Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks

Erfan Mozaffari Ahrar, Mohammad Nassiri, Fabrice Theoleyre

Multipath aware Scheduling for High Reliability and Fault Tolerance in Low Power Industrial Networks

Erfan Mozaffari Ahrar[a], Mohammad Nassiri*[a], Fabrice Theoleyre[b]

[a] *Computer Department, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran*
[b] *ICube Laboratory, CNRS / University of Strasbourg, Pole API, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France*

Abstract

The Industrial Internet of Things is expected to enable the Industry 4.0 through the large deployment of low-power devices. However, industrial applications require most of the time high reliability close to 100%, and low end-to-end delays. Thus, most industrial wireless networks rely on a strict schedule of the transmissions to avoid collisions, and implement frequency hopping to combat external interference. In multihop topologies, the network has to decide both when the transmissions have to be scheduled, and which router can forward the packets. To be fault-tolerant, multipath routing consists in exploiting several paths in parallel. We exploit here a braided path routing structure, where each router has several possible next hops. Thus, we can cope with any fault along the path, while still providing a remaining operational path. We propose also a scheduling algorithm, where multiple transmitters are attached to a single cell, to the same receiver. The schedule is constructed such that only one transmitter is active at a time, and is consequently collision-free. Mutualizing the same cell for several transmitters reduces the energy consumption and increases the network capacity. Our approach is still fully compliant with the standard while minimizing idle listening. Our simulation results show the relevance of such solution to provide high-reliability and fault-tolerance. While the single and disjoint paths solutions achieve a very low reliability (20%) when two nodes crash, we keep on providing a packet delivery ratio above 80%, whatever the conditions. Besides, our scheduling algorithm is particularly energy efficient since it presents the same energy consumption as the classical single path routing scheme.

Keywords: high-reliability, multipath, opportunistic forwarding, scheduling algorithms, energy efficiency, capacity

1. Introduction

Industry 4.0 aims to constitute the novel industrial revolution, making the industrial processes more adaptive, through reconfigurable production lines. To reach such flexibility, Industry 4.0 relies heavily on the Internet of Things (IoT) [1]. A collection of sensors and actuators is disseminated to monitor and control industrial processes. Because they are battery-powered, they can be easily deployed and reconfigured. We expect a growing adoption of wireless technologies, and thus very dense deployments [2].

*Corresponding author
Email addresses: e.mozaffari@alumni.basu.ac.ir (Erfan Mozaffari Ahrar), m.nassiri@basu.ac.ir (Mohammad Nassiri), theoleyre@unistra.fr (Fabrice Theoleyre)

Industrial applications have to react in real-time to the environments. Typically, a control loop relies on a controller which collects a set of measurements, and which controls a collection of actuators [3]. To prove the control application is *safe*, the communication network has to provide high-reliability close to 100% and a bounded end-to-end latency [4].

Deterministic MAC layers aim to provide these guarantees. By carefully scheduling the transmissions, the communication network is able to avoid collisions, making the medium access deterministic. A scheduling matrix defines when each device can receive or transmit a packet; during the rest of the time, they can turn off their radio chipset to save energy. The schedule can be constructed by a centralized controller which has a complete knowledge of the topology, and the traffic pattern. Alternatively, each device has to preempt some resources, and to resolve possible collisions distributively [5].

To provide strict guarantees with a distributed scheduling algorithms is actually very challenging since each device reacts autonomously to the traffic or link variations [6]. Similarly, RPL [7] is often used to construct distributively the paths, provoking the re-allocation of the cells all along the novel paths. Finally, the devices have to detect the collisions, so that the corresponding cells are relocated in a pseudo-random manner [8]. On the contrary, centralized scheduling algorithms are deterministic, and are able to provide rather stable performance.

Because of reliability constraints, the communication network has to be fault-tolerant. Typically, an industrial application must still operate when a specific device crashes, or when a link becomes unavailable because of e.g. external interference [1]. Thus, we need to provision additional resources to handle any link or node's failure. In multihop topologies, it consists in reserving several paths for the same flow, and to allocate enough transmission opportunities along each of the paths. Several heuristics have been proposed to identify these disjoint paths [9].

The challenge consists in exploiting these paths in parallel to provide high-reliability. Leapfrog [10] proposes to replicate the packets through different paths, and then to eliminate the duplicated packets based on a sequence number. Overhearing helps to increase the number of reception opportunities, at the price of a larger energy consumption. However, the schedule is not adapted to multipath, the devices being sequentially scheduled, without cell re-utilisation. Thus, the schedule length can become very large.

Link-layer anycast represents an alternative technique to improve the reliability while reducing the energy consumption [11]. Several receivers are attached to the same cell: the transmission is considered successful if at least one of the receivers is able to decode and acknowledge the packet. However, it needs to modify the link layer acknowledgement mechanisms, and the solution would not be anymore standard-compliant.

Constructing an efficient schedule to take benefit from multiple paths is particularly challenging. We have both to reduce idle listening to reduce the energy consumption, and also to mutualize some cells to make the scheduling matrix compact. If the routes and schedules are handled independently, orthogonal resources have to be allocated per path. Thus, we would obtain a very long schedule since we cannot *share* the cells between the principal and backup paths.

In this paper, we propose a centralized scheduling algorithm tailored for multipath routing. It aims to provide a compact schedule, mutualizing cells for a collection of transmitters. The contributions of this paper are as follows:

1. we propose an algorithm to construct braided multi-paths, so that each intermediary router has several possible next hops toward the border router. These paths are constructed to maximize the number of common routers, to be able to exploit a compact schedule;
2. we propose a centralized scheduling algorithm, tailored for multipath. It allocates the cells for the primary and alternative paths simultaneously, so that the same cell can be used by several transmitters with a packet for the same receiver. By forcing only one transmitter to be active at the same time, we reduce idle listening and we make the schedule more compact. To the best of our knowledge, this represents the first scheduling algorithm able to take benefit from multiple paths without having to replicate the packets;
3. we evaluate the performance of this multipath aware scheduling algorithm to demonstrate its ability to improve the reliability while being fault-tolerant. It is particularly efficient to handle nodes failure, keeping on providing a high reliability even with the presence of multiple faults.

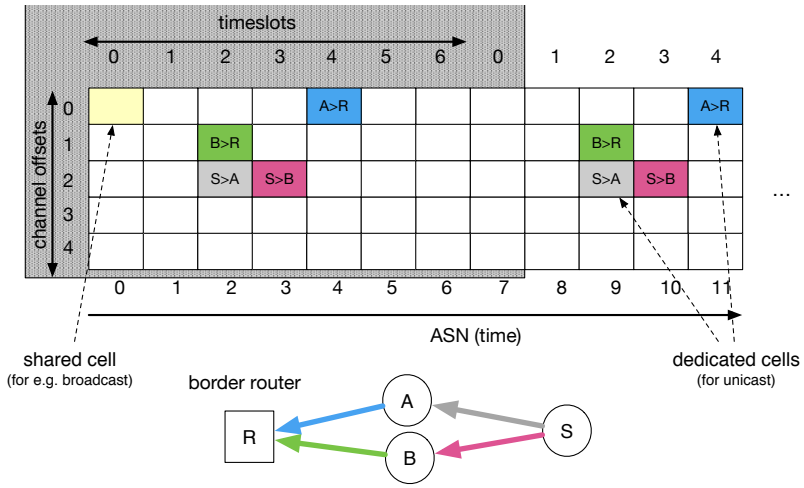


Figure 1: Schedule with a 4 nodes topology

2. Related Work and Background Knowledge

We present here the basic knowledge of slow channel hopping MAC, and how the transmissions can be organized into a scheduling matrix. Then, we detail the related work on multipath routing and fault-tolerance.

2.1. IEEE802.15.4-TSCH and 6TiSCH

IEEE 802.15.4-2015 has proposed the TSCH mode, which relies on a strict schedule of the transmissions [12]. The *slotframe* contains a fixed number of timeslots, labelled with an Absolute Sequence Number (ASN) which counts the number of timeslots since the PAN coordinator started. Based on the schedule, a node can decide its role (transmitter/receiver/sleeping mode) at the beginning of each timeslot. During a timeslot can take place at most one frame and its acknowledgment.

