Notice

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.
BDPC: Controlling Application Delay in the Industrial Internet of Things for 6TiSCH networks

Lucas Aimaretto*, Diego Dujovne†
*Facultad de Ciencias Exactas, Físicas y Naturales, Universidad Nacional de Córdoba
Email: laimaretto@mi.unc.edu.ar,
†Escuela de Informática y Telecomunicaciones, Universidad Diego Portales
Email: diego.dujovne@mail.udp.cl

Abstract—One of the essential requirements of the Industrial Internet of Things (IIoT), is to have an extremely high packet delivery rate, generally over 99.9%. However, packets which arrive after a predefined deadline shall be considered lost too. Industrial applications require a predictable delay. To solve this problem, we propose a new mechanism, called BDPC (Bounded Delay Packet Control). BDPC combines the knowledge of a node’s delay to its root with the time budget of a data packet traversing the IoT network, to allocate cells in 6TISCH slotFrames in order to fulfill the application’s maximum delay requirements in a controlled manner: the application packets must arrive before the deadline, but not faster. This is achieved by allocating cells from a parent node to a child node, thereby adapting the cell capacity and attaining the bounded delay goal, by the analysis of the new variable latePags. In other words, the resource allocation is a function of latePags. Moreover, the number of packets arriving after the predefined deadline can be controlled by two thresholds: sfMax and sfMin. Our results show that using BDPC, the number of packets arriving before the deadline can be improved more than 2.6 times compared to the case when using the default Minimal Scheduling Function from the standard. As a further advantage, BDPC involves minor modifications to the 6TISCH protocol stack, which makes it compatible with current implementations.

Index Terms—6TISCH, Bounded, Control, Delay

I. INTRODUCTION

In the context of the the Industrial Internet of Things (IIoT), it is fundamental to guarantee an extremely high packet delivery rate [1], and to satisfy realtime performance, data packets must also arrive within a limited time frame. A packet which is delayed after the expected deadline is discarded by the application, and should be considered lost too. Currently, the standardized Architecture of Deterministic Networks [2] include both minimum Packet Delivery Rate requirements and time-bounded data packet delivery, or bounded latency.

However, Deterministic Networks are generally defined for wired solutions, while wireless Industrial IoT has been following a parallel path. One of the most developed wireless Industrial IoT implementations is the Time Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4 standard, which defines how nodes reserve resources according to a channel-timeslot schedule matrix, called slotframe. The slotframe matrix determines a number of cells, where a pair of nodes over a communication link can exchange packets. The IPv6 over the TSCH mode of IEEE 802.15.4e (6TISCH) stack establishes the mechanisms for resource allocation, leaving to the Scheduling Functions (SFs) the implementation of the algorithms that decide when and how to allocate resources [3].

Scheduling Functions are the core of the allocation intelligence in 6TISCH-based IoT networks. SFs implement an algorithm to improve a specific network characteristic such as delay, power consumption, network lifetime and packet delivery rate. Among the wide range of proposed SFs, there is a subgroup where the aim is to reduce the end-to-end delay by dynamically allocating resources in the data plane. However, to the best of our knowledge, there is no SF currently designed to maximize the Packet Delivery Rate within a certain application-defined deadline. As a consequence, we have compiled in the state of the art the most representative SFs which minimize packet delay, as the closest reference.

First, MSF [4] is the Minimal Scheduling Function, which is the full-featured standard SF used by default within the 6TISCH stack. MSF recommends increasing the number of cells in the data plane to decrease latency, always at the cost of higher power consumption.

Second, the Low Latency Scheduling Function (LLSF) [5] is a SF designed to decrease end-to-end latency in 6TISCH multi-hop networks, using a daisy-chain approach. The transmission sequence is defined by a succession of allocated timeslots incrementally shifted in time for each node in the network. Once a packet is received in a rx timeSlot, it is automatically transmitted to the next node in the following rx timeSlot. The results in [5] show that the end-to-end delay is 82.8% lower than using SF0 [6] for a 5-hop linear topology.

