Hindawi
Journal of Sensors
Volume 2023, Article ID 9910809, 16 pages
https://doi.org/10.1155/2023/9910809

Research Article

An Efficient Anycast Mechanism for 802.15.4-TSCH to Improve QoS in IIoT

Sahand Amiri,[1,2] Mohammad Nassiri,[1,2] Reza Mohammadi,[1,2] and Fabrice Theoleyre[3]

[1]*Computer Engineering Department, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran*
[2]*Center for International Scientific Studies & Collaboration (CISSC) Ministry of Science, Research and Technology, Tehran, Iran*
[3]*ICube, CNRS/University of Strasbourg, Strasbourg, France*

Correspondence should be addressed to Reza Mohammadi; r.mohammadi@basu.ac.ir

Received 3 January 2023; Revised 24 April 2023; Accepted 19 September 2023; Published 21 October 2023

Academic Editor: Rajkishor Kumar

The Industrial Internet of Things (IIoT) has emerged as a technology that automates industrial processes. In IIoT networks, data are collected from various nodes and sent to a base station for managerial purposes. However, in the industrial environment, network reliability and delay are significant challenges due to the high likelihood of packet loss in radio networks. Anycast is a link layer mechanism that increases reliability and reduces delay by allowing multiple receivers to be connected to a sender, and a single packet is simultaneously sent to all receivers. The receivers decode the packet based on their priorities, and transmission succeeds if at least one receiver can decode the packet. Moreover, mechanisms exist to limit the number of duplicates. This paper proposes a novel centralized anycast aware scheduling algorithm (AASA), which implements anycast based on the 802.15.4e-time-slotted channel hopping (TSCH) standard and in the stack of the 6TiSCH protocol. The goal of AASA is to improve IIoT networks more reliable and reduce end-to-end delay. To do this, upon a link failure, AASA chooses an alternative link and the packet is sent without any delay via that link through the same time slot. We implemented AASA in 6TiSCH simulator and carried out different scenarios to investigate its efficiency under various conditions. Results from simulations show that AASA effectively increases reliability by reducing repetitive packet transmissions and, thus, decreasing the delay in packet delivery.

1. Introduction

Industrial Internet of Things (IIoT) or Industry 4.0 is an industrial revolution based on the Internet of Things (IoT) to make production lines more scalable and adaptable [1]. These networks are composed of a large collection of wireless nodes; each node consists of a processor, a power supply (battery), a radio (for communications and data exchange), and several physical sensors (temperature, pressure, humidity, place, and so on) [2]. IIoT networks are used for specific purposes such as environmental surveillance, target tracking, or alarm systems in an industrial environment. These networks can remarkably improve connectivity, productivity, scalability, time-saving, and cost-effectiveness in industrial organizations [3].

The major challenges in the IIoT include delay reduction, power consumption, reliability, and the lifespan of nodes [4]. Unfortunately, in most cases, radio links are not reliable. For instance, interference between the radio channels may lead to the loss of the packets transmitted over a link [5]. There is a need for developing a mechanism for increasing reliability and reducing delay and energy consumption. To this end, we should maximize sleeping time to reduce energy consumption. To overcome these limitations, the time-slotted channel hopping (TSCH) MAC behavioral mode of 802.15.4e was proposed. TSCH is based on time-slotted mechanism, where nodes in 802.15.4e-TSCH networks communicate by following a time division multiple access (TDMA) [6]. The medium access control (MAC) layer controls the node's radio for a fair access to a shared channel [7]. Therefore, the MAC sublayer requires an appropriate sleep/awake mechanism. What is important in these networks is end-to-end reliability, and adequate resources should be allocated all along the path between the sender and the receiver while considering the issue of power consumption. This issue can be managed by multipath routing in the routing layer [8]. To this end,

separate paths should be created between the sender and the receiver to make the network more fault tolerant and increase its reliability. Thus, the packet is lost only if all the links between the sender and the receiver failed.

In most IIoT networks, scheduling helps to implement reliable transmissions. A schedule is defined by the time slot and channel on which a node should transmit/receive data to/from its neighbors. The basic unit of bandwidth scheduled is called cell. In the scheduling matrix, each node is awake in a specific time interval in which it is either sending a packet or waiting to receive a packet. To improve reliability, multiple paths may be provisioned between a source and a destination. However, it also means that some extra time has to be reserved in the scheduling matrix for these redundant paths [9], which may be used only if a failure occurs. Alternatively, additional cells may be provisioned for the retransmissions to increase the network's reliability [10, 11]. In this case, the awake time of nodes increases further, as well as the network's required resources and power consumption.

The 802.15.4e-TSCH standard explains how the MAC layer executes a schedule, but it does not specify how such a schedule is built [12]. IETF 6TiSCH (Ipv6 over TSCH) working group defined a sublayer built over TSCH called 6top (6TiSCH operational sublayer), which enables scheduling in TSCH networks [13]. This paper investigates the anycast mechanism in IIoT and presents a centralized topology and anycast-aware algorithm (with two and three receivers) based on the 6TiSCH stack. In particular, we assign several receivers to the same cell in the scheduling matrix to maximize sleeping time. This minimizes the number of retransmissions. Besides, it also reduces the delay and increases the reliability, thus making the network fault tolerant. Moreover, we improve the network lifetime compared with multipath and retransmission cells. In summary, the contribution of this paper is as follows:

(1) We propose a centralized topology-aware scheduling method, where the nodes are scheduled from the leaves to the root to reduce the end-to-end delay.
(2) We select (up to two) additional parents for any node. We consider the number of children of other nodes to balance the energy consumption and the link reliability to the possible parents to also optimize the reliability.
(3) We propose an implementation of the anycast mechanism with up to two and three parents on the scheduling algorithm. In particular, acknowledgments are used to remove possible duplicates.
(4) We evaluate our scheduling algorithm through simulations.

The rest of this paper is organized as follows. Section 2 describes some background on 802.15.4e-TSCH, 6TiSCH, and reviews some recent related works. The proposed solution is presented in Section 3. A comprehensive performance evaluation is conducted in Section 4. Finally, a conclusion is given in Section 5.

2. Background and Related Works

This section briefly details the behavior of the 802.15.4-TSCH and 6TiSCH standards.

2.1. An Overview of 802.15.4-TSCH and 6TiSCH. The 802.15.4 standard proposed the TSCH mode enable scheduled access in multihop topologies. TSCH is a combination of time synchronization, frequency division, and channel hopping. The time is divided into slot frames or superframes that are repeated cyclically over time. Each slot frame is divided into a specific number of equal slots and iterates periodically. The duration of a time slot is equal to the duration of the transmission of a data packet and its possible acknowledgment. All nodes in the network share this time structure.

Obviously, all the nodes must be synchronized, so that they wake up and sleep simultaneously. Fortunately, any transmission can be used in TSCH for resynchronization since time offsets are fixed. Thus, it is sufficient to compute the time difference between the actual and expected times of arrival. All the nodes maintain an absolute sequence number (ASN) that counts the number of time slots since the network bootstrapped.

TSCH supports also channel hopping to exploit 16 orthogonal channels. More precisely, a cell is defined in the scheduling matrix by a time slot and a channel offset. This channel offset is then derived into a frequency at the beginning of the time slot according to the Equation (1) [10]:

$$f = ((\text{ASN} + \text{Channel offset}) \% N_{\text{Channels}}), \quad (1)$$

where ASN denotes the absolute slot number, channel offset denotes the channel number allocated to each communication, and N_{channel} is the total number of usable channels.

As aforementioned, this standard defines two types of cells:

(1) Dedicated cells (colored cells in Figure 1): There is no contention avoidance mechanism, and the transmission is triggered without any waiting time. The clear channel assessment (CCA) may be used by the sender only to detect external interference; therefore, these cells should be allocated to noninterfering nodes.
(2) Shared cells: Random access is implemented to solve contention. However, the backoff is not implemented *inside* a cell but rather *among* cells. More precisely, a sender which doesn't receive an acknowledgment assumes that a collision occurs. It chooses a random backoff value, and skips the corresponding number of shared cells before triggering retransmission. This mechanism is expensive (increasing the delay), but is required to use any transmission for the resynchronization.

