Duocast for Wireless Industrial Networks: an Experimental Study

Fabrice Theoleyre

Duocast for Wireless Industrial Networks: an Experimental Study

Fabrice Theoleyre
ICube Lab, CNRS / University of Strasbourg
Illkirch, France
theoleyre@unistra.fr

ABSTRACT

Many Internet of Things (IoT) applications have increasingly stringent requirements: messages have to be delivered to the destinations before a given deadline. Unfortunately, radio networks are known to be lossy, and retransmissions and acknowledgements help to improve the end-to-end reliability. To provide high-reliability, most wireless industrial networks schedule the transmissions to reduce the collisions, and to make the medium access deterministic. In these conditions, duocast helps to improve both the reliability and the fault-tolerance: two receivers are associated with one transmission, so that the transmission fails only if both receivers fail to decode the packet. While anycast has been widely used in simulations, we provide here a thorough experimental evaluation. Indeed, radio links present practically variations, may be asymmetrical, and the hidden receiver problem may practically reduce the gain of duocast. We demonstrate in our experimental evaluation that duocast is really efficient to provide high-reliability while limiting the number of (re)transmissions. It also helps the network to be fault-tolerant: even if a device crashes, or if a given link quality degrades, an alternative path exists to forward the packets, without any additional delay.

CCS CONCEPTS

• **Networks → Link-layer protocols**; **Network experimentation**; **Network measurement**; **Network reliability**; *Network dynamics*; *Wireless personal area networks*; Packet scheduling.

KEYWORDS

IIoT; 6TiSCH; IEEE802.15.4-2015-TSCH; Duocast; Experimental analysis;

ACM Reference Format:
Fabrice Theoleyre. 2021. Duocast for Wireless Industrial Networks: an Experimental Study. In *Proceedings of the 24th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '21), November 22–26, 2021, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3479239.3485696

1 INTRODUCTION

Industry 4.0 is the next industrial revolution, that aims to enable smart factories, making them more flexible and reconfigurable [1]. In this context, Internet of Things (IoT) is a key enabler: devices use a radio chipset to communicate, removing the needs for expensive, fixed cables [2]. Machines may also comprise mobile parts, where cables are prone to breakdowns. In particular, sensors and actuators may use wireless transmissions, even if they are exploited by a critical application.

5G may help to setup a reliable infrastructure for industrial applications [3]. However, it relies on an expensive infrastructure, operating in the licensed band. Moreover, it implies long range transmissions, that limit the scalability: the radio capacity has to be shared among a possibly very large number of devices. Thus, we consider in this paper wireless industrial sensor networks, where strict guarantees have to be respected, but operating in the unlicensed band.

Unfortunately, radio networks are known to be lossy. Depending on the radio channel, a packet may or may not be decoded by the receiver. Even worse, the link quality is time-variant: a burst of packets may be lost because e.g. external interference. In these conditions, providing high reliability and low delays, as required by most industrial applications, is particularly challenging.

To be battery powered, the devices must implement a low duty-cycling strategy, turning off their radio most of the time. Short-range communications reduce also the energy consumption, but multihop topologies may be the rule to limit the deployment costs. In these conditions, providing end-to-end reliability is even more challenging since enough radio resource have to be provisioned all along the path. We may handle the reliability problem at the routing layer with routing diversity. In particular, multipath routing [4] constructs several disjoint paths from the source to the destination. Thus, the network is then fault-tolerant: the packet is lost only if all the paths are *broken*. However, to handle unreliable radio links, we have still to rely on retransmissions: for each path, the packet needs to be retransmitted if no acknowledgement is received.

Most industrial wireless networks rely on scheduling to avoid collisions, and to make the network predictive. In that situation, multipath routing requires to provision many radio resources for all the different paths. Braided paths provide the highest degree of redundancy, but we have to allocate carefully the cells to minimize the end-to-end delay since a packet may follow any sequence of nodes in these partially overlapping paths [5].

We may choose rather to improve the robustness directly at the link layer. DUO-CAST in ISA100.11a-2011 [6] schedules two receivers: the transmission fails only if none of the receivers is able to acknowledge a data packet. Such technique improves the end-to-end reliability, and enables the network to be fault-tolerant. Even better, it does not consume additional radio resources: the two receivers are scheduled during the *same* timeslot.

However, it assumes implicitly that packet losses are independent: if both receivers fail to decode the packet, duocast has no practical benefit. Teles *et al.* already demonstrated that some radio links are sufficiently independent [7]. However, they only exploit

MSWiM '21, November 22–26, 2021, Alicante, Spain

ACM ISBN 978-1-4503-9077-4/21/11…$15.00
https://doi.org/10.1145/3479239.3485696

an experimental dataset, and do not implement anycast at the link layer. Practically, we need the mechanisms to negotiate anycast cells, and to select relevant parents (for braided paths). Moreover, we should quantify the hidden receiver effect: when the second receiver fails to receive the ack of the first receiver, a ack collision arises, and the data packet has to be uselessly retransmitted.

To the best of our knowledge, anycast was never implemented to be evaluated experimentally. The contributions of this paper are as follows:

(1) we provide here a link-layer implementation of duocast (i.e. anycast with two ordered receivers). The secondary receiver will send an acknowledgement only if the primary receiver fails to decode a data packet, and to send an ack. The whole transmission (data and ack) fits in a single timeslot;
(2) we detail how we can modify the 6TISCH stack to enable duocast. In particular, we exploit 6P [8] to be able to negotiate point-to-multipoint cells, and modify RPL to exploit two parents;
(3) we quantify experimentally the gain of duocast, focusing both on the packet loss independency, and the hidden receiver problem;
(4) we provide a full open dataset, as well as the implementation of the duocast feature, for the sake of reproducibility.

2 BACKGROUND & RELATED WORK

We will first expose how IEEE 802.15.4-TSCH and the 6TiSCH stack operate, and then detail the solutions proposed to make the forwarding process fault-tolerant in wireless sensor networks.

