CoopStor: a cooperative reliable and efficient data collection protocol in fault and delay tolerant wireless networks

Georgios Papadopoulos, Alex Mavromatis, Antoine Gallais, Fabrice Theoleyre

Noname manuscript No.
(will be inserted by the editor)

CoopStor: A Cooperative Reliable and Efficient Data Collection Protocol in Fault and Delay Tolerant Wireless Networks

Georgios Z. Papadopoulos · Alex Mavromatis · Antoine Gallais · Fabrice Theoleyre

Received: September 28, 2020

Abstract Internet of Things (IoT) consist in the deployment of constrained and battery-powered devices with a radio interface. Most industrial applications require to respect strict reliability and delay constraints. Lossy links imply mechanisms to retransmit the packets to improve the reliability. However, transient faults (e.g. obstacles, interference) may also impact the reliability, and require to change the routes. In this article, we present CoopStor (Cooperative Storing mode), that detects the faults and triggers action before a packet is dropped. In particular, the packets are cooperatively stored in the network, until the routing protocol re-converges. The congested zone gradually increases to serve as a local storage facility. CoopStor relies on a selection of relay and leaf nodes to reduce the energy consumption while providing a low end-to-end delay in steady state. Our performance evaluation campaign highlights a reduction of packet drops and an improved energy-efficient data collection procedure.

1 Introduction

More and more smart objects are now connected to the Internet, to create the so-called Internet of Things (IoT). Typically, smart buildings rely heavily on collecting a large volume of data in real-time [30]. The sensors send their measurements to a cloud infrastructure, through a border router.

Most IoT networks exhibit a convergecast traffic pattern, where all the packets are forwarded to a border router. If the routing protocol balances the load around

G.Z. Papadopoulos
IMT Atlantique, IRISA, France, E-mail: georgios.papadopoulos@imt-atlantique.fr

A. Mavromatis
Faculty of Engineering, University of Bristol, UK, E-mail: a.mavromatis@bristol.ac.uk

A. Gallais
LAMIH Laboratory, CNRS / Polytechnic University Hauts-de-France, France, E-mail: antoine.gallais@uphf.fr

F. Theoleyre
ICube Laboratory, CNRS / University of Strasbourg, Pole API, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France, E-mail: theoleyre@unistra.fr

the border router, many nodes have to forward packets, resulting in a high contention [4]. It is rather more efficient for the link layer to select only a small set of nodes to forward most of the traffic [33]. This way, we reduce the number of collisions and, thus, increase of network reliability and energy efficiency. Hence, energy consumption is balanced by changing the role (relay / passive) of the nodes regularly through the routing protocol.

However, the routing protocol has also to construct efficient routes on top of a lossy topology. In particular, external interferences and collisions require to retransmit the packets for some links. Epidemic routing proposes to flood the network to reduce the delivery delay and to optimize the reliability [7]. However, these schemes generate a large volume of packets and, thus, collisions, and require a large buffer size to be efficient. Traditional unicast routing protocols are often preferred in such environments. Multipath routing tries to exploit multiple paths in parallel to provide high-reliability [5], [16], [21]. However, replicating the packets consume both bandwidth and energy.

Last but not least, many critical applications need to respect strict end-to-end constraints. However, the end-to-end delay is proportional to the wake-up period of each relaying node: the transmitter has to wait that the receiver is awake before transmitting its packet. Thus, maintaining a low end-to-end delay requires to increase the energy consumption since it needs a higher duty-cycle ratio (i.e. smaller wake-up period).

In this article, we propose CoopStor, a forwarding scheme that provides high reliability even in presence of faults. Instead of provisioning a priori resources, we propose a mechanism to store cooperatively the data packets when a fault occurs. A congested area grows gradually, where all nodes cooperate to store the packets until the network re-converges. Each decision is localized, based on the state of the neighbors, and the buffer occupancy of the node.

The contributions presented in this paper are as follows:

1. We compute the probability of potential buffer overflow occurrence, which depends on the traffic rate, the number of transmission opportunities toward the next hop, the wake-up interval in the link layer, and the buffer occupancy;
2. CoopStor relies on differentiated medium access schemes to make the network energy efficient. Only a subset of nodes forward the packets to reduce the energy consumption while still limiting the end-to-end delay. These relay nodes implement a high sampling rate (i.e. they wake up frequently to receive possibly packets). Inversely, the leaf nodes may implement an ultra low duty cycle approach to reduce their energy consumption;
3. CoopStor uses all the neighbors when the probability of a buffer overflow is not null. We cooperatively store the packets in the different neighbors, to let the network re-converge. This way, we limit the number of packet drops, and we improve the reliability;
4. CoopStor reduces the traffic rate gradually, by forcing the nodes to hold their packets back once their next hop is congested. This congested area gradually increases, and is used as a cooperative storage area;
5. We thoroughly evaluated our solution with COOJA (an emulator for Contiki OS) to highlight the relevance of CoopStor to deal with transient faults.

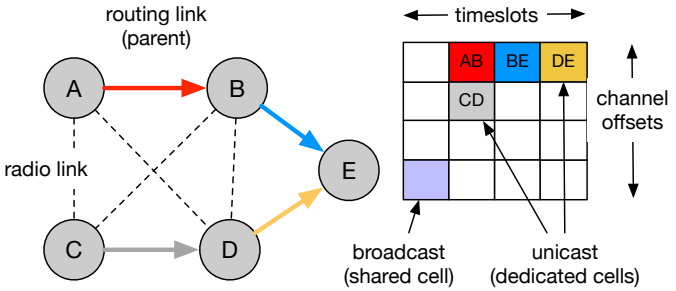


Fig. 1: TSCH schedule for a 5 nodes topology.

2 Background

We focus in this paper on synchronized wireless networks, where the devices implement a low duty cycle to save energy. Thus, we will describe first IEEE 802.15.4-TSCH (Time slotted Channel Hopping) [12], a slow channel hopping MAC protocol designed to provide high reliability and robustness against external interference. Then, we will detail RPL (IPv6 Routing Protocol for Low Power and Lossy Networks) [36].

2.1 IEEE 802.15.4-TSCH

The standard relies on a strict schedule of the transmissions [12]. The *slotframe* contains a fixed number of timeslots, during which at most one frame and its acknowledgment are transmitted. Each timeslot is labelled with an Absolute Sequence Number (ASN) which counts the number of timeslots since the Personal Area Network (PAN) coordinator started. Based on the schedule, a node can decide its role (transmitter/receiver/sleeping mode) at the beginning of each timeslot.

IEEE 802.15.4-2015 TSCH implements a channel hopping approach to combat external interference and signal fading and, thus, to achieve high reliability. At the beginning of a timeslot, a device verifies in the schedule if it has to stay awake. If the cell is allocated to the device, the physical frequency to use for transmission/reception is derived from the ASN of the timeslot and the channel offset assigned to this cell. Nodes regularly broadcast Enhanced Beacons (EB) to disseminate control information for unattached devices. The EBs may also be used as a mean for nodes already part of the network to re-synchronize their clocks. To this aim, an EB frame may contain synchronization of ASN and Join Metric, channel hopping sequence identifier and TSCH slotframe.

