Downlink Scheduling in LoRaWAN: ChirpStack vs The Things Stack

Carlos Fernandez Hernandez, Oana Iova, Fabrice Valois

Downlink Scheduling in LoRaWAN: ChirpStack vs The Things Stack

Carlos Fernandez Hernandez
INSA Lyon, Inria, CITI
UR3720, 69621 Villeurbanne, France
carlos.fernandez-hernandez@insa-lyon.fr

Oana Iova
INSA Lyon, Inria, CITI
UR3720, 69621 Villeurbanne, France
oana.iova@insa-lyon.fr

Fabrice Valois
INSA Lyon, Inria, CITI
UR3720, 69621 Villeurbanne, France
fabrice.valois@insa-lyon.fr

***Abstract*—LoRaWAN is a key enabler for Internet of Things applications, providing long range communication at a low energy consumption. However, extensive use of downlink communication can reduce network capacity and increase uplink loss, as shown in several studies. The network server is responsible for scheduling downlink packets, while respecting strict timing constraints on the end devices and on the gateways. This paper presents the first evaluation of downlink scheduling implemented by two widely used LoRaWAN network servers: ChirpStack and The Things Stack. Using the ELoRa emulation tool, we inject real LoRaWAN traffic into these network servers and analyze their downlink scheduling algorithms based on defined performance metrics such as reliability and schedule efficiency. Our findings indicate that The Things Stack has a more accurate scheduler, but more conservative, as it delivers less downlink packets. ChirpStack, on the other hand, with its simpler algorithm, performs comparably and even equals The Things Stack as the number of end devices increases.**

***Index Terms*—LoRaWAN, downlink scheduling, emulation, fairness, performance**

I. Introduction

LoRaWAN [1] is one of the main communication technologies for the Internet of Things (IoT), due to its long radio range and its energy efficiency. LoRaWAN supports both uplink communication (UL), where end devices send information to the application and network servers, and downlink communication (DL), where application and network servers send information to end devices. The UL is massively used to collect data from IoT devices, while the DL is used for control purpose, acknowledgements, device activation, and firmware update, as well as for information sent from application servers to end devices.

Several studies have shown that the use of DL communication impacts overall network performance, such as loss of UL packets and reduced network capacity, especially in dense networks [2–4]. The main reasons for this drop in performance are: (i) half-duplex radio: the gateway cannot transmit and receive data at the same time; (ii) single transmitter: the gateway can receive multiple ULs in parallel, but can only transmit one DL at a time; (iii) duty cycle and dwell time: the time spent by a gateway sending DL packets is limited by frequency regional regulations [5].

We notice that the bottleneck in DL communication is the LoRaWAN gateway. For this reason, several studies have focused on evaluating the communication between gateways and end devices [6]. However, the gateway only acts as a relay between the network server and end devices. As we can see in Fig. 1, the UL sent by an end device is received by all the gateways in its communication range, and then it is forwarded to the network server for de-duplication. When a DL packet needs to be sent to an end device, the network server is in charge of the scheduling by selecting one of the gateways, and the time at which it should be sent. Therefore, it is of uttermost importance to study and evaluate the performance of the network server and its DL scheduler.

In this paper, we present the first evaluation of the DL scheduler implemented by two of the most used LoRaWAN network servers: ChirpStack V4 (CS) [7] and The Things Stack V3.3 (TTS) [8]. To have a realistic environment, we used the ELoRa network emulator [9] that connects the ns-3 network simulator [10] to the CS network server. Our contributions are:

1) We extend the ELoRa network emulator by adding a second network server from TTS, which implements a more complex LoRaWAN architecture and DL scheduler [11].
2) We add a traffic analysis model to ELoRa, allowing us to have the exact same logs for both network servers, providing a fair comparison.
3) We make the first performance evaluation of both CS and TTS network servers and their respective DL schedulers.