Third, the Low-latency Distributed Scheduling Function (LDSF) [7] algorithm is oriented to minimize the end-to-end delay while providing high reliability. LDSF is able to generate a schedule for a large number of transmissions, also managing the buffering delay when oversubscription cells are chained. In order to achieve this goal, LDSF divides a long slotframe into small blocks that repeat over time. Each node then selects the right block corresponding to its hop distance to the border router which minimizes delay.

The concepts of primary and ghost cells are also defined. The primary cells correspond to the earliest expected reception time of a data packet from the previous hop. To ensure network reliability, ghost cells correspond to the retransmission opportunity in the event that a receiver has not received the data.
packet during the primary cell. The ghost cells are allocated at the same timeSlot offset in subsequent subFrames.

By performing simulations, LDSF delivers a $PDR_{2c}$ above 98.5% with a network lifetime of 4 years, for a traffic flow of 1 packet every 30s. In this scenario, the end-to-end delay is 250ms.

Finally, the Recurrent Low-Latency Scheduling Function (ReSF) [8] is a SF that focuses on optimizing the delay of applications with periodic traffic.

This function uses a distributed method to allocate resources through 6P [9], focusing on building a resource reservation mechanism that minimizes recurring communication latencies, aimed towards IoT systems where devices perform periodic data transmission. ReSF assumes that the source node is aware of the traffic period pattern. With this information, ReSF is able to generate a recurring path, which consists of cells that are activated only when traffic is expected and deactivated otherwise. ReSF allows nodes to make reservations that are based on the ASN where it started generating data, the ASN where it stops generating data and the repetition period. This information is sent to the next hop until it reaches the destination, thus reserving resources for each node along the path.

By performing simulations, ReSF shows that the latency, for a network size of 25 elements and a traffic flow of 1 packet per minute, is close to 0.5 seconds with a $PDR_{2c}$ of 100%.

Although the former proposals show that delay can be reduced, they do not consider first, a maximum delay goal i.e. the application-level deadline; this is expected in a deterministic environment, for example, if an application running on a 6TiSCH-based network requires that the packets must arrive to destination before a maximum deadline. Second, they do not consider the effect of jitter of the arriving packets, which increment packet loss for those arriving later than the deadline; and third, even though the former proposals provide a high $PDR_{2c}$, they do not take into account the $PDR_{2c}$ for a specific application deadline, considering that late packets are lost packets.

The most important characteristic in time-related tasks, is the requirement of time expressed as a function of its maximum limit, called the deadline. There are two kinds of deadlines [10]: hard deadlines and soft deadlines. The hard deadline case establishes a conditioning over the maximum limit in which the task must be completed, while if the condition is a soft deadline, the task is less strict on the arrival time. The hard deadline case is mostly related to real time applications, such as industrial control systems. Thus, for the hard deadline case, if any task is completed after the maximum deadline, the outcome is discarded, even if the outcome is correct. Hard and soft Time Utility Functions (TUF) can be observed in Fig. 1.

Our proposal, BDPC (Bounded Delay Packet Control) deals with real-time applications with hard timing requirements and solves this problem combining upstream and downstream information in order to allocate cells in the reverse direction, i.e. on the child node link, so as to provide enough resources to let the highest number of packets reach the destination before the expected deadline. This is a completely different approach from the former delay optimization proposals: the application packets must arrive before the deadline, but not necessarily faster. Further, BDPC is agnostic to traffic pattern, flow isolation or topology structure.

Our solution comprises the following contributions:

- **BDPC** is the first SF that combines the estimated delay to the root node (upstream information, $d_{2r}$) with the packet time budget ($time_{Left}$, downstream information) to allocate resources. Thus, the $latePaqs$ variable is the result of the combination of $d_{2r}$ and $time_{Left}$.

- Unlike most SFs, which allocate data cells to the Next-Hop, BDPC assigns data cells to the previous node, or Pre-Hop, based on the analysis of the $latePaqs$ variable.

- **BDPC** enables the control over the percentage of packets allowed to arrive after the pre-established deadline at their destination, with only two parameters $sfMax$ and $sfMin$.

The rest of the paper is organized as follows: Section II presents the current mechanisms and provides motivation for this work; section III presents the BDPC algorithm and the full compatibility with current standards. In section IV, we present the results based on simulation work; and finally in section V we conclude our work.