6TiSCH aims to implement the protocols to execute a low-power network on top of the protocol stack [11]. The 6TiSCH stack exploits IPv6 routing protocol for low-power and lossy networks (RPL) in multihop networks for routing [14]. It also

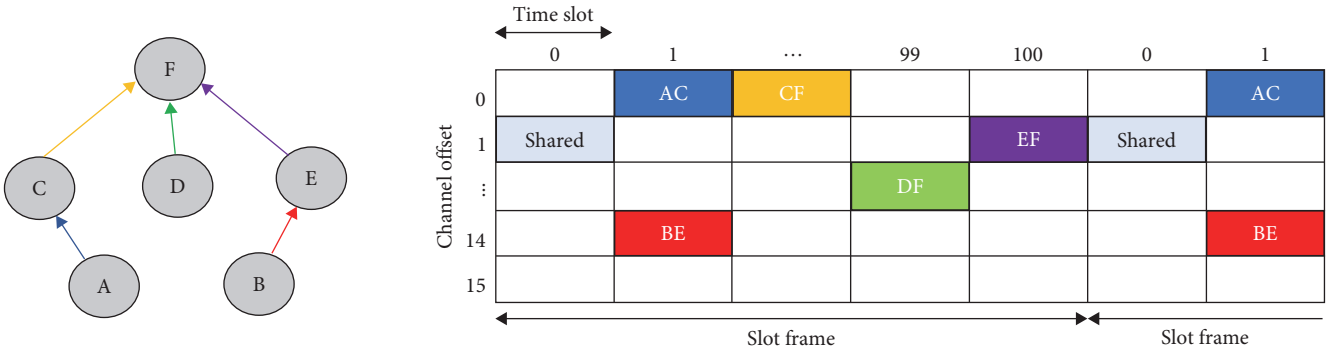

Figure 1: A sample TSCH scheduling matrix.

defines the 6P protocol to change dynamically the schedule. 6P supports both end-to-end instructions from a controller and autonomous local negotiations between any pair of nodes [15]. 6TiSCH defines a minimal configuration (6TiSCH minimal) based on which all the nodes stay simultaneously awake in the shared cells. These cells are used by default to transmit control packets such as enhanced beacon (EB) or DODAG information object (DIO) [16].

Many centralized and distributed algorithms have so far been proposed to overcome the challenges of IIoT concerning the stack of the 6TiSCH protocol [17]. Centralized algorithms need to be aware of the network topology, link quality, and the traffic produced by every node. These algorithms are dependent on a central controller that runs the scheduling algorithm and informs all the nodes about it. Palattella et al. [12] developed a centralized traffic-aware scheduling algorithm called TASA, which solves scheduling problems using the graph theory. In contrast to distributed algorithms, the scheduling algorithm is executed locally between each two nodes. For instance, 6top scheduling function (SFX) is a default distributed algorithm in 6TiSCH that keeps the number of cells at least equal to the number of the sent packets and relies on the hysteresis function to limit the number of retransmissions [18].

2.2. Anycast Transmissions. Transmission of data packets over wireless networks is inherently a broadcast transmission, so that all the neighboring nodes that are listening to the network may capture and decode the transmission. In TSCH, a scheduling matrix is defined detailing the cells assigned to each link. More precisely, a link consists of a pair of sender/receiver, using unicast transmissions.

Lampin et al. [19] proposed an opportunistic method of packet forwarding based on RPL to use long-range low-reliability radio links. The propose is to assign a transmission to multiple receivers instead of a single one. However, their proposition was not tailored for TSCH, with its strict timing property. The transmissions could be scheduled simultaneously within a single time slot if they have the same transmitter. Also, the existing opportunistic schemes for ad hoc networks, such as ExOR [20], rely on adding a list of potential forwarders and their associated priorities in each data packet. Since the IEEE 802.15.4 standard provides a maximum transmission unit (MTU) of 127 bytes, ExOR reduces significantly the part reserved for the payload.

Figure 2 depicts a scheduling matrix. As shown in Figure 2, node A has two parents, C and D, and node B has also two parents, E and D. With anycast, a node can use a single time slot to send its packet to its two parents at the same time. As shown in Figure 2, there are three types of cells:

(1) Anycast cells correspond to anycast transmissions, that is, transmissions that go from one sender to multiple receivers.
(2) Dedicated cells are for one-to-one transmissions, that is, a single receiver sends data specifically to one receiver (identical to what is described in 802.15.4e).
(3) Shared cells are allocated for control packets, typically transmitted in broadcast.

Huynh et al. [21] indicated the efficiency of anycast scheduling in the reduction of energy consumption without compromising reliability. They demonstrate that there exists an optimal scheduling policy in which the same reliability is obtained with a reduction in energy consumption and delay if anycast is used at the link layer. Hosni and Theoleyre [22] developed a routing solution in which k different receivers can be chosen according to the link packet delivery ratio with the sender (k-cast). The solution, however, was evaluated by simulation in a way that the probability of packet loss was independent for all links. Recently, Hermeto et al. [23] examined the performance of anycast by means of an experimental dataset (packet loss/reception for a set of multipoint links). They simulated a wireless network through replaying test datasets. Finally, duocast (i.e., anycast with two receivers) has been implemented in a 6TiSCH stack [24].

To the best of our knowledge, the anycast mechanism has not so far been implemented on centralized scheduling algorithms with the aim of enhancing reliability and reducing the delay.

3. Problem Statement and Proposed Solution

As shown in Figure 2, in the anycast mechanism, a sender transmits its data to two receivers in the same time slot. The

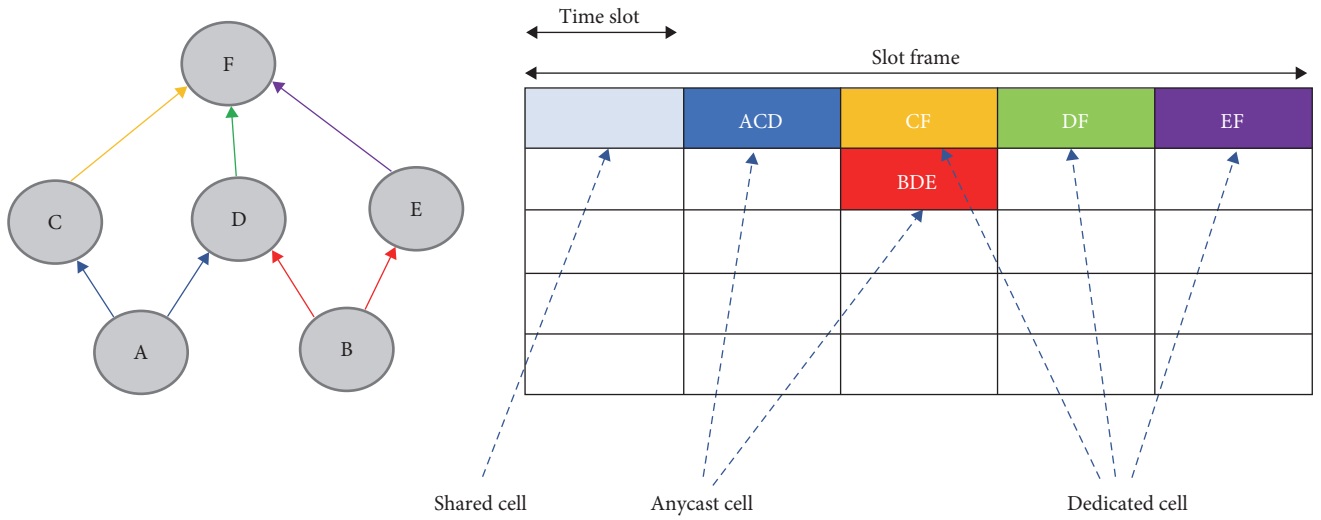

Figure 2: A sample TSCH scheduling matrix containing anycast cells.

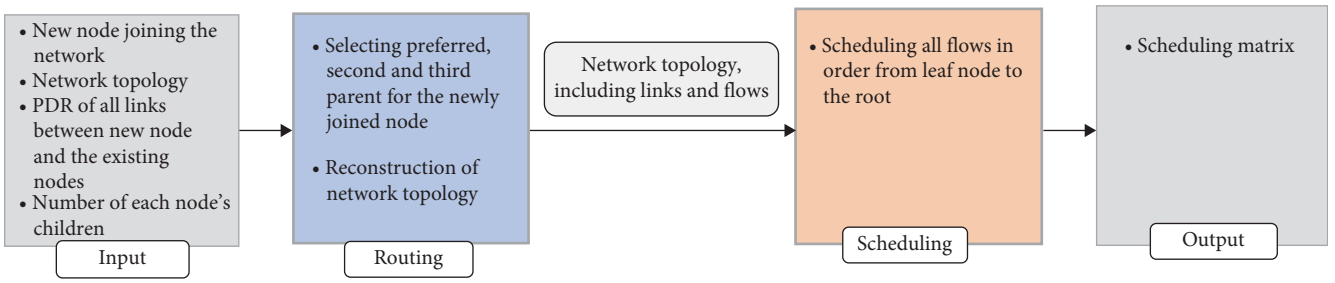

Figure 3: Workflow of creating scheduling matrix in the proposed solution.

priority of transmission is given to the preferred parent and if acknowledgment is not received from the preferred parent, this will be recognized by the next parent or parents via the CCA mechanism and they forward the packet. As shown in Figure 2, for example, let's suppose that, for node A, the preferred parent is C and the secondary parent is D. We may face the following situations:

(1) C receives the packet initially: It forward the packet without considering node D and D removes the packet after receiving it.
(2) C does not receive the packet initially: D which has already received the packet discovers via the CCA mechanism that C has not sent acknowledgment to A and then forward the packet.
(3) D receives the packet initially: It waits to see if C receives the packet or not. If C receives the packet, D will remove it; otherwise, it will forward the packet.
(4) None of the parents receives the packet: A retransmission is triggered.