2.1 IEEE 802.15.4-TSCH & 6TiSCH overview

To allow high reliability in industrial networks, IEEE 802.15.4-2015 has proposed the TSCH mode. It combines slow channel hopping (per packet) to combat external interference with a scheduled access to avoid collisions.

TSCH is scheduled-based: each device is allowed to access the medium during specific *timeslots*, organized in a slotframe, that repeats over time (cf. Fig. 1). More specifically, the network allocates to each device a collection of *cells*, defined by a timeslot and a channel offset. To enable channel hopping, the channel offset is in reality translated in a different frequency at the beginning of each timeslot, according to the number of timeslots since the network has bootstrapped. This number of timeslots, or Absolute Sequence Number (ASN), defines a global clock in the network. It is worth noting that if the length of the slotframe and the number of frequencies are mutually prime numbers, the same cell uses a different frequency in consecutive slotframes.

The standard defines two types of cells:

dedicated cells do not implement any contention resolution algorithm (e.g. cell for the link (BE) in Fig. 1). The transmitter just starts its transmission without any random backoff. Optionally, a clear channel assessment (CCA) may be triggered by the transmitter, but just to detect external interference. Thus, dedicated cells have to be allocated to non-interfering transmitters, else their transmissions will collide *for sure*, and *repetitively*.

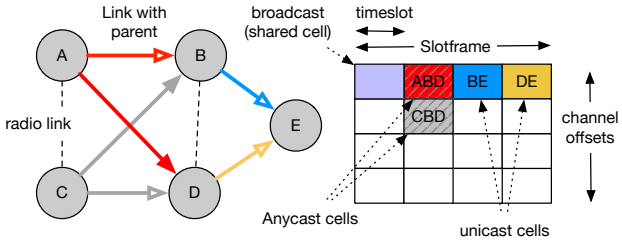


Figure 1: Simple TSCH schedule mixing anycast and unicast cells

shared cells implement a contention resolution for unicast packets. When an acknowledgement is required but not received, the transmitter assumes a collision occurred. In that case, it chooses a random backoff value and skips the corresponding number of shared cells. If a collision occurs, the timeslot is wasted. Thus, the contention resolution is quite expensive, and the amount of traffic to be forwarded through shared cells should be kept to very reasonable values [9].

TSCH supports both centralized and distributed scheduling algorithms [10]. Centralized approaches need to know the link quality, the topology, and the traffic generated by each device to allocate enough cells to each node. Inversely, distributed algorithms are executed locally by each device. Because of the reliability issue, the shared cells [11] are often used to exchange only control packets to bootstrap the network. Thus, most of the solutions rely on allocating distributively non colliding dedicated cells.

In particular, the protocol 6P [8] defines how to modify the schedule. A pair of nodes typically negotiates the cells to use to exchange later data packets. The scheduling algorithm is let unspecified in the standard.

2.2 Opportunistic and anycast forwarding

Opportunistic forwarding has received much attention in the past. Kulkarni *et al.* [12] propose to combine MAC and routing, so that a packet is forwarded to the first neighbor which wakes up, in direction of the sink. Preamble-sampling is also particularly relevant, since the first receiver which wakes-up can acknowledge the preamble to start the reception [13]. However, industrial networks rely on a scheduled access: nodes do not wake up randomly. In other word, opportunistic forwarding has no interest when the transmitter knows exactly when the receivers are awake.

Multipath may help to improve the end-to-end reliability [4]: if the primary path is broken, another backup path is automatically present. Mozafari *et al.* [5] try to reduce the cost of such multipath scheduling. They allocate different links of the same multipath flow to the same cell while avoiding collisions: only one path is practically followed by a given packet. However, retransmissions are still required, and the transmitter has to switch to the backup path after it detects a failure. This has an impact on the bandwidth usage, as well as the end-to-end delay.

PAREO [14] rather exploits multiple paths by introducing replicating points: the corresponding nodes forward a given packet to several nodes. This redundancy helps to improve the reliability.

We rather propose to exploit anycast at the link layer, scheduling several receivers *concurrently*.

ISA100.11a-2011 exploits DUO-CAST, when two receivers are attached to the same cell [6] (cf. anycast cell in Figure 1). The receiver with the lowest priority waits for the ack of the first receiver: it will send an acknowledgement only if nothing is captured. Thus, DUO-CAST enables fault-tolerance: a second receiver automatically takes the lead. However, this feature is not entirely specified, and was to the best of our knowledge not evaluated practically. Huynh *et al.* [15] demonstrate the efficiency of anycast scheduling to reduce the energy consumption with the same level of reliability.

Harms *et al.* [16] implemented an opportunistic routing approach in TSCH. A centralized scheduler allocates both the transmission opportunities and the rank of each node. The receiver with the highest rank acknowledges each correctly decoded transmission. However, they do not describe how the contention among acknowledgements is handled (i.e. timing inside a timeslot).

Surprisingly, anycast scheduling was not evaluated experimentally. Hosni *et al.* [17] propose a routing solution, so that k different receivers can be selected (k-cast), based on their Packet Delivery Ratio of the link with the transmitter. However, the solution is evaluated by simulations, where packet loss probabilities are independent for all the links. More recently, Teles *et al.* [7] investigate the efficiency of anycast by exploiting an experimental dataset (Packet Loss/Reception for a collection of multipoint links). They emulate a wireless network, by replaying the experimental dataset. However, the real anycast is not implemented, and the authors do not emulate the channel between the receivers, which is however fundamental to quantify the anycast gain.

3 PROBLEM STATEMENT

In IEEE 802.15.4-TSCH, duocast consists in associating two different receivers to the same (anycast) cell. Let us consider the simple schedule illustrated in Figure 1. In particular, A transmits its packets that may be acknowledged by either B or D, by using an anycast cell. In our example, the nodes B and D are neighbors of the dagroot (E) and do not have an alternative parent. That is typically the case if the link quality is very close to 100%.

At a first glance, anycast is always relevant: if several receivers are awake to receive a packet, less packets are lost. However, several characteristics may deeply impact the efficiency.