## II. Motivation

The resource scheduling algorithms for the Industrial Internet of Things must satisfy the requirements of each data flow, specially in control networks. SFs which reduce delay generally achieve this goal as a global process, where no packet delay information is taken into consideration to make a decision. We propose to use the packet time budget, plus the estimation of a node’s delay to the root, to allow the SF allocate resources and adapt to the dynamic data flow requirements.
III. Proposed Solution

A. Overview

BDPC is a system based on a simple and modular architecture contained within each of the network intermediate nodes (Fig. 3). It is designed to run as part of the 6TiSCH protocol stack and interacts with 6P and the TSCH MAC layer (Fig. 5). Algorithms 1 and 2 are complementary: they calculate and decide packet allocation respectively. The $DB_{data}$ and $DB_{dio}$ databases store data packet statistics, and delay to root measurements carried by the DIO RPL [13] packet. The $sfMax$ and $sfMin$ thresholds are externally configurable parameters.

Algorithm 1 estimates the late packet rate, $latePaqs$. When an intermediate node’s SF receives a data packet, it analyzes the 6LoRHE of the packet and verifies if the remaining time, $timeLeft$, is enough to reach the root node. The comparison is done against the delay of the intermediate node towards the root node, $d2r$. If the time is not enough, the $delayed$ variable is increased; otherwise, $inTime$ is increased. Finally, $latePaqs$ is calculated and stored in a database. The late packet rate is fundamental to the operation of BDPC because it gives an intermediate node information about each child’s current overtime packet arrival statistics.

Algorithm 2 uses $latePaqs$, and compares it against the $sfMax$ and $sfMin$ thresholds. Using 6P, it performs one of these possible options:

- add a RX cell towards the previous node;
- remove a RX cell towards the previous node.

\[
\text{alg1}(\text{deadLine}, d2r) \xrightarrow{\text{store}} \text{latePaqs} \\
\text{alg2}(\text{latePaqs}, sfMax, sfMin) \xrightarrow{6P} (\text{addCell/delCell})
\]

Algorithm 1 Estimation of $latePaqs$

\[
timeLeft = \text{deadLine}_{\text{packet}} - \text{currentTime}_{\text{asn}} \\
d2r \xleftarrow{\text{get}} DB_{dio} \\
inTime \xleftarrow{\text{get}} DB_{data} \\
delayed \xleftarrow{\text{get}} DB_{data}
\]

if $timeLeft >= 0$ and $timeLeft >= d2r$ then

\[
inTime = inTime + 1
\]
else

\[
delayed = delayed + 1
\]
end if

\[
latePaqs = \frac{delayed}{(delayed + inTime)}
\]

\[
\text{latePaqs} \xrightarrow{\text{store}} DB_{data} \\
inTime \xrightarrow{\text{store}} DB_{data} \\
delayed \xrightarrow{\text{store}} DB_{data}
\]
Algorithm 2 Cell assignment to the PreHop

\[
\text{latePaqs} \leftarrow \text{get } DB_{data} \\
\text{if } \text{latePaqs} \geq \text{sfMax} \text{ then} \\
\text{addCell(PreHop)} \\
\text{else if } 0 \leq \text{latePaqs} \text{ and latePaqs} < \text{sfMin} \text{ then} \\
\text{delCell(PreHop)} \\
\text{end if}
\]

![Figure 3. BDPC system at an intermediate node](image)

**B. System Model**

The single path end to end data delay, is the difference between the arrival time of a data packet at the root node, and the departure time from an intermediate node: \( d = t_{rx} - t_{tx} \).

It can be shown by fitting the results of Fig. 14, that the delay is beta distributed, i.e., \( d \sim \text{Beta}(\alpha, \beta) \).

Further, the end to end data delay can be modeled as:

\[
d = \sum_{i=1}^{N-1} t_{q_i}
\]

Where \( d \) is the end to end data delay, \( N \) is the amount of nodes forming a single path towards the root node and \( t_{q_i} \) is the queuing time, for each consecutive link \( i \).

The queuing time is the time a packet waits in the queue before being put into the radio interface towards the next hop.