3.1. Description of the Approach. In this section, we describe our proposed anycast mechanism specifically with two and three parents on a network with tree topology. The proposed solution uses centralized scheduling, so that, for each node that initially connects to the network, the central controller an ordered list of parents (based on the packet delivery ratio (PDR) of links attached to it, as well as the number of the children of other nodes). Next, the network topology (including all the links along with the nodes and their parents) is given to the scheduling algorithm and the scheduling matrix is created using the existing information for this network. The procedure is shown in Figure 3. According to this figure, our proposed method encompasses four input parameters. These include new node joining the network, network topology, PDR of all links between the new node and existing nodes, as well as the number of children associated with each node. The aforementioned inputs are then subjected to Algorithms 1 and 2, which subsequently facilitate the operation of routing and selection of new parents for newly added nodes within the network (as depicted in the second box). Following this, the network undergoes a transformation resulting in a new topology, including the necessary links and flows. This newly transformed network is then utilized by the scheduling algorithm—Algorithm 3—in order to schedule all flows accordingly. Upon completion of this process, the output is a correctly scheduled algorithm containing all relevant links.

Our selected topology is similar to the directed acyclic graph (DAG), as shown in Figure 2, but it is worth mentioning that our proposed method can be applied to any type of topologies and it is not limited to only DAG topologies. We assume

```
Inputs:
    linksPDR: List of all links with their PDR
    nodes: List of all nodes in topology
Outputs:
    parentList: List of three parents, respectively
Initialization:
    parentsList ← ()
    maxChild ← 2 ▷ maximum number of child a parent can have
    maxParents ← 3 ▷ maximum number of parent a node can have
sortedParentsByPDR ← sortParentByPDR (nodes, linksPDR)
▷ descending sort of all parents by PDR joining node can have
for parent ∈ sortedParentsByPDR do
    if numChild(parent) < maxChild then
        parentsList.insert(parent)
        if parentList.size() = maxParents then
            break
        end if
    end if
End if
```

Algorithm 1: Parent selection in the proposed method.

```
Inputs:
    links: List of all links in the topology
    nodes: List of all nodes in topology
Outputs:
    routesList: List of all routes to root, respectively
routesList ← ()
for node ∈ nodes do
    route ← ()
    depth ← getDepth(node)
    route.insert(node)
    for i ← 1,depth do
        parent ← route(-1)
        node ←getParent(parent)
        route.insert(node)
    end for
    routeList.insert(route)
End for
Return sort(routeList)
```

Algorithm 2: Route list creation in the proposed method.

that the controller has to allocate the transmissions for a convergecast traffic pattern: the network traffic flows from all the nodes toward the root node (node F, as shown in Figure 2).

3.2. Parent Selection Algorithm. In 6TiSCH networks, any node that is not yet connected to the network waits for EB from other nodes. After receiving the EB, it sends a RPL connection request (DIS) to the sender node and, on the other side, the sender node sends a DIO message to the requesting node. The DIO message contains the rank of each node along with information about the DODAG. In general, two methods of parent selection are defined in 6TiSCH:

(1) Using the rank of other nodes: The new node selects as parent the neighbor with the lowest rank. The new node then computes its own rank based on the DIO

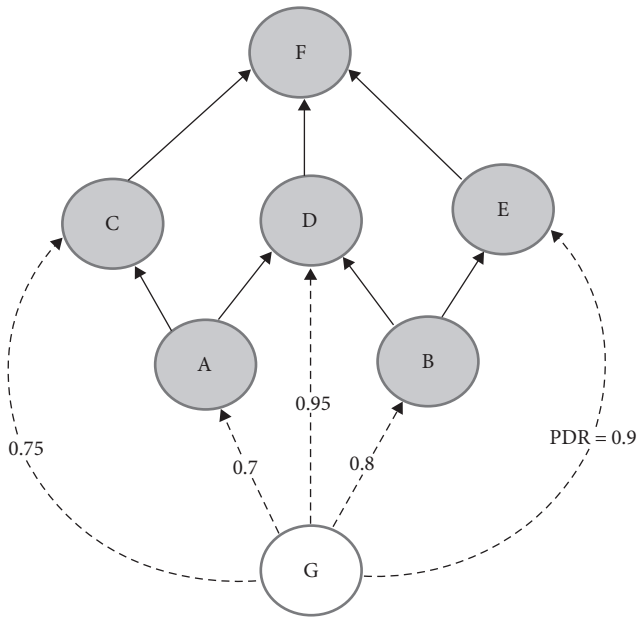

Figure 4: Links and their PDR values between node G and other nodes.

of the parent and the objective function. The next DIO will advertise its novel rank [14].

(2) Using the PDR of links that are connected to the new node: In this method, a node measures the average PDR with all its already associated neighbors. Then, the node selects parents that have link PDR greater than ACCEPTABLE_LOWEST_PDR value. This value can be predefined by an administrator [11].

In fact, after the implementation and evaluation of both methods, we concluded that, under identical circumstances and in similar topologies, the second method would have a better performance in terms of delay and reliability. Therefore, the second method is used to select parents in this study. But we made a few changes in this method. In 6TiSCH networks, when the topology is formed and the coordinates of each node are determined, the RSSI value is calculated based on the distance of the nodes, and then the initial PDR value (calculated in the Connectivity.py file in the 6TiSCH simulator) is calculated based on the RSSI. This PDR is updated during the lifetime of the network and is not fixed, so parents change during the lifetime of the network (exactly when only rank is used to select the parent). The selection of parents is described in Algorithm 1. For any node that connects to the network, the PDR of all its links with its neighbors is calculated and listed in order from highest to lowest (line 4). Next, nodes are selected for the parents from the sorted list whose children are not more than or equal to 2 (in order to prevent the formation of queues in the node's memory; lines 6–11).

In the default 6TiSCH parent selection method, only the parent whose communication link with the child node has the highest PDR is selected. If the same process is used in parent selection (when the anycast mechanism is used), an intermediate node may be selected as a parent for multiple nodes, and another node may not be selected as a parent for any node.

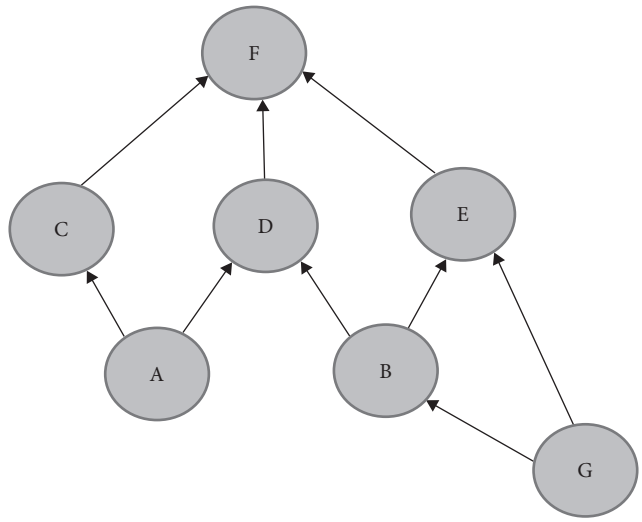

Figure 5: Final topology after the joining of node G.

The improvement made in the presented parent selection algorithm also considers the number of children when selecting the parent, and when the number of children of a node exceeds a certain value (mentioned as 2), that node is not selected as parent and the next node in the sorted list of parents based on PDR is checked. This value can be changed by the administrator, but to improve network performance, this value should be larger in larger networks and smaller in smaller networks.

To make it clearer, we illustrate the process of parent selection for the topology, as shown in Figure 4. In this figure, values on edge show PDR and directed edge shows parent–child relationship nodes and here, for simplicity and to prevent network complexity, we assume that the maximum number of selected parents and children is 2. In this figure, C and D are the first and second parents of node A, and the new node G seeks to connect to the network and select two parents for itself. As shown in Figure 4, each edge is labeled by its corresponding PDR. In the parent selection phase, all the nodes are put into a sorted list named SortedParentsByPDR = [D, E, B,C, A] according to the PDR of the links between them and the new node. Then, node G starts the parent selection process from the list SortedParentsByPDR based on the number of children of each node. The number of children is included both in the DIO message (only in the packet responded to the DIS (DODAG information solicitation) that the joining node sends) and in the destination advertisement object (DAO) message. In DIO, the joining node knows the number of children of the parent and in DAO, the root node can know the structure of the network.