The standard supports both distributed and centralized scheduling algorithms. Scheduling for slow channel hopping industrial networks has received much attention in the past [32]. 6TiSCH-minimal [38] proposes to use some common (shared) cells. Since all the nodes wake-up synchronously, a node is sure that the receiver is awake. Besides, such shared cell is also useful to send a broadcast packets: every neighbor will be able to decode the transmission. The shared cells should be uniformly distributed in the slotframe helps to reduce the number of collisions [34].

2.2 RPL

The Routing Protocol for Low Power and Lossy Networks (RPL) [36] is a distance vector routing protocol, initially designed for topologies of thousands of nodes. RPL focuses mostly on the convergecast traffic pattern, where all the packets are generated toward the sink (called border router in RPL).

RPL is based on a *Destination Oriented Directed Acyclic Graph (DODAG)*, rooted at the border router. Each node computes its *rank*, denoting its virtual distance from the border router. By selecting as parent only a neighbor with a strictly lower rank, RPL constructs a loop-free routing structure. More precisely, RPL defines an *Objective Function* (OF) to compute the rank, and a node selects as parent its neighbor which gives it the lowest rank. Basically, an objective function takes as arguments the rank of a neighbor and its link and node metrics.

The DODAG Information Objects (DIO) are control packets that contain the rank of the transmitter, as well as the metrics required to compute the rank for neighbors. Typically, we may use the popular Expected Transmission Count (ETX) [8] metric as link quality metric. It counts the average number of (re)transmissions required before receiving an acknowledgement (ACK). The objective function may sum the ETX value of all the links of the path toward the border router.

3 Related Work

A Wireless Sensor Network (WSN) should be fault-tolerant, able to handle e.g., a node crash, or a sudden change of the link quality. In these conditions, several approaches exist in the literature to limit the impact on the reliability.

3.1 Data Redundancy

Data redundancy mechanisms increase the number of messages the transmitter sends to the next hop. With the HARQ (Hybrid Automatic Repeat reQuest) strategy, packets which cannot be decoded are stored in the receiver to be recombined later with the next copies [22]. The strategy to adopt to reduce both the packet losses and the delay depends on the activation of acknowledgments.

Ruan *et al.* [28] propose to deal with byzantine faults. They consider the wireless infrastructure as a distributed storage system, storing the measurements, and replying to the external queries. They use erasure codes to recover from a network disconnection. However, the technique is computational intensive for our scenario, and does not consider real-time transmissions to the sink.

Network coding can also help to introduce redundancy, and thus reliability. Wu *et al.* [37] propose a per-link network coding strategy, adapted to the link quality. When the packet loss rate exceeds a threshold, redundant and independent packets are generated. Static coding vectors are used to simplify the decoding. Swamy *et al.* [31] also uses network coding to improve the link reliability. However, these approaches can only handle link quality variations, i.e. they cannot react efficiently to a node failure. Alternatively, the network may adopt an end-to-end network coding strategy [11].

In Time Division Multiple Access (TDMA) schemes, the transmissions are strictly scheduled to avoid collisions. The problem consists in reinforcing the schedule to allocate more transmissions opportunities to the different flows while still respecting the end-to-end delay constraints [9]. Redundancy has a cost in bandwidth and energy: more packets have to be transmitted, whatever the conditions [23]. In other words, they are tailored for the *worst-case*. We propose rather a reactive approach: more resources are consumed *only* when a *fault* is detected.

3.2 Congestion Detection and Control

Low power and lossy networks support a combination of event-triggered, continuous and query based traffic. Congestion occurs when the number of packets exceeds the number of transmission opportunities, and creates collisions and packet drops. Jan *et al.* [15] present a comprehensive survey of the existing congestion detection and control schemes. Typically, congestion can be detected by analyzing the buffer occupancy. If the number of packets exceeds a threshold value, it is highly probable that some packets would be dropped either because of congestion, or because a route failure.

Kafi *et al.* [18] propose a congestion control based scheduling algorithm. By appropriately scheduling the transmission, intra and inter-path interference can be mitigated. But such scheme does only operate with TDMA schemes, which are expensive with very low, or unpredictable traffic conditions. Pappas *et al.* propose to selectively drop the oldest packets in the queue [27], considering old information as less relevant. However, this approach assumes implicitly periodic redundant transmissions, which is not always the case.

3.3 Intermittently Delay Tolerant Networks

Epidemic routing such as *Spray and Wait* [29] played a key role for routing in delay-tolerant mobile ad hoc networks. Several copies circulate in the network to guarantee a minimum reliability in intermittently connected topologies. Wang *et al.* explore the optimal number of replicas to generate and how to schedule them to increase the reliability [35]. They also select appropriately the packets to drop when a buffer overflow occurs. However, the authors focus on a mobile topology of devices, and use their contact-duration distribution to derive the optimal strategy. These epidemic protocols are particularly efficient, but only in extreme conditions, where the topology is intermittently connected and the links very lossy.

3.4 Content Centric Networks

Named Data Networking (NDN) tries to change the Internet paradigm to handle directly data [39]. A producer generates data packets, that are cached in the network. Symmetrically, a consumer generates an interest, routed toward one copy of the chunk. Extensive caching strategies improve the scalability since some data are very popular, and do not need to be forwarded independently for all the consumers.

It has been recently extended to handle also IoT applications [6]. Indeed, IoT applications rely on a large collection of producers which generate measurements, consumed by controllers and actuators. To support data streams, a device may subscribe to an interest, so that the producers automatically push the novel measurements [25] (periodically, or threshold based). However, a NDN cache needs a large amount of memory for small embedded devices.

3.5 Routing Mechanisms

A first approach consists in balancing the load among the different nodes. Ahmed *et al.* [3] propose to construct multiple routes, and to distribute the load on the less congested ones. Iova *et al.* proposed to modify RPL to support multi-parent forwarding [13]. While it balances well the load (and the energy consumption), congestion may still occur locally when a route change occurs. Kim *et al.* [19] provide a load-balancing technique for RPL. Each node selects its parent considering their queue utilisation, so that the load is spread among the different nodes, to reduce the congestion. Furthermore, in [17], the authors defined the Traffic Aware Objective Function (TAOF), which balances the traffic load at each node locally to provide node lifetime maximization and high network reliability.

While these approaches balance well the load, they cannot handle sudden faults. In particular, global repairs would require a global network reconfiguration, wasting time and energy. With RPL, a node may use siblings or switch its parent for a fast recovery [20].

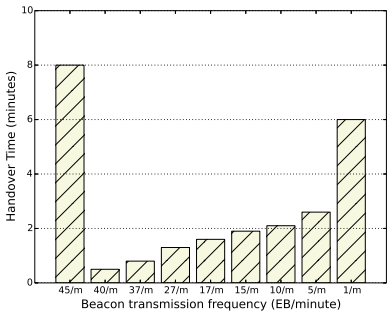
4 CoopStor: Cooperative Storage before the Routing Convergence

In this article, we consider that a fault may occur unpredictably (e.g. mobility of nodes, failures of devices, failures of links). Routing tables may become inconsistent, and the packets may be accumulated in the buffers, leading possibly to packet drops. Using multipath techniques, or replicating the packets may help to alleviate the problem, but is expensive: more packets have to be transmitted.