In a 6TiSCH network, the TSCH slotFrame is composed of cells that can be of type SHARED, TX, RX, or a combination of those. According to [4], the delay can be reduced by adding cells into the schedule so data packets will have extra capacity for their journey. The idea of the paper is to modify the parameters \( \alpha \) and \( \beta \) of \( d \sim \text{Beta}(\alpha, \beta) \), using algorithm 1: the mean of the beta distribution, \( \alpha / (\alpha + \beta) \), is equal to the variable \( \text{latePaqs} \). Using algorithm 2 we can add RX cells from a parent node to a child node and as such, increment \( \beta \) (Fig. 5). Thus, the probability of the delay being lower than or equal to its maximum deadline, \( \text{maxDelay} \), is equal to \( (1 - \text{sfMax}) \).

Formally:

\[
P(d \leq \text{maxDelay}) = 1 - \text{sfMax} \tag{1}
\]

Hence, \( (1 - \text{sfMax}) \) is the probability that packets traversing the network towards the root node, will make it in time.

**C. Delay to the root node estimation**

The DODAG Information Object (DIO) messages of the IPv6 Routing Protocol for Low-Power and Lossy Networks can be used to provide an intermediate node with a delay reference to the root, from the packet structure provided by the RPL standard [13].

In this proposal, a node generating DIO messages includes additional information to allow the child nodes become aware of the delay to the root. When a child node receives a DIO message with timing information from its parent node, it updates the delay to the root node value in the internal \( DB_{dio} \) database. Timing information in DIOs must only be considered by a node when coming from a parent.

The DIO messages traverse the network using the Minimal-Cell [4] specification. Consequently, the intermediate nodes will not observe any changes in the delay to the root over time (Fig. 9). However, this phenomena does not constitute a problem from a global performance point of view, since only the result of the combination of the system parameters mentioned earlier is what triggers the cell allocation in the data plane to the previous node.

**D. Cell assignment to the PreHop**

The usual allocation type implies the assignment of a TX cell from a child to its parent. BDPC initiates a RX cell request to each child, where the child is identified by the source MAC address of the sent frames. This means that each parent can seamlessly manage cell allocation among children independently. In other words, BDPC is agnostic to topology structure.

![Algorithm 2](image)

Fig. 6 shows the allocation procedure to the PreHop along the network. The transmitted IP packet includes a field inside 6LoRHE (Fig. 2) with the \( \text{deadline} \) value required by the application. The packet departs from the leaf node towards the root node. On each intermediate node, algorithms 1 and 2 will process the deadline as one of the inputs. Depending on the outcomes of the algorithms 1 and 2, a RX type cell can be added or removed in the PreHop direction (i.e. the previous hop). We must emphasize that the addition of a RX type cell to the PreHop automatically instructs the local TSCH layer of the PreHop to install a TX type cell towards the next-hop (in the direction of the root node).

The assignment process uses the 6P [9] protocol, which is the same mechanism that MSF is already using when adding or removing cells towards the next-hop. Both MSF and BDPC use the same two-way 6P standard hand-shake for the assignment.

Every time a node needs to assign cells, it triggers a 6P request to its neighbor (if MSF, the next-hop; if BDPC, the pre-hop) to perform such an assignment. The client will offer a list
of possible cells, out of which the server will decide and confirm back. Fig. 4 shows the 6P two-way handshake for cell-assignment between nodes.

The cell allocation from the parent to the child node (Pre-Hop), managed by algorithm 2 interacts seamlessly with the allocation mechanism of cells to the preferred parent (Next-Hop) as defined on MSF [4]. This means that, if required, the number of added cells by BDPC in the reverse direction can be removed by MSF in the child-parent direction of the same link.

Thus, the solution integrates two complimentary behaviors: (i) the one defined by the standard, MSF, towards the preferred parent and (ii) the one performed by BDPC, towards the previous node. This is shown in Fig. 5.

E. latePaqs

In a 6TiSCH network, the path to the root node is handled by the RPL protocol [13]. RPL uses the well-known Rank value. The Rank value represents an abstract distance to the root of the DODAG tree. The Rank value is exchanged between the nodes using DIO messages, and allows other RPL nodes to avoid loops and to build the routing topology of the network that will enable data packets to arrive at the root node [14]. As such, the Rank value is an always increasing strictly positive number that gives the node an approximation of its topological situation within the network.

