decreased by one when the RREQ is broadcast by a LOADng Router. The following parameters are used:

- MNB_START. The initial value of MNB. This is a small value to limit the initial search range of Route Discovery. It is set to 1 in this study.
- MNB_INCREMENT. The MNB increment when a previous search failed. It increases the search range by number of hops. It is set to 2 in this study.

[10]Either for interfering with or hindering path construction, or to clandestinely attract traffic for inspection, before relaying it to the ultimate destination so as to exist unnoticed in the network.

- MNB_THRESHOLD. The maximum number of hops allowed for expanding ring search, beyond which network-wide classical flooding is used. It is set to 7 in this study.

When initiating a Route Discovery in LOADng and with Expanding Ring flooding enabled, the originating LOADng Router includes an MNB TLV with a value of MNB_START. If the timeout (normally two times the network traversal time) expires without a matching RREP having been received, a new RREQ is broadcast with an MNB TLV with a value incremented by MNB_INCREMENT. This continues until the value of the MNB TLV in the RREQ reaches MNB_THRESHOLD, beyond which the Route Discovery can either be declared a failure or continued with an MNB with a value of MAX_HOP_COUNT (*i.e.,* 255), which corresponds to a network-wide flooding.

Combined with SmartRREQ, as introduced in section 3, Expanding Ring Route Discovery can be divided into two parts: (i) broadcast RREQs until a LOADng Router with a valid path to the sought destination is encountered, then (ii) unicast RREQs towards the destination. Expanding Ring flooding tries to limit the number of LOADng Routers impacted, and the number of messages required, by (i).

When an intermediate LOADng Router receives an RREQ, it performs the following procedure before transmitting the RREQ:

1. If, the intermediate LOADng Router r has an available path to the destination. The RREQ message is unicast to the destination by using SmartRREQ. The RREQ MNB field is left unchanged.
2. Otherwise If the value of the included MNB is equal to 0, then the RREQ is discarded.
3. Otherwise, the RREQ is re-broadcast as, with the value of the included MNB TLV is decreased by one.

Figure 4 illustrates Expanding Ring flooding in LOADng: LOADng Router S initiates a Route Discovery for D, the LOADng Routers with double circles already have a valid path to D. In figure 4(a), MNB_START is set to 1, and the MNB_INCREMENT is set to 2, thus no RREP results from the first RREQ with MNB = 1. Then, in figure 4(b), S increases MNB by 2 – the RREQ reaches two LOADng Routers that already have valid paths to D, and that therefore by way of *SmartRREQ* unicast the RREQ to D – which will respond by returning an (unicast) RREP.

4.2. Interoperability Considerations

This extension defines the MNB TLV, to be inserted into RREQs. Unextended LOADng routers will not be able to recognise the MNB TLV, and thus cannot reduce the value of the MNB TLV when forwarding an RREQ. This will merely "extend" the search range – in the worst case – simply degrade Expanding Ring flooding to classical flooding. Thus, the benefit of this extension be limited if LOADng Routers in the network do not support the Expanding Ring extension – but extended and unextended routers will interoperate.

4.3. Security Considerations

The value of the MNB TLV is mutable, *i.e.,* it is changed, on every broadcasting hop, and thus cannot be covered by a digital signature generated by the originator of the RREQ. Consequently, a malicious LOADng Router, interception an RREQ, can modify the value of an MNB TLV undetected, *e.g.,*set it to MAX_HOP_COUNT (disable the expanding ring) or to 1 (cause Route Discovery failure, akin to if it didn't forward the RREQ).

5. Collection Trees for LOADng

LOADng (extended, or not, with SmartRREQ and Expanding Ring) discover paths between any (orginator,destination) pairs, for carrying point-to-point traffic. In some LLNs, another traffic pattern, called multipoint-to-point, prevails - where one or more devices act as data sink for all traffic – and where and all the other devices in the network communicate with the data sink. Discovering all these paths to the data sing individually may be inefficient, motivating a LOADng extension allowing efficient construction of a "collection tree", whereby all routers are provisioned with paths towards the data sink (the "root" of the collection tree).

Denoted LOADng-CTP, this extension is based on the operation and packet format of LOADng.

5.1. Collection Tree Signalling

LOADng-CTP introduces two flags to RREQ messages

- RREQ_Trigger: when set, a receiving LOADng Router will be triggered to discover with which of its neighbours it has bi-directional links.
- RREQ_Build: when set, a receiving LOADng Router will build a route to the root.

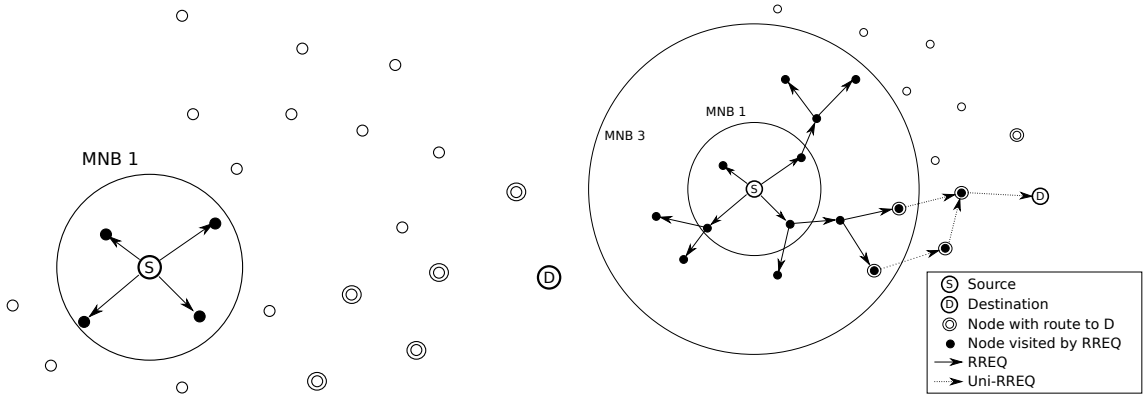
In addition, an additional HELLO message is defined, in order to permit verification of bidirectionally of links before admitting them to the collection tree. The HELLO message is generated when receiving an RREQ_Trigger, and serves to ensure that only bi-directional links are included in the collection tree.

5.2. Collection Tree Construction

The LOADng Router, wishing to be the root of the collection tree generates an RREQ with *RREQ_Trigger*. Both the originator and destination of the *RREQ_Trigger* are set to an interface address of the root.