The reminder of this paper is organized as follows. Section II presents the technological background as well as the challenges and considerations of DL scheduling. Section III presents the tools and metrics used for this study. Section IV present the results of the performance evaluation. Finally, Section V concludes our work.

II. Background

In this section, we provide a big picture of Long Range (LoRa) and LoRaWAN technology, and we discuss their most relevant characteristics for our study.

A. LoRaWAN in a Nutshell

LoRaWAN uses LoRa at the physical layer, which is a Chirp Spread Spectrum (CSS) modulation. The amount of bytes that the signal is able to modulate is defined by the Spreading Factor (SF) parameter, which also directly influences the data rate and the sensibility at the receiver. The Medium Access (MAC)

layer and the network architecture have been standardized by the LoRa Alliance in the LoRaWAN standard [1]. In our study, we focus on Class A, as it is the default class and one of the most used class due to its focus in energy efficiency.

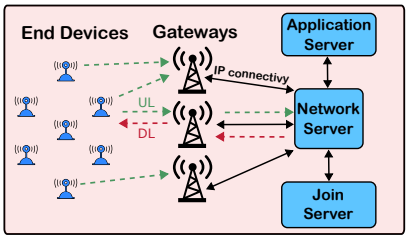


Fig. 1: LoRaWAN architecture.

As we can see in Fig. 1, the UL sent by an end device is received by all the gateways in its communication range, then it is forwarded to the network server for de-duplication, and sent afterward to the application server. Note that end devices share the medium using pure ALOHA. When a DL packet needs to be sent (e.g., to acknowledge an UL data packet), the network server selects one of the gateways to forward the packet, and also it schedules the packet in a given receive window. As shown in Fig. 2, Class A defines a first receive window (RX1) that has the same parameters (SF, frequency channel, and transmission power) as the UL communication, and a second receive window (RX2), which uses a different frequency channel, with an increased transmission power and SF (default values: 27 dBm and SF12) to increase the reception probability for end devices.

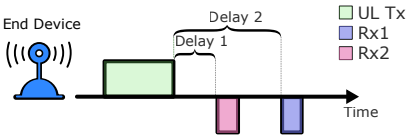


Fig. 2: Class A communication where the end device opens two distinct receive windows after the transmission of an UL.

LoRa is defined for ISM sub-GHz and ISM 2.4 GHz bands, which are unlicensed radio bands. To operate in sub-GHz bands, regional regulations should be respected. In order to allow a fair use of the channels to all users, several constraints are imposed. For example, in Europe, EU868 defines a duty cycle (1% for Rx1, 10% for Rx2), whereas in several Latin America countries (Argentina, Brazil, Mexico, Chile, etc), AS915 and US902 define a dwell-time limit (400 ms) for both end devices and gateways.

B. LoRaWAN Downlink Scheduler

The DL scheduler is a critical part of the network server, as it needs to ensure that there are no conflicts between packets to be scheduled, since the gateway can transmit only one DL at a time. To understand the DL scheduler, it is important to be aware of the communication between the gateway and the network server. Both components run a packet forwarder software: either the UDP Packet Forwarder [12], or the Basic Station [13]. Although the former is the first software available, Basic Station is replacing it progressively. Because of the wide use of UDP Packet Forwarder due to it simplicity, our work is based on it.

In the UDP Packet Forwarder, the gateway regularly polls the network server to check if there is any DL that needs to be transmitted. The network server responds by sending a PULL_RESP to the gateway with the information of the packet to be transmitted. The gateway responds to the network server with an acknowledgement, which can also contain a message with the possible errors encountered in the scheduling. The most common error is a time scheduling conflict if two DL packets need to be scheduled at too close times.

To check for a scheduling conflict between two DL packets, the UDP Packet Forwarder verifies if the two packets overlap in time. As illustrated in Fig. 3, two packets are not in a schedule conflict if the difference in their transmission time (T1 and T2) is higher than the sum of their respective Time on Air (ToA) and pre-delay (delay for transmission), plus a margin. Packets that pass this check are sent to a just-in-time queue on the gateway, where they wait to be sent to the LoRaWAN concentrator for transmission.