3.1 Link quality

Obviously, anycast does not increase the reliability with perfect links, i.e. a Packet Delivery Ratio close to 100%. However, it helps the network to be fault-tolerant: a second path to the destination is automatically available. To improve the reliability, the two receivers should be independent, i.e. the packet losses for the two receivers should be independent. Fortunately, Teles *et al.* [7] already studied experimentally a smart building environment, and highlighted that independent receivers often exist.

3.2 Energy Consumption vs. Reliability

Without anycast, we can choose to increase the number of retransmissions to provide the same end-to-end reliability. In that situation, the energy consumption is different:

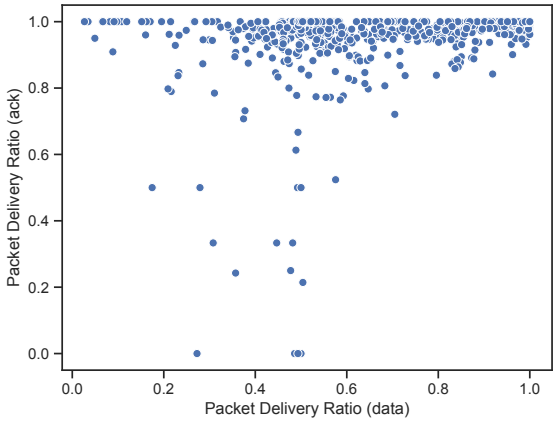


Figure 2: Ack vs. Data packet reliability

TX (retx): the transmitter consumes a quantity of energy proportional to the number of retransmissions;

TX (anycast): the transmitter needs less retransmissions, and consumes on average less energy;

RX (retx): additional cells have to be provisioned for retransmissions. While the transmitter will not wake-up during these additional cells, the receiver must wake-up at the beginning of the timeslot, and will go sleeping only if no Start of Frame Delimiter (SFD) is detected;

RX (anycast): with anycast, all the receivers are awake for the whole timeslot when a packet is transmitted. If additional cells are used for retransmissions, the receivers have also to listen to the beginning of the timeslot to detect a SFD.

3.3 Unidirectional vs. Bidirectional Reliability

Expected Transmission Count (ETX) is a popular metric to estimate the link quality [18]. Practically, it counts the expected number of transmissions before the packet is decoded correctly by the receiver. More precisely, with acknowledgements, ETX is often the quantity of packets to transmit before receiving an acknowledgement. The difference is often very reduced since the data packet had already captured the medium: the ack is transmitted just after the data packet. However, the Wi-Fi backoff may be shorter, and external interference may arise in practice. The problem is even exacerbated in anycast since the secondary receiver has first to capture the ack of the primary receiver, before sending its own ack. Collisions in IEEE 802.15.4-TSCH are reduced since the access is timeslotted, but other colocated IEEE 802.11 or IEEE 802.15.4 networks may negatively impact the performance.

We run in the FIT IoT-Lab experiments a sequence of preliminary experiments to quantify this impact. Our experimental setup is described in details in section 5. We have very diverse link qualities in the testbeds: the Packet Delivery Ratio is mostly comprised between 5% and 100% for data packets. However, when the data packet is correctly decoded, the probability to not be able to decode the acknowledgement is for most links very low. This means that the second receiver will also be probably able to decode the ack.

This property typically differs greatly from the default propagation model implemented in the simulation mode of openwsn, that uses the Pister-Hack Model [19]. While it predicts well the reliability for data packets, it tends to highly over-estimate the losses for ack packets: it has not been designed for this purpose. Thus, we definitely need an experimental evaluation for duocast transmissions.

3.4 Hidden Receiver Problem

Finally, the hidden terminal problem is well known in wireless networks: two transmitter cannot detect their mutual transmissions (carrier sense is inefficient), while they create a collision at the receiver. RTS/CTS has been typically proposed to limit the impact: short reservation packets help to limit the duration of the collision.

Unfortunately, creating additional control frames to just send an acknowledgement for anycast is inappropriate. Since the network is synchronized, we would have to schedule the acknowledgements back-to-back to avoid collisions, which would increase the timeslot duration. Besides, both receivers would forward the packet since the second receiver cannot know if the primary receiver decoded it: this replication increases the resource consumption. We expect here to quantify this hidden receiver problem in an indoor environment: can the probability to miss the ack of the primary receiver be neglected?

4 DUOCAST FOR THE 6TISCH STACK

We proceed to the following modifications to support efficiently duocast in IEEE 802.15.4-TSCH:

- detection of the ack of the primary receiver by the secondary one. Indeed, only one acknowledgement must be transmitted to avoid duplicates and collisions. The secondary receiver implement both a Start of Frame Delimiter (SFD) detection, and triggers a Clear Channel Assessment (CCA) before transmitting its ack to avoid false negatives;
- timing in a timeslot to support multiple receivers: the secondary receiver must capture first the transmission of the primary receiver. If nothing is detected, it transmits an ack. Since we have also to consider the delay to load the packet in the radio chipset, the CCA duration, etc. we have modified the timing of the timeslot, as well as the finite state machine for the secondary receiver;
- in a duocast cell, three different devices are involved: the transmitter and the two receivers. We have re-used the protocol 6P so that the transmitter can interleave the negotiations with the two receivers, to support a consistent 3-tiers reservation.
- to handle possible control packet losses, exacerbated by the fact that three different nodes are involved in the negotiation, and that unreliable links may be exploited, we modify the maintenance of the schedule to avoid inconsistency, without relying on sequence numbers;
- modification of the routing plane to take benefit of duocast. In unicast, each device selects a preferred parent, and all its packets are forwarded to this single next hop. With duocast, we are able to take benefit of unreliable links, to forward the packets farther (and thus reduce the route length). More precisely, a device selects as primary receiver a neighbor which is closer to the DAG root, even if the reliability metric is quite low. However, it selects as secondary receiver its preferred parent, with a high link quality. In that way, the packet will be forwarded by the referred parent if the primary receiver fails to decode (and acknowledge it).