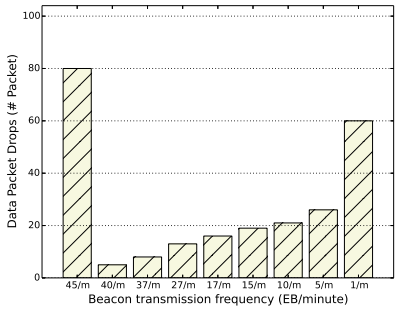
We aim here rather to propose an adaptive scheme: energy should be consumed **only** if a fault is detected. However, we must be particularly reactive since some nodes may quickly drop data packets. Thus, we propose to use all nodes to cooperatively store the packets in order to reduce the potential losses. Hereafter, we present our architecture, and we present in section 5 how each mechanism is used during and after the routing convergence.

4.1 Motivation: duration of the convergence

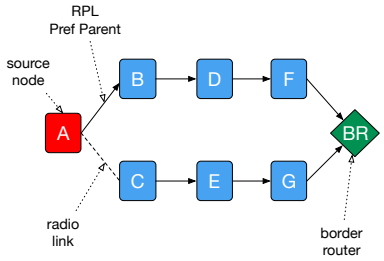
Let us first simulate a simple topology as depicted in Figure 2c, using Cooja and 6TiSCH minimal [38]. We use the parameters detailed in Table 2, as in the performance evaluation. We let the network converge and we turn off the node B which is the preferred parent of A. We denote by handover the process for the node A to: i) detect its parent is not reachable, ii) find another parent, iii) reinitiate



(a) Handover Time investigation.



(b) Data packet drops and the EB frequency.



(c) Topology Tested in Cooja.

Fig. 2: Impact of the EB period on the disconnection time.

possibly the synchronization if it has been desynchronized, iv) change its rank and configure its default route.

Typically, Enhanced Beacons (EB) are required to allow the network resynchronization and reconfiguration. In particular, a smaller EB transmission frequency period helps a device to find faster another time reference, but it also means that the energy consumption is higher. As it can be observed in Fig. 2a, a large number of EBs means that a device has to wait on average for shorter time before receiving a valid EB. However, very high EB rates (≥ 45 per minute) mean a very large number of collisions, which jeopardize the convergence. Inversely, less EB transmissions reduce the collisions, but increase the convergence time. Typically, the node A still needs 6 minutes to have a valid route with the default EB value (every 60 seconds). Meanwhile, several dozens of packets can be dropped (see Fig. 2b): the node keeps on generating packets, and a buffer overflow occurs before having a valid parent. This problem may be even worse if the node has packets to forward: its buffer will fill up faster.

4.2 A two-layer topology to store packet when a fault occurs

In time-synchronized networks such as 6TiSCH minimal [38], each device has a fixed number of transmission opportunities, equally interspaced, that defines its duty cycle ratio. It is straightforward that increasing the duty cycle ratio means

a smaller end-to-end delay: a node has shorter wait times before the next transmission opportunity. However, it also increases the energy consumption.

We propose consequently in CoopStor to exploit two types of sensor nodes:

1. **relay nodes** forward the traffic of their subtree (i.e., subDODAG in RPL terminology). They are similar to the Full Function Devices (FFD) of ZigBee [1].
2. **leaf nodes** are nodes which are not required for the routing task. Here, we consider leaf nodes as the routing leaves only, similar to the Reduced Function Device (RFD) of ZigBee [1]. With a convergecast traffic pattern, the leaf nodes only wake up to transmit the sensed data to one of their parents, a relay node.

The relay nodes represent the routing backbone, i.e., the RPL DODAG, and forward all the packets toward the border router (i.e., the sink station). All the leaf nodes implement an ultra low duty cycle mode since they can only wake-up when they have a date packet to transmit, without any impact on the end-to-end delay. However, we keep on maintaining a low end-to-end delay since each node uses only the active nodes as next hops.

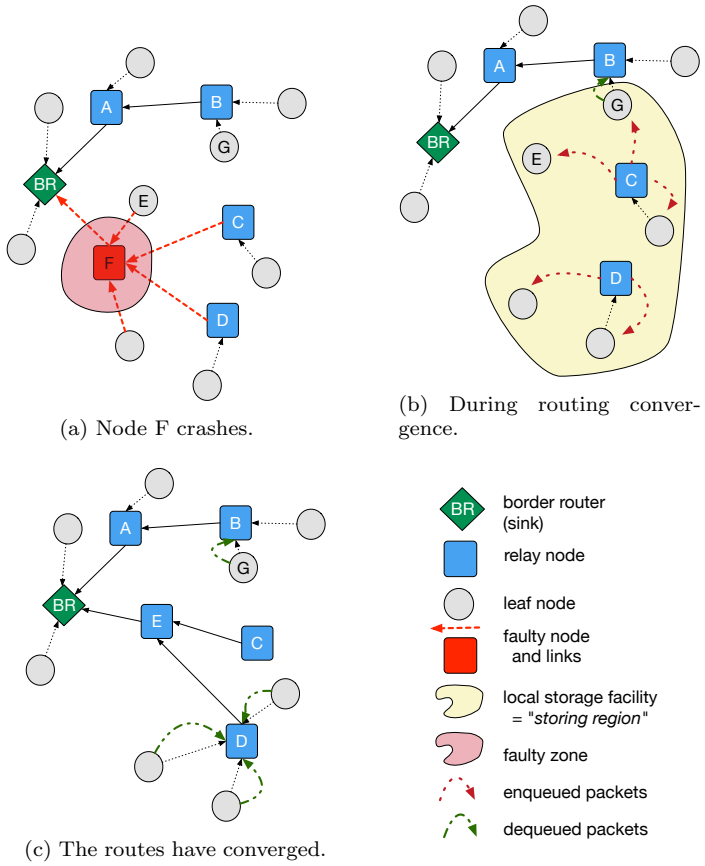
Let us now consider a node crashes (for instance the node F in Fig. 3). The relay nodes may quickly run out of memory since they need to store many packets before having a valid route. To avoid dropping packets (e.g., on node D), we propose to employ the neighboring leaf nodes as a temporary storage facility. When a fault or a congestion is detected, all the neighboring leaf nodes have to stay awake to receive packets. The packets are stored until the leaf nodes have a valid, non-congested parent. In Figure 3b, node G has a valid parent (B) from the beginning and can forward the packets immediately. The other leaf nodes have to wait for the routing reconvergence (Fig 3c)).

4.3 Using leaf nodes when congestion occurs

We assume here that the wireless network is based on 6TiSCH-min [38], that uses shared cells uniquely. Regarding the leaf nodes, they only wake up to transmit their own data packets. Upon packet generation, they wake up and transmit their data packet during the next available shared cell toward their preferred parent according to the RPL protocol. As leaf nodes send their own data only, their queues are empty most of the time and they do not suffer from congestion.