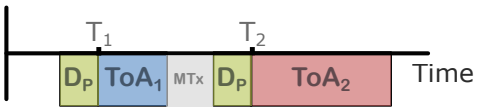


Fig. 3: Example of two DL packets correctly scheduled at times T_1 and T_2. ToA represents the Time on Air of a packet, D_p the pre-delay, and MTx the transmission margin.

C. Related Work

The works that studied the impact of DL traffic in LoRaWAN focused mainly on the link between end devices and gateways [6]. Capuzzo et al. showed how the performance of LoRaWAN can significantly degrade due to the duty cycle regulations and the asymmetry in the sensitivities of the radio chips in gateways and end devices [4]. Van den Abeele et al. pointed out the detrimental impact of DL traffic on the delivery ratio of confirmed ULs [2]. A few works looked at the the impact of gateway selection on network throughput [14] or proposed a novel gateway selection algorithm for improving DL delivery efficiency [15] . All these studies were conducted in a simulator environment, and did not account for the communication between the gateway and the network server.

To the best of our knowledge, there is no work that compares the performance of the scheduling algorithms used in real network server implementations such as CS and TTS. Our work focuses on this and tries to shred light on the aspects that need to be improved in order to enhance the performance of LoRaWAN networks.

III. Emulation of a LoRaWAN Network

To study the performance of DL schedulers, we choose to use the only LoRaWAN network emulator: ELoRa [9], which

combines the flexibility of a simulator to generate traffic from thousands of end devices and gateways, with the complexity of a real implementation of a LoRaWAN network server. Next, we briefly present the ELoRa emulator, followed by our proposed improvements: the support of the TTS network server and traffic analysis modules.

A. ELoRa - First Generation with ChirpStack

ELoRa integrates the ns-3 simulator [10] with the known CS [7] network server, as illustrated in Fig. 4. To recognize LoRaWAN packets and to manage the simulated LoRaWAN network, end devices must be registered on the network server. ELoRa uses libcurl [18] to communicate with the REST API from the network server, and implements the Activation by Personalization (ABP) to register and activate end devices.

The Tap-Bridge node enables the forward of LoRaWAN packets encapsulated in UDP, outside the simulation environment, to the operative system where the network server is running. Here, the packets are managed by the server infrastructure as if they were real traffic.

The structure inside CS that interacts with the gateway is called the gateway bridge and implements the UDP Packet Forwarder [12]. This component receives the upstream UDP datagrams, extracts the LoRaWAN packets and forwards them to a MQTT broker (messaging protocol for IoT devices) [19] that later sends it to the network server. This component is also responsible for encapsulating DL packets and sending them downstream. CS network server schedules DL to the gateway in a first-come, first-serve manner, without checking for any possible conflicts (as illustrated in Fig. 3).

B. ELoRa Extension for The Things Stack Support

We extended ELoRa to support the open source implementation of The Thing Stack [8] maintained by The Things Industries [20]. The software architecture is described in Fig. 4. The main changes are related to the registration of end devices, as the commands to communicate with the REST API are different, as well as the parameters to be configured.

Unlike the CS, the component that communicates with gateways is the Gateway Server. This structure also implements the UDP Packet Forwarder to forward and receive the DL and UL packets to and from the gateway, respectively. Moreover, the Gateway Server implements more operations than the CS Gateway Bridge, such as monitoring the time on air, the duty cycle and the dwell time, depending on the region and restrictions impose. The Gateway Server also monitors the DLs transmitted and to be transmitted, which allows for a more complex and accurate scheduling than in the CS, as TTS takes into account the possibility of scheduling conflicts before actually scheduling the DL packets (as illustrated in Fig. 3).