4.1 Acknowledged duocast transmissions

Since we target high reliability, we consider only dedicated cells to implement duocast. Besides, duocast is used only for data packets. Thus, one transmitter sends a frame simultaneously to two different receivers. We may use overhearing for the second receiver, or even unacknowledged broadcast layer-2 transmissions [20]. But this technique presents to our mind two major limits:

(1) if both receivers decode correctly the transmission, they forward both the packet, consuming more resources. Duplicated packets should be eliminated along the path, which is not an easy task if paths are disjoint;
(2) with overhearing, the transmitter cannot know if the packet has been correctly decoded. To provide high-reliability, the transmitted needs to retransmit packets, with a number of retransmissions designed for the worst-case. It consumes unnecessary radio and energy resources.

Thus, we propose rather to use acknowledged duocast transmissions. The second receiver must detect the acknowledgement of the first one. For this purpose, we have two possible options:

Clear Channel Assessment (CCA): the second receiver triggers a CCA when the first acknowledgement is expected (the offset from the frame's reception is fixed). We propose to use an energy detection threshold-based CCA: if the reception power is above a threshold value, the second receiver assumes an ack is already in transmission;

Start of Frame interruption is triggered in listen mode when the preamble of a frame is detected. Thus, when the second receiver receives a start of frame interrupt after the reception of the data packet, it assumes that the first receiver started the ack transmission.

It is worth noting that to minimize the probability of the hidden receiver problem, we may implement both techniques: the second receiver stops the transmission if any of these two events occur.

4.2 Timeslot organization

Obviously, we have to change the timing of the timeslot for both the transmitter and the receiver:

- the transmitter needs to stay awake longer to wait for the second ack;
- the second receiver must change slightly its finite state machine to capture the ack transmission.

We finally obtain the timeline described in Fig. 3. We can note that the first part of the timeslot remains unchanged: the packet is transmitted to the two receivers, that may decode it if no error is present. In particular, if the Frame Check Sequence (FCS) is invalid, the packet is dropped by the receiver.

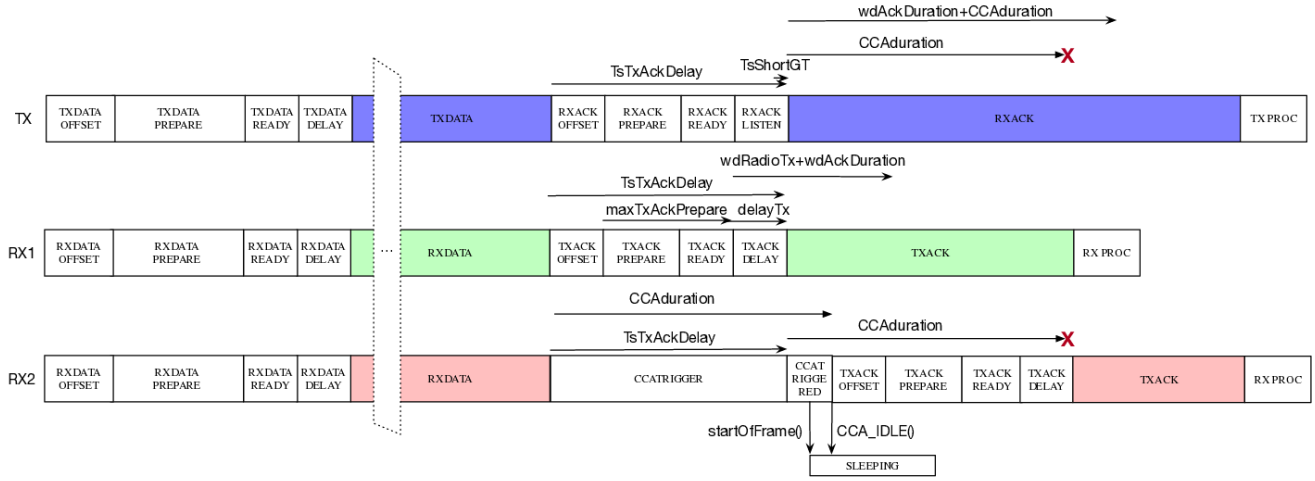


Figure 3: Timeline of a duocast cell.

The second part of the timeslot changes: the transmitter has to stay awake longer, to receive possibly the ack from the second receiver, shifted in time. This offset corresponds to the CCA duration.

The state machine for the first receiver remains also unchanged, and the ack is transmitted in the same way in unicast and in duocast. The second receiver has to detect the first ack before transmitting its own ack. Thus, it stays in CCATRIGGER state after having received the data frame. Just before the expecte time of arrival of the ack, the second receiver triggers a CCA by the radio chipset and changes its state to CCATRIGGERED. If either a start of frame interrupt is received, or the CCA returns a busy state, the second receiver goes directly in sleeping mode. Else, it has to load its ack to the radio, and to trigger its transmission. In our implementation, we had first to trigger the CCA, and then to load the ack frame in the radio chipset.

Finally, because of the margin, we exploit a timeslot of 25 ms, instead of the original timeslot of 20ms when duocast is not enabled. This corresponds to a overhead of 25% at most. Practically, the transmitter and the two receivers are often asleep before the end of the timeslot: 25 ms corresponds to the worst case, i.e. the second anycast receiver was really needed, and another retransmission would have been required in unicast.

4.3 Cell negotiation

Duocast implies that three devices (the transmitter and the two receivers) need to agree on a common dedicated cell to use. Practically, this negotiation is time consuming since they must use shared cells to negotiate, prone to collisions. This corresponds to the reason why we focus here on duocast: increasing the number of receivers involved in the negotiation increases the volume of control packets for the reservation, with a small gain in reliability.

We re-use here 6P [8] for negotiating duocast cells. For this purpose, we mix the two-way and three-way handshakes of 6P, originally tailored for unicast, transmitter vs. receiver oriented negotiation.