The relay nodes must know when their neighboring leaf nodes will wake-up. Leaf nodes still send in broadcast EBs, every T_{EB} seconds. All their neighbors, including the relay nodes, receive these periodical EBs. Thus, a relay node can predict the instant of EB transmission since the EB period is fixed in the network. Therefore, a relay node has to store the last EB reception time for each leaf neighbor, to know when each of its neighbors will wake-up.

To serve as a local storage facility when a congestion occurs, we propose that a leaf node remains awake the N_{scan} shared cells following its EB transmission. Thus, idle listening is strictly limited: leaf nodes have only to listen to N_{scan} shared cells every T_{EB} seconds. When it intercepts a message from a neighboring relay node with a `storing mode` flag, the leaf node has not anymore the right to sleep during the next shared cells: it must stay awake until all the neighboring relay nodes have recovered. It triggers the `storing mode`.



(a) Node F crashes.

(b) During routing convergence.

(c) The routes have converged.

Fig. 3: Using the leaf neighbors as a temporary storage facility to cope with transient faults.

Let us consider the N_{sh} shared cells that are uniformly distributed in the slotframe (with a slotframe length SF_{length}) [34]. Since a packet may be generated at any time, the source has to wait on average for $\frac{SF_{length}}{2*N_{sh}}$ timeslots before being able to transmit its packet. Then, the packet has to be forwarded in the next shared cell, i.e., $\frac{SF_{length}}{2*N_{sh}}$ timeslots later. Similarly, if the packet has to be retransmitted, a random number of shared cells has to be skipped before the retransmission. In conclusion, the end-to-end delay is directly proportional to the number of shared cells. A large number of shared cells decreases the end-to end delay but increases the energy consumption for relay nodes with idle listening.

5 Engaging and recovering from the storing mode

We now define when the `storing mode` has to be triggered. To avoid packet drops, a node engages the `storing mode` upon *any* fault detection (e.g., no next hop,

buffer overflow). In particular, we will compute the probability that a buffer overflow occurs, according to the schedule of the neighbors and the buffer occupancy. Finally, we will describe how the network recovers when the routing protocol has converged.

5.1 Storage Phase

A node decides to activate the `storing mode` in CoopStor when at least one of the following conditions holds:

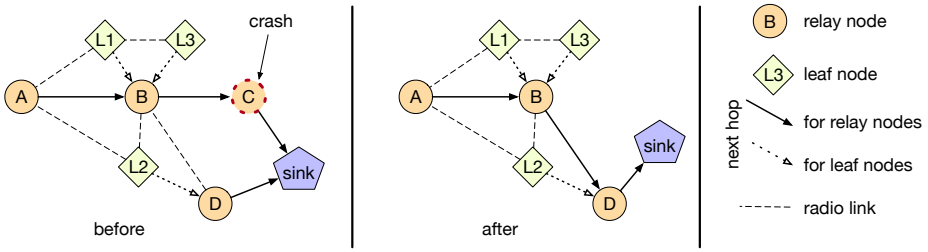
1. a routing inconsistency. Neighbor Unreachability Detection (NUD) helps to detect that the RPL parent is not anymore up. In the same way, RPL detects a loop when a packet is received in the upward direction from a neighbor with a lower rank;
2. the probability of a buffer overflow before the next wake-up of a neighboring leaf node exceeds a threshold. Each time a relay node receives an EB from a leaf node, it has to compute the probability of a buffer overflow. We will detail in section 5.2 how to compute this probability. If this probably exceeds a threshold value, a relay node engages the `storing mode` step, and uses neighboring leaf nodes to store temporarily the packets, and to avoid a buffer overflow;
3. a neighboring relay node is in `storing mode`. Typically, it reflects a routing inconsistency, and all the nodes must stay awake until the network recovers.

Let us consider the Figure 4. The node C has crashed, and the relay node B will be probably saturated before being able to find a correct next hop. Figure 4b illustrates the exchanges of packets at the MAC layer. For the sake of simplicity, we only represent the active period of the slotframe (i.e., shared cells). The relay nodes have to wake-up for every shared cell, and consume more energy.

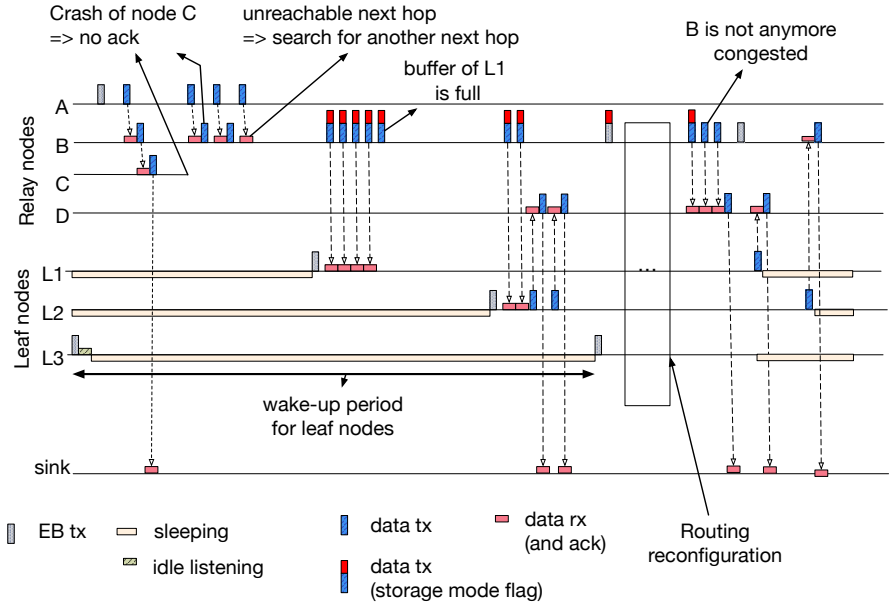
The node B detects its next hop (node C) has become unreachable. However, due to its buffer occupancy and its limited storage resources (i.e., queue size), the node B can only store a limited number of packets. Thus, it has to find local neighbors to help it *storing* temporarily the packets. $L1$ is the first leaf node which sends an EB, and it remains awake in the next shared cell. The relay node B starts emptying its buffer by transmitting its packets to $L1$ with the `storing mode` flag to force $L1$ to remain awake. B send all its packets (EB and `data` packets) with the `storing mode` flag. Step by step, all the neighboring leaf nodes are notified and remain awake during all the shared cells. Besides, a leaf node stops forwarding its packet when it is notified that its parent has activated the `storing mode`.

To avoid buffer overflows, the relay nodes adopt a *hot potato* strategy. They send their packets to the first leaf neighbor which sends an EB. Since this leaf node has then to remain awake, the relay node can reduce its buffer occupancy relatively fast. At certain point, a leaf node may also become congested, and it will stop acknowledging packets (label "*full*" in Figure 4b). Thus, the relay node has to search for another leaf node, and waits for the next EB.