As shown in Figure 4. D has two children; it is not accepted as the first parent. Instead, E and B are taken as the first and second parents, respectively. Finally, the network's topology takes the form of what is displayed in Figure 5.

```
1:  Inputs:
        routesList: List of all routes to root, respectively
2:  Outputs:
        MX: 2D scheduling matrix
3:  for route ∈ routeList do
4:      slot ← getFirstAvailableTimeSlot(MX) ▷ allocation of the first timeslot
5:      for hop ∈ route do
6:          slot++
7:          while slot < slotFrameLength do
8:              if timeSlotCollision(hop, slot) then
                    ▷ Checks whether there is an collision in timeslot or not
9:                  slot++
10:                 continue
11:             else
                    ▷ Return available channeloffset for hop(sender, receiver) in timeslot slot
12:                 channel ← getChannelOffSetAvailable(hop, slot)
13:                 if channel then
                        ▷ set hop(sender, receiver) in timeslot slot and channeloffset channel
14:                     setMX(hop, slot,channel)
15:                 else
16:                     slot++
17:                     continue
18:                 end if
19:             end if
20:         end while
21:     end for
22: End for
23: Return MX
```

Algorithm 3: Scheduling process in the proposed method.

In this article, the number of parents is two and three, and the maximum number of children of each node is two. These values depend on the opinion of the network administrator. One of the things that can be investigated in the future is determining these values optimally.

3.3. Computing Routes. TSCH relies on a scheduling matrix, assigning cells to each link (i.e., pair of sender/receiver). In anycast, several receivers are scheduled in the same cell. All the receivers are in listening mode before the transmitter triggers a transmission. A node that is not taking part in any transmission/reception can sleep without creating deafness. By minimizing the number of cells assigned to each node, the controller can reduce the energy consumption and, thus, can increase the network lifetime. Thus, we have to carefully select the set of parents for each node to balance the energy consumption and, thus, improve the network lifetime (the parent selection algorithm presented in Section 3.2 implies the same thing). Because when choosing parents, it should be done in such a way that the number of lost packets does not happen as much as possible because each packet lost means retransmission and waking up the sender and receiver nodes, and this itself causes an increase in energy consumption and a decrease in network life. Obviously, enough transmission opportunities have to be allocated to respect a high end-to-end reliability.

We proceed in the following way to create the list of flows and their routes (Algorithm 2):

(1) We calculate the depth of each source node in the network (line 6). getDepth function calculates the depth of a node from the path of both of its parents and returns the maximum depth as the depth of the node.
(2) The links on the path to the root are calculated (lines 8–12). getParent function returns all parents of node, and algorithm 2 performa a kind of breadth first search (BFS).
(3) The total path is inserted in the list routeList (line 13).
(4) We sort the routeList to return (line 15). More precisely, it is sorted according to the length of the flows in descending order.

Then, we schedule the first flow on this list and then remove it from the list (Algorithm 3). We reiterate until all the flows in the routeList have been scheduled.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0		GBE	BDE	EF	DF	EF	BDE	DF	EF	DF	CF	DF	EF
1		ACD	CF										
2													
3													
4													

Figure 6: The proposed scheduling matrix for the topology of Figure 5 when traffic is initiated at node G.

Finally, a list like the following is created in which longer flows appear first. As shown in the topology in Figure 5, node G is located at the lowest depth of the tree; therefore, all the paths from G to F should be scheduled first.

$$\text{routeList} = ((\text{GBE}, \text{BDE}, \text{EF}, \text{DF}, \text{EF}), (\text{BDE}, \text{DF}, \text{EF}), (\text{ACD}, \text{CF}, \text{DF}), (\text{EF}), (\text{DF}), (\text{CF})). \quad (2)$$

3.4. Adressing Parents in the Packet Header. In the 6TiSCH protocol stack, in the network layer, and in the 6lowpan sublayer, when sending the packet and header compression, source mac and destination mac are added to the packet header. In the approach presented in this sublayer, the mac address of the second and third parents (if any) is also added. Also, an anycast bit is added to identify that the source mac of this packet (sender) has second and third parents. On the receiving side, in the 802.15.4-TSCH sublayer, the anycast bit is first checked, and if this bit is set, the destination macs in the packet header are compared with the mac of the node that received the packet. If the mac of the node that received the packet is among the destination macs, it means that a MAC layer frame has been received and this frame is delivered to the higher layer (6lowpan). Otherwise, it will be dropped. In layer 6lowpan, it is checked whether this packet should be forwarded or dropped (based on the four situations mentioned in Section 3).

3.5. Proposed Scheduling Algorithm (AASA). The network exploits a DAG topology, and we consider the following constraints when constructing the scheduling matrix:

(1) We schedule longer flows first. Indeed, longer flows create more constraints because of the multihop topology.
(2) We maximize the reliability and consider uniquely dedicated cells. Thus, a node can be either in RX mode or in TX mode [25].
(3) We try to minimize the end-to-end delay by minimizing the buffering delay. Indeed, any intermediary hop has to buffer the packet until a TX cell is scheduled for the next hop. Thus, consecutive links are scheduled in consecutive time slots.

We now compute the schedule along the paths computed through Algorithm 2. We schedule the flows one by one. For instance, the flow from G is scheduled first because it contains four links. Some of the links are anycast (one transmitter and two receivers), and the other ones are unicast.

Besides, the links have been scheduled back-to-back to reduce the end-to-end delay: the buffering time is minimized. We also consider the order constraint specific to anycast. Indeed, a packet may come from several paths. Thus, the TX cell has to be scheduled after all the RX cells. It is worth noting that two transmission opportunities have to be allocated for the link EF, for instance, because its link quality is lower.

By convention, we consider that the cell (0,0) is reserved for control packets transmitted in broadcast (Figure 6). All the nodes have to stay awake to receive EB, DIO, etc. As shown in Figure 6, for the scheduling of the G-to-F flow, all links (blue cells) have been scheduled in consecutive time slots.

Next, the second flow of routeList (B to F) should be scheduled. According to the scheduling matrix, the BDE link should be scheduled in the sixth time slot (lines 8–10 of Algorithm 3) because nodes B, D, and E were in the TX or RX mode in the first and second time slots; also, DF and EF have been scheduled in the third, fourth, and fifth time slots.

Afterward, in the same way, DF and EF are scheduled in the seventh and eighth time slots.

Then, the flow from A to F should be scheduled. As shown in Figure 6, the ACD link can be scheduled in the first time slot. The reason is that none of A, C, or D has been scheduled in this time slot. Afterward, the CF is scheduled in the second time slot and DF in the eighth time slot.

Next, the scheduling of the EF, DF, and CF flows is selected from routeList and scheduled in the appropriate cells.

The presented solution we named anycast aware scheduling algorithm (AASA) uses many cells because it schedules each route separately from the origin to the destination and it is possible that part of several routes are shared, in which case the common links are scheduled several times. Also, for example, when we want to schedule the route from B to F (orange cells, as shown in Figure 6), we cannot place the BDE link in cells 1–5 due to time-slot collision; therefore, we schedule it in cell 6. Finally, although in this algorithm, reliability is improved and latency is reduced, as nodes stay awake for more time slots, network energy consumption will increase.

3.6. Compact Mode of AASA. Importantly, to make the scheduling matrix more compact and make more effective use of the cells, we could only consider the main parents when applying constraint 2 for scheduling named compact mode of anycast aware scheduling algorithm (CMAASA), which would give us a scheduling matrix, as shown in Figure 7.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0		GE(B)	EF	DF	CF	EF	DF	EF					
1		BD(E)											
2		AC(D)											
3													
4													

Figure 7: The compact scheduling matrix for the topology of Figure 5.

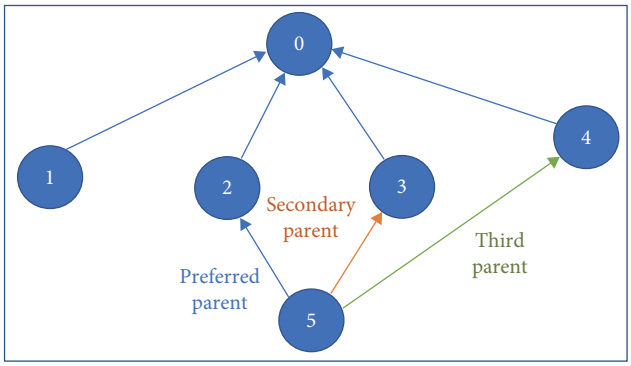

Figure 8: Part of static 16 nodes topology.

For example, when we want to schedule route from B to F, we can place BD link in time slot 1 because the B and D are not scheduled in this time slot. In fact, in this algorithm, only the first parent is considered when checking interferences, but both parents (if any) are scheduled. For further explanation, we consider two scenarios:

(1) In the first scenario, we assume that the packet sent from node B is not received by the first parent (D) but is received by the second parent (E). In this case, this packet remains in the queue of a node and must be forwarded in time slot 2 or 5. In these time slots, there is a packet of node G and the data sent by node E itself, so the sending of packet B is delayed.