Unlike the Rank value, latePaqs is not exchanged between nodes, but inferred by each intermediate node by using algorithm 1. Inside the $DB_{data}$ database, each intermediate node stores a version of latePaqs for each of their children. Each child is identified by the source MAC address of the receiving data frames in their path to the root node and each version of latePaqs gets updated whenever the intermediate node receives a data frame from its children.

In the same way that the Rank provides a node information about its topological situation, latePaqs provides an intermediate node the instantaneous number of packets that will not arrive before the deadline at the root node. A packet is considered a late packet when the estimated time lapse to arrive at the root node from an intermediate node, is larger than the packet’s own remaining time. latePaqs accumulates all the packets arriving fulfilling the former condition from each of its children. In other words, latePaqs informs an intermediate node the rate of packets that are already late at that precise and moment.

Consequently, latePaqs becomes the fundamental element of BDPC: It is calculated by algorithm 1 and it is used as an input to the Scheduling Function to define if RX cells must be added or removed to the appropriate child (Fig. 5).

Fig. 7 shows an example of the different latePaqs measurements that each intermediate node obtains from its children. When MSF is used, latePaqs is hardly reduced and hence an important number of data packets arrive late at the root node. On the other hand, when BDPC is used, starting from the third group, the latePaqs starts to decrease because of the added cells in the parent-child direction, which is coherent with Fig. 11.

IV. SIMULATION

A. Setup

The 6TiSCH simulator [15] includes the standard protocol implementations for MSF, 6P, and RPL in the 6TiSCH network stack. The RPL DIO message was modified to add the delay to root information.

We performed three experiments with 30 different seeds each. The first two experiments evaluate the performance of BDPC for different values of $sfMax$ and $sfMin$, while the remaining experiment is used as a reference, running the original MSF implementation.

The length of each simulation run is 10000 slotFrames (approximately 2.8 hours in simulation time) in order to allow the system stabilize after the initial network convergence stage.

The network is a hierarchy of groups of nodes (Fig. 8), each group containing three nodes. In the first group, each node has a link towards the root node, which is shown in green. In the rest of the groups, each node has links towards every node in
the group above, and also towards every node in the group below: nodes in even groups cannot be seen among them. This topology allows for a dynamic parent change situation along the simulation run. Also this setup of links gives the topology a certain depth that will allow to visualize the effect of delay on the data packets traversing the network. The values of PDR = 100% and RSSI = −10dB for each of the links, correspond to the default values provided by the simulator.

All the nodes (except for the root) generate data traffic where the destination is the root node. This means that there are multiple flows of data packets traversing the network in parallel, going from the child nodes towards the root node. Each node generates 90-Byte data packets that are transmitted every 30s, with 0.05 variance. The application maximum delay threshold (maxDelay) is configured to 1.5s. The timeSlot duration is 10ms and the slotFrame is composed of 101 timeSlots, both default values from the standard. Table I summarizes the simulation parameters.

### B. Results

Fig. 9 shows the average delay observed by a node within its group, with respect to the root node, based on the information provided by the DIO messages. The horizontal dotted line shows the maximum delay supported by the application (maxDelay). The delay to the root, d2r, remains constant across the experiments because the DIO messages use the Minimal-Cell, which belongs to the control plane, while BDPC manages the Data plane delay.

The application packets delay measured at the root node is shown in Fig. 10. Using MSF, the average delay exceeds the maximum application allowed delay when the nodes reside on group-2 and backwards; however, when BDPC is enabled, the average delay stays below the maximum delay until the packets arrive at the root node: the packets transmitted by a node within group-5, show an average delay measured at the
Figure 7. Example of the evolution of latePqgs along the network. When MSF is used, latePqgs is hardly reduced. On the other hand, when BDPC is used, starting from the third group, the latePqgs starts to decrease because of the added cells in the parent-child direction. The width of the links represent the traffic load, showing the path from leaf nodes to the root node. The darker the links, the better latePqgs measurement. For example, when using MSF, 77% of the packets received by the root node from node 3, were late packets. Conversely, when using BDPC, only 1% of the packets that the root node received from node 3, were late packets. It is important to note that the outgoing arrows mean aggregated traffic: the packets delivered by node 3 to the root node is a compound of its own packets plus the packets received from its child nodes.