The relay node may find a neighboring leaf node which has still a valid, non-congested parent. For instance, $L2$ has a valid parent (D) in Figure 4a: it is not impacted by the crash of C. However, it will be solicited by the relay node C and will start buffering its packets. Since the relay node D is still valid, and is its parent, $L2$ can safely keep on forwarding its data packets to its parents. It is worth



(a) Routing Topology before and after a node crashed



(b) MAC exchanges with CoopStor

Fig. 4: Illustration of CoopStor with a faulty relay node

noting that the relay node C can empty its buffer thanks to $L2$, but remains in `storing mode`, until it has a correct parent which is a relay node (node D in our example).

Relay nodes forward most of the packets. Thus, they should stop sending their packets when their parent (also a relay node) is congested. Similarly to the leaf nodes, the relay nodes enter in `storing mode` as soon as they receive any packet from their parent. Thus, the `storing mode` is activated hop-by-hop, propagated in the subtree of the congested node. More precisely, we block all the subtree rooted at the relay node that became congested. However, the routing protocol will reconfigure the routes. Thus, as soon as a relay node has non congested parent, it can safely starts re-forwarding its packets toward the sink.

$\mathcal{N}eigh_p(k)$:	Set of leaf neighbors of the node k
$T_{up}(i)$:	Wake-up period of the node i
$rx_{EB}(j)$:	Last time since the reception of an `EB` from the leaf node j
Δ_{clock}:	Average clock drift (typically 30ppm)
$per(i,j)$:	packet error rate for the link (i,j)
$Queue_{max}$:	Maximum number of packets which can be stored in the queue
$Queue$:	Number of packets currently in the queue
λ_{in}:	Number of packets received per second (= 0 for leaf nodes)

Table 1: Notation

5.2 Buffer Overflow Probability

We propose to monitor the queue occupancy and to compute the probability a buffer overflow occurs before the next leaf node wakes-up. We consider also the link reliability toward each leaf neighbor to upper bound the probability of a packet drop. We first analyse here the conditions that each relay node must respect to avoid a buffer overflow.

Each leaf node sends periodically an EB, and then stays awake after its EB to receive possibly sollicitations. We denote by T_{up} the wake-up period of each node, i.e. the period at which it wakes up to receive packets from its neighbors (cf. Tab. 1 for our notation). To save energy, the wake-up period of leaf nodes is much larger than for relay nodes.

By memorizing the schedule of its leaf neighbors, each relay node is able to predict when each of them will wake up to store its packets if a congestion occurs. If a relay node N_r monitors the instant at which it received the last EB from each of its neighbors, it can compute the expected wake-up time for each of it neighboring leaf node. A leaf node N_l sends periodically its EB every $T_{up}(N_l)$. Thus, the relay node N_r can precisely estimate the next wake-up time of each neighboring relay node:

$$T_{wake-up}(N_l, t) = rx_{EB}(N_l) + k \times T_{up}(N_p) \tag{1}$$

with $rx_{EB}()$ being the time of the last EB received from N_l and $T_{up}()$ its wake-up period.

One can note that we consider the next EB transmission, even if the previous EBs have not been received correctly (because of e.g. collisions). More precisely, the integer k is derived to respect the following condition:

$$rx_{EB} + (k-1) \times T_{up}(N_l) \leq t \leq rx_{EB} + k \times T_{up}(N_l) \tag{2}$$

with t being the current time clock.

We must also take into account the clock drift since the last reception of the `EB` from the neighbor N_l. Indeed, the maximum time difference between the two clocks of N_l and N_p is:

$$T_{diff}(N_l) = \big(T_{wake-up}(N_l, t) - rx_{EB}(N_l)\big) \times \Delta_{clock} \tag{3}$$

Finally, the maximum time to wait before N_l is able to receive the data packets, is finally:

$$T_{wait}(N_l) = T_{wake-up}(N_l) + T_{diff}(N_l) \quad (4)$$

Let us now consider the packet error rate (PER) of the radio links: some data packets may be lost because the radio link is unreliable. In this case, the leaf neighbor would go back to the sleeping state without buffering the expected packets.

We aim here at upper bounding the probability $P[buffer\ overflow]$ that the node undergoes a buffer overflow. Typically, $P[buffer\ overflow]$ should not exceed $\Delta_{thr.}$ (e.g. threshold reached in maximum 1% of the time). Thus, we have to consider the maximum waiting time which will be guaranteed in $(1-P[buffer\ overflow])$ of the cases.

We have to determine which neighbor will be the first to acknowledge the data packet transmissions from the congested relay node. The probability that a given neighbor will be the first leaf node to acknowledge the anycast packet is the probability that all the previous neighbors (with a lower waking-up time) failed, and that the transmission of the considered neighbor is a success. Finally, this neighbor $N_{first} \in \mathcal{N}eigh_p(N_r)$ must wake up sufficiently early such that:

$$P[buffer\ overflow] \leq \Delta_{thr.} \quad (5)$$

with

$$\Delta_{thr.} \leq (1 - PER(W)) \times \prod_{k \in \mathcal{N}eigh_p(N_a)}^{T_{wait}(k,t) \leq T_{wait}(N_{first},t)} (PER(k)) \quad (6)$$

If a temporary disconnection occurs next, a relay node will consequently undergo a buffer overflow if it receives too many packets until the waking-up of this leaf neighbor:

$$(Queue_{max} - Queue) \leq \lambda_{in} \times (T_{wait}(N_{first}, t) - t) \quad (7)$$

with λ_{in} the average rate of packet reception.

5.3 Recovery Phase

When none of the conditions in section 5.1 holds, a node can consider safely the network has converged and the steady state has been reached. A node engages the recovery phase only when its parent announces it has recovered, i.e., the `storing mode` flag is inactivated in the overheard packets. This way, the relay nodes close to the recovered region will first empty their buffers. Step by step, the storing region will be reduced, and the network will finally entirely turn into steady state.

In Figure 4, the relay node B has to change its parent (from C to D). Before the change, the relay node uses its neighboring leaf nodes. After it has a correct parent, it starts sending its data packets to D, and can remove the `storing mode` flag from its packets as soon as its buffer occupancy is sufficiently low to respect the condition of eq. 7. Similarly, the leaf node $L1$ observes that its parent B is not congested anymore (the `storing mode` flag is off from its packets). Thus, it can safely starts emptying its buffer by sending its packets to B. When its buffer is empty, $L1$ re-engages its low power mode, and wakes up only every T_{EB} seconds.

Since the buffer of a recovering node is probably full, each node will transmit several packets to its parent. However, its parent is not anymore in `storing mode` and can receive multiple packets without suffering from a buffer overflow.

5.4 Overhead

Our protocol relies on Beacon and routing packets, that already exist for most of the MAC or routing protocols, and are defined either in IEEE 802.15.4-2015 [12] or RFC 6550 [36] standards. In particular, `Enhanced Beacons` packets keep on being generated according to the same period. Each node computes the expected wake-up time of its leaf neighbors by taking into account the instant of reception of the last EB, and the EB period in the network.