On receiving an RREQ_Trigger, a LOADng Router:

- Records the address of the sending LOADng Router (*i.e.,*the neighbour, from which it received the RREQ_ Trigger message) in its *neighbour set*, with the status *HEARD*.
- If no earlier copy of that same RREQ_Trigger has been previously received:



(a) MNB = 1 (b) MNB = 3

Figure 4: An example of Expanding Ring flooding initiated by S for D. The white LOADng Routers are not visited by RREQs, but would have been without Expanding Ring flooding enabled.

  - The RREQ_Trigger is retransmitted, subject to a jitter of *RREQ_Jitter* and according to [21], so as to reduce the probability of collisions.
  - Generates a HELLO message, subject to jitter of *HELLO_Jitter*, also according to [21] [11]. When the scheduled HELLO message is generated, it includes the addresses of all the neighbours, from which it has received an RREQ_Trigger.

On receiving a HELLO message, a LOADng Router:

- If its own address is included in the HELLO message, it records the address of the sending LOADng Router (*i.e.,* the neighbour, from which it received the HELLO) in its *neighbour set*, with the status *SYM* (bi-directional).

Thus, each LOADng Router will learn with which among its neighbours it has a bi-directional (SYM) or uni-directional (HEARD) link.

2×Net_Traversal_Time after having generated the the RREQ_Trigger, the root generates and broadcasts a RREQ_Build. On receiving a RREQ_Build, a router:

- Verifies if the RREQ_Build was received from a neighbour to which it has a bi-directional link. If not, 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*.

[11] Where *HELLO_Jitter* > *RREQ_Jitter*

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

If also paths from the root to other routers (sensors) inside the network is required, each LOADng Router receiving an RREQ_Build will unicast an RREP to the root, transmitted and processed according as normal RREP message. In this way, downward traffic is also enabled.

5.3. Collection Tree Maintenance

During the process described in section 5.2, control messages may be lost, causing some LOADng Routers to not be included in the resulting collection tree. Furthermore, the routing entries may expire because of not being updated in a timely fashion. Both of those result in a path to the root not being available in some of the LOADng Routers.

Worst case, a LOADng Router with data traffic to send to the root will initiate Route Discovery – however, if a collection tree is present in the network, it is likely that a neighbour will have a path to the root, and thus in order to avoid network-wide RREQ broadcast, the SmartRREQ extension introduced in section 3 can be employed.

When a link on an active path 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 an may trigger a new Route Discovery.

5.4. Interoperability Considerations

An unextended LOADng Router will forward RREQ_Trigger and RREQ_Build message as normal RREQ messages, however cannot generate HELLO messages. As a consequence, while unextended LOADng Routers will not be able to be verified as bi-directional neighbours, and will as such not be participating in a collection tree. Thus, the benefit of this extension be limited to the connected set of LOADng Routers that support LOADng-CTP – with unextended LOADng Routers (or, extended LOADng

Routers which are separated from the root by one or more unextended LOADng Routers) falling back to Route Discovery for finding paths to the root.

5.5. Security Considerations

The collection tree building process relies on strictly ordered message sequences: RREQ_Trigger message for triggering the building process, then HELLO message for bidirectionality check of neighbours, and RREQ_Build message for collection tree build in the end. The message emission is controlled by router parameters Net_Traversal_Time, RREQ_Jitter, and HELLO_Jitter.

The correct receiving order can be expected if those parameters are set properly – however, in deployments, mis-configured routers, or even compromised routers that emit messages out of order, may exist. For example, if a router sends a HELLO message before it receives all the RREQ_Trigger messages from its neighbours, or an RREQ_Build message is received before the HELLO message exchange finished, the router cannot identify its bidirectional neighbours correctly – thus is not able to join the collection tree as expected. In that case, *i.e.,* when faced with a misconfigured or malicious router preventing the collection tree from being built, the protocol falls back to Route Discovery as described for LOADng-Core (possibly with SmartRREQ).

6. Depth-First Forwarding with LOADng

The second "L" in LLN means "lossy", *i.e.,* communication channels are of low capacity, time-varying and with high loss rates.

Routing protocols for LLNs, such as LOADng, are typically designed to limit the routing overhead imposed to networks as much as possible, and to be adapted to the varying nature of communication media. However, even once paths have been found, these paths may be unusable from time to time due to different reasons: presence of noise or interferences, low power supply in certain devices, uni-directional links, etc. From a routing protocol point of view, when such link failure is detected, it needs some extra signalling and/or time to recover and discover new, valid paths. During this recovery phase, data packets being sent over the broken link must either be buffered and wait for the path recovery, or be dropped because of lack of memory in constrained devices.

To alleviate the effects of inevitable random link failures in LLNs, a set of data forwarding mechanisms have been proposed [22]. Those mechanisms that work in the "forwarding plane" use data packets to detect loops, update routing tables, and reroute data packets through alternative paths when the primary paths are broken. By doing so, the packets that are originally forwarded through failed links can be recovered, instead of being dropped.

This section studies integration of a Depth-First Forwarding (DFF) extension for LOADng, to improve the data delivery reliability over lossy links.

6.1. DFF Overview

"Depth-First Forwarding in Unreliable Networks" (DFF) [15] is an experimental data forwarding standard by the IETF, which proposes a mechanism for rapid and localised 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.

When a packet is to be forwarded by a router using DFF, the router creates an ordered list of *Candidate Next Hops* for that packet. DFF proceeds to forward the packet to the first next hop in the list. If the transmission was not successful (as determined by the underlying link layer) or if the packet was "returned" by a next hop to which it had been sent before, the router will try to forward the packet to the subsequent next hop on the list based on "depth-first searching". A router "returns" a packet to the router from which it was originally received once it has unsuccessfully tried to forward the packet to all elements in the "Candidate Next Hop List" (CNHL). If the packet is eventually returned to the originator of the packet, and after the originator has exhausted all of its next hops for the packet, the packet is dropped.

To support duplicate packet detection and loop detection, DFF specifies a DFF header to be used in data packet, which is processed by each intermediate router. The header mainly includes:

- Sequence number, containing an unsigned integer to identify the packet.
- DUP field, a "duplicate" flag tagging a duplicate packet.
- RET field, a "return" flag tagging a returned packet.