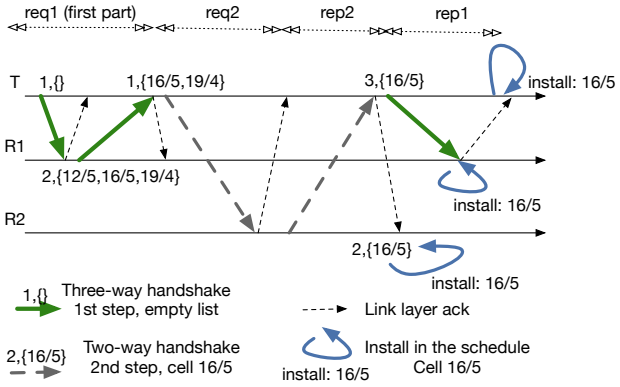


Figure 4: 6P negotiation of a duocast cell.

We proceed in six steps, interleaving the two-way and three-way handshakes (Fig. 4):

(1) the transmitter T sends a three-way handshake ($req1$) to the first receiver $R1$. By definition, the list of cells is empty;
(2) if $R1$ accepts the request $req1$, it piggybacks a list of available cells to the 6P request, replied to T;
(3) T extracts the list of cells from $req1$, and removes from the list the cells that are already busy in its schedule. If not enough cells remain in the list, the negotiation stops here. Else, T engages a two-way handshake ($req2$) with the second receiver $R2$, with the modified list;
(4) $R2$ extracts the list in $req2$ and selects the cells that are available in its schedule. It finally sends a 6P reply to T (corresponding to $rep2$). These duocast cells are now reserved in its schedule;
(5) T receives the reply of $rep2$ from R, and sends the reply $rep1$ (last handshake) to $R1$. As soon as it receives the ack for the $rep1$, the anycast cells are installed in its schedule;

(6) $R1$ installs the cells of $rep1$ in its schedule.

A device piggybacks in its 6P requests a list which is larger than the number of cells to insert really in the schedule (by default 3 cells are present in the list). We force our distributed scheduling function to reserve the cells one by one, so that the probability that all the cells of the lists are already reserved is very low. Besides, we also modify (MSF) [21] to reserve only duocast cells when a device has multiple parents.

Control packet may be lost, leading to inconsistencies in the schedule of the three involved motes. Thus, we chose a timeout-based maintenance scheme, as described in the next section. In particular, a TX cell is inserted in the schedule only if **both** negotiations were successful. Formulated differently, a transmitter can be sure that the cells are present in the two receivers (cf. Fig. 4). When any other control packet is lost, some receivers may have modified their schedule while the transmitter didn't. We will explain in the next section how such situation is handled without creating any data packet loss because of schedule inconsistency.

4.4 Schedule Maintenance

6P exploits sequence numbers when exchanging requests to detect inconsistencies in the schedule. A device maintains a 6P sequence number per neighbor. When a 6P negotiation terminates successfully, the sequence number is incremented. Different sequence numbers reflect a schedule inconsistency: the two nodes then reset their schedule (i.e. they flush all the mutual dedicated cells). However, such approach is very agressive and the schedule has to be reconstructed from scratch when an inconsistency arises (e.g. an ack is not received) [22].

This problem is exacerbated by duocast: the transmitter must flush the schedules with the **two** receivers when a difference of 6P sequence number is detected. Thus, similarly to [22], we prone a timeout-based maintenance: when a cell is not used for a while, the cell is simply removed silently from the schedule. The timeout in reception is larger than in transmission to guarantee that no packet is lost, i.e. the cell is removed first from the transmitter, and then from the receivers. A cell is considered unused by the transmitter if no ack is received, and by the receiver if no frame is received.

We also modified MSF to reflect this behavior. When not enough cells are present in the schedule, additional duocast cells are inserted, negotiated as detailed in the previous section. If too many cells are reserved, the transmitter picks the cell which was not used for the longest time (i.e. no ack received), and removes it silently from its schedule. The receivers will remove it later asynchronously, after the timeout. While the receivers will wake-up for this cell in the next slotframes, they will not detect any reception, and will sleep early. We consider this is much less expensive than the number of control packets to generated to update explicitly the schedule.

4.5 Routing: Preferred / Secondary parent selection

We consider here a convergecast network: the packets are generated and transmitted to the Internet though a single node, the dagroot. In that situation, a device must select two parents to implement duocast. We could measure the independency of transmissions [7], but this estimation is very expensive in practice: we would need probes packets, sent periodically to **all** neighbors.

We propose here rather a greedy solution, so that we can mix the link qualities:

- we select the preferred parent with the orignal RPL version, using the ETX metric. In that way, we privilege a parent with a good link quality;
- the second parent is the neighbor with the smallest rank in our neighborhood table, except the preferred parent. The second parent MUST have a lower rank than ourself to avoid routing loops.

Duocast represents an opportunity to exploit efficiently medium link qualities without impacting too much the reliability. Thus, the primary receiver is the parent with the lowest rank (i.e. largest progress toward the dagroot), and the secondary receiver is the other parent. Practically, the preferred parent (with a good link quality) is often the secondary receiver. That is an advantage: if the transmission to the primary receiver (with the greatest progress) fails, we fallback automatically without any additional cost to a good neighbor. In that way, we reduce the risk to exploit long, and possibly worse, radio links [23].

Table 1: Parameters

#devices	[5, 15]
Period for CoAP packets	800ms * nbDevices
Queue length	20 packets
Queue reserved for control packets	5 packets
Sixtop timeout	65,535s
Slotframe length	101 slots
Duration of each experiment	60min

5 EXPERIMENTAL EVALUATION

We implemented the duocast feature using the openwsn* project. We modified the schedule to be able to account for anycast cells, attaching two receivers to a single cell. The packets that need to be forwarded upward (to the sink) are stamped with the specific anycast multicast link layer destination address. Thus, a device that receives a packet considers it valid in any of the two following cases:

(1) its link layer address is present in the destination field,
(2) the destination is the anycast address, and the source address of the packet is the neighbor attached to the current anycast cell.

We consider a convergecast traffic: each device (except the sink), generates a CoAP packet, forwarded to the sink (i.e. RPL border router). We rely on RPL already implemented in openwsn, modified to choose a secondary parent as described in section 4.5. All the parameters are described in Table 1, and are detailed exhaustively in the configuration file of openwsn-iotlab†.