Industrial environments are prone to noise, shadowing, multipath fading and interference [13]. Thus, IEEE 802.15.4-2015 TSCH implements a channel hopping approach to combat external interference and signal fading and, thus, to achieve high reliability [14]. For this purpose, each *cell* in the schedule is defined by a pair of timeslot and channel offset. At the beginning of each timeslot, the actual frequency to use is derived from the channel offset and the ASN.

IEEE802.15.4-TSCH supports two medium access approaches:

shared cells are allocated to a group of nodes (e.g. for broadcast). They implement a slotted Aloha mechanism to solve contention, with a random backoff when a collision is detected;

dedicated cells are allocated to non-interfering transmitters. Thus, the transmitter just starts the transmission after a fixed offset from the beginning of the timeslot.

Let's consider the topology and the schedule illustrated in Fig. 1. The first cell is typically used for beacons, e.g. DIO in RPL and Enhanced Beacons in IEEE802.15.4-TSCH. Oppositely, all the unicast transmissions often use dedicated cells. The same timeslot may be allocated to two different radio links, but with different channel offsets (e.g. links BR and SA) to avoid collisions. Thus, unicast packets can only be dropped because of link unreliability or external interference, not because of internal collisions.

6TiSCH defines a set of protocols to execute IPv6 above IEEE802.15.4-TSCH [15]. It relies on RPL [7] to construct the paths, and uses a minimal shared schedule to send control packets when the network bootstraps. Then, dedicated cells may be either allocated by a centralized scheduler or distributively, each

pair of node negotiating the timeslots and channel offsets to use. 6TiSCH implements service differentiation, so that multiple applications with different requirements may cohabit in the same infrastructure [16].

2.2. Scheduling Algorithms

Scheduling for slow channel hopping industrial networks has received much attention in the past [5]. A sufficient number of cells has to be allocated for each pair of nodes, while avoiding collisions to limit the number of retransmissions. In a convergecast network, where every packet has to be delivered to the border router, the scheduling process allocates the cells from one node to its next hop, aka its *parent* in a tree rooted at the border router.

Many distributed solutions have been proposed to allocate cells in a 6TiSCH network. SFx is one of the default scheduling solution for 6TiSCH [17]. For each of its neighbors, it maintains a number of cells larger than the number of packets to forward during the last slotframes. It relies on a hysteresis function to limit the number of de/re-allocations.

Centralized scheduling algorithms rely on a complete knowledge of the radio topology and the traffic requests. Usually, they first designate a path for each flow, and then allocate the bandwidth, hop by hop along each path. Traffic aware scheduling algorithm (TASA) [18] relies on matching and graph coloring techniques to allocate cells for each pair of nodes. KAUSA adopts a per-flow strategy: the scheduler allocates enough cells along the path of each flow to respect end-to-end reliability and delay [19].

Several techniques have been proposed to improve the reliability in this kind of deterministic architecture. Over-provisioning consists in reserving additional cells to retransmit the packets if the transmission has failed. For instance, the number of cells may be inversely proportional to the Packet Delivery Ratio for the considered radio link [20]. Dobslaw *et al.* [21] fortify a centralized schedule by allocating additional cells to the most unreliable links until the deadline constraint cannot be anymore respected. Hashimoto *et al.* [22] allocate shared cells for the retransmissions. However, shared cells are prone to collisions, and thus impact negatively the reliability of retransmissions.

To the best of our knowledge, no scheduling algorithm has been proposed to handle efficiently multipath routing to increase the fault-tolerance.

2.3. Energy Efficient Routing in industrial networks

Energy efficient routing has attracted much attention in the past since the IoT devices are battery powered and have to optimize their energy consumption [23]. Most solutions focus on minimizing the average energy consumption. However, we increase the network lifetime by rather minimizing the energy consumption of the most loaded device. It leads to the energy-balanced routing problem [24].

RPL is the most widely used routing standard for the Internet of Things [7]. It constructs a Destination Oriented Directed Acyclic Graph (DODAG), rooted at the border router. For this purpose, each node has to compute a rank, denoting its *virtual* distance from the border router. The rank is computed with an *objective function*, taking as argument the rank and the link quality metric of its next hop (parent). The Objective Function Zero (OF0) [25] typically considers a linear additive path metric. The Expected Transmission Count (ETX) metric [26] is widely used to select the best parents, minimizing the average number of packets to transmit along the path before a copy is received correctly by the border router.

However, RPL has been proved to present stability and reliability issues [27], which jeopardize its utilization for sensitive applications. Similarly, it handles only poorly actuators, since downward paths are practically less reliable [28]. Software-Defined Networks based approaches seem promising to construct QoS-aware paths for industrial networks [29].

2.4. Multipath Routing

Multipath routing helps to increase the fault tolerance: a backup path is in that case available [9]. Since constructing perfectly node-disjoint paths is very expensive, many routing schemes rely on braided multipath (link disjoint). At each hop, an alternative next hop exists to recover. RPL has already been modified so that each node can forward its packets to several parents [30, 31].

In industrial networks, REALFLOW forces each device to send a collection of parents to the controller [32]. Then, a schedule is computed in a centralized manner. However, this schedule is not adapted to multipath, the controller reserving dedicated cells independently for each path. DiGS [33] proposes rather a distributed algorithm, where each device picks two parents. The schedule is then derived automatically from the ID. However, only 75% of the flows achieve a PDR higher than 95%, and the fault-tolerance has not been evaluated.

Leapfrog proposes to replicate the packets, through disjoint paths to improve the reliability [10]. The solution relies on overhearing, forcing the parents to stay awake to overhear the transmissions. Besides, all the devices are scheduled consecutively, according to their distance from the border router. By exploiting all possible overhearing opportunities, they reduce the jitter. However, the energy consumption is increased even when no fault occurs, since all the nodes have to stay awake for overhearing, replicating the packets, etc.

Here, we propose rather to construct a schedule tailored for multipath, without modifying the link layer. We do not force a node to stay awake for overhearing. Besides, the packets are not replicated: an alternative, unicast path is used automatically only when the transmission to the primary parent fails. Thus, transmissions keep on being acknowledged to control finely the forwarding process.

3. Multipath Aware Scheduling Algorithm

We aim here to increase the reliability and the fault tolerance by exploiting multiple paths. Let us focus on a bi-connectivity approach, which can easily be generalized to a k-connectivity (i.e. k different next hops for each device, except the neighbors of the sink). For traffic isolation, we have to allocate for a given flow a collection of cells along each of the two paths. We adopt the following strategy: if a packet is not acknowledged by the primary next hop, a node uses its secondary one, instead of retransmitting to the same next hop. We can note that the cell to the secondary next hop is not used when the transmission is successful.

3.1. Problem Statement

Let us consider the example in Fig. 1, where a source S has to transmit a packet along two node-disjoint paths toward the border router (R). We consider here flow isolation, where each flow has *its* own dedicated cells. The schedule is clearly inefficient:

1. delay: cells should be allocated consecutively in the path. In particular, the cell for the link (B,R) should be allocated *after* the cell for the link (S,B), to decrease the buffering delay. In a more complex scenario, we have to consider all the combinations since each intermediary hop may select any of its parents;
2. redundancy: only one of the two paths is practically used. In particular, if A correctly acknowledges the packet, the green cell (BR) would be unused. The cells allocated to the other path are wasted. In particular, we may have here allocated the same cell to the links (AR) and (BR): either A or B will receive the packet from S. Besides, R is the receiver for both links, and we would reduce idle-listening.

3.2. Description of the approach

We propose here a scheduling algorithm able to take benefit from multipath routing while still exploiting a compact schedule. We assume the centralized scheduler has the knowledge of the topology (list of links), and of traffic requests. To simplify the description, we consider only a convergecast traffic pattern, where all the devices send their packets to a common border router. However, the scheme can be easily extended to handle both directions, by considering the actuators as destinations.