Each router running DFF maintains a *Processed Set*, which records sequence numbers of previously received data packets, as well as a list of next hops to which each data packet has been successively sent, as part of the depth-first forwarding mechanism. The "Processed Set" consists of "Processed Tuples", of the form:

`(P_orig_address, P_seq_number, P_prev_hop, P_next_hop_neighbor_list, P_time)`

where:

- `P_orig_address` is the originator address of the received packet;
- `P_seq_number` is the sequence number of the received packet;
- `P_prev_hop` is the address of the previous hop of the packet;

- `P_next_hop_neighbor_list` is a list of addresses of next hops to which the packet has been sent previously, as part of the depth-first forwarding mechanism;
- `P_time` specifies when this tuple expires and must be removed.

6.2. Integrating DFF with LOADng

DFF requires that a LOADng Router has a list of all its bi-directional neighbours available for constructing the CNHL for a data packet. [15] specifies that an external mechanism is to be in place to provide that list, and suggests the use of [23] – which is implemented and used for the purpose of the performance studies in this paper.

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 CNHL 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 CNHL 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 DFF mechanism is activated when:

- A LOADng Router receives a data packet from another LOADng Router, for which it does not have a corresponding entry in the routing table, OR
- Forwarding of a data packet to the next hop, as indicated by LOADng (*i.e.,* the first entry in the CNHL) fails (either by way of the packet being returned by DFF, or by a link layer acknowledgement being absent).

When a routing failure is detected, the LOADng Router performs the following steps:

- data packets are sent according to the DFF forwarding rules, as described in section 6.1; AND
- an RERR is sent to the originator of that data packet, as described in section 2.3.2.

An RERR message is sent since while DFF tries to 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).

Figure 5 gives an example of how LOADng works with DFF. Router *A* is sending data packets to LOADng Router *D*. The path initially discovered by LOADng is *A-B-F-D*. The CNHL at LOADng Router *B* is (*F, C, E. G*). *F* is the first element in CNHL because that is the next hop suggested by the routing table. Without further topology information, the remainder of the list is, simply, a lexicographically ordered list of B's remaining neighbors (excluding *B*'s previous hop *A*).

If, the link between LOADng Routers *B* and *F* breaks, as detected by *B* failing to deliver a data packet to *F* *B* would remove *F* from the CNHL, and forward the data packet to the next entry in the CNHL – to *C*. As *C* is not on any path to LOADng Router *D*, the packet would eventually be returned to device *B*, with *RET* (return) flag set, after depth-first searching the "cloud" in figure 5. Getting the data packet returned, LOADng Router *B* attempts delivery via the next element in CNHL, *E*, which happes to have a path towards *D* through *H*.

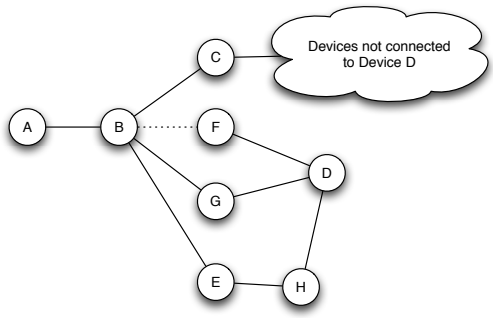


Figure 5: An example of DFF. Router *A* sends packets to LOADng Router *D*. The dashed line represents a broken link.

[15] specifies that the CNHL is constructed per data packet. Therefore, in the example illustrated above, before the routing protocol recovers from the path failure, *all* the subsequent data packets from *B* to *D* will follow the order (*C, E, G*), and explore the same "blind alley" in the network by way of *C*.

6.3. The DFF++ Destination Field Extension

Section 6.2 describes the integration of DFF and LOADng. According to [15], without further topology information, all the data packets sent along a broken path can only try its neighbours "blindly", as illustrated in the example of figure 5.

This section considers a simple extension to DFF, henceforth DFF++, for establishing "memory" across several data packets for the same destination. This extension (i) piggy-bags information already maintained by DFF, and (ii) maintains information only temporarily, for as long as DFF otherwise maintains information pertaining to forwarded packets.

The DFF++ extension adds an element to *Processed Tuple*, thus:

`(P_orig_address, P_seq_number, P_prev_hop, P_next_hop_neighbor_list, P_time, P_dest_address)`

where:

- `P_dest_address` indicates the destination address of the received packet.

The proposed DFF++ extension also imposes an additional constraint on P_next_hop_neighbor, which is that:

- `P_next_hop_neighbor` must be ordered such that the last element (`P_next_hop_neighbor_list[LAST]`) of that list contains the last neighbour to which delivery to P_dest_address was attempted (and all previous entries in that list contain successively earlier attempts, with the first element of the list containing the first neighbour to which delivery was attempted).

On receiving a data packet not destined to a LOADng Router itself, DFF++ defines the following process for selecting an ordered CNHL, within the constraints and guidelines from section 11 in [15].

Find the (unique) Processed Tuple, where:

- `P_dest_address` == the destination address of the data packet; AND
- which has the greatest `P_time`.

Using that tuple, the CNHL is constructed thus (where $\oplus$ indicates list concatenation, \ indicates list exclusion, RT(address) is the next hop on the shortest path to the destination from the routing table – if any, and NS indicates the set of neighbours of the device):

1. CNHL = RT(`P_dest_address`)
2. CHNL = CHNL $\oplus$ `P_next_hop_neighbor_list[LAST]`
3. CHNL = CHNL $\oplus$ {NS\ {`P_prev_hop`} \ `P_next_hop_neighbor_list`}
4. CHNL = CHNL $\oplus$ `P_next_hop_neighbor_list`

Where 1) satisfies the requirement from [15] that the first element in the CNHL is the next hop, indicated by a routing table (if present). Items 2) and 3) capture "pick up where the most recent data packet delivery to the same destination left off". Specifically, 2) is the neighbour, last tried for the most recent packet to the same destination, and which is not yet confirmed as having failed (in which case there would be a subsequent entry in the list, except if all neighbours had been tried and failed), 3) includes all other so far untried (by the most recent data packet delivery for this destination) neighbours. Finally, 4) – which is an optional step in DFF++ – includes all previously (by the most recent data packet delivery) tried neighbours – excluding, of course, the one from which the data packet was received – capturing the fact that a previous failure may have been due to transient losses.