C. Traffic Analysis Module

To study the performance of both CS and TTS DL schedulers, we need a common traffic measurement tool that would collect statistics on the network servers and on the simulated environment (end devices and gateways). To this end, we implemented the traffic analysis module as shown in Fig. 4.

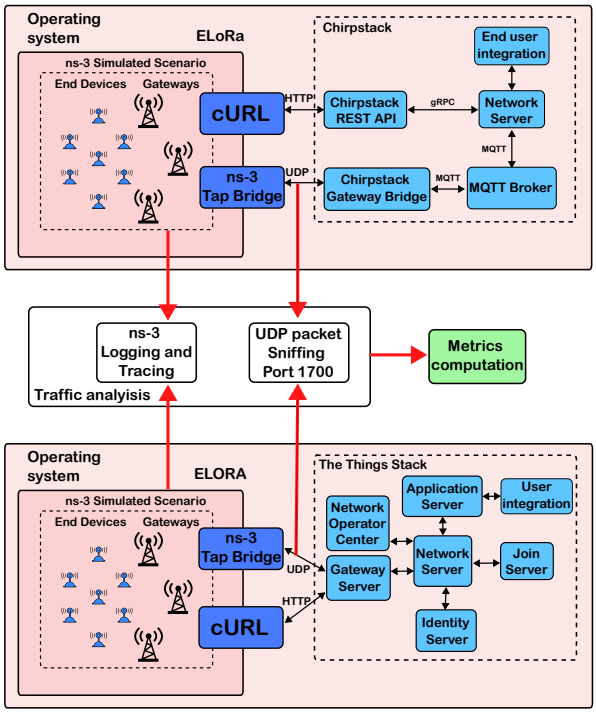


Fig. 4: Architecture of ELoRa.

Firstly, we add a sniffer in the UDP port that connects the ns-3 Tap-Bridge with the network server. This allows us to collect the datagrams exchanged between the network server and the gateways through the UDP Packet Forwarder. We are able to obtain the raw information regarding the UL packets arriving at the network server, including the duplicated packets, which gives us precious information on the selection gateway for DL. Secondly, we also take advantage of the logging and tracing system implemented by the ns-3 simulator. We are able to gather further information on the DLs being scheduled, such as, if they are effectively transmitted and if they arrive to the end devices as well as the causes of packet loss in both flows. Finally, the traffic analysis module is connected to a metrics computation module which allows to process the information collected during the emulation and calculates several metrics of interest:

- **Traffic statistics**: amount of UL and DL packets flowing through the gateways;
- **Packet Deliver Rate (PDR)**: metric calculated for UL and DL packets. Along with this, loss causes are explored. This may be detailed for each gateway and for each receiving window in case of DL;
- **Scheduling result**: when scheduling a DL in a gateway, this can be successful or unsuccessful due to scheduling conflicts (a previous DL was schedule for the same time);
- **Transmission result**: before being transmitted, the DL must be sent from the gateway to the concentrator, where it can be dropped because the timestamp has expired, or skipped because the concentrator is transmitted.

We present next the results of these evaluation metrics for

both the CS and TTS network servers.

IV. Performance Evaluation

In this section, we make a performance evaluation of both CS and TTS network servers and their respective DL schedulers. We start by presenting the emulation scenario.

A. Emulation Scenario

We consider a network topology with 3 gateways placed through hexagonal tiling, and a variable number of end devices placed in range around the gateways uniformly. The number of end devices is chosen to represent a variety of scenarios in the context of dense IoT applications (e.g., 500, 1000, 2000 and 5000). Each end device transmits UL packets using a constant bit rate (CBR) of 150 s, which ensures an intensive traffic scenario, while respecting the duty cycle constraints of the EU868 frequency band. The LoRa spreading factor (SF) is configured on the end devices according to the distance, so the minimum SF value allows connectivity with at least one gateway. We obtain an SF distribution as presented in Table I.