When selecting a set of motes on the platform, we select randomly the first mote. Then, we select randomly a novel mote at each

*http://www.openwsn.org
†https://bit.ly/3BYvtcb

Table 2: Timing of a duocast timeslot

Offset	Duration
TsTxAckDelay	5,521 μs
CCAduration	3,200 μs
wdAckDuration	3,000 μs
delayTx	549 μs
TsShortGT	700 μs

step. If the closest id in the selected set differs by at most 9 from the mote id of the novel mote, we keep it. Else, we select randomly another mote. Since motes with a close id are located close geographically, we maintain a connected network topology. The same topology is used to compare all the different protocols/variants to limit the possible bias.

Table 2 regroups the timing for a timeslot (cf. Fig. 3), for our specific hardware (Cortex M3, with the STM32F103REY MCU, that includes 64 KB of RAM and 256 KB of ROM).

We measure the following metrics:

Packet Delivery Ratio (PDR): we measure the ratio of the number of packets received and the number of packets generated. We make the distinction between:

per link: we consider the link layer transmissions (each frame). Thus, each link layer retransmission counts for one generated packet;

end-to-end: we consider application layer messages. While the message is delivered to the sink, it may have been retransmitted *en route*.

Number of (link layer) transmissions: the ratio of the number of link layer (re)transmissions and the number of messages generated by the source. A large value means typically that many motes have to forward and retransmit the packet before it reaches the sink.

Ratio of false negatives (acks): ratio of acks actually detected by the secondary receiver, and the number of acks transmitted by the primary receiver. Because of the hidden terminal problem or because of a different radio channel state, the secondary receiver may not be able to detect the ack. This situation is detrimental since the two acks will cause a collision: their transmission will overlap.

5.1 Dataset and Implementation

For the sake of reproducibility, we provide a full access to all our implementation, experiments plan, and data analysis:

openwsn-iotlab [24]: experiment plan, and all the python scripts to reserve a FIT IoT-Lab experiment, flash motes, run the server part, store all the log files for post-processing;

openwsn-fw [25]: firmware in C that implements the CCA feature (medium access, queue management, sixtop negociation, schedule management);

openvisualizer [26]: server (in Python) connected to the motes, with a visualization part (for anycast cells), with a customized library to retrieve all the statistics, and store them in a sqlite3 database;

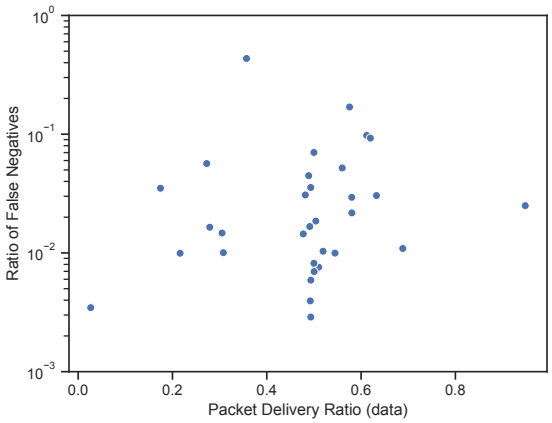


Figure 5: Frequency of the hidden receiver problem (per link)

openwsn-data [27]: data analysis pipeline that extracts information from the database.

dataset [28] is fully available for the community (packets transmissions/receptions, sixtop events, schedule modifications, RPL changes, etc.)

5.2 Hidden Receiver Problem

We first quantify the impact of the hidden receivers problem. If the secondary receiver fails to capture the ack of the primary receiver, both will transmit an ack, that will collide. It is worth noting that it does not impact the reliability: the packet will be retransmitted. However, it is energy inefficient since such retransmissions will be useless. We measured the ratio of acks that collide, considering individually each anycast cell (Fig. 5). Practically, the number of false negatives is very low: the worst situation (3%) comes from a radio link with a very low reliability (30%). In most cases, the ratio of false negatives is below 1%, which means a very limited increase of the number of retransmissions. This efficiency is corroborated when measuring the raw number of link layer transmissions (cf. section 5.4).

5.3 CCA vs. Start of Frame

The secondary receiver may detect the ack from the primary receiver either with a Start of Frame Delimiter (the radio chipset triggers an interruption after having detected the preamble), or with a busy Clear Channel Assessment (the radio chipset reports an energy detection above the threshold value). We measured for all the anycast cells the ratio of time the ack is detected with a CCA (Figure 6). It is worth noting that SFD interruptions are largely sufficient to detect very accurately the transmission of the ack by the primary receiver. Thus, we could reduce the implementation complexity by removing the CCA detection: the transmission may be scheduled just after the expected time of arrival of the ack preamble. A SFD interrupt would just cancel the data transmission.

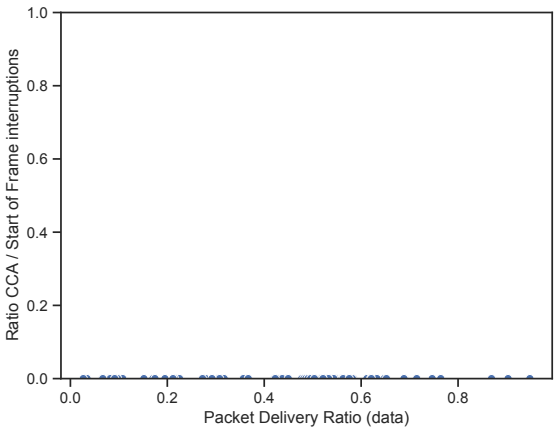
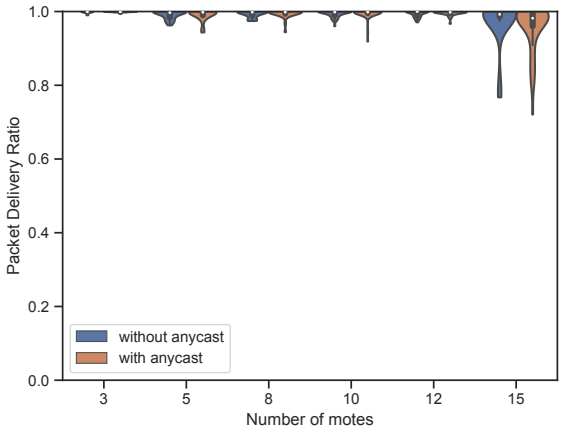
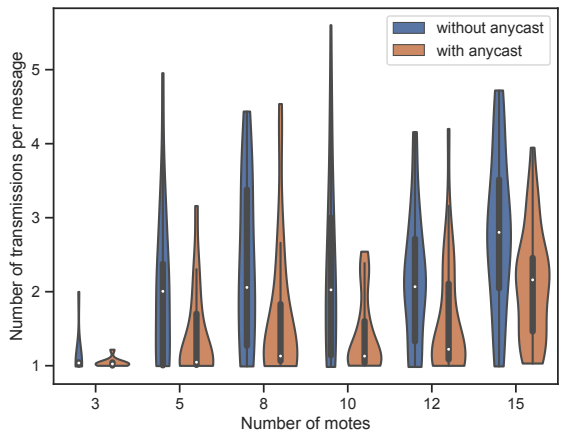