Returning to the example in figure 5, one of the issues raised in section 6.2, is alleviated:

1. The initial CNHL for the first data packet arriving at B for destination D will – using the same ordering (routing table entry first, then the "worst-case" lexicographical order) – be {F, C, E, G}.
2. Initial delivery is attempted via F (which is added to the end of P_next_hop_neighbor_list) and fails, and delivery via C is attempted (which is added to the end of P_next_hop_neighbor_list).
3. Delivery via C also fails (no path via C to D), and delivery is now attempted via E (which is added to the end of P_next_hop_neighbor_list) – as there is a valid path to D via E, delivery succeeds, and the P_next_hop_neighbor[LAST] for that processing tuple now contains E.
4. Other data packets for D, arriving at B, *before* the routing protocol (if any) has recovered and provided an entry in the routing table for D, will, using the DFF++ CNHL construction rule, result in a CNHL of:
   - If they arrive after step 3), {E, G} – thus avoiding the "broken link" to F, as well as the "blind alley" that would be attempting delivery via C.
   - If they arrive after step 2) but before step 3), {C, E, G} – thus avoiding the "broken link" to F, but not the "blind alley" that would be attempting delivery via C
   - If they arrive before step 2), {F, C, E, G} – thus offering no improvement over DFF, but also no additional penalty.

Note that DFF++ avoids the problem of repeatedly attempting delivery to a given destination via "blind alleys" and over "recently detected broken links", but does not attempt at offering "shortest paths" – that remains under the auspices of a routing protocol (if any) in the network. Also, DFF++ does not affect interoperability: the extension does not introduce any new signals or any new external behaviours, but simply offers guidance for how to order the CNHL for a data packet. The specification of DFF [15] specifically encourages an intelligent ordering, and DFF++ does just that. As that ordering of the CNHL for a data packet concerns only internal processing of a device, DFF and DFF++ remain interoperable. DFF++ can furthermore be deployed with exactly the same (or no) unicast routing protocols as DFF.

6.4. Interoperability Considerations

DFF requires a proactive neighbourhood discovery mechanism in order to identify bi-directional neighbours, and additional DFF header information for duplicate detection. Therefore, the DFF extension is limited in scope to the routing domain in which DFF is used.

However, it is possible for data traffic from outside the DFF routing domain traversing the DFF domain. Given the example in figure 6, LOADng Router S sends data packets to LOADng Router D (both are without DFF extension), across the DFF routing domain in the middle. If IPv6 is used, the border LOADng Router *R1*, can encapsulate the data packet using IPv6-in-IPv6 tunnels, according to RFC2473 [24]. The DFF header is also added

with fields defined in section 6.1. The packet can then be forwarded with DFF extension to border LOADng Router *R2*, where the inner IPv6 packet is de-capsulated, and forwarded to *D*.

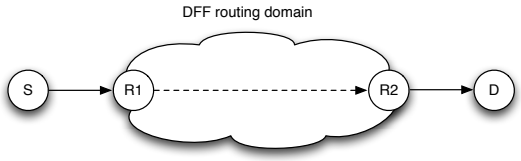


Figure 6: Traffic traverse the DFF routing domain. *S* is the originator, *D* is the destination. *R1* and *R2* are border LOADng Routers

The ability to traverse a DFF routing domain is actually very important, thus for a given deployment DFF can be enabled in particular "lossy" areas of a network, to alleviate packet loss, without interfering with other parts of the network.

6.5. Security Considerations

DFF relies on sequence number of a data packet to detect duplicate packets and loops. A malicious LOADng Router may modify the sequence number to disrupt the packet forwarding: if the sequence number is changed to a number of previously sent packet of the same originator, this packet may be wrongly perceived as a duplicated packet.

Denial-of-Service (DoS) are also possible, by exceeding the memory capacity of a LOADng Router. The *Processed Set* is used to keep the information of all recently forwarded packets. A malicious LOADng Router can generate large number of packets and, thus, exhaust the memory capacity of a LOADng Router. An even worse situation is when a malicious LOADng Router sends packets to a non-existing address in the network, in which case DFF would perform a depth-first search of the entire network – or, until the hop limit has reached zero.

Those attacks can be mitigated by applying link layer security: if the malicious LOADng Routers do not possess valid credentials, other LOADng Routers will not process and forward data from the malicious LOADng Routers.

7. Simulation and Performance Study

7.1. Simulation and Evaluation Settings

The performance of LOADng and different extensions described in previous sections, is evaluated by way of network simulations using NS2 (Network Simulator 2). This section introduces the settings used in network simulations.

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 (data-link layers, distribution, number of routers, etc.).

Simulations are conducted using the TwoRayGround propagation model [25] and the IEEE 802.11b data-link layer. 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.), given that LOADng (and its extensions) are agnostic of the underlying link layer, general behaviours of a protocol can be inferred from these simulation. One possible difference, however, could be fragmentation when using smaller MTUs, such as in 802.15.4, as described in [26].

The general characteristics of the scenarios tested are as follows: n (from 63 to 500) routers are placed randomly in a square field of a size so as to maintain a constant network density. Depending on the scenarios, Constant Bit Rate (CBR) flows with originator-destination pair are generated. Each CBR flow sends one packet of 512 octets every 5 seconds from the originator to the destination.

Following scenarios are considered in the simulations:

- **Point-to-point traffic (P2P) scenarios**: 30 concurrent CBR traffic flows in the network, each from one random originator LOADng Router to another random destination LOADng Router. Ten iterations are run for each scenario, *i.e.,* each data point in the figures represent an average of 300 CBR flows.
- **Multipoint-to-point traffic (MP2P) scenarios**: with a single "root" in the network, acting as sink for all data flows. All the other LOADng Routers generate a CBR traffic flow to the "root", *i.e.,* there are $n-1$ (n is the number of LOADng Router in the network) concurrent CBR traffic flows in the networks. Ten iterations are run for each scenario, *i.e.,* each data point in the figures represent an average of $10(n-1)$ CBR flows.
- **Lossy network scenarios**: the simulations enforce a packet loss probability of 20% to simulate a network with unreliable and lossy links.

Following protocol settings are evaluated:

- **LOADng**: the LOADng core specification based on [13].
- **LOADng SmartRREQ**: the LOADng with smart RREQ extension based on section 3.2.
- **LOADng ExpRing** : the LOADng with smart RREQ extension and expanding ring extension based on section 4.
- **LOADng-CTP**: the LOADng with collection tree extension based on section 5.

- **LOADng DFF** : the LOADng with DFF extension based on section 6.2.
- **LOADng DFF++**: the LOADng with DFF++ extension with destination field prediction based on section 6.3.
- **AODV**: the AODV protocol based on [4].