We proceed in the following way (Fig. 2):

1. we first construct the braided paths. We select two next hops per device toward the border router, while maximizing the number of common ancestors to reduce the schedule's length;

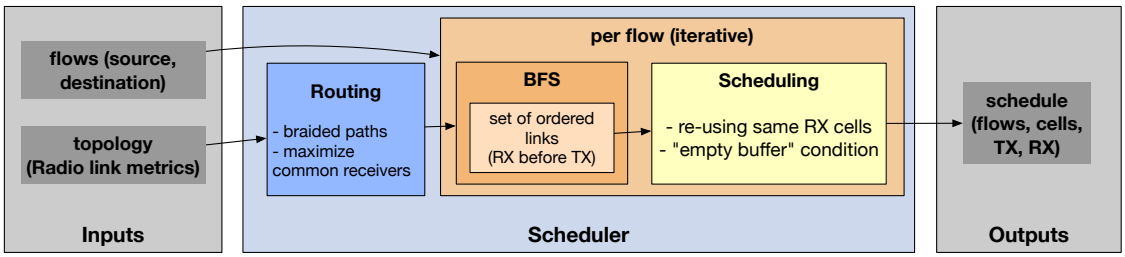


Figure 2: Workflow to construct a multipath aware schedule.

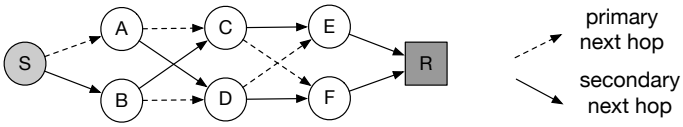


Figure 3: Ideal braided paths for the RPL DODAG (dualpath case)

2. for each flow, we order the different hops in the braided paths. This way, we can in the next step iterate over this set to allocate the cells while guaranteeing a packet is received before being forwarded;
3. we allocate the cells to each link, while maximizing the number of cells which can be shared without collisions. Typically, we search for a cell toward the same receiver which was previously allocated to the same flow. This way, we can minimize idle listening.

We will detail now each of these steps.

3.3. Selecting the right set of next hops

We have first to construct the different redundant paths. We select for each device two different next hops toward the destination (i.e. the border router).

To reduce the number of cells to allocate, we should ideally construct braided paths, as illustrated in Figure 3. For instance, A and B are the two next hops of the source S toward R. To be able to re-use the same cells, and to exploit a compact schedule, the nodes A and B should have the same set of next hops. If this characteristic is topologically impossible, the node will choose *independent* next hops, at the cost of a larger schedule (i.e. some cells cannot be mutualized).

A node first selects its preferred next hop (PP), using any routing metric, and computes its *rank* accordingly. We use typically the Expected Transmission Count (ETX) metric to select the most reliable paths, providing the smallest number of transmissions. Then, the node selects its secondary next hop among the neighbors with a lower rank. It selects the neighbor with the lowest ETX, according to the following conditions (in descending importance):

1. the neighbor has exactly the same set of next hops as those of PP;
2. the neighbor has one common next hop with those of PP;
3. the neighbor has a non overlapping set of next hops.

3.4. Ordering the links to respect the scheduling constraints

We have now to construct a schedule, on top of this routing topology. We adopt a per-flow scheduling approach: a collection of cells is allocated iteratively to each flow. The scheduling algorithm has to cake care of the multipath organization.

Indeed, we should re-use the same cells for the two paths used in parallel. This helps to reduce the energy consumption (less idle listening) and to provide a more compact schedule. Not considering the multiple paths would be unscalable, and the schedule size would grow exponentially with the flow length.

Algorithm 1: Construction of an ordered set of links (modified BFS).

```
   Data: G(V,E) routing graph for the considered flow
   Result: set of ordered links Links
   // indegree of each node
 1 for v ∈ V do
 2 |  deg[v] ← getInDegree(v);
 3 end
   // until all the nodes have been handled
 4 Nodes ← {src};
 5 while |Nodes| > 0 do
 6 |  for (u, v)|u ∈ Nodes ∧ (u, v) ∈ E do
   |  |  // the TX cells can be allocated only when all the RX cells have been handled
 7 |  |  if deg(u) == 0 then
 8 |  |  |  add(Links, (u, v));
 9 |  |  |  deg(v) ← deg(v) − 1;
10 |  |  end
11 |  end
12 end
```

Indeed, we would have to consider all the possible combinations, each hop having two different choices to forward the packets of a flow.

Let us consider the topology depicted in Fig. 3. The same cell may be allocated safely to the links $(A \rightarrow C)$ and $(B \rightarrow C)$. Because of flow isolation, B will receive a packet to forward only if the transmission through the preferred next hop has failed. Thus, A and B will not be active simultaneously during this cell. This approach is still perfectly standard compliant:

- the cell is dedicated (TX mode) for the nodes A and B, and the node C is defined as the only one destination;
- the cell is shared (RX mode) for the node C, without specifying the source address.

Because the cell is dedicated, the transmitters will pop the first packet in their queue to the node C and transmit it without contention. Thus, the scheduler has to take care that A and B do not have a packet in their queue at the same time.

Based on this characteristic, we order the set of links so that the scheduler can then allocate iteratively the cells. We construct properly an ordered set of the links in the following way (algo. 1):

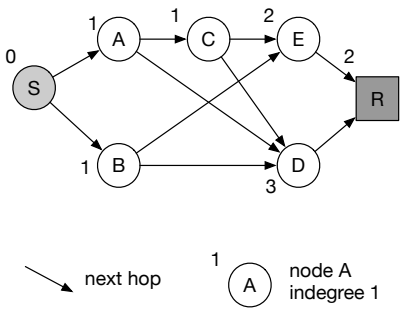
1. we compute the indegree of each node, i.e. number of incoming edges for which it is the head (lines 1-3);
2. we construct iteratively the FIFO, inserting the links for which all the RX cells have been already scheduled. In our case, their indegree is null (lines 5-7).

Thus, the allocation of a radio link does depend only on the previously allocated links. This way, we can respect that a packet is received before being forwarded. Since no deadlock can be created, we don't need any fallback strategy as other per-flow schedulers, such as [19]. Thus, our algorithm converges faster.

Let us consider the flow illustrated in Figure 4. It represents a non ideal case, where the mutualization is not maximal (e.g. A and B do not have the same next hops). We first compute the indegree of each node, reported in the first column. Then, we start from the source. The links toward A and B are inserted into $\mathcal{L}inks = \{(S, A), (S, B)\}$ during the first round since the indegree of S is equal to 0. The indegrees of A and B are automatically updated (0). We can note that during the second round, the link (D,R) cannot be inserted since the indegree of D is not equal to 0, it will be handled during the 3^{rd} round.

3.5. Per Flow Cells Allocation

After having ordered the links for each flow, we have now to schedule a collection of cells for each radio link, and for each flow. Long paths tend to create more constraints, since a packet has to be relayed toward the destination. Thus, we order the flows by the maximum hop distance from the source to the destination. If



	Round	0	1	2	3	4
	links inserted		(S,A) (S,B)	(A,C) (A,D) (B,E) (B,D)	(C,D) (C,E)	(E,R) (D,R)
indegree	S	0				
	A	1	0			
	B	1	0			
	C	1	1	0		
	D	3	3	1	0	
	E	2	2	1	0	
	R	2	2	2	2	0

Figure 4: Illustration of the reverse BFS to compute an ordered set of links.

a message has to be fragmented [19], the scheduler generates several packets, which are handled as separated flows. Thus, we assume an end-to-end fragmentation / reassembling.

The scheduler iterates on the flows, and allocates cells to each radio link. For this purpose, it considers the following constraints:

half-duplex: a node cannot simultaneously receive or transmit a packet [18];

conservation: a packet must be received before being forwarded. For a given flow, all its RX cells are scheduled in a node **before** its TX cells;

empty buffer: a RX cell can be allocated to a node N only if its buffer is either empty or may contain a packet from the same flow. In other words, its RX cells canot be located between a RX and TX cell for any already allocated flow. Else, this would mean that a packet was received for another flow, and may not have been forwarded yet.

Actually, we have to verify that no cell for another flow is located between the first RX and the last TX cells (worst-case scenario). This way, at most one packet is present at any time in each queue: we guarantee flow isolation;

multiplexing: when the same receiver is involved for several transmission opportunities, they should be allocated in the same cell. Since we are sure that a buffer contains packets only from one flow (empty buffer condition), only one path is actually used at a time. Thus, only one of these transmissions will be used in a given cell, and no collision is created;

compactness: to limit the delay, the scheduler picks the first available cell in the schedule which respects the previous conditions.

We adopt a greedy allocation which proceeds in the following way:

1. we handle iteratively the links in the FIFO ($\mathcal{L}inks$) (line 2) obtained through Algo. 1. By construction, we respect the conservation constraint if we consider the links in FIFO order;
2. we search for a cell allocated to the same receiver, and for the same flow (line 4). We have two possibilities:
   (a) if such cell exists, the current radio link can be safely assigned to the same cell (line 17). Indeed, this cell has been allocated to another transmitter, which, by definition cannot transmit simultaneously. Indeed, the packet is not replicated along the path: it is retransmitted only if the primary next hop is not able to decode (and acknowledge) it;
   (b) else, we cannot mutualize the cells, and have to find an empty cell (i.e. it has not been allocated to another flow during this timeslot). Besides, a channel offset has to be available for this link, i.e. the same cell has not been allocated to another transmission in the network (line 8);

Algorithm 2: Greedy allocation of the cells.

Data: $\mathcal{L}inks$: list of links, fid: flow id considered in this step $\mathcal{S}ched$: schedule for the previously allocated flows, $buffer[node][slot]$: content of the buffer of each node
Result: SUCCESS or FAIL, depending if a cell was inserted correctly in the schedule

```
1  ts ← 0 ;                                                  /* start with the first cell of the slotframe */
2  for (tx, rx) ∈ Links do
3      tmp ← ts;
       /* does a cell exist with the same flow id and receiver and after the rx cell?                  */
4      (ts, ch) ← getCellFor(rx, fid, Sched, buf, ts)
5      if ts == NULL then                  /* else, pick the first available cell after the receiving cell */
6          ts ← tmp ;                                          /* search after the previous rx cell */
7          while ts ≤ SFLength do
               /* the buffer is empty for the receiver, and a channel offset exists without interference */
8              if ((buf[rx][ts] == 0 or buf[rx][ts] == fid) and ∃ch|noInterference(Sched, ch, tx, rx)) then
9                  break;
10             end
11             ts ← ts + 1;
12         end
13     end
14     if ts > SFLength then            /* no candidate cell was found, the scheduling process has failed */
15         return FAIL
16     end
17     Sched = Sched ∪ (ts, ch, rx, tx) ;                /* the cell is identified:  we can modify the schedule */
18     for k ∈ [ts, SFLength] do                          /* the buffer  is reserved for the whole slotframe */
19         buf[rx][k] ← fid ;                                        /* (until the last TX cell is scheduled) */
20     end
21     if ∄(tx, ∗) ∈ Links then                            /* this is the last TX cell for this transmitter */
22         for k ∈ [ts + 1, SFLength] do                  /* reset the remaining slots:  free for other flows */
23             buf[tx][k] ← 0
24         end
25     end
26 end
27 return SUCCESS;
28 Function getCellFor(addr, fid, Sched, buffer[][], ts_prev):
29     for (ts, ch, tx, rx) ∈ Sched do        /* same address, correct flow id, and after the previous rx cell */
30         if rx == addr and buf[rx][ts] == fid and ts ≥ ts_prev then
31             return (ts, ch)
32         end
33     end
34     return (NULL, NULL);
```

3. after having modified the schedule, we reserve all the slots until the end of the slotframe for this flow. This means that the buffer cannot be re-used for another flow until the packet has been forwarded to the next hop (lines 18-20). Indeed, we don't know yet the last TX cells for this transmitter for the flow fid.
   A buffer can be re-used safely for another flow when the last transmission opportunity has been scheduled to the next hop (worst-case scenario). Thus, we search if another link from the same transmitter has to be allocated later in the FIFO (line 21). If this condition is false, this means that the current link corresponds to the last TX cell, and we reset the corresponding timeslots until the end of the slotframe so that they can be re-used by another flow (lines 21-25).

We can note that we search for the first available cell placed after the cells already allocated in the same flow (lines 3, 6 and 7). Thus, by construction of the $\mathcal{L}inks$ set, we cannot violate the conservation constraint.

For the sake of clarity, we consider in our explanations that all the packets have to be delivered before the end of the slotframe. To consider a cyclic slotframe, we just have to modify slightly algo. 2. More precisely, we have to consider a cyclic slotframe (line 11, $ts \leftarrow (ts + 1) \pmod{SFLength}$), and we have to stop after

Table 1: Simulation Parameters

Parameters	Value
Simulator	OpenWSN
CBR	1 pkt / slotframe
Max. Hop distance	[1-7]
Topology	ladder
Number of nodes	1 border router + 6, 10 or 14 devices
Timeslot duration	15 ms
Topology size	7, 11, 15
Duration	400 pkts
# of Experiments	5 topologies per scenario
# of Shared cells	3
n_{cells} (retx cells per path)	1, 2
Physical	802.15.4 2400MHz OQPSK
Link layer	802.15.4e-TSCH
Radio	CC2420
Slotframe length	117

SFLength steps if we don't find a possible allocation.

4. Performance Evaluation

We consider here uniquely the two next hops scenario (i.e. dualpath), where each node maintains a preferred and a secondary next hop. We implemented Braided Paths[3] in the OpenWSN protocol stack [34]. We assume no label-switching solution (e.g. [20]) is available at the link layer, and we force our scheduling algorithm to enqueue at most one packet at once in the buffers to provide flow isolation. This represents a worst-case for our multipath scheduling algorithm.

4.1. Experimental Setup

To cope with unreliable links, we implemented overprovisionning. More precisely, we consider the following multipath strategies:

Single Path (SP): only one path is constructed, as RPL does by default. Each device receives $2n_{cells}$ cells (n_{cells}=1 or 2) to transmit its packets to its preferred next hop;

Disjoint Paths (DPs): the packets are replicated on the two paths. We provision independently n_{cells} transmission opportunities along each of the two paths;

Braided Paths (BPs): our approach, described in section 3. Each device receives n_{cells} cells toward its preferred parent, and the same quantity toward its backup parent;

To assess the performance of the different multipath solutions, we simulate a ladder topology as depicted in Figure 5. The ladder comprises between 7, 11 or 15 devices (1 border router, and 6, 10, or 14 devices). Each device (except the neighbors of the sink) have two possible next hops, to provide a sufficient diversity. Thus, two node-disjoint paths exist from any device to the sink. The routing topology and the schedule are fixed statically at compilation time, to focus on the long-term performance when a fault occurs using a centralized schedule.

[3]Our implementation is freely available at https://github.com/erfanmozaffari/Braided_4TX with all our scripts for a sake of reproducibility Our raw results are available at www.theoleyre.eu/tmp/results_multipath.zip (will be integrated in the GitHub repository after acceptance)

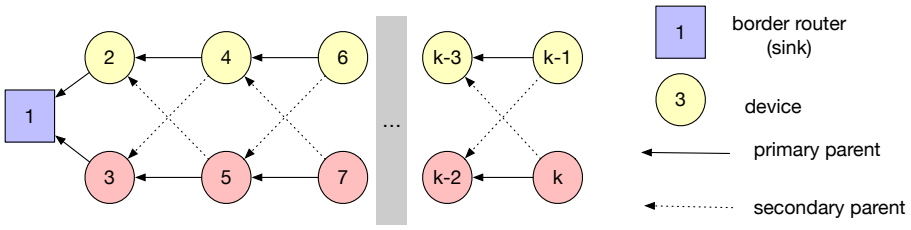


Figure 5: Ladder topology to evaluate the performance of the multipath solution (1 sink, and k-1 devices)

For each network size, we select randomly the Packet Delivery Ratio for the primary parent between 85%-95% (resp. 75%-85% for the backup parent). We report the 95% confidence intervals using 5 different topologies with random PDR values. Each device generates 1 packet per slotframe, just before the TX slot assigned to the corresponding flow by the scheduler. Thus, the end-to-end delay considers only the forwarding delay.

Because we exploit long slotframes, control packets may collide, during the shared cells. Thus, 3 shared cells are reserved at the beginning of the slotframe for synchronization and control messages. The default parameter values are summarized in Table 1.

We measured the following metrics:

Packet Delivery Ratio (PDR): the ratio of the number of packets received correctly by the border router, and the number of packets generated by the sources;

Delay: amount of time between a packet is generated, and received by the border router;

Allocated Cells: number of cells allocated in the slotframe;

Jain Index [35] to estimate the fairness among the different flows:

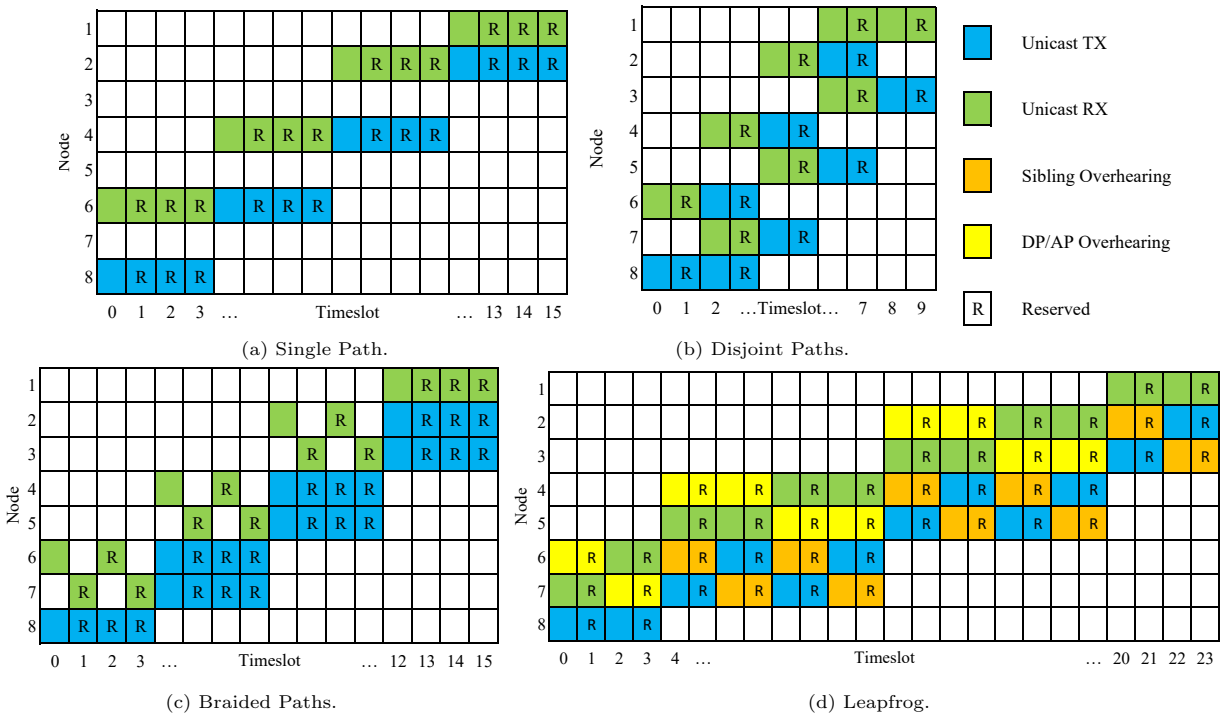
$$Jain\ Index = \frac{\left(\sum_{u\in\mathcal{F}} f(u)\right)^2}{|\mathcal{F}|\sum_{u\in\mathcal{F}} f^2(u)} \quad (1)$$

with f(u) the performance result (here the Packet Delivery Ratio) associated with the flow $u \in \mathcal{F}$.

4.2. Schedule Length

Figure 6 illustrates the schedule obtained for the flow generated by the device 8 (Fig. 5). We can note that Braided Paths (BP) (Fig. 6c) may schedule the same cell to two different transmitters. For instance, the fourth cell is allocated to the transmitters 6 and 7 to the same receiver (4). This mutualization makes the schedule very compact. Scheduling the cells for the single path scenario is expensive (Fig. 6a): cells have to be allocated consecutively, increasing the schedule's length. Exploiting Disjoint Paths is more efficients since the paths are scheduled in parallel (using different channel offsets). However, it needs to replicates the packets, which has a high energy cost.

We also represented here the long schedule produced by Leapfrog [10] to understand finely how it works (Fig. 6d). Since it relies on overhearing, the cells cannot be mutualized. These consecutive cells have a cost: Leapfrog produces a very long schedule, and the transmitters have to stay awake more frequently, e.g. 4 overhearing cells for the device 4. The schedule length is significantly larger (+50% compared with Braided and Single paths), which impacts very negatively the network capacity. Thus, we focus in the rest of this paper on the Braided, Single and Disjoint paths solutions.



(a) Single Path.

(b) Disjoint Paths.

(c) Braided Paths.

(d) Leapfrog.

Figure 6: Schedule length of the different approaches ($n_{cells} = 2$).

4.3. Static topology

We first measure the Packet Delivery Ratio for the three solutions (Fig. 7a). All the multipath solutions are able to provide high reliability, whatever the topology is. Using a single path (SP) allows the retransmission opportunities to be mutualized: $2 * n_{cells}$ cells are available for each hop. Similarly, BraidedPaths (BPs) allocates $2 * n_{cells}$ transmission opportunities to each node in the path: we have many possible retransmissions. With DisjointPaths (DPs), the packets are forwarded through two independent paths to increase the robustness, with n_{cells} cells per path. We can note that disjointPaths has to forward the packets through both paths. Typically, the secondary path is used even if the primary one succeeds to deliver the packet.

We also measure the end-to-end delay (Fig. 7b). Braided and single paths solutions provide a very low jitter: the cells are consecutively scheduled for a flow. DPs exhibits a lower delay: each path is scheduled independently. Thus, half of the bandwidth is assigned to each path, reducing the delay by one half. However, when the primary path fails to deliver the packets, the packet is received only through the second path, increasing the jitter, particularly for 15 nodes.

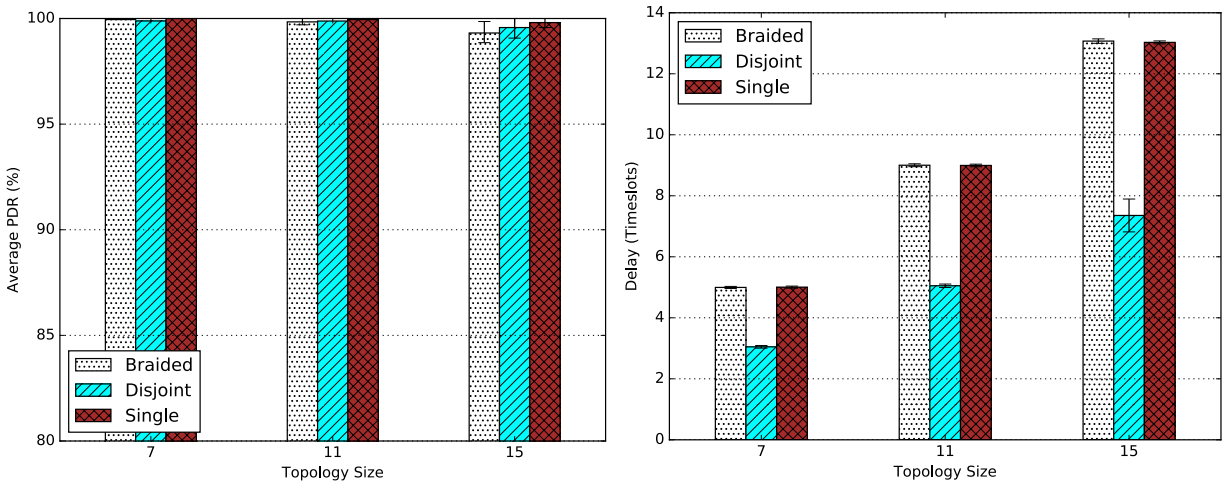
Finally, we measure the Jain Index, which is equal to 1 for all the conditions and algorithms. Thus, we didn't reproduce here the graph. All three solutions are efficient, and provide a perfect fairness for all the flows. Provisioning retransmitting cells seems sufficient.

4.4. Fault Tolerance

To assess the performance of our solution in faulty environments, we consider the following scenario:

- at time T_1, a neighbor of the sink crashes (Fig. 5 – node 2);
- at time T_2 ($T_2 > T_1$), a second node (Fig. 5 – node 5) crashes.

This way, we evaluate the robustness of the different multipath solutions. What is the reliability achieved after one or two nodes crashed suddenly? Does the system still deliver most of the packets?



(a) Average PDR.

(b) Average delay.

Figure 7: Ladder topologies of 7 to 15 nodes.

In our ladder topology, a second path exists through the second side rail. Thus, the system should keep on delivering some packets for all the devices located under the faulty node. When the second node crashes, the two side rails are impacted. Obviously, the Packet Delivery Ratio will decrease since the cells allocated through the faulty nodes cannot be used. However, the flows should exhibit a sufficient reliability (at least to report the novel statistics to the controller), until the novel schedule can be recomputed, and disseminated in the network.

4.4.1. One Node (Id 2)

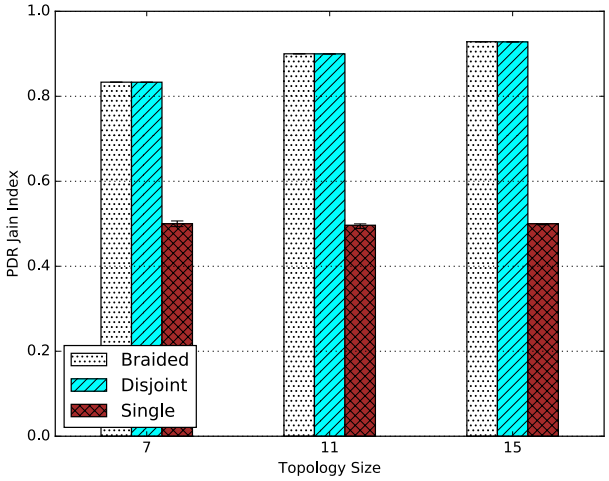
We first focus on the one-faulty node scenario (Fig. 8), provisioning 2 cells through each path ($n_{cells} = 2$). The single path solution is very unfair (Fig. 8a): all the flows which were forwarded by the crashed node exhibit a null PDR. On the contrary, the multipath solutions (Disjoint and Braided Paths) achieve a much higher fairness. However, long flows keep on presenting a lower end-to-end reliability with these harsh conditions, and the Jain Index is not equal to 1.

When considering the Packet Delivery Ratio (Fig. 8b), we can note that Braided Path achieves a higher reliability. In particular, exploiting two disjoint paths does not allow mutualization: all the cells assigned to the faulty node are wasted, and cannot be used by the secondary path. Still, we don't achieve a 100% packet delivery ratio for braided paths since a single node (3) has to forward all the flows, and the number of transmission opportunities is not sufficient to achieve alone a perfect delivery. However, the reliability seems reasonable: the system is in degraded mode. It delivers one part of the packets, until the novel schedule is computed and disseminated.

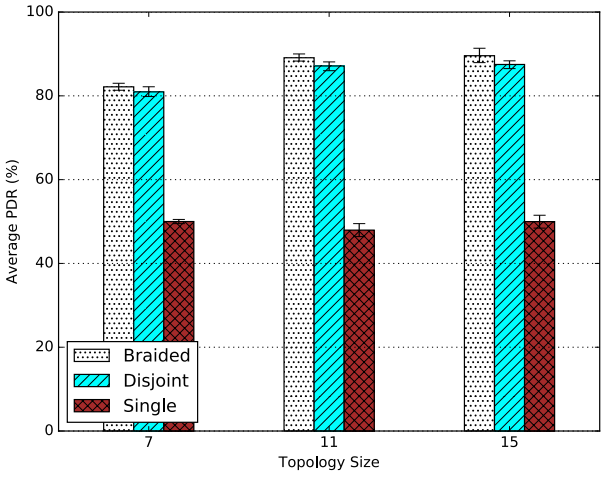
A larger topology means a longer delay when a node crashes: more packets are present in the queue, and the forwarding delay increases when the nodes are farther from the sink (Fig. 8c). Disjoint paths keep on presenting a lower delay: all the flows which don't pass through the node 2 keep on presenting a lower delay.

4.4.2. Two Nodes (ids 2 and 5)

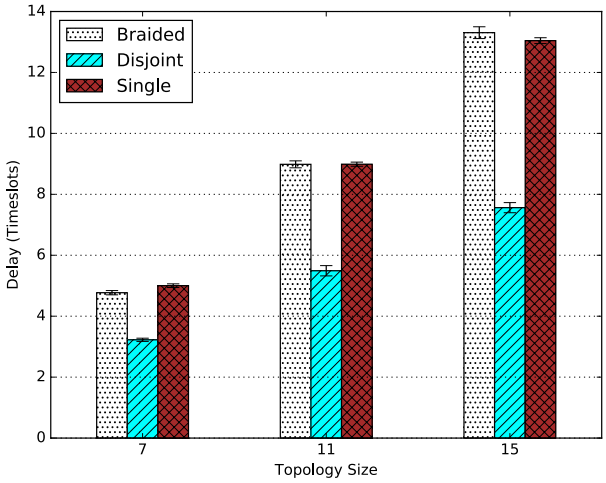
Finally, we consider here two consecutive faults (Fig. 9). We represent in Figure 9a the instantaneous packet delivery ratio with the 15-nodes topology. We can clearly identify the two events (at 72s and 216s). While the disjoint path solution accommodates quite well one fault, the PDR falls very significantly with two faulty nodes. Indeed, the network is not able to recombine two different paths: if each path is broken at different places, the network cannot recover. Thus, if two faults occur, on the two different paths, the systems stops delivering correctly the packets. On the contrary, braided paths allow the network to exploit



(a) Jain Index.



(b) Average PDR.



(c) Average delay.

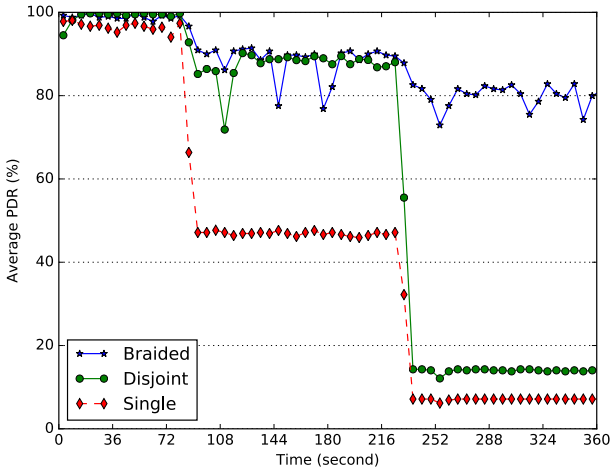
Figure 8: Fault-tolerance evaluation (one node is turned off) ($n_{cells} = 2$, number of transmission opportunities for each device per path).

all the possible subpath combinations, achieving a much higher reliability. It succeeds to deliver still almost 80% of the packets in very harsh conditions.

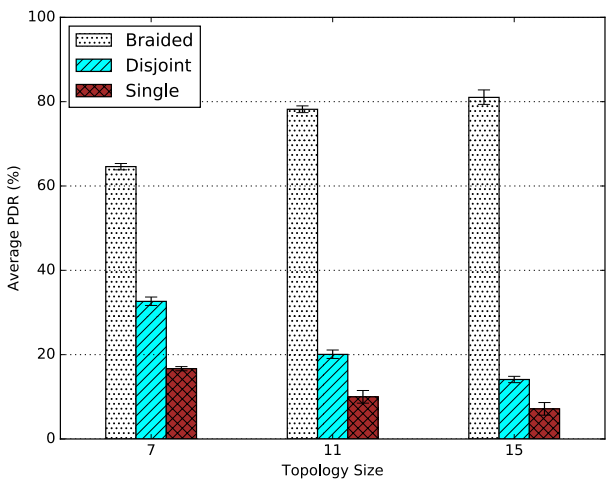
We make the same conclusions when considering the other topologies (Fig. 9b). Single and disjoint path strategies perform very bad when two faults occur in the network. With braided paths, we are much robust, allowing each device to report its measurements (and possibly its novel radio environment characteristics) to the controller, before the network is reconfigured.

Fairness is very low for SP and DPs (Fig. 9c): some flows which pass through the crashed nodes do not succeed to delivery any of their packets, providing a very high unfairness. On the contrary, braided path succeeds to still provide a very good fairness, even with 15 devices.

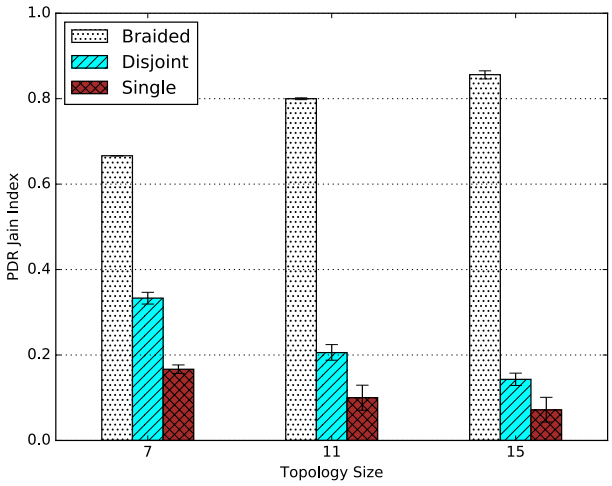
Finally, we stress out our solution, provisioning here only two cells, i.e. one per path (Fig. 10). As expected, Disjoint and Single paths solutions provide a very bad Packet Delivery Ratio as soon as a fault occurs in the network. Even without fault, disjoint paths provide a lower reliability, since each flow is handled independently, without any retransmission opportunity. Braided paths provides a 60% packet delivery ratio even with 2 faulty nodes. The cells are interleaved and efficiently mutualized, allowing more retransmissions. Obviously, the reliability is not perfect (i.e. 100%) since we have lossy links.



(a) Variations of the Packet Delivery ratio for a 15 nodes topology.



(b) Average PDR(two nodes crash).



(c) Jain Index.

Figure 9: Fault-tolerance evaluation – two nodes are turned off sequentially.

4.5. Schedule Compactness

Finally, we measure the number of cells allocated for the three algorithms (Fig. 11). Indeed, the reinforcement of reliability should not be too expensive in the schedule. In particular, more cells in the schedule means also that the energy consumption is increased the devices have to stay awake longer.

Single and Disjoint Paths solutions allocate the same number of cells. Indeed, we allocate the same number of cells: when allocating retransmission cells for a single path, this is similar to allocating half of the cells for each disjoint paths. We also handle the braided paths case in the same way, allocating half of the transmission cell for each possible next hop. We can note that our scheduling algorithm (BraidedPaths) is very scalable: it uses the same number of cells, while authorizing each intermediary hop to use its two next hops. Thus, the size of the schedule is the same for all the (multi)-path solutions, denoting the same schedule fingerprint.

We can note that using disjoint paths increases the energy consumption: the packet is replicated by the source, whatever the conditions are. Thus, **all** the cells are used, even with almost perfect links. On the contrary, braided and single paths solutions provision only retransmission cells, and are used only when the primary next hop fails. The energy consumption of these idle cells is much smaller [36], and the energy

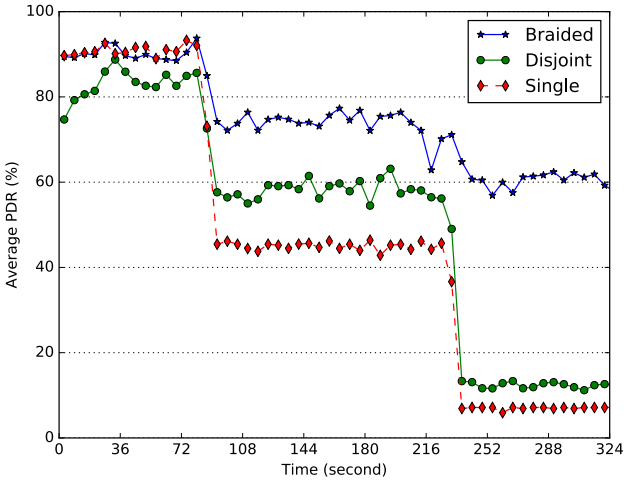


Figure 10: Variations of the Packet Delivery ratio for a 15 nodes topology with only 2 provisioned cells ($n_{cells} = 1$).

efficiency of single and braided paths is consequently higher.

5. Conclusion and Perspectives

We propose here a scheduling algorithm tailored for multipath in IEEE802.15.4-TSCH networks. To improve the reliability and the fault-tolerance, each node maintains two different next hops toward the border router. Our scheduling algorithm mutualizes the same cell when different transmitters may send their packet to the same receiver. By properly implementing traffic isolation at the link level, we are still able to guarantee a collision-free schedule. Our performance evaluation highlights the relevance of our solution to provide high-reliability and fault tolerance. In particular, the different flows keep on presenting a high Packet Delivery Ratio, even when several routers are turned off, simulating a crash.

In the future, we plan to address a heterogeneous scenario, where the number of next hops depends on the Packet Delivery Ratio of each radio link. Typically, two next hops may be sufficient for very high link qualities, while a larger number of next hops may be relevant for medium or bad links. We also plan to investigate how label switching such as G-MPLS like approaches may help to reduce the schedule length. Indeed, we would remove the *empty queue* constraint, since the cells can be associated with a specific flow label. This way, we can still forbid collisions while reducing the schedule length. Finally, we aim to combine our standard-compliant solution with a schedule exploiting anycast. By allocating several receivers to the same cell, we may increase the delivery probability for a single transmission. By allocating the same cells to different receivers and transmitters, we expect to increase the reliability and the fault-tolerance for no cost.

Acknowledgements

We thank Dr. Tengfei Chang for his valuable assistance in resolving problems related to the OpenWSN simulator. We really appreciated his willingness to give his time so generously.

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

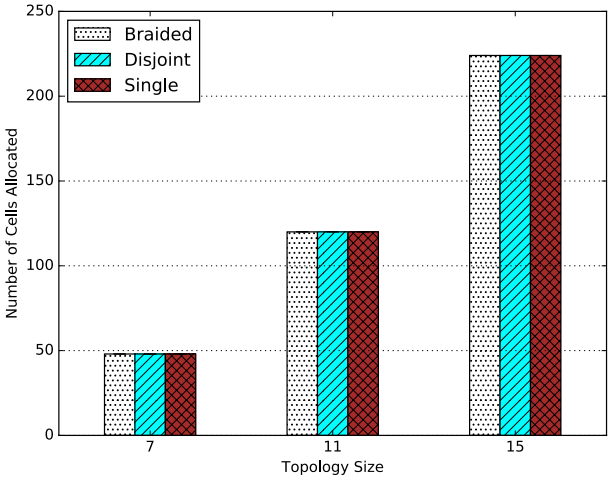


Figure 11: Number of cells allocated by the scheduling algorithms.

References

References

[1] M. Wollschlaeger, T. Sauter, J. Jasperneite, The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0, IEEE Industrial Electronics Magazine 11 (1) (2017) 17–27. doi:10.1109/MIE.2017.2649104.

[2] G. Chopra, R. K. Jha, S. Jain, A survey on ultra-dense network and emerging technologies: Security challenges and possible solutions, Journal of Network and Computer Applications 95 (2017) 54 – 78. doi:10.1016/j.jnca.2017.07.007.

[3] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, Y. Chen, Real-time wireless sensor-actuator networks for industrial cyber-physical systems, Proceedings of the IEEE 104 (5) (2016) 1013–1024. doi:10.1109/JPROC.2015.2497161.

[4] G. Zhao, M. A. Imran, Z. Pang, Z. Chen, L. Li, Toward real-time control in future wireless networks: Communication-control co-design, IEEE Communications Magazine 57 (2) (2019) 138–144. doi:10.1109/MCOM.2018.1800163.

[5] R. Teles Hermeto, A. Gallais, F. Theoleyre, Scheduling for IEEE802.15.4-TSCH and Slow Channel Hopping MAC in Low Power Industrial Wireless Networks, Comput. Commun. 114 (C) (2017) 84–105. doi:10.1016/j.comcom.2017.10.004.

[6] T. Chang, M. Vucinic, X. Vilajosana, 6TiSCH Minimal Scheduling Function (MSF) , draft, IETF, https://tools.ietf.org/html/draft-ietf-6tisch-msf-02 (2019).

[7] T. Winter, el at., RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550, IETF (March 2012). doi:10.17487/RFC6550.

[8] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, T. Engel, On-the-fly bandwidth reservation for 6tisch wireless industrial networks, IEEE Sensors Journal 16 (2) (2016) 550–560. doi:10.1109/JSEN.2015.2480886.

[9] M. Z. Hasan, H. Al-Rizzo, F. Al-Turjman, A Survey on Multipath Routing Protocols for QoS Assurances in Real-Time Wireless Multimedia Sensor Networks, IEEE Communications Surveys Tutorials 19 (3) (2017) 1424–1456. doi:10.1109/COMST.2017.2661201.

[10] R. Koutsiamanis, G. Z. Papadopoulos, X. Fafoutis, J. M. D. Fiore, P. Thubert, N. Montavont, From best effort to deterministic packet delivery for wireless industrial iot networks, IEEE Transactions on Industrial Informatics 14 (10) (2018) 4468–4480. doi:10.1109/TII.2018.2856884.

[11] I. Hosni, F. Théoleyre, Adaptive k-cast Scheduling for High-Reliability and Low-Latency in IEEE802.15.4-TSCH, in: International Conference on Ad Hoc Networks and Wireless (ADHOCNOW), 2018, pp. 3–14. doi:10.1007/978-3-030-00247-3_1.

[12] IEEE Standard for Low-Rate Wireless Networks, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), Tech. rep., IEEE (April 2016). doi:10.1109/IEEESTD.2016.7460875.

[13] D. V. Queiroz, M. S. Alencar, R. D. Gomes, I. E. Fonseca, C. Benavente-Peces, Survey and systematic mapping of industrial wireless sensor networks, Journal of Network and Computer Applications 97 (2017) 96 – 125. doi:10.1016/j.jnca.2017.08.019.

[14] T. Watteyne, A. Mehta, K. Pister, Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense, in: Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN), ACM, 2009, pp. 116–123. doi:10.1145/1641876.1641898.

[15] D. Dujovne, T. Watteyne, X. Vilajosana, P. Thubert, 6TiSCH: deterministic IP-enabled industrial internet (of things), IEEE Communications Magazine 52 (12) (2014) 36–41. doi:10.1109/MCOM.2014.6979984.
[16] S. Kharb, A. Singhrova, Fuzzy based priority aware scheduling technique for dense industrial iot networks, Journal of Network and Computer Applications 125 (2019) 17 – 27. doi:10.1016/j.jnca.2018.10.004.
[17] D. Dujovne, L. Grieco, M. R. Palattella, N. Accettura, 6TiSCH Experimental Scheduling Function (SFX), Internet-draft, IETF, draft-ietf-6tisch-6top-sfx-01 (March 2018).
[18] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, T. Engel, On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH, IEEE Sensors Journal 13 (10) (2013) 3655–3666. doi:10.1109/JSEN.2013.2266417.
[19] G. Gaillard, D. Barthel, F. Theoleyre, F. Valois, Kausa: KPI-aware Scheduling Algorithm for Multi-flow in Multi-hop IoT Networks, in: International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW), Vol. 9724, 2016, pp. 47–61. doi:10.1007/978-3-319-40509-4_4.
[20] F. Theoleyre, G. Z. Papadopoulos, 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), ACM, 2016, pp. 102–110. doi:10.1145/2988287.2989133.
[21] F. Dobslaw, T. Zhang, M. Gidlund, End-to-End Reliability-aware Scheduling for Wireless Sensor Networks, IEEE Transactions on Industrial Informatics 12 (2) (2014) 758–767. doi:10.1109/TII.2014.2382335.
[22] M. Hashimoto, N. Wakamiya, M. Murata, Y. Kawamoto, K. Fukui, End-to-end reliability- and delay-aware scheduling with slot sharing for wireless sensor networks, in: International Conference on Communication Systems and Networks (COMSNETS), 2016, pp. 1–8. doi:10.1109/COMSNETS.2016.7439984.
[23] J.-H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, IEEE/ACM Transactions on Networking 12 (4) (2004) 609–619. doi:10.1109/TNET.2004.833122.
[24] O. Iova, F. Theoleyre, T. Noel, Improving the network lifetime with energy-balancing routing: Application to rpl, in: 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), 2014, pp. 1–8. doi:10.1109/WMNC.2014.6878864.
[25] P. Thubert, Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL), RFC 6552, IETF (March 2012). doi:10.17487/RFC6552.
[26] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, Wirel. Netw. 11 (4) (2005) 419–434. doi:10.1007/s11276-005-1766-z.
[27] O. Iova, F. Theoleyre, T. Watteyne, T. Noel, The love-hate relationship between ieee 802.15.4 and rpl, IEEE Communications Magazine 55 (1) (2017) 188–194. doi:10.1109/MCOM.2016.1300687RP.
[28] T. Istomin, C. Kiraly, G. P. Picco, Is RPL Ready for Actuation? A Comparative Evaluation in a Smart City Scenario, in: European Conference on Wireless Sensor Networks (EWSN), ACM, 2015, pp. 291–299. doi:10.1007/978-3-319-15582-1_22.
[29] J. W. Guck, M. Reisslein, W. Kellerer, Function split between delay-constrained routing and resource allocation for centrally managed qos in industrial networks, IEEE Transactions on Industrial Informatics 12 (6) (2016) 2050–2061. doi:10.1109/TII.2016.2592481.
[30] O. Iova, F. Theoleyre, T. Noel, Using multiparent routing in RPL to increase the stability and the lifetime of the network, Ad Hoc Networks 29 (2015) 45 – 62. doi:10.1016/j.adhoc.2015.01.020.
[31] M. Nassiri, M. Boujari, S. V. Azhari, Energy-aware and load-balanced parent selection in rpl routing for wireless sensor networks, International Journal of Wireless and Mobile Computing 9 (3) (2015) 231–239. doi:10.1504/IJWMC.2015.073105.
[32] K. Yu, M. Gidlund, J. Åkerberg, M. Björkman, Performance evaluations and measurements of the realflow routing protocol in wireless industrial networks, IEEE Transactions on Industrial Informatics 13 (3) (2017) 1410–1420. doi:10.1109/TII.2016.2587842.
[33] J. Shi, M. Sha, Z. Yang, Digs: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks, in: International Conference on Distributed Computing Systems (ICDCS), IEEE, 2018, pp. 354–364. doi:10.1109/ICDCS.2018.00043.
[34] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, K. Pister, Openwsn: a standards-based low-power wireless development environment, Transactions on Emerging Telecommunications Technologies 23 (5) (2012) 480–493. doi:10.1002/ett.2558.
[35] R. Jain, W. Hawe, D. Chiu, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, Research Report arXiv:cs/9809099, DEC, https://arxiv.org/abs/cs/9809099 (1984).
[36] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, K. S. J. Pister, A realistic energy consumption model for tsch networks, IEEE Sensors Journal 14 (2) (2014) 482–489. doi:10.1109/JSEN.2013.2285411.

Erfan Mozaffari Ahrar is a research assistant in the Networking Research Laboratory at the Computer Department of Bu-Ali Sina University. He received his BSc and MSc degrees in Computer Networking both from Bu-Ali Sina University in 2013 and 2016, respectively. He focuses on the research of Wireless Sensor Networks and the Industrial Internet of Things.

Mohammad Nassiri is an assistant professor and the head of Networking Research Laboratory at the Computer Department of Bu-Ali Sina University. He received his BSc and MSc degrees from Iran University of Science and Technology (IUST) and Sharif University of Technology, respectively, in 2000 and 2002. He also received his PhD in Computer Engineering from Grenoble INP, France, in 2008. His research interests mainly concern experimental design for High Throughput Wireless LANs and MAC improvement for Industrial Internet of things.

Fabrice Théoleyre is a researcher at the CNRS. After having spent 2 years in the Grenoble Informatics Laboratory (France), he is part of the ICube lab (Strasbourg, France) since 2009. He received his PhD in computer science from INSA, Lyon (France) in 2006. He was a visiting scholar at the University of Waterloo (Canada) in 2006, and a visiting researcher at INRIA Sophia Antipolis (France) in 2005. He serves in about ten TPC per year, and is area editor for Ad Hoc Networks since 2018. He has been associate editor for IEEE Communications Letters and guest editor for Computer Communications and Eurasip JWCN. His research interests mainly concern distributed algorithms and experimental design for the Internet of Things.