The overhead generated by our solution consists of:

1. a `storage` flag in the data packets, indicating the emitter is in `storing mode`. In particular, no packet is required to maintain the nodes awake in the *storing region*: a node cannot sleep until all its neighbors indicated they recovered, either in `EB` or `data` packets;
2. two additional transmissions to store a packet in a neighboring leaf node (and retrieve it). However, the corresponding data packet would have been dropped without our solution.

Our scheme may also be adapted for different MAC layers. With Low Power listening (LPL) [24], leaf nodes would have a very large preamble wake-up period, while relay nodes would wake up more frequently to reduce the forwarding delay.

6 Performance Evaluation

We assess here the performance of CoopStor through a set of simulations, to evaluate its efficiency for using all the nodes as a temporary storage infrastructure.

6.1 Emulation Setup and Parameters

For the setup of our experiments we utilize the RIME communication stack (`http://www.contiki-os.org/`), with a gradient based routing protocol (similar to RPL). We construct a tree rooted at the sink, and use the number of hops as routing metric to select the next hop nodes. Table 2 regroups all the default values of our simulation setup. We compare our work with:

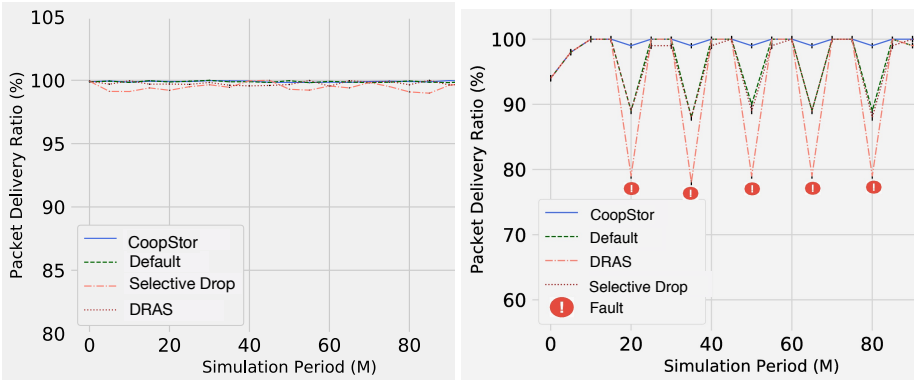
Selective Drop generates several replicates for each packet, and selects carefully the packets to drop when a buffer overflow occurs [35]; The aforementioned strategy proves to be counter-effective in large dense WSNs.

DRAS (DistRibuted Adaptive Scheme) adapts its transmission rate depending on the congestion [26]. It is similar to a back-pressure algorithm, reducing efficiently the outgoing rate hop by hop toward the sources. Nevertheless, DRAS does not perform always as expected and results to delays in large networks.

Default represents the default strategy. All the packets are enqueued until the buffer is full (32 packets by default);

Simulation setup	**Default Value**
Duration	1.5 *hours*
Topology	Random
Number of nodes	40 nodes
Number of source nodes	20 nodes
Simulation Area (in COOJA)	150x100 meters
Routing	
Routing protocol	RPL [36]
Routing metric	Expected Transmission Count (ETX)
Objective Function	MRHOF [10]
MAC	
MAC protocol	IEEE 802.15.4-TSCH [2]
Scheduling algorithm	6TiSCH-minimal [38]
N_{sh} (number of shared cells)	1
SF_{length} (slotframe length)	7
N_{scan}	1
Maximum retries	3
Queues / Traffic	
Queue size	32 *pkts*
Application model	CBR, 1 $pkt/12$ s
Payload size	102 *Bytes*
PHY	
Radio chipset model	CC2420
Radio propagation	2.4Ghz
Radio model	Unit Disk Graph Medium (UDGM)
Modulation model	O-QPSK
Transmission power	-10 dBm

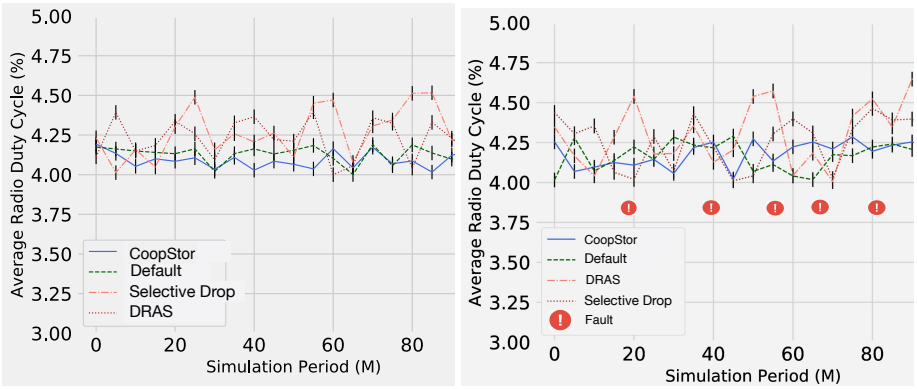
Table 2: Simulation setup.



(a) No fault (b) With faults

Fig. 5: Reliability achieved by the different forwarding strategies.

CoopStor is the strategy presented in this paper. It exploits all the nodes to store the packet when a fault (i.e. buffer overflows) occurs.



(a) No fault (b) With faults

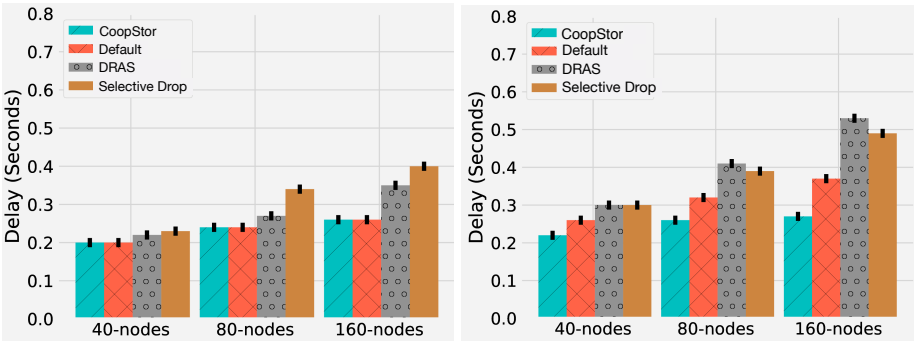
Fig. 6: Duty Cycle ratios achieved by the different strategies.

6.2 Emulation of Faults

Our approach is to depict average results from 200 samples of experiments and with 95% confidence interval used. Firstly, we measure the reliability achieved by the different solutions when exploiting a static topology where everything is stable (Fig. 5a). Initially we consider a topology with 40 nodes. We can verify that all the solutions achieve high reliability, delivering most of the packets to the sink. DRAS and Selective Drop tends to overreact, dropping some packets because they consider false congestion. Typically, DRAS may over-estimate the congestion, forcing the upward nodes to reduce their traffic rate, increasing the probability of a buffer overflow.