7.2. Point-to-point Traffic Scenarios

Figure 7 illustrates the average delay, average overhead and number of collisions in P2P scenarios. The data delivery ratio is not depicted, as it was identical and close to 100% in these scenarios.

LOADng with SmartRREQ reduces protocol overhead, by limiting the number of broadcasts in the network. The use of Expanding Ring yields a lower overhead because it can limit the scope of flooding at the begin of the route discovery. In order to compare the performance impact of eliminating gratuitous/intermediate RREPs, figure 7 also includes plots for AODV. In the P2P traffic scenario, (figure 7(a)) LOADng (with or without extensions) systematically provides less control traffic overhead.

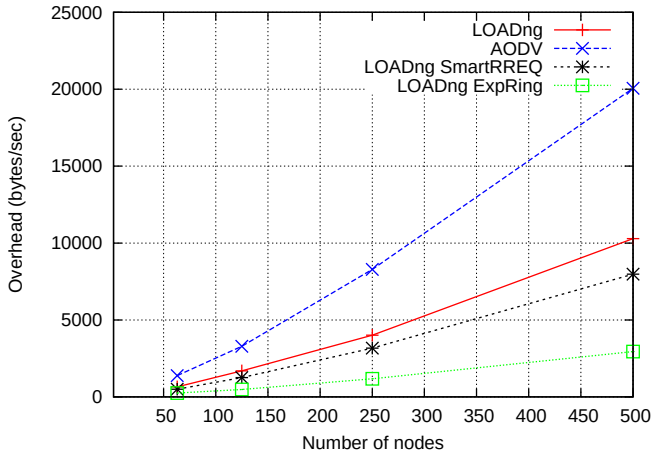
LOADng with SmartRREQ yields shorter delays than LOADng alone – although it is worth noting that the delay grows as the network size grows. This is because – all else equal – a RREQ message always needs to reach the final destination before an RREP is generated. The expanding ring has the longest delay: if the sought destination is out of the expanding ring search scope, the originator has to wait for a timeout before initiating a subsequent RREQ with an increased search scope, which is also the cost of less overhead compared to the others. AODV has the shortest delay thanks to the gratuitous RREP, but it has also security concern as discussed in section 3.4.

Figure 7(c) depicts the number of collisions during the simulations. LOADng SmartRREQ and LOADng ExpRing have less collisions due to the low overhead it generated.

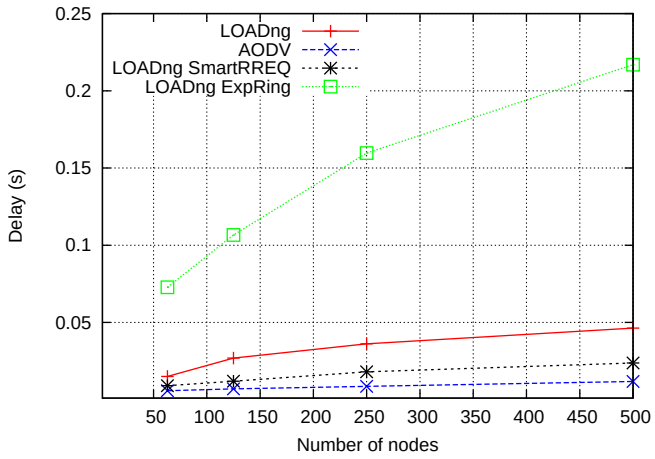
7.3. Multipoint-to-point Traffic Scenarios

In Multipoint-to-point (MP2P) scenarios, the delivery ratio of LOADng drops significantly as the network size increases as shown in figure 8(a). This, as for every Route Discovery, the RREQ is broadcast to the whole network and thus imposing a significant network load (see figure 8(b)), and a higher collision rate in the network (figure 8(c). In this scenario, since every LOADng Router has to maintain a path to the root, when a Route Discovery is initiated by a LOADng Router, its neighbours are likely to still have an active path to the root. The other mechanisms that take benefits of existing routing information can thus have lower overhead and less collisions compared to base LOADng. The LOADng-CTP, which is specially designed for such MP2P scenarios, has the best performance.

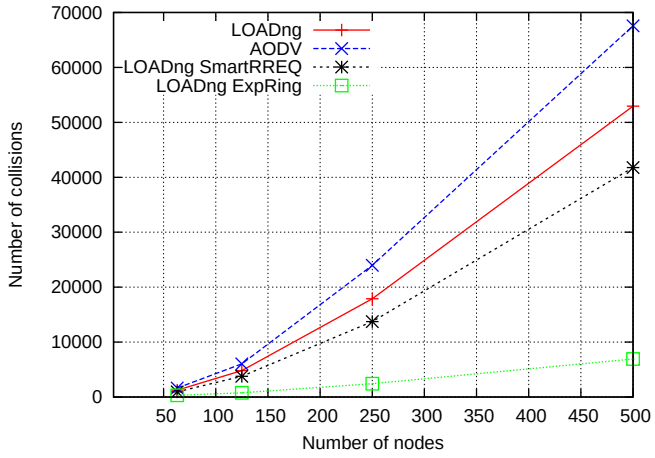
Due to the high overhead and collisions of "blind" RREQ flooding, LOADng also incurs higher delays, as shown in figure 8(d). The LOADng ExpRing has lower delay than