Figure 8. Simulation topology

Figure 9. DIO Delay to root measured at each node. The horizontal black-dashed line represents the maximum application delay, maxDelay. The shaded area around the average delay to root, is the standard deviation for the series.

Fig. 12 describes the number of late packets arriving to the root from each source group. As such, 44% of the packets which were generated from group 2 arrived after the deadline when using MSF as the Scheduling Function. On the other hand, when using BDPC, this rate drops down to less than 6% for packets generated at the same group. Thus, when BDPC is used, the smaller sfMax is, the lower the percentage of packets arriving late at the destination. For example, for data packets leaving a node within group-5, the average rate of packets arriving late at root is 97.34% when using MSF; when
Figure 10. End-to-End Data packet delay measured at the root node. The horizontal black-dashed line represents the maximum application delay, maxDelay. The shaded area around the average delay, is the standard deviation for the series.

Figure 11. Evolution of latePaqs along the network. The shaded area around the average latePaqs, is the standard deviation for the series. An example over the network topology can be seen in Fig. 7.

Using BDPC this rate drops to 14.06% and 0.29% for sfMax values of 0.1 and 0.0001, respectively.

By comparing the results of Fig. 11 and Fig. 12, i.e., latePaqs versus the actual rate of packets arriving late at the root node, it can be observed that even though latePaqs is high in groups away from the root, the number of packets originated in such groups (i.e., group-5) that actually arrive late when using BDPC, is less than 20% while for the case of MSF is close to 100% of late packets. Since latePaqs measures the rate of packets arriving late at an intermediate node, the lower latePaqs becomes means lesser late packets arriving late at their destination. As a consequence, BDPC has the objective of reducing latePaqs by using algorithm 2 and thus adding RX cells to the pre-hop. This results in a latePaqs decrease in nodes closer to the root node and, as a consequence, a decrease in the actual rate of packets arriving late at their destination. Table II, shows in detail the mean percentage of packets arriving late versus its corresponding value of latePaqs for data packets originated in groups for different number of hops from the root node.

Fig. 13 shows a different analysis from a combination of what can be observed on Fig. 11 and Fig. 12. Fig. 13 is a scatter plot that depicts the distribution of the percentage of late packets at the root node for the full range of values for latePaqs. It can be observed that latePaqs has a different distribution for each group: The measurements of latePaqs are concentrated to the right for groups away from the root node (i.e., group-5), which follows an increasing percentage of late packets. However, an increase of latePaqs rate using BDPC does not follow an increase of the number of data packets arriving late at the root node. However, this is not the case for MSF, where an increment of latePaqs results in an increment of the rate of late packets at the root node. The average packet delivery rate values for the three configurations can be observed in Table II.

The overall E2E delay performance of the network can be observed in Fig. 14. The vertical dotted line shows the maximum delay defined by the application, while the horizontal dotted lines represent the value of (1 − sfMax). According to
Figure 13. Evolution of latePaqs along the network with the corresponding late packets arrival rate, at the root node. Table II summarizes the graphics’ average values for groups 1 and 5.

Figure 14. End-to-End data packets delay. The vertical black-dashed line represents the maximum delay tolerated by the application, maxDelay. The horizontal red and green dashed lines, represent the values of \((1 - sf\text{Max})\), for sfMax values of 0.1 and 0.0001, MSF performance shows that only 38.5% of the data packets will arrive within the deadline, while BDPC guarantees that the packets arriving in term will be 92.5% and 99.8% for sfMax values of 0.1 and 0.0001, respectively.

Eq. (1), this value represents the overall rate of packets that will arrive at the destination within the deadline.

Moreover, MSF performance shows that only 38.5% of the data packets will arrive within the deadline, while BDPC guarantees that the packets arriving in term will be 92.5% and 99.8% for sfMax values of 0.1 and 0.0001, respectively. If MSF is expected to achieve a packet delivery (before deadline) rate of 90%, the maximum delay tolerated by the application must be 4s or longer. Consequently, not only BDPC enables a better deadline compliance, but also the sfMax parameter can finely control the overall rate of packets that will arrive within the deadline.