ED	SF7 (%)	SF8 (%)	SF9 (%)	SF10 (%)	SF11 (%)	SF12 (%)
500	28.8	7.00	9.60	11.80	17.00	25.80
1000	27.60	8.80	9.1	12.00	16.40	26.90
2000	27.00	7.95	9.55	12.70	17.40	25.35
5000	26.72	7.90	9.60	13.16	17.20	25.38

TABLE I: Distribution of SF per simulation scenario.

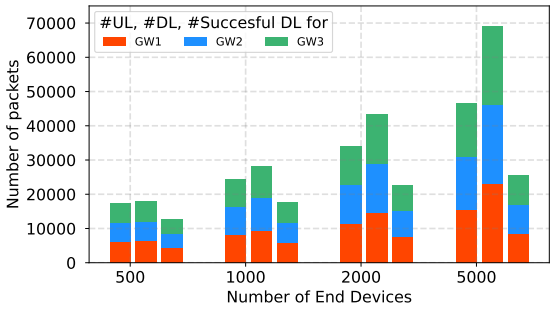
The propagation model is based on the Okumura-Hata model for open areas [21]. In case of collisions, the signal-to-interference ratio matrix is used for the calculation [22]. This matrix allows taking into consideration the interspreading factor interference. Finally, we consider that UL and DL are orthogonal, hence they do not interfere with each other. Table II summarizes the rest of the simulation parameters.

	Configuration
Number of end devices	500, 1000, 2000, 5000
Number of gateways	3
Max distance to a gateway	2154 m
Tx period	150 s
% Confirm traffic	100
Retransmissions	No
Simulation duration	120 min
Duty Cycle	Only for end devices
Rx1 parameters(SF, P_{Tx})	SF_{UL}, 14 dBm
Rx1 parameters(SF, P_{Tx})	SF12, 27dBm

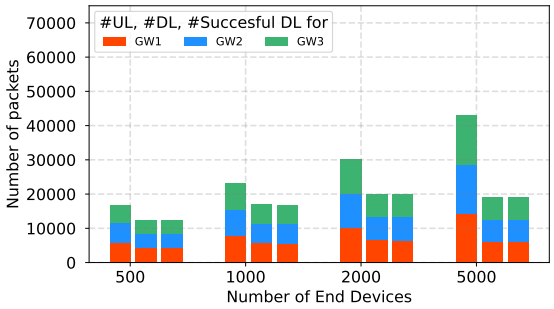
TABLE II: Configured parameters for the emulation.

B. Scheduling Evaluation

First, we take a look at the statistics of the traffic that goes through each gateway and arrives at the network server, and vice versa. Fig. 5 shows for each network topology, the number of ULs arriving to each gateway and forwarded to the network server, the number of DLs scheduled by the network server for each gateway, and the number of successfully scheduled DLs (after the resolution of scheduling conflicts).



(a) ChirpStack (CS).



(b) The Things Stack (TTS).

Fig. 5: Traffic statistics at each gateway for each network topology. The columns represent, from left to right: number of ULs forwarded to the network server, number of scheduled DLs and number of successfully scheduled DL.

We can notice that both the received UL traffic and the scheduled DL traffic are well-balanced among all the gateways. This is clearly a consequence of the uniform distribution of the end devices around the gateways. Considering the DL traffic, we see a significant difference in the amount of DLs scheduled by the two network servers: CS appears to be able to schedule much more DLs than TTS, especially as the network size increases. However, only a part of these DLs are successfully scheduled by CS. This is explained by the fact that: (1) CS does not check for possible scheduling conflicts and simply schedules all DLs on Rx1, and (2) CS immediately reschedules an unsuccessful transmission in Rx2 if the Rx1 schedule has failed. TTS on the other hand, has a strict verification for the conflicts (as illustrated in Fig. 3) and directly schedules a DL on Rx1 or on Rx2, depending on their availability. If we look at Table III, we notice that TTS actually encounters more scheduling conflicts than CS, as it has a more conservative approach, but also more accurate. These packets do not appear in Fig. 5, as they never reach the gateway, reducing the traffic from the TTS network server to the gateway.