(a) Average overhead



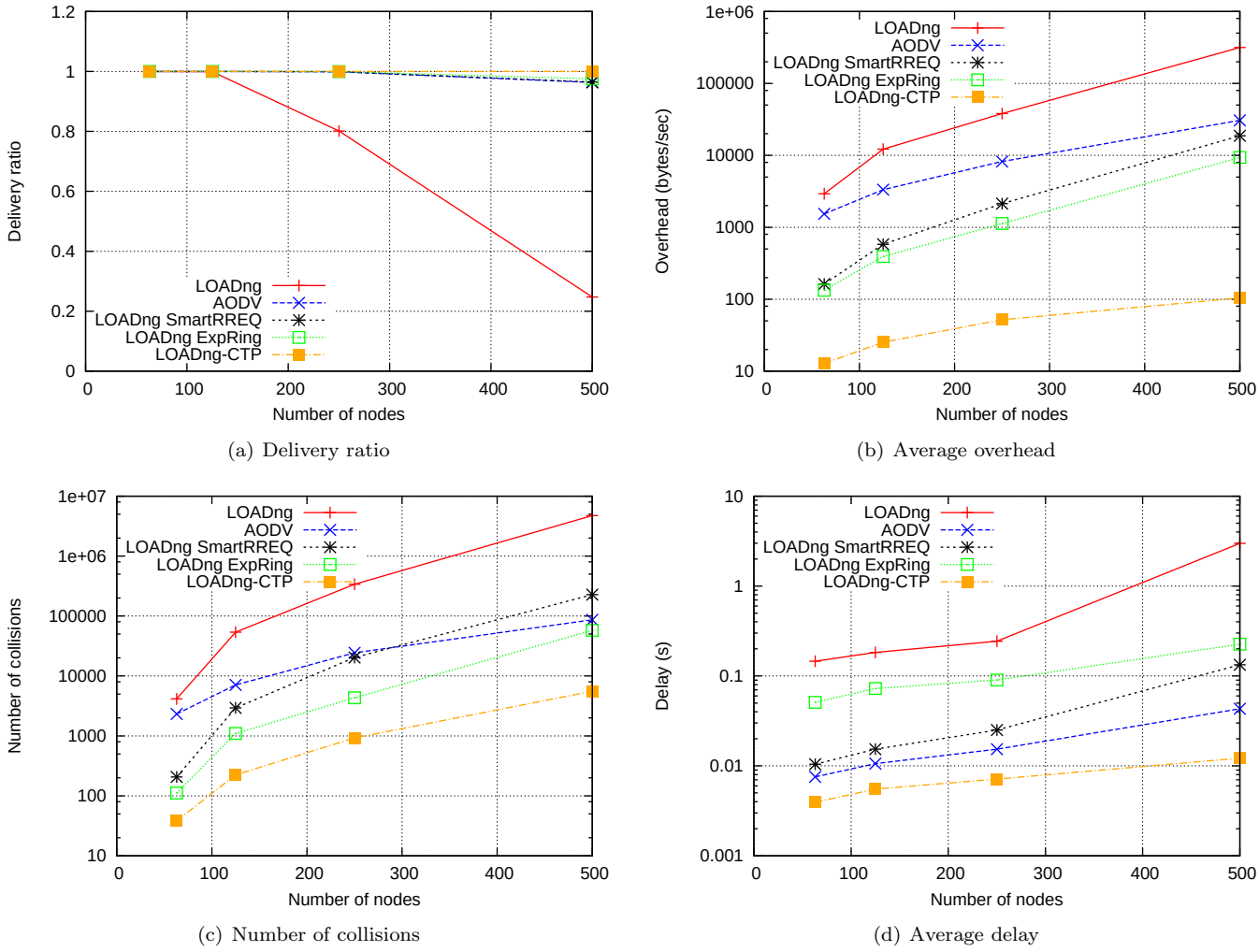
(b) Average delay



(c) Number of collisions

Figure 7: Point-to-point traffic pattern

the LOADng SmartRREQ and LOADng, which is in contrast to the figure 7(b). This is due to the fact that all the routers search for a common destination in the network, in which case the neighbour routers have large chance to have available paths to the destination already. The router



(a) Delivery ratio

(b) Average overhead

(c) Number of collisions

(d) Average delay

Figure 8: Simulation results of multipoint-to-point traffic pattern

discovery delay can thus be greatly reduced without the need of "expanding" the ring. The LOADng-CTP has the lowest delay because the paths to the single root in the network have been set up before they are actually needed. There is no route discovery delay – only the packet forwarding delay is relevant for LOADng-CTP.

For the MP2P traffic scenario (figure 8) the absence of gratuitous/intermediate RREPs in AODV is immediately visible on the performance of LOADng – however, as can be seen, introducing SmartRREQto LOADng alleviates this, and provides, with simpler protocol mechanisms and simpler protocol messages, a performance identical to that of AODV.

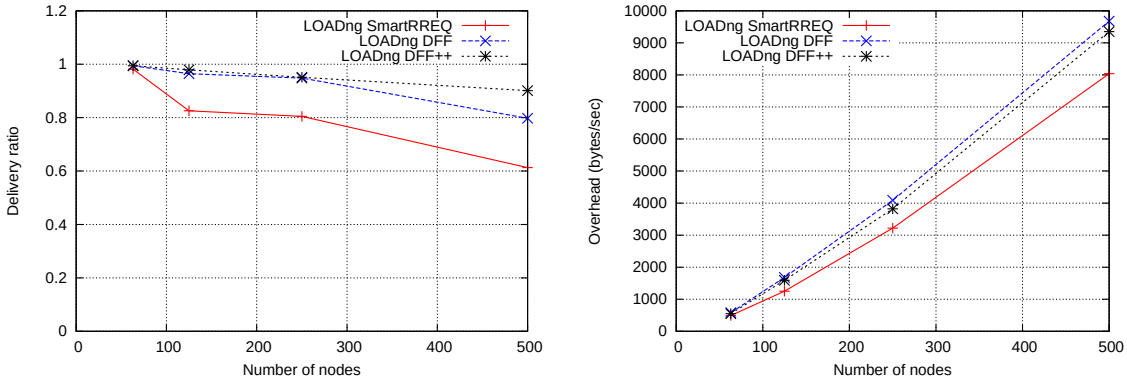
7.4. Lossy Network Scenarios

To evaluate the performance of LOADng with the DFF extension, simulations with point-to-point CBR traffic have been conducted. As DFF is supposed to be particularly beneficial in lossy networks, the simulations enforce a packet loss probability of 20%. The implementations with DFF extension uses [23] for neighbourhood discovery, with HELLO interval set to 1 second – it represents a "very frequent" HELLO message exchange and therefore a good "worst case" example. LOADng with the SmartRREQ extension is chosen as reference protocols.

Figure 9 depicts the performance of LOADng with the SmartRREQ extension, as well as LOADng with the two versions of DFF. DFF, used with LOADng, yields about 20 percentage points improvement of the delivery ratio, as compared to LOADng alone, and DFF++ used with LOADng further improves the data delivery ratio. The improvement comes at the expense of longer delay, and average path length, because more data forwarding is required to perform the depth-first searching. By providing a refined CNHL, DFF++ can reduce the average delay and path length, with no penalty on other performance metrics.