Figure 6: CCA vs. Start of Frame Delimiter Detection of acknowledgements (per link)



(a) End-to-end reliability



(b) Number of link layer transmissions

Figure 7: Efficiency of duocast for Constant Bitrate Traffic

5.4 Scalability

Figure 7 illustrates the performance at the application end-to-end level. The end-to-end reliability is similar with and without duocast (Fig. 7a): a packet is retransmitted until an ack is received. Since the maximum number of transmissions at the link layer is sufficiently high, most of the packets are correctly delivered. However, we can note that the number of link layer transmissions is reduced with duocast (Fig. 7b). More precisely, we combine two positive effects:

(1) a packet is retransmitted only if **both** receivers are unable to decode the packet. The radio channel toward the two receivers is sufficiently different to exhibit sufficiently independent links. This reduces on average the number of transmissions;
(2) a transmitter has two receivers, including a neighbor with the greatest progress toward the dagroot. This means that with anycast, the route length is on average smaller to reach the dagroot. Hopefully, with anycast, exploiting medium link qualities does not impact negatively the reliability. In other words, duocast limits the risk of selecting a bad quality parent.

We can note that that the median number of transmissions is very low, even in the 12-nodes topology. With duocast, we can exploit unreliable links while still providing a high reliability.

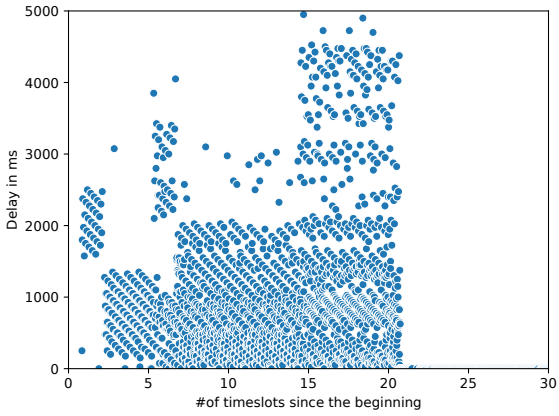
5.5 Faut Tolerance

Finally, we focus on the fault-tolerance property of our duocast solution. We focus on a fixed topology with 4 motes. We manually emulate the crash of a neighbor of the dagroot that forwards all the packets, and keep on measuring the delay all along the experiment. When we turn off the mote after 20 minutes without anycast, the packets stop to be delivered (Fig. 8a). Even worse, the devices become unsynchronized, and are disconnected from the network, searching for an alternative parent for a long time.

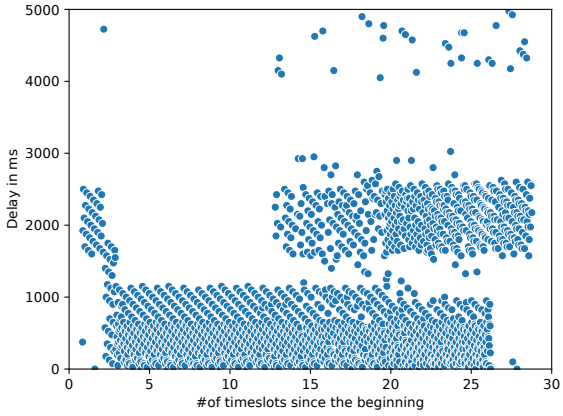
With anycast, a device is turned off after 26 minutes. However, the packets keep on being delivered to the dagroot through the second preferred parent (Figure 8b). We can note that the delay increases, with a bimodal distribution: the secondary parent is farther from the dagroot than the primary parent. This means that the route is longer (1 hop), and that the delay mechanically increases. However, the device uses the second parent to stay synchronized, and it keeps on being able to forward its packets to the remaining forwarding node. Besides, the end-to-end delay may change even if the routes through the two parents are equal: the forwarding nodes have different TX cells to forward the packets.

6 CONCLUSION & FUTURE WORK

We propose here an implementation of duocast in scheduled wireless networks. In a single timeslot are transmitted a data packet, and an ack from either the primary receiver or the secondary receiver. We also propose to enhance 6P to be able to negotiate a timeslot and a channel offset to use for anycast cells, among a group of 3 devices (the transmitter and the two receivers). We conducted a thorough experimental evaluation in an indoor testbed to quantify the gain of duocast to improve the reliability and to reduce the number of link layer transmissions. In particular, the frequency of the hidden receiver problem is very low, and false negatives do not impact



(a) Without anycast: crash at the 20^{th} minute



(b) With anycast: crash at the 27^{th} minute

Figure 8: Fault-tolerance

significantly the reliability. Besides, duocast helps to improve the fault-tolerance: a secondary receiver is automatically here when the primary receiver fails. Our whole dataset, our ducocast implementation, as well as our data analysis pipeline are freely available for the sake of reproducibility.