(2) In the second scenario, we assume that the GF flow is not received at the first link by the first parent (E) but by the second parent (B), the packet must be forwarded by node B. As a result, it should wait until the next slot frame and the first time slot because it has been scheduled only in the first time slot of the BD (E) link. At the same time, however, B has its own packet on the transmission queue and, therefore, the packet received from G has to wait in B's queue before being sent in the next slot frame. This will increase delay in the transmission process.

Therefore, in the CMAASA, although the number of used cells will decrease, the amount of delay will increase.

4. Performance Evaluation

We implemented the scheduling process described in Algorithm 3 (AASA and CMAASA). We implemented two scenarios using the parent selection algorithm presented in Section 3.2:

Table 1: Simulation parameters.

Value	Parameter
6TiSCH	Simulator
1, 2, 3, 4, and 5 pkt/s	CBR
DoDAG (tree)	Topology
Pister–Hack model	Propagation model
8, 12, 16, and 20	Topology size
Converging toward the root	Traffic pattern
10 ms	Time slot duration
101 time slots	Slot frame length
16	Number of physical channel
OF0	Rpl OF
1 pkt/60 s	DIO transfer rate
9 hr	Simulation time
6.4 μC	Idle listen
54.5 μC	TxDataRxAck
49.5 μC	TxData
32.6 μC	RxDataTxAck
22.6 μC	RxData
0.0 μC	Sleep
2,821 mAh	Battery capacity

Scenario 1: We create a static topology with 16 nodes (Figure 8). We evaluate and discuss the performance of the compact mode of AASA.

Scenario 2: Twenty topologies are generated randomly. The average of the execution results is taken as the final results. We study the impact of the network size (number of nodes) and of the packet interarrival. In this scenario, we use AASA to schedule the links.

The number of parents is a parameter of the simulation (comprised between 1 and 3). We consider a convergecast traffic pattern with constant bitrate flows.

We performed the simulations using the 6TiSCH standard simulator [11], which is an event-based simulator based on Python. This simulator was developed by the IETF 6TiSCH work group and is used as a simulator for standard experiments. The source code of our simulation is available [26]. Table 1 shows the simulation parameters.

We evaluate our proposed solution in terms of the following metrics:

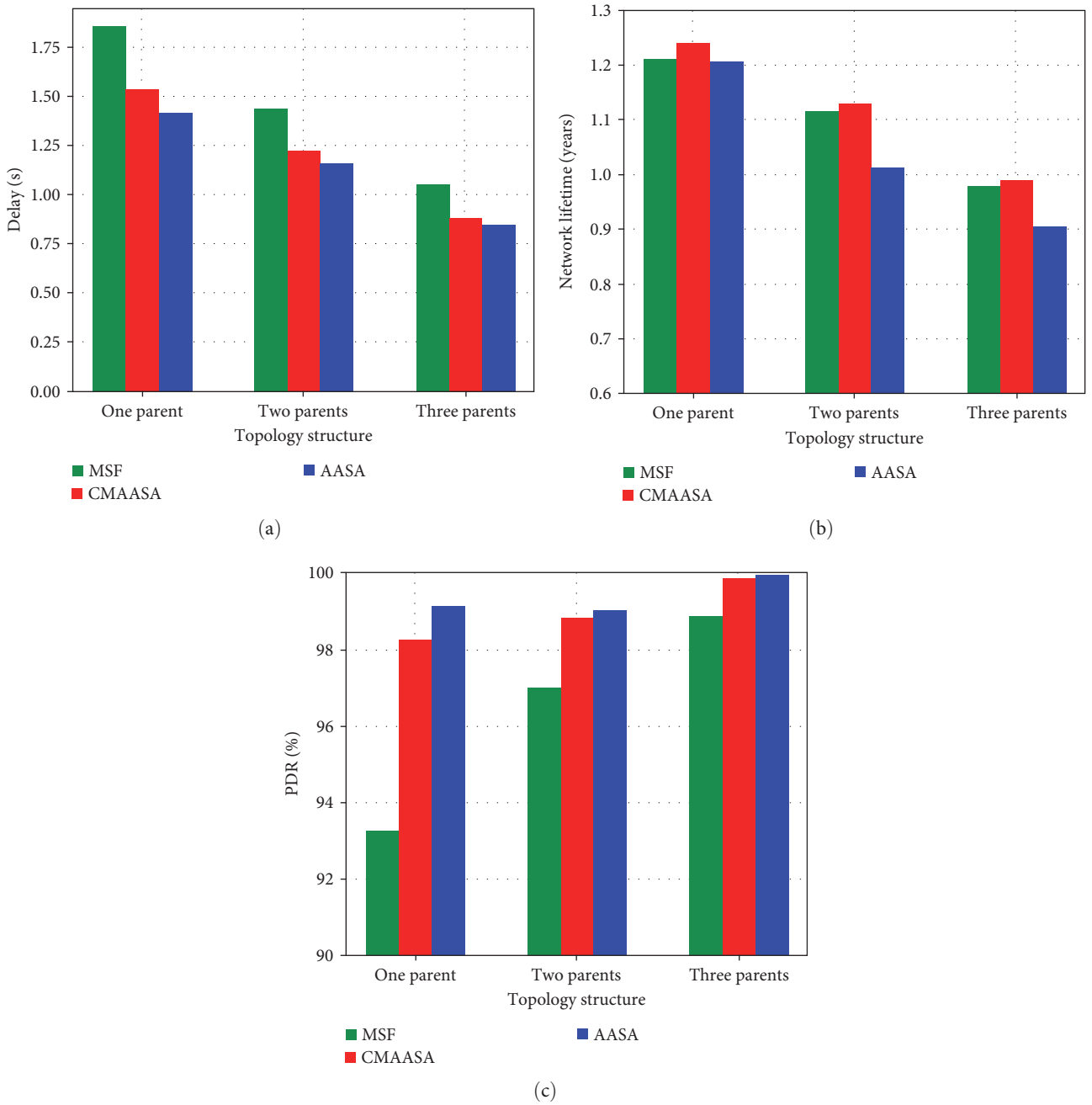

Figure 9: Evaluation of static topology for two scheduling implementations: (a) delay; (b) network lifetime; (c) packet delivery.

(1) End-to-end PDR: Ratio of the number of packets received correctly by root to the number of packets produced by the sources.

(2) End-to-end delay: Amount of time between the generation of the packet and its reception by the root node.

(3) Network lifetime: We consider the first death. Thus, we measure the amount of time until one node out of energy.

(4) Average number of packets in the queue: We count the number of packets that each second-level and lower nodes have in their queue.

4.1. Scenario 1: Static Topology. We first measure the impact of the number of parents in the DAG (Figure 9). In this section, we have compared the proposed algorithms with the MSF algorithm that is implemented by default in the 6TiSCH simulator. MSF defines both the behavior of a node when joining the network and how the communication schedule is managed in a distributed fashion. As shown in Figure 9(a), our scheduling algorithm schedules efficiently the links back-to-back even in AASA. Thus, we greatly reduce the end-to-end delay. Having a compact schedule has a cost on the delay. Also, as can be seen, the MSF algorithm has a higher delay than the proposed algorithms. This