Fig. 15 and Fig. 16 show the packet delivery rate (PDR) at the application level. The first figure shows the rate of packets that successfully arrived at the root node, for each of the series. Since the PDR at the link level is configured to 100%, the PDR at the application level is close to 99.975%. However, when the packet deadline is used as a criteria to discard packets, Fig. 16 shows that BDPC guarantees the arrival of more than 90% of the packets, compared to MSF which guarantees less than 40% of the packets arriving on time. Table III summarizes the graphics’ values as a reference.

Fig. 17 correlates the results of Fig. 15 and 16. The horizontal red and green dashed lines, represent the values of \((1 - sf\text{Max})\), for sfMax values of 0.1 and 0.0001 respectively. In this figure it can be observed that even though all the alternatives deliver a \(PDR_{e2e}\) above 99.8%, only when using BDPC the packets arrive before the deadline. Furthermore, the smaller sfMax is, the lesser jitter can be obtained for the packets arriving before the deadline.

But yet, why does it work? The previous shown results have presented the performance in terms of \(PDR_{e2e}\) before \(maxDelay\) (Fig. 17), end-to-end delay distribution (Fig. 14) and the distribution of the latePaqs variable (Fig. 7 and 13). Fig. 18 shows the dedicated reserved TX cells along the simulation run. For the case of MSF, after 2000th slotFrame, the scheduling function has already clustered all the necessary cells for the current traffic load. On the other hand, BDPC senses the traffic value every time a packet traverses a node, and precisely adjusts the required cells to agree to the application deadline requirements (Fig. 6). Once the traffic has converged, BDPC only triggers little modifications to the schedule, by means of algorithm 2. In general BDPC with sfMax=0.1 requires less TX cells to cope with the application requirements, while BDPC with sfMax=0.0001 requires the double of capacity, on average, for the same purpose.

Finally, Fig. 19 shows the expected lifetime for each group in the network, for each of the experiments\(^4\). The life of the

\(^4\)To calculate the node lifetime, the charge of a 2821.5mAh AA battery is considered as a reference. The energy consumption is calculated by counting the number of TX/RX/ACK/Listen/Sleep radio operations performed by a node.

| run           | PDR_{e2e} w/deadline | PDR_{e2e} w/o deadline |
|---------------|-----------------------|-------------------------|
| BDPC-L2 sfMax=0.0001 | 0.99766               | 0.99972                 |
| BDPC-L2 sfMax=0.1       | 0.92459               | 0.99968                 |
| MSF             | 0.38329               | 0.99971                 |
network. Table III summarizes the graphics’ average e2e PDR values as a reference.

From this point of view, all experiments behave similarly\(^5\). Consequently, the average lifetime of the network using BDPC is 3.06 and 3.11 years, for the experiments with \(sfMax\) values of 0.1 and 0.0001, respectively while the network would run 3.48 years, with MSF, a lifetime increase of only 4.4 months.

Nodes closer to the root with a lower power consumption than nodes located farther away is counter-intuitive; this is a result of the network cell allocation process using management frames. Nodes closer to the root need less packet management exchanges given that the traffic is already clustered, while power consumption at the extremes requires more negotiation instances.

By sacrificing 11.8% of the network lifetime when using MSF, 99.8% of the packets could reach their destination within the deadline.

V. CONCLUSION

This paper proposes a new approach to achieve realtime capabilities in wireless IIoT. In the industrial context, two critical factors define performance: an extremely low packet loss rate and deadline compliance at application data packets. Packets arriving late are discarded at the application layer: guaranteeing the delivery of data packets within a maximum and predictable deadline is of utmost importance. In other words, it is not only important to guarantee an extremely high
**Figure 19.** Node lifetime. The lifetime of the network is defined by the period until at least one of the nodes in the network depletes its energy storage and stops responding.

$PDR_{2e}$, it is also important to guarantee an extremely high $PDR_{e2e}$ within the application deadline.