In the future, we plan to investigate the relevance of anycast with a centralized schedule: how could we optimize the choice of parents to improve the reliability and the diversity? We also plan to port this implementation with different hardware, to study the impact of the SFD interruptions delay on the timing inside a timeslot. Finally, we expect to investigate how to modify our implementation to support other radio chipsets and to reduce the timeslot duration, removing the unnecessary CCA for the secondary receiver.

ACKNOWLEDGMENT

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

REFERENCES

[1] Luis Moutinho, Paulo Pedreiras, and Luis Almeida. A real-time software defined networking framework for next-generation industrial networks. *IEEE Access*, 7:164468–164479, 2019.

[2] Georgios Z. Papadopoulos, Fabrice Theoleyre, Pascal Thubert, and Nicolas Montavont. Ietf reliable and available wireless (raw): Use cases and problem statement. In *Ad-Hoc, Mobile, and Wireless Networks*, pages 303–314, Cham, 2020. Springer International Publishing.

[3] Jan García-Morales, M. Carmen Lucas-Estañ, and Javier Gozalvez. Latency-sensitive 5g ran slicing for industry 4.0. *IEEE Access*, 7:143139–143159, 2019.

[4] Mohammed Zaki Hasan, Hussain Al-Rizzo, and Fadi Al-Turjman. A survey on multipath routing protocols for qos assurances in real-time wireless multimedia sensor networks. *IEEE Communications Surveys Tutorials*, 19(3):1424–1456, 2017.

[5] Erfan Mozaffari Ahrar, Mohammad Nassiri, and Fabrice Theoleyre. Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks. *Journal of Network and Computer Applications*, 142:25–36, 2019.

[6] Mark Nixon. A comparison of wirelesshart and isa100.11a. White Paper HCF-SPEC, Emerson, Sept. 2012.

[7] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Théoleyre. Is link-layer anycast scheduling relevant for ieee 802.15.4-tsch networks? In *LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, pages 133–140, 2019.

[8] Q. Wang, X. Vilajosana, and T. Watteyne. 6TiSCH Operation Sublayer (6top) Protocol (6P). RFC 8480, IETF, November 2018.

[9] Fabrice Theoleyre and Georgios Z. Papadopoulos. Experimental validation of a distributed self-configured 6tisch with traffic isolation in low power lossy networks. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 102–110, 2016.

[10] 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*, 114:84 – 105, 2017.

[11] X. Vilajosana, K. Pister, and T. Watteyne. Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration. RFC 8180, IETF, May 2017.

[12] S. Kulkarni, A. Iyer, and C. Rosenberg. An address-light, integrated MAC and routing protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 14(4):793–806, Aug 2006.

[13] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. Let the tree Bloom: scalable opportunistic routing with ORPL. In *SenSys*, page 2. ACM, 2013.

[14] Remous-Aris Koutsiamanis, Georgios Z. Papadopoulos, Tomas Lagos Jenschke, Pascal Thubert, and Nicolas Montavont. Meet the pareo functions: Towards reliable and available wireless networks. In *International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[15] Thong Huynh, Fabrice Theoleyre, and Won-Joo Hwang. On the interest of opportunistic anycast scheduling for wireless low power lossy networks. *Computer Communications*, 104:55 – 66, 2017.

[16] Oliver Harms and Olaf Landsiedel. (poster) overtake: Opportunistic routing and concurrent transmissions for tsch. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 141–143, 2020.

[17] I. Hosni and F. Theoleyre. Adaptive k-cast Scheduling for high-reliability and low-latency in IEEE802.15.4-TSCH. In *ADHOC-NOW*, volume 11104 of *LNCS*, pages 1–12. Springer, Sep 2018.

[18] Simon Duquennoy, Joakim Eriksson, and Thiemo Voigt. Five-nines reliable downward routing in rpl, 2017.

[19] Esteban Municio, Glenn Daneels, Mališa Vučinić, Steven Latré, Jeroen Famaey, Yasuyuki Tanaka, Keoma Brun, Kazushi Muraoka, Xavier Vilajosana, and Thomas Watteyne. Simulating 6tisch networks. *Transactions on Emerging Telecommunications Technologies*, 30(3):e3494, 2019. e3494 ett.3494.

[20] Remous-Aris Koutsiamanis, Georgios Z. Papadopoulos, Xenofon Fafoutis, Julián Martín Del Fiore, Pascal Thubert, and Nicolas Montavont. From best effort to deterministic packet delivery for wireless industrial iot networks. *IEEE Transactions on Industrial Informatics*, 14(10):4468–4480, 2018.

[21] Tengfei Chang, Mališa Vučinić, Xavier Vilajosana, Simon Duquennoy, and Diego Roberto Dujovne. 6TiSCH Minimal Scheduling Function (MSF). RFC 9033, May 2021.

[22] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. Experimental in-depth study of the dynamics of an indoor industrial low power lossy network. *Ad Hoc Networks*, 93:101914, 2019.

[23] Tao Liu and Alberto E. Cerpa. Data-driven link quality prediction using link features. *ACM Trans. Sen. Netw.*, 10(2), January 2014.

[24] Fabrice Theoleyre. Experiment plan for openwsn with durocast. https://github.com/ftheoleyre/openwsn-iotlab/tree/duocast, https://doi.org/10.5281/zenodo.5499988, Sept. 2021.

[25] Fabrice Theoleyre. Openwsn firmware with ducocast. https://github.com/ftheoleyre/openwsn-fw/tree/duocast, https://doi.org/10.5281/zenodo.5499997, Sept. 2021.

[26] Fabrice Theoleyre. Openvisualizer for openwsn with duocast. https://github.com/ftheoleyre/openvisualizer/tree/duocast, https://doi.org/10.5281/zenodo.5499993, Sept. 2021.

[27] Fabrice Theoleyre. Data processing pipeline. https://github.com/ftheoleyre/openwsn-data/tree/duocast, https://doi.org/10.5281/zenodo.5499999, Sept. 2021.

[28] Fabrice Theoleyre. Indoor wireless deterministic anycast transmissions data from the fit iot-lab testbed. Zenodo dataset, https://doi.org/10.5281/zenodo.5341805, August 2021.