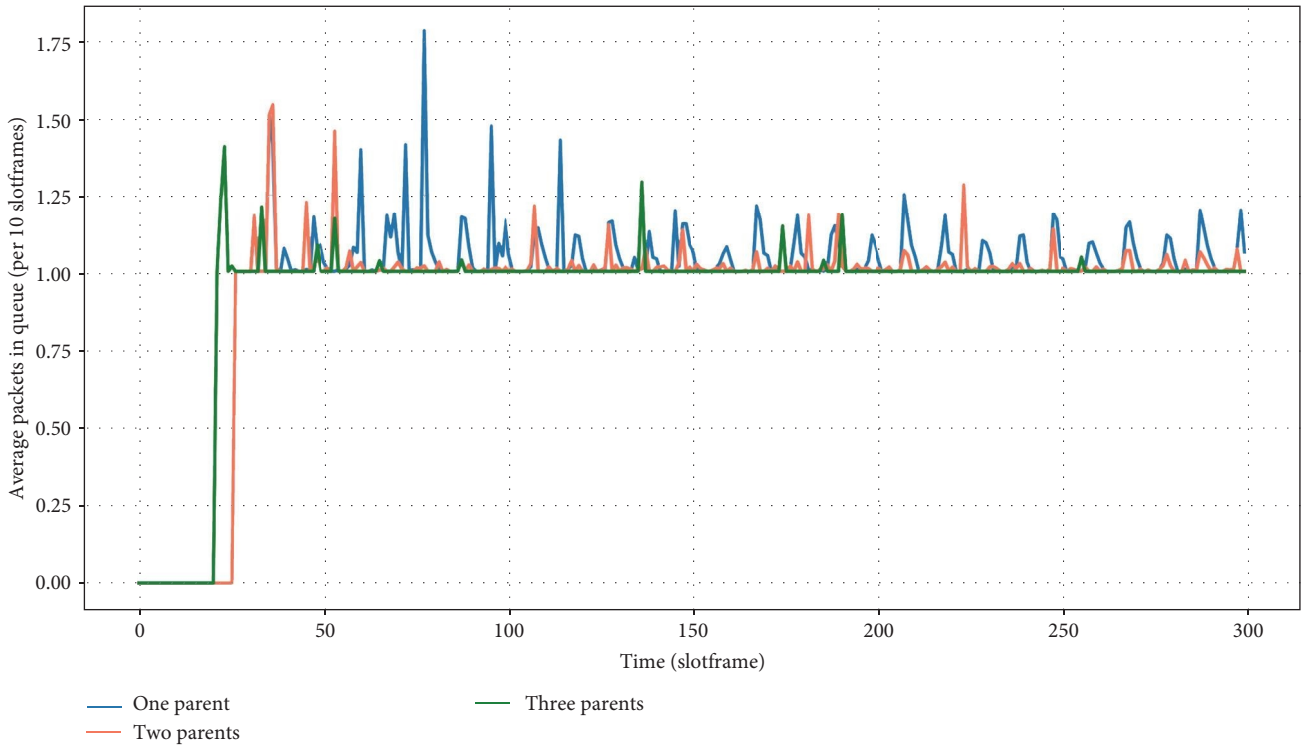

Figure 10: Average packets in queue for node 5.

difference is even greater when only one parent is used because adding another parent will prevent retransmission. Therefore, the difference in end-to-end delay between the AASA and CMAASA will become smaller but not disappear since sometimes all the parents do not receive the packet simultaneously and this will cause retransmission. As in the AASA repetitive paths are again scheduled in the same slot frame, the packet will be retransmitted immediately in the same slot frame at the next link without waiting for the next slot frame (a case where CMAASA is used for scheduling). Therefore, the delay will be reduced.

Figure 9(b) shows the average network lifetime. The network lifetime is longer when the CMAASA is used because the scheduling algorithm allocates fewer cells to each of link without tentative cells. Since a node can sleep during the unscheduled cells, the CMAASA decreases energy consumption. Also, it can be seen in MSF that the network lifetime is less than that of CMAASA, because in MSF, the allocation of cells in the scheduling matrix is random and there is no particular order, so the number of retransmissions will be more. Obviously, lower reliability and a larger delay are the prices to pay for this energy gain. We can note that the decrease in the network lifetime remains acceptable for two parents or less. After three parents, more receivers are scheduled, that have to stay awake, and the additional parents are very seldom used.

In this study, the destination of all packets is the sink node. Using the anycast mechanism, if a link fails, the packet can be sent through an alternative link and transmitted through the same time slot. Therefore, this feature guarantees high reliability in the transmission of packets. As shown in Figure 9(c), the network's reliability is higher when AASA is used. The reason is that, in this algorithm, a packet that is lost has much more cells available for retransmission and will have higher chances of being delivered to the sink node. Also, adding a new parent will increase the network's reliability because packet loss is sometimes even probable in retransmissions and this could be prevented by the secondary and third parents. Thus, we can maintain a CMAASA and reduce the retransmissions with only a small impact on the network lifetime.

4.1.1. Number of Queue Packets of Multiparent Nodes. We measure the queue size of the different nodes. We consider three modes for this scenario: one parent, two parents, and three parents and use AASA for scheduling method. The topology size is 16 nodes, and packet transmission rate is 2 pkt/s. The rest of the parameters are the same, as shown in Table 1. Figure 8 shows a part of the network topology focusing on node 5. In this part, we want to check the number of packets in this node's buffer, as shown in Figure 10. In this figure, the average number of packets in the queue of node 5 in every 10 slot frames is shown over time. For example, between slot frames 500 and 1,000, the average number of packets in the buffer of node 5 is much higher in single-parent mode than in two- and three-parent modes. As shown in Figure 10, the number of packets in the queue may present a few bursts with only one parent. Indeed, retransmissions may occur, and the node has to postpone its retransmissions to the next slot frame. The queue may possibly accumulate the packets before the retransmission succeeds. Inversely, with more parents, the retransmissions are much less frequent.

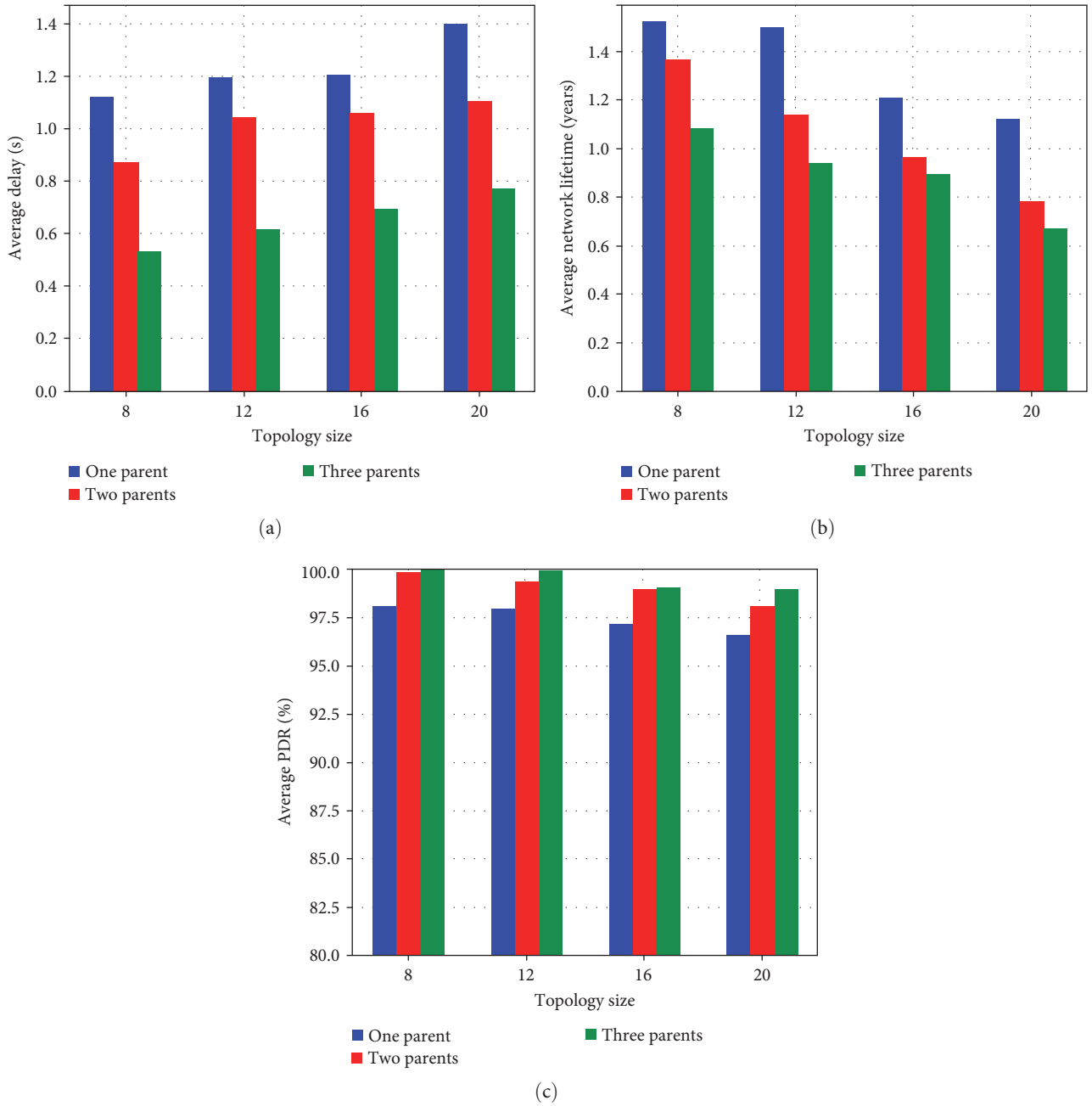

Figure 11: Impact of topology size: (a) average delay; (b) average network lifetime; (c) average PDR.

We remind that a transmission fails if no parent is able to decode it. Thus, considering multiple parents helps to reduce the maximum queue size, which corresponds to the worst case. Thus, anycast helps to reduce the constraints on the queue length, and the number of buffer overflows.

4.2. Random Topology. In this scenario, we make use of the AASA algorithm.

4.2.1. Impact of Topology Size (Number of Nodes). The average end-to-end delay in relation to the topology size is shown in Figure 11. Figure 11(a) indicates that, as the number of network nodes increases, the end-to-end delay increases as well whatever the number of parents. However, exploiting multiple parents reduces the end-to-end delay. This is because with the increase in the number of parents, the number of packet loss decreases and as a result the number of retransmissions decreases again, which leads to lower delay in three and two parent modes compared to single parent mode. The counterpart is an impact on energy consumption (Figure 11(b)). Indeed, more receivers are scheduled for each transmission and have to stay awake to decode the packet. This has an impact on the network lifetime. Inversely, more parents mean also fewer retransmissions. On average, the network lifetime is slightly decreased with a larger number of parents.