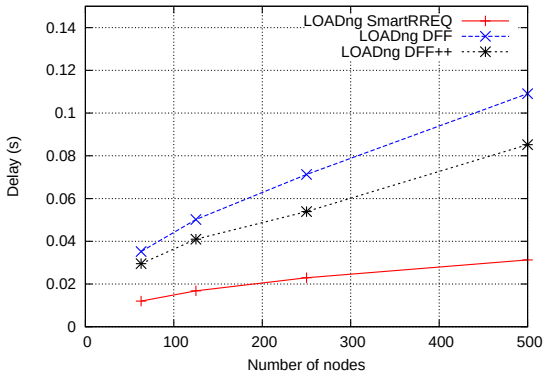
7.5. Discussions

An extension for RREQ message forwarding, SmartRREQ makes use of paths available in the local LOADng

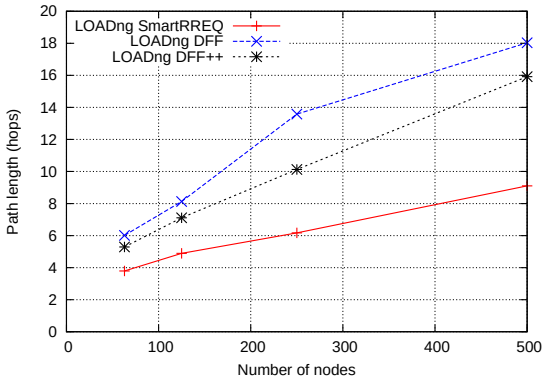


(a) Delivery ratio

(b) Average overhead



(c) Average delay



(d) Average path length

Figure 9: Depth-first forwarding simulations, point-to-point traffic pattern

Router to carry RREQ message over unicast whenever possible – without requiring additional signals nor state. Figure 7 and 8 show that this extension can considerably reduce the routing overhead in common scenarios (*e.g.,* P2P traffic), and is especially efficient if most of the LOADng Routers are sending data packets to a few common destinations in the network (*e.g.,* MP2P traffic). This reduced overhead is obtained without punishing other performance metrics, such as data delivery ratio, average end-to-end delay, etc. – thus, rendering this extension highly recommended especially in data collection scenarios.

The Expanding Ring extension for LOADng limits the range of RREQ broadcasting to reduce the Route Discovery overhead. If a Route Discovery with limited range fails, the searching range is extended, and a new Route Discovery is initiated. The reduction of overhead comes at the expense of increasing Route Discovery delay, especially in point-to-point scenarios, where there is no "single" destination in the network. On the other hand, in MP2P scenarios, the Expanding Ring can actually reduce the Route Discovery delay. Therefore, this extension is advantageous if there are few "common" destinations in the network, and where delays are non-crucial.

The collection tree extension for LOADng is designed to build multipoint-to-point paths with reduced overhead. It inherits the main characteristic of LOADng, and retains LOADng as fall-back in case of heterogenous networks with also unextended LOADng Routers – but, at the same time, enables LOADng Routers in the network to discover bi-directional routes to the root, making it an attractive protocol for data acquisition network deployments.

The DFF is beneficial only in lossy scenarios. In the simulations, the 20+ percentage points gain on the data delivery ratio, makes DFF and DFF++ interesting – albeit, with increased delays as the obvious side-effect, recommended only where data traffic is (at least, somewhat) delay tolerant. Over a low-capacity, but not particularly lossy, channel, DFF will not yield any advantages, but will consume network and other resources for bi-directional neighbour discovery.

8. Securing LOADng

As network devices and networks, emerge in increasingly less controlled environments with less "physical protection" of the infrastructure (*e.g.,* limited access to a building where the network equipment is deployed), security requirements increase: in a wireless network, simply being within radio-range of a router may suffice to launch an attack – and sensor networks are deployed where there's

interesting data to sense, not where it's easy to prevent physical access to the sensor devices.

LOADng, as a reactive routing protocol, is prone to attacks that are discussed in the literature (*e.g.,* [27][28]), including black-hole or spoofing attacks, jamming of wireless channels, etc. However while LOADng faces these same security threats, LOADng is easier to protect because of the design decisions of LOADng, in particular the decision to prohibit intermediate/gratuitous RREPs and thereby to render all LOADng control messages "end-to-end" – and this section proposes a simple framework for securing LOADng.

8.1. Integrity Protection

One of the main objectives in developing LOADng was to maintain a modular architecture with a core, but easily extensible, protocol. The rationale for this decision was that rarely "one-size-fits-all" in the area of constrained networks – and, this is particularly true for security extensions: some networks may not require any level of Layer 3 security, *e.g.,* because physical access is limited, or lower layer protection is sufficient. Other networks require integrity protection with a lightweight cipher suite due to limited precessing power and memory of routers. In some cases, security requirements are tighter and confidentiality as well as strong cryptographic ciphers are required. And constrained networks may exhibit different constraints in terms of MTU sizes – allowing inclusion of smaller or larger digital signatures in control messages.

In addition to modularity, reuse of existing standards was another important design consideration for LOADng. "Reinventing the wheel" by specifying a standalone security extension for LOADng limits reuse of existing code. To this end, the IETF has standardised a security framework for use by protocols, using the message and packet format defined in [20] – such as LOADng. [29] specifies a syntactical representation of security-related information in TLVs for use with [20] addresses, messages, and packets. That specification does not represent a stand-alone protocol, but is intended for use by MANET routing protocols, or security extensions thereof, such as LOADng.

Figure 10 depicts the architecture of a security module for LOADng that provides integrity and non-repudiation for LOADng, using the framework specified in [29].

Incoming RFC 5444 packets are first parsed by the RFC 5444 parser that demultiplexes messages and sends them to the protocol "owning" the message type. As each RFC 5444 packet may contain multiple messages that are used by different protocols on a router, the message type is used to demultiplex and send the message to the appropriate protocol instance. A message intended for LOADng will then be forwarded to the security extension module that verifies the signature contained in a signature TLV inside the message. As the TLV contains additional information, such as the hash function (*e.g.,* SHA-256) and the cryptographic function (*e.g.,* RSA), the module can choose the correct key and verify the integrity protection. If the

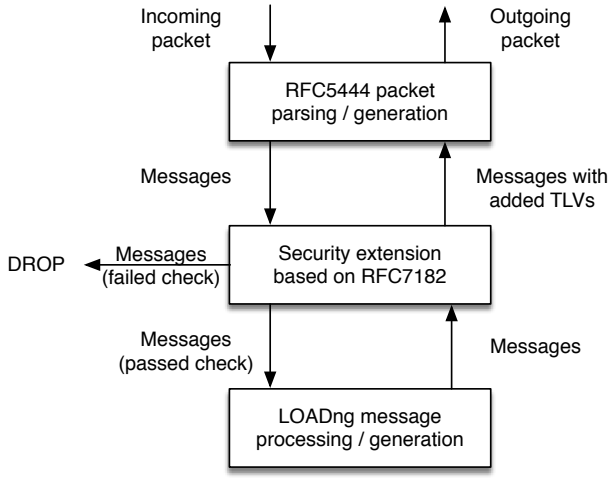


Figure 10: Relationship with RFC5444, RFC7182 and LOADng

message signature is correct, the message is handed over to the core LOADng module, otherwise it is rejected. Similarly, outgoing messages from LOADng are handed over to the security module, which in turn adds the TLV containing the digital signature of the message. Then the message is handed over to the RFC5444 module that multiplexes it into a packet.

During the message signature generation, as well as the verification process, [29] takes special consideration for mutable fields, such as hop count and hop limit. In addition to hop count and limit, the route metric contained in a metric TLV is also updated along the path of a message, and can therefore not be protected by a digital signature. LOADng lists these mutable fields explicitly. While this is a security problem that needs to be addressed in addition to a pure message signature (and is not discussed in this paper), based on the message format of LOADng messages, at least the calculation of a digital signature is easy. This is because the message size does not change as no field is added or removed during the forwarding process of a message through the network (and therefore no other fields, such as message size or TLV block size, need to be recalculated). The metric can simply be replaced by a sequence of zeros before calculating the signature, and is then restored afterwards.

In addition to *message* integrity, *packets* may also be digitally signed. As packets are used hop-by-hop, *i.e.,* are never forwarded, this is useful to authenticate the previous hop along the path of a message. Otherwise, a router not having any credentials may, for example, simply forward a correctly signed RREP message from one adjacent router to another and increase the hop count. As the hop count is excluded from the signature calculation, the message integrity would still be valid. Packet signatures mitigate this problem at the expense of increased overhead on the channel. Note also that it is difficult to detect simple forwarding of a frame without modifying the content, also

known as "wormhole attack".

The security extension described above, using [29] framework, does not encrypt messages, only digitally sign them. The rationale is that the information about the topology itself is in many cases not as confidential as the data traffic between routers. Even if messages were encrypted, an observer may deduce information about the topology by listening to the incoming and outgoing traffic of a router and correlating message pairs that belong together (RREQ and RREP) based on, *e.g.,* timing as well as lower layer header information. That said, LOADng also supports security extensions that provide confidentiality, if such is desired.

9. Conclusion

This paper presents the protocol design and various optional extensions to, the "Lightweight On-Demand Ad Hoc protocol – Next Generation" (LOADng). A reactive routing protocol, LOADng is part of the ITU-T G3-PLC standard, and was designed with core principles of modularity, extensibility, as well as small footprint – and deployment-tuneable efficiency by way of (interoperable) extensions.

On the altar of "simple and compact", LOADng has sacrificed several protocol functions, commonly found in reactive routing protocols: intermediate/gratuitous RREPs being one of these protocol functions. This paper has demonstrated that not only did their removal yield a benefit (overall lower control traffic overhead by way of mechanically smaller control traffic messages), in the scenarios where intermediate/gratuitous RREPs would have been beneficial, a simple protocol extension – SmartRREQ – was able to provide the same benefits without the control traffic overhead penalty.

The simple design of LOADng, where all control messages are "end-to-end", adds another benefit: the ability to adapt an existing security framework for providing integrity and non-repudiation of control messages.

As an example of the principles of modularity and extensibility, this paper also considers functional extensions: providing more than just "point-to-point" routes, a Collection Tree extension is studied, allowing efficiently deploying a LOADng network for data acquisition, with low overhead and high reliability. And for increasing reliability even across lossy networks, this paper discusses the integration of LOADng with DFF – below-layer-3 fast rerouting mechanism, allowing a network to continue to (attempt to) deliver data, even the during the convergence time required for LOADng to react to and recover from a link breakage.

For all extensions and protocol elements discussed in this paper, performance, interoperability, and security considerations are presented, and analysed.

Acknowledgements

The authors would like to gratefully acknowledge the following people for their precious contributions during the specification and development of LOADng and its extensions: A. Colin de Verdiere, A. Bas (Ecole Polytechnique), C. Lavenu (EDF R&D), T. Lys (ERDF), Justin Dean (NRL, USA), A. Niktash (Maxim Integrated Products), Y. Igarashi, and H. Satoh (Hitachi YRL).

References

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

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

[3] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "Optimized Link State Routing Protocol Version 2 (OLSRv2)," RFC 7181 (Proposed Standard), Internet Engineering Task Force, Apr. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7181.txt

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

[5] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," Experimental RFC 4728, February 2007.

[6] R. Ogier, M. Lewis, and F. Templin, "Topology Broadcast based on Reverse Path Forwarding (TBRPF)," Experimental RFC 3684, IETF, February 2004.

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

[8] 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.

[9] 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.

[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] T. Clausen, U. Herberg, and M. Philipp, "A Critical Evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks." Proceedings of the 5th IEEE International Conference on Wireless & Mobile Computing, Networking & Communication (WiMob), October 2011.

[13] 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.

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

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

[16] J. Yi, T. Clausen, and A. C. de Verdiere, "Efficient Data Acquisition in Sensor Networks:Introducing (the) LOADng Collection Tree Protocol." Proceedings of the 8th IEEE International Conference on Wireless Communications, Networking and Mobile Computing., October 2011.

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

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

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

[20] T. Clausen, C. Dearlove, J. Dean, and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format," RFC 5444, IETF, Feburary 2009.

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

[22] S. Cespedes, A. Cardenas, and T. Iwao, "Comparison of Data Forwarding Mechanisms for AMI Networks." Proceedings of 2012 IEEE Innovative Smart Grid Technologies Conference (ISGT), January 2012.

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

[24] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, IETF, December 1998.

[25] The VINT Project, "The ns Manual," http:www.isi.edunsnamnsdocns_doc.pdf, April 2011.

[26] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4944.txt

[27] W. Wang, Y. Lu, and B. K. Bhargava, "On vulnerability and protection of ad hoc on-demand distance vector protocol," in *Telecommunications, 2003. ICT 2003. 10th International Conference on*, vol. 1. IEEE, 2003, pp. 375–382.

[28] P. Ning and K. Sun, "How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols," *Ad Hoc Networks*, vol. 3, no. 6, pp. 795–819, 2005.

[29] U. Herberg, T. H. Clausen, and C. Dearlove, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)," RFC 7182, Apr. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7182.txt