Then, we generated faults, by turning a device off to study the convergence for the reconstruction (Fig. 5b). Faults are inserted every 20 seconds. This period is selected in order to let the network re-converge. DRAS achieves the lowest reliability, dropping almost 20% of the packets when a fault occurs. Reducing the rate is not enough: packets fill very quickly the buffers all along the routes, which results to dropping some packets. The recovery is also quite slow, when increasing the number of packets: upward nodes did not yet recover. RPL achieves a slightly higher Packet Delivery Ratio, delivering more packets. However, still 10% of the packets are dropped. Selective Drop behaves similarly to RPL and cannot accurately limit the packet drops. On the contrary, CoopStor achieves the highest reliability, delivering almost all the packets even when a fault occurs. The forwarding nodes store successfully the packets in the queues of their neighbors before a buffer overflow occurs, limiting the packet drops.

Then, we measure the radio duty cycle (i.e. percentage of time the radio chipset is turned on) in Figure 6. We can notice that RPL and DRAS are very volatile, even without faults (Fig. 6a): the routes may change even if the topology is static, because of a dynamic routing metric [14]. CoopStor achieves the highest stability, and a device is awake only 4% of the time on average. Packets are enqueued until the novel next hop is chosen, and the queue is then emptied in bursts, increasing



(a) Without faults (b) With faults

Fig. 7: Delay in large-scale topologies

the radio duty cycle regularly. The recovery is sufficiently fast to not penalize too much the duty cycle ratio of leaf nodes.

Then, we injected faults in the topology, every 20 seconds, as previously (Fig. 6b). The radio duty cycle is slightly increased: a device stays awake in every shared cell until its buffer is empty. However, the impact on the energy consumption is very reasonable. We identify a higher variability, the maximum duty cycle ratio being slightly higher.

6.3 Scalability

We now study the scalability, with larger networks (between 80 and 160 devices) in the same simulation area. We insert faults following a Poisson distribution with range of [5-10] faults per minute. We first measure the delay with and without faults (Fig. 7) and the simulation period is doubled to 3-hours in total. The end-to-end delay increases with more nodes: the network has more packets to forward, increasing possibly the congestion, and the number of transmissions in a given shared cell. CoopStor keeps on providing the lowest delay compared with all the other solutions. Typically, DRAS may introduce useless buffering delay by over-estimating the amount of congestion. Even when no fault occurs, some packets undergo a longer delay. The presence of faults only slightly increase the delay for CoopStor: the packets are temporarily stored in neighboring nodes, and de-queued as soon as a valid route exists. CoopStor achieves to reduce by 30% the end-to-end delay compared with the Default strategy, in presence of faults.

Then, we measure the Packet Delivery Ratio (Fig. 8). With only 40 nodes, the topology is less redundant, and each relay node has a smaller number of leaf neighbors to store its packets. However, with 80 or 160 nodes, the PDR is close to 100%: faults are handled transparently by the network infrastructure, and the packets are correctly delivered to the sink. DRAS is much less scalable: a waterfall effect may force the nodes to decrease their traffic rate, dropping more packets for high densities. Similarly, the Default strategy cannot deliver all the packets to the sink when the network has to re-converge to a legal state. Finally, Selective Drop also drops more packets, and is inefficient to handle local routing modifications.

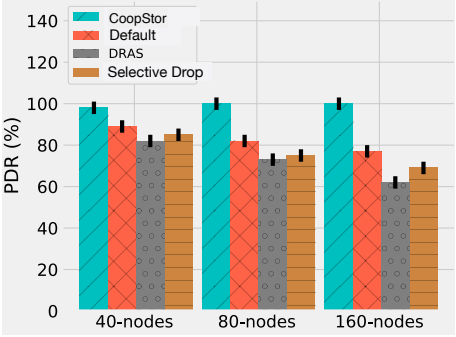


Fig. 8: Efficiency of large-scale networks in presence of faults.

7 Conclusions and Perspectives

In this paper we have presented CoopStor that targets at achieving in a reliable data collection in fault and delay tolerant wireless networks. We have shown to what extent such a mechanism would be profitable during the data collection phase of a time-driven and event-driven monitoring application and even more, in many-to-one topologies where the congestion is prominent around the root of the tree. Our approach considerably improves the reliability in presence of faults. Even when the routing protocol has to reconstruct some routes with e.g. a local repair, the network infrastructure is able to store efficiently the packets to still provide high reliability. CoopStor reduces also the delay since it triggers a `storing mode` only when required, not overreacting to changes.

We aim in a future work to investigate how our solution may be adapted to support other MAC layers. In particular, preamble sampling protocols may implement different sampling periods for relay vs leaf nodes. We need to modify the equations in section 5.2 to formulate the buffer overflow probability, depending among other parameters on the density and the sampling period. We also expect to implement our prototype on the FIT IoT-LAB large-scale testbed to assess the performance of our buffering strategy in a smart building scenario, in presence of external interference, with sudden radio environment changes. We also expect to implement our prototype on the FIT IoT-LAB large-scale testbed to assess the performance of our buffering strategy in a smart building scenario, in presence of external interference, with sudden radio environment changes.

Acknowledgement

This work was partly supported by the French National Research Agency (ANR) project Nano-Net under contract ANR-18-CE25-0003.

References

1. 802.15.4-2006 - IEEE Standard for Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer

(PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). IEEE Std. 802.15.4-2006 (2006)
2. IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. IEEE Std. 802.15.4e-2012 (2012)
3. Ahmed, A.M., Paulus, R.: Congestion detection technique for multipath routing and load balancing in wsn. Wireless Networks **23**(3), 881–888 (2017). DOI 10.1007/s11276-015-1151-5
4. Ahn, G.S., Hong, S.G., Miluzzo, E., Campbell, A.T., Cuomo, F.: Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks. In: SenSys, pp. 293–306. ACM, New York, NY, USA (2006). DOI 10.1145/1182807.1182837
5. Ahrar, E.M., Nassiri, M., Theoleyre, F.: Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks. Journal of Network and Computer Applications **142**, 25 – 36 (2019). DOI https://doi.org/10.1016/j.jnca.2019.05.013. URL http://www.sciencedirect.com/science/article/pii/S1084804519301808
6. Arshad, S., Azam, M.A., Rehmani, M.H., Loo, J.: Recent advances in information-centric networking-based internet of things (icn-iot). IEEE Internet of Things Journal **6**(2), 2128–2158 (2019). DOI 10.1109/JIOT.2018.2873343
7. Chen, P., Sung, M., Cheng, S.: Buffer occupancy and delivery reliability tradeoffs for epidemic routing. CoRR **abs/1601.06345** (2016). URL http://arxiv.org/abs/1601.06345
8. De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. Wirel. Netw. **11**(4), 419–434 (2005). DOI 10.1007/s11276-005-1766-z
9. Dobslaw, F., Zhang, T., Gidlund, M.: End-to-end reliability-aware scheduling for wireless sensor networks. IEEE Transactions on Industrial Informatics **12**(2), 758–767 (2016). DOI 10.1109/TII.2014.2382335
10. Gnawali, O., Levis, P.: The Minimum Rank with Hysteresis Objective Function. RFC 6719 (Proposed Standard) (2012). URL http://www.ietf.org/rfc/rfc6719.txt
11. Gorla, E., Ravagnani, A.: An algebraic framework for end-to-end physical-layer network coding. IEEE Transactions on Information Theory **64**(6), 4480–4495 (2018). DOI 10.1109/TIT.2017.2778726
12. IEEE Standard for Low-Rate Wireless Networks: IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) (2016)
13. Iova, O., Theoleyre, F., Noel, T.: Using multiparent routing in RPL to increase the stability and the lifetime of the network. Ad Hoc Networks **29**, 45 – 62 (2015). DOI 10.1016/j.adhoc.2015.01.020
14. Iova, O., Theoleyre, F., Watteyne, T., Noel, T.: The love-hate relationship between ieee 802.15.4 and rpl. IEEE Communications Magazine **55**(1), 188–194 (2017). DOI 10.1109/MCOM.2016.1300687RP
15. Jan, M.A., Jan, S.R.U., Alam, M., Akhunzada, A., Rahman, I.U.: A comprehensive analysis of congestion control protocols in wireless sensor networks. Mobile Networks and Applications (2018). DOI 10.1007/s11036-018-1018-y
16. Jenschke, T.L., Papadopoulos, G.Z., Koutsiamanis, R.A., Montavont, N.: Alternative Parent Selection for Multi-Path RPL Networks. In: Proceedings of the 5th IEEE World Forum on Internet of Things (WF-IoT), pp. 533–538 (2019)
17. Ji, C., Koutsiamanis, R.A., Montavont, N., Chatzimisios, P., Dujovne, D., Papadopoulos, G.Z.: TAOF: Traffic Aware Objective Function for RPL-based Networks. In: Proceedings of the Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–5 (2018)
18. Kafi, M.A., Ben-Othman, J., Ouadjaout, A., Bagaa, M., Badache, N.: Refiacc: Reliable, efficient, fair and interference-aware congestion control protocol for wireless sensor networks. Computer Communications **101**, 1 – 11 (2017). DOI 10.1016/j.comcom.2016.05.018
19. Kim, H.S., Kim, H., Paek, J., Bahk, S.: Load balancing under heavy traffic in rpl routing protocol for low power and lossy networks. IEEE Transactions on Mobile Computing **16**(4), 964–979 (2017). DOI 10.1109/TMC.2016.2585107
20. Korte, K.D., Sehgal, A., Schönwälder, J.: A study of the rpl repair process using contikirpl. In: International Conference on Autonomous Infrastructure, Management, and Security (AIMS), pp. 50–61. IFIP, Luxembourg, Luxembourg (2012). DOI 10.1007/978-3-642-30633-4_8

21. Koutsiamanis, R.A., Papadopoulos, G.Z., Jenschke, T.L., Thubert, P., Montavont, N.: Meet the PAREO Functions: Towards Reliable and Available Wireless Networks. In: Proceedings of the IEEE International Conference on Communications (ICC) (2020)
22. Lien, S.Y., Hung, S.C., Deng, D.J., Wang, Y.J.: Efficient ultra-reliable and low latency communications and massive machine-type communications in 5g new radio. In: GLOBECOM, pp. 1–7. IEEE (2017). DOI 10.1109/GLOCOM.2017.8254211
23. Mahmood, M.A., Seah, W.K., Welch, I.: Reliability in wireless sensor networks: A survey and challenges ahead. Computer Networks **79**, 166 – 187 (2015). DOI 10.1016/j.comnet.2014.12.016
24. Merlin, C.J., Heinzelman, W.B.: Duty cycle control for low-power-listening mac protocols. IEEE Transactions on Mobile Computing **9**(11), 1508–1521 (2010). DOI 10.1109/TMC.2010.116
25. Muralidharan, S., Sahu, B.J.R., Saxena, N., Roy, A.: Ppt: A push pull traffic algorithm to improve qos provisioning in iot-ndn environment. IEEE Communications Letters **21**(6), 1417–1420 (2017). DOI 10.1109/LCOMM.2017.2677922
26. Papadopoulos, G.Z., Pappas, N., Gallais, A., Noel, T., Angelakis, V.: Distributed Adaptive Scheme for Reliable Data Collection in Fault Tolerant WSNs. In: World Forum on Internet of Things (WF-IoT), pp. 116–121. IEEE (2015)
27. Pappas, N., Gunnarsson, J., Kratz, L., Kountouris, M., Angelakis, V.: Age of information of multiple sources with queue management. In: Communications (ICC), 2015 IEEE International Conference on, pp. 5935–5940 (2015). DOI 10.1109/ICC.2015.7249268
28. Ruan, Z., Luo, H., Chen, Z.: Improving reliability of erasure codes-based storage paradigm under correlated failures for wireless sensor networks. International journal of communication systems **29**, 992–1011 (2016)
29. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In: Workshop on Delay-tolerant Networking (WDTN), pp. 252–259. ACM SIGCOMM, Philadelphia, Pennsylvania, USA (2005). DOI 10.1145/1080139.1080143
30. Sun, Y., Song, H., Jara, A.J., Bie, R.: Internet of things and big data analytics for smart and connected communities. IEEE Access **4**, 766–773 (2016). DOI 10.1109/ACCESS.2016.2529723
31. Swamy, V.N., Rigge, P., Ranade, G., Sahai, A., Nikolić, B.: Network coding for high-reliability low-latency wireless control. In: Wireless Communications and Networking Conference (WCNC), pp. 1–7. IEEE (2016). DOI 10.1109/WCNC.2016.7564660
32. Teles Hermeto, R., Gallais, A., Theoleyre, F.: Scheduling for IEEE802.15.4-TSCH and Slow Channel Hopping MAC in Low Power Industrial Wireless Networks. Comput. Commun. **114**(C), 84–105 (2017). DOI 10.1016/j.comcom.2017.10.004
33. Theoleyre, F.: A route-aware MAC for wireless multihop networks with a convergecast traffic pattern. Computer Networks **55**(3), 822 – 837 (2011). DOI 10.1016/j.comnet.2010.10.018
34. Theoleyre, F., Papadopoulos, G.Z.: Experimental validation of a distributed self-configured 6tisch with traffic isolation in low power lossy networks. In: International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pp. 102–110. ACM (2016). DOI 10.1145/2988287.2989133
35. Wang, E., Yang, Y., Wu, J.: A knapsack-based message scheduling and drop strategy for delay-tolerant networks. In: International Conference on Embedded Wireless Systems and Networks (EWSN), pp. 120–134 (2015). DOI 10.1007/978-3-319-15582-1_8
36. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Tech. rep., IETF (2012)
37. Wu, C., Ji, Y., Xu, J., Ohzahata, S., Kato, T.: Coded packets over lossy links: A redundancy-based mechanism for reliable and fast data collection in sensor networks. Computer Networks **70**, 179 – 191 (2014). DOI 10.1016/j.comnet.2014.05.010
38. X. Vilajosana, et al.: Minimal 6tisch configuration. `https://datatracker.ietf.org/doc/draft-ietf-6tisch-minimal/`, IETF (2015)
39. Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B.: Named data networking. SIGCOMM Comput. Commun. Rev. **44**(3), 66–73 (2014). DOI 10.1145/2656877.2656887