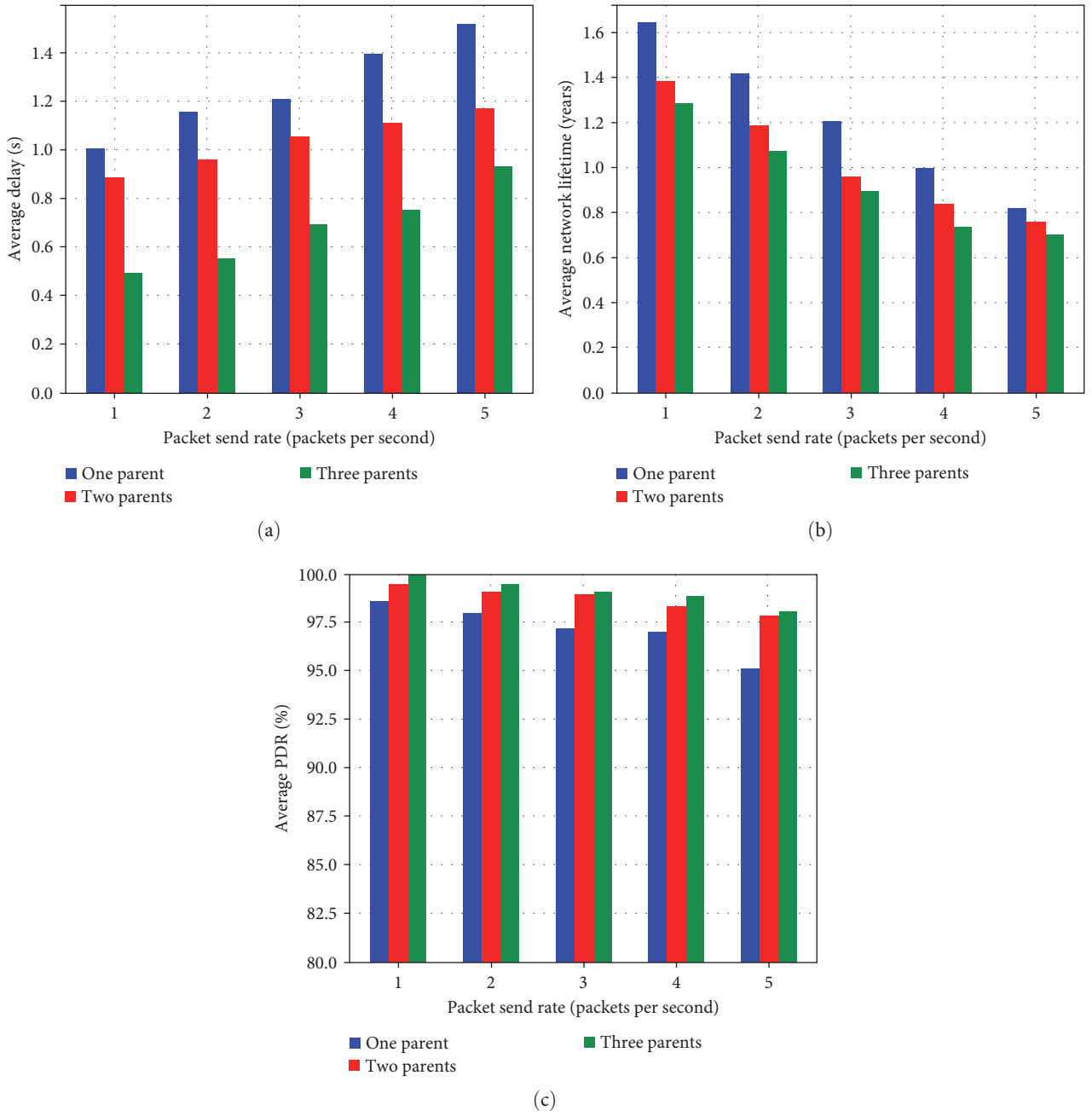

Figure 12: Impact of packet rate: (a) average delay; (b) average network lifetime; (c) average PDR.

However, we consider this is the price to pay for high reliability achieved by anycast (Figure 11(c)). Indeed, a larger number of receivers means that fewer packets are lost. We need fewer retransmissions to obtain high end-to-end reliability. Even with 20 nodes in the network, we are able to provide reliability higher than 99%, which should be sufficient for most applications. If we want to consider even higher reliabilities, our scheduling algorithm may allocate retransmission opportunities when allocating cells.

4.2.2. Impact of Packet Rate. The average end-to-end delay in relation to the packet send rate, as shown in Figure 12(a), indicates that, as the packet rate grows in the network, the end-to-end delay increases. Indeed, more packets imply more collisions and retransmissions. Thus, the buffering delay increases as well. Exploiting multiple parents helps to reduce the delay: retransmissions are only due to collisions and not due to a low link quality. Indeed, exploiting multiple receivers decrease the probability to not be able to decode the packet at the receiver side.

As shown in Figure 12(b), as the network's packet rate grows, its lifetime decreases. Indeed, more packets have to be transmitted, which increases mechanically the energy consumption. When the packet send rate increases, the possibility of collision in the network increases, and as a result, the number of retransmissions increases. This condition causes

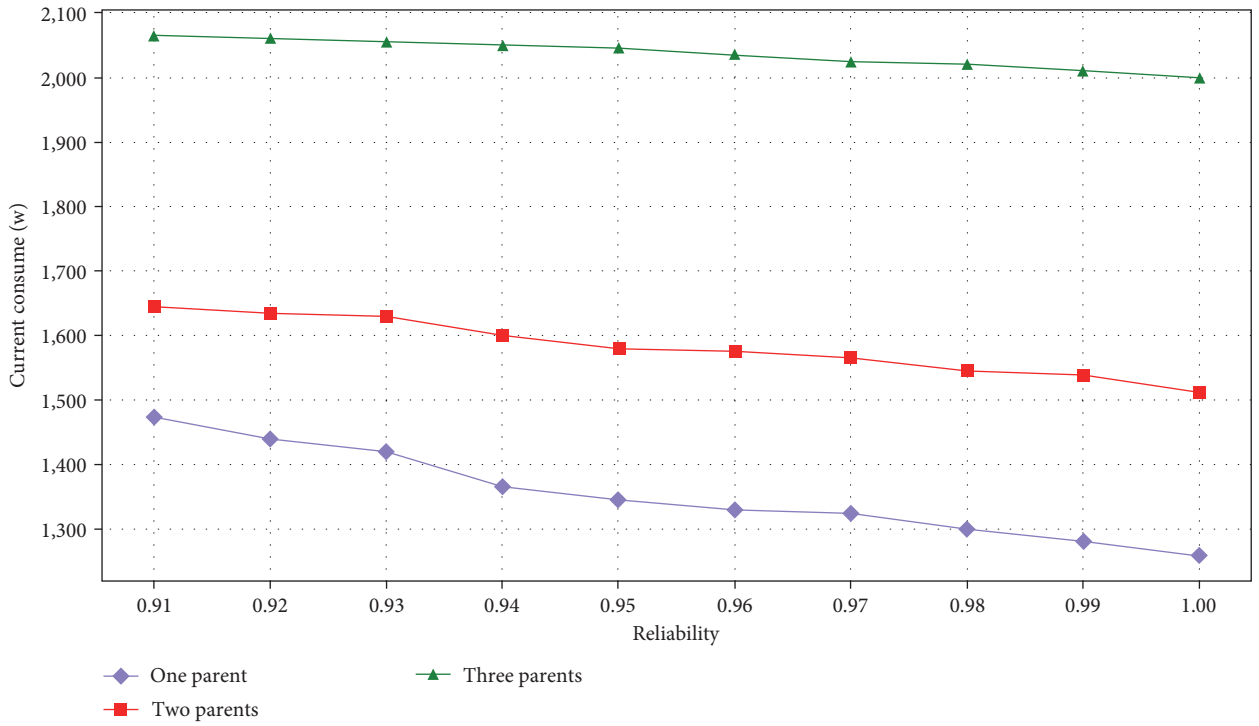

Figure 13: Average current consume based on given reliability.

the nodes to stay awake in more time slots, which leads to an increase in energy consumption. When multiple parents are used, the energy for maintaining several receivers awake increases. Thus, the network lifetime decreases. However, this loss seems acceptable with regard to the improvement in reliability and end-to-end delay. When the number of packet loss decreases with the increase of the number of parents, it increases the reliability in the network. As shown in Figure 12(c), the number of packet losses is significantly decreased with multiple parents. However, the reliability in the network decreases with the increase in network latency. The reason is clear. As mentioned, with the increase in the number of packets sent per unit of time, the amount of collision in the network increases; as a result, the value of PDR decreases, which means that fewer packets reach their destination with the increase of interference in the network.