We solve this problem with BDPC (Bounded Delay Packet Control). BDPC is a mechanism that combines the knowledge of the maximum delay tolerated by the application with the observed delay towards the root node to allocate network resources. Additionally, we create the variable latePaqs. latePaqs informs the intermediate node about the rate of late packets traversing it. We also define two thresholds called sfMax and sfMin to establish the number of packets allowed to arrive later than the deadline, and finally, we establish a new allocation request in the inverse, parent-child direction (PreHop). On top of that, BDPC is agnostic to traffic pattern, flow isolation or topology structure.

BDPC shows a substantial improvement over the standard behavior of MSF, where the network delivers 99.8% of the packets within the deadline, for a value of $sfMax = 0.0001$, compared to MSF, which guarantees on-time delivery of only 38.5% of the packets, for the same network and application with only a network lifetime decrease of 11.8% (equivalent to 4.4 months).

Finally, this proposal has the potential to be implemented in the current standard, given that the solution only involves minor modifications to the 6TiSCH protocol stack.

**ACKNOWLEDGMENT**

The authors would like to thank the CYTED AgIoT Project (520rt011), CORFO CoTH2O “Consorcio de Gestión de Recursos Hídricos para la Macrozona Centro-Sur”, Proyecto Asociativo UDP “Plataformas Digitales como modelo organizacional” and 19STIC-09 “WirelessWine” STIC-AmSud, for their support during the development of this research work.

**REFERENCES**

[1] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, “6tisch: deterministic ip-enabled industrial internet (of things),” *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36-41, 2014.
[2] N. Finn, P. Thubert, B. Varga, and J. Farkas, “Deterministic Networking Architecture,” RFC 8655, Oct. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8655.txt
[3] X. Vilajosana, K. Pister, and T. Watteyne, “Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration,” RFC 8180, May 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8180.txt
[4] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, “6TiSCH Minimal Scheduling Function (MSF),” RFC 9033, May 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc9033.txt
[5] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, “Lsf: Low latency scheduling function for 6TiSCH networks,” in *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2016, pp. 93-95.
[6] D. R. Dujovne, L. A. Grieco, M. R. Palatella, and N. Accettura, “6TiSCH 6top Scheduling Function Zero (SF0),” Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-6top-sf0-05, Jul. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-6top-sf0-05
[7] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, “LDSF: Low-latency distributed scheduling function for industrial internet of things,” *IEEE Internet of Things Journal*, 2020.
[8] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey, “Rsf: Recurrent low-latency scheduling in IEEE 802.15.4e TSCH networks,” *Ad Hoc Networks*, vol. 69, pp. 100–114, 2018.
[9] Q. Wang, X. Vilajosana, and T. Watteyne, “6TiSCH Operation Sublayer (6top) Protocol (6P),” RFC 8480, Nov. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8480
[10] S. Ryu, B. Ryu, H. Seo, and M. Shin, “Urgency and efficiency based packet scheduling algorithm for ofdma wireless system,” in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, vol. 4. IEEE, 2005, pp. 2779–2785.
[11] L. Thomas, S. Ananalamudi, S. Anand, M. Hegde, and C. E. Perkins, “Packet Delivery Deadline Time in the Routing Header for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs),” RFC 9034, Jun. 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc9034.txt
[12] P. Thubert and R. Cragie, “IPv6 over low-power wireless personal area network (6lowpan) paging dispatch,” *IETF RFC 8025*, 2016.
[13] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks,” RFC 6550, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt
[14] P. Thubert, “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL),” RFC 6552, Mar. 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6552
[15] E. Municio, G. Daneels, M. Vučinić, S. Latré, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, “Simulating 6tisch networks,” *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 3, p. e3494, Mar. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3494
[16] S. K. Sunori, P. K. Juneja, M. Chaturvedi, and J. Mittal, “Dead time compensation in sugar crystallization process,” in *Proceeding of International Conference on Intelligent Communication, Control and Devices*, Springer, 2017, pp. 375–381.
[17] K. Lim and K. Ling, “Generalized predictive control of a heat exchanger,” *IEEE Control Systems Magazine*, vol. 9, no. 6, pp. 9–12, 1989.