To better understand the difference between the two network servers, we look next at the reasons why a DL may be dropped by a gateway, after being successfully scheduled by the network server. As the gateway concentrator can only transmit one packet at a time, the UDP Packet Forwarder

Number of EDs	500		1000		2000		5000	
Network server	CS	TTS	CS	TTS	CS	TTS	CS	TTS
Successful scheduled (%)	88.96	90.29	83.46	83.16	74.13	72.59	57.96	46.39
Scheduling conflict (%)	11.14	9.71	16.54	16.84	25.87	27.41	42.04	53.61
Timestamp expired (%)	0.01	0.01	0.04	0.05	0.57	0.60	3.80	2.68
Concentrator Tx (%)	3.91	0.00	6.37	0.00	8.01	0.03	10.16	0.47
Tx (%)	85.04	90.28	77.05	83.11	65.55	71.96	44.01	43.23
Received (%)	76.90	81.88	68.46	72.64	54.70	59.03	33.02	32.64
Interfered(%)	0.82	1.44	1.73	2.86	2.28	4.25	2.04	3.94
Under sensitivity (%)	7.32	6.96	6.85	7.60	8.56	8.69	8.95	6.66

TABLE III: Results of the DL scheduling for each experiment and network server. Values normalized by number of DLs that need to be scheduled.

Number of EDs	500		1000		2000		5000	
UL	24000		48000		96000		240000	
Network server	CS	TTS	CS	TTS	CS	TTS	CS	TTS
Received	14127	13714	21070	20175	30289	27186	44022	40787
Received (%)	58.86	57.14	43.89	42.03	31.55	28.32	18.34	16.99
Interfered (%)	1.81	1.75	1.53	1.39	1.4	1.08	2.00	1.78
No demodulator (%)	0.03	0.09	0.85	0.80	4.9	3.86	31.05	25.71
Busy gateway (%)	39.19	40.93	53.62	55.69	62.09	66.70	48.59	55.50
Under sensitivity (%)	0.10	0.09	0.1	0.09	0.05	0.04	0.01	0.01

TABLE IV: Uplink statistics.

implements a just-in-time queue to hold the packets that were correctly scheduled. While waiting to send packets from this queue, the gateway can drop a DL if: (1) the timestamp of the packet has expired, being too late for the packet to be transmitted (it would arrive at the end device outside of its Rx window); (2) the packet needs to be transmitted while the concentrator is already transmitting. We can see in Table III that since TTS checks for scheduling conflicts, its DL loss due to the concentrator already transmitting is close to 0. In the case of packets dropped because of expired timestamp, the performance is similar between the network servers, as the UL traffic (which defines the timing of the DL in class A devices) has the same periodicity and load in both scenarios.

Finally, after transmission by the gateway, the DL can still get lost before reaching the end device, due to under-sensitivity at the receiver and interference between packets. The latter can only happen when two DLs that overlap in time are sent in the same frequency channel and with the same SF. This will tend to happen more in Rx2, since all DLs are sent with the same SF. We can see in Table III that TTS has more DL loss due to interference than CS, suggesting a higher use of the Rx2 window. However, this packet loss can simply be avoided if the network severs would account for this phenomenon.

Fig. 5 also highlights a difference in the number of received ULs, with CS being able to receive more ULs than TTS. Table IV gives us an overview of the reasons behind the UL loss, with the main one being the busy gateway: ULs cannot be received while the gateway is transmitting a DL (half-duplex gateway). TTS has slightly higher busy gateway than CS, and the difference increases with the number of end devices. This result shows that the TTS gateways spend more time transmitting DLs, even though they transmit fewer packets.

Fig. 6 presents the percentage of successfully scheduled DLs, which gives us an idea of the efficiency of each scheduler. When the network is small (500 end devices), it looks like TTS is slightly more efficient than CS. However, as the number of end devices increases and the gateways start to become