4.3. Energy Consumption. In this section, we present the energy consumption model for our proposed model. According to the given reliability, we investigate the average current consumption for the successful transmission of packets in different scenarios of one, two, and three parents. Table 1 shows the amount of current consumed in terms of microcoulombs for each state (sleeping, awake, idle, transmitter, receiver, transmitter–receiver Ack, receiver–transmitter Ack) of the node. Each state is explained separately as follows:

Idle: Time slot during which a node listens for data but receives none.

TxDataRxAck: A time slot during which the node sends some data frame and expects an acknowledgment (ACK).

TxData: Similar to TxDataRxAck, but no ACK is expected. This is typically used when the data packet is broadcast.

RxDataTxAck: A time slot during which the node receives some data frame and sends back an ACK to indicate successful reception.

RxData: Similar to the RxDataTxAck but no ACK is sent (for a broadcast packet).

Sleep: Time slot during which the node's radio stays off.

In the proposed model to calculate the current consumption, the total number of slots when the node in each which of the introduced modes, has been multiplied by the corresponding value of the same state and the results are added together.

$$\text{Charge} = \text{Idle listen slots} \times 6.4, \tag{3}$$

$$\text{Charge} += \text{TxDataRxAck slots} \times 54.5, \tag{4}$$

$$\text{Charge} += \text{TxData slots} \times 49.5, \tag{5}$$

$$\text{Charge} += \text{RxDataTxAck slots} \times 32.6, \tag{6}$$

$$\text{Charge} += \text{Rx slots} \times 22.6, \tag{7}$$

$$\text{Charge} += \text{Sleep slots} \times 0. \tag{8}$$

Next, the value of charge is divided by the total time the node has been in the network to calculate the average amount of current consumed by the node in terms of Microamp.

$$\text{avgCurrent} = \frac{\text{Charge}}{\text{asn} \times \text{tschSlotDuration}}. \quad (9)$$

To calculate the energy consumed in Watts, the amount of current consumed must be multiplied by the voltage:

$$\text{Consumed energy} = \text{avgCurrent} \times \text{Voltage}. \quad (10)$$

In the following, the amount of energy consumed for the rip sensors [27] is used in the IoT; based on the energy consumption calculation model, we will check the average energy consumption for transmitting packets based on different PDRs in one, two, and three parent scenarios. The simulation is based on the parameters, as shown in Table 1. The packet transmission rate is 5 pkt/s, the topology size is 20, and the algorithm used is AASA. In this simulation, values between 0.91 and 1 for network reliability are given manually.

As shown in Figure 13, the slope of the line corresponding to one parent is higher and the lowest slope is related to three parents. The reason is clear. In the case where all nodes have a parent, the number of retransmissions increases rapidly when network reliability decreases and the nodes stay awake in more time slots, which according to the provided energy consumption model, the energy consumption of the nodes will increase at a faster rate. But in the two-parent and three-parent modes, retransmissions rarely happen and the energy consumption of the nodes does not change significantly. It can also be seen that, in general, the energy consumption related to the three-parent mode is more than the other modes. Because, as mentioned in Sections 4.1 and 4.2, adding a parent increases the current consumption of nodes and ultimately reduces the network lifetime.

5. Conclusion

This paper implemented and evaluated an anycast mechanism on a centralized scheduling algorithm based on the 802.15.4-TSCH and 6TiSCH protocol stack. We propose as well a specific scheduling algorithm to take benefit from the anycast feature. Indeed, each transmitter sends its packet simultaneously to two or more receivers in a single time slot. If the packet acknowledgment is not received from the receiver with higher priority, the second, third, etc., receiver will decode the packet, send its acknowledgment to the sender, and forward the packet further. Our findings suggest that by using the anycast mechanism end-to-end delay is significantly reduced while the network's reliability increases. Since several nodes are awake instead of two nodes in each cell, these improved performances have a negative impact on the network lifetime: that's the price to pay for high reliability.

In future work, we expect to measure the performance of our system in a real environment (also known as testbed). We also expect to investigate the time–variant characteristics of the environment and how to adapt the schedule to changes, when, e.g., the link quality varies over time. Anycast should help us to keep high reliability: even if the link quality of a specific link evolves, the other parents may serve as fallback alternatives, until the schedule is patched.

Data Availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Acknowledgments

This work has been supported by the Center for International Scientific Studies & Collaboration (CISSC) Ministry of Science, Research and Technology of Iran (Grant Number 1453).

References

[1] L. Moutinho, P. Pedreiras, and L. Almeida, "A real-time software defined networking framework for next-generation industrial networks," *IEEE Access*, vol. 7, pp. 164468–164479, 2019.

[2] G. Chopra, R. K. Jha, and S. Jain, "A survey on ultra-dense network and emerging technologies," *Journal of Network and Computer Applications*, vol. 95, pp. 54–78, 2017.

[3] C. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[4] IEEE, "IEEE Standard for Local and metropolitan area networks —Part 15.4: low-rate wireless personal area networks (LR-WPANs) Amendment 1: MAC sublayer," in *IEEE Std.802.15.4e-2012*, April 2012.

[5] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: automation networks in the era of the internet of things and industry 4.0," in *IEEE Industrial Electronics Magazine*, vol. 11, pp. 17–27, March 2017.

[6] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in *2016 IEEE Wireless Communications and Networking Conference, Doha, Qatar*, pp. 1–6, 2016.

[7] IEEE, "IEEE standard for local and metropolitan area networks—Part 15.4: low-rate wireless personal area networks (LR-WPANs)," in *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, September 2011.

[8] M. Z. Hasan, H. Al-Rizzo, and F. Al-Turjman, "A survey on multipath routing protocols for qos assurances in real-time wireless multimedia sensor networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1424–1456, 2017.

[9] E. M. Ahrar, M. Nassiri, and F. Theoleyre, "Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks," *Journal of Network and Computer Applications*, vol. 142, pp. 25–36, 2019.

[10] T. Watteyne and M. Palattella, "Using IEEE 802.15.4e time-slotted channel hopping (TSCH) in the internet of things (IoT): problem statement, document draft-ietf-6tisch-tsch, Internet Engineering Task Force," May 2015.

[11] E. Municio, G. Daneels, M. Vuini et al., "Simulating 6tisch networks," *Transactions on Emerging Telecommunications Technologies*, vol. 23, pp. 480–494, 2012.
[12] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications—(PIMRC)*, pp. 327–332, Sydney, NSW, 2012.
[13] T. G. Venkatesh Lokesh Bommisetty, "Dynamic programming based low-latency scheduling (DPLLS) for 6TiSCH Networks," *Ad Hoc Networks*, vol. 124, no. 816, Article ID 102708, 2022.
[14] R. Alexander, A. Brandt, V. Jp et al., "*RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, document draft-ietf-roll-rpl," in *Internet Engineering Task Force*, March 2012.
[15] Q. Wang, X. Vilajosana, T. Watteyne, and D. Dujovne, "*6TiSCH Operation Sublayer Protocol*(*6P*), document draft-ietf-6top-protocol-12," in *Internet Engineering Task Force*, June 2018.
[16] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF), document draft-ietf-6tisch-msf-01," in *Internet Engineering Task Force*, October 2018.
[17] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: a survey," *Computer Communications*, vol. 114, pp. 84–105, 2017.
[18] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura, "6TiSCH experimental scheduling function(sfx), document draft-ietf-6top-sfx-01," in *Internet Engineering Task Force*, June 2018.
[19] Q. Lampin, D. Barthel, I. Auge-Blum, and F. Valois, "Exploiting long-range opportunistic links to improve delivery, delay and energy consumption in wireless sensor networks," in *International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pp. 483-484, IEEE, 2012.
[20] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 133–144, ACM, 2005.
[21] T. Huynh, F. Theoleyre, and W.-J. Hwang, "On the interest of opportunistic anycast scheduling for wireless low power lossy networks," *Communications*, vol. 104, pp. 55–66, 2017.
[22] I. Hosni and F. Theoleyre, "Adaptive k-cast Scheduling for high-reliability and low-latency in IEEE802.15.4-TSCH," in *ADHOC-NOW*, vol. 11104 of *LNCS*, pp. 1–12, Springer, Sep 2018.
[23] R. T. Hermeto, A. Gallais, and F. Théoleyre, "Is link-layer anycast scheduling relevant for ieee 802.15.4-tsch networks?" in *LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, pp. 133–140, 2019.
[24] F. Theoleyre, "Duocast for wireless industrial networks: an experimental study," in *International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 99–107, Association for Computing Machinery, New York, NY, USA, 2021.
[25] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3655–3666, 2013.
[26] https://github.com/sahand-amiri/6TiSCH-AnyCast.
[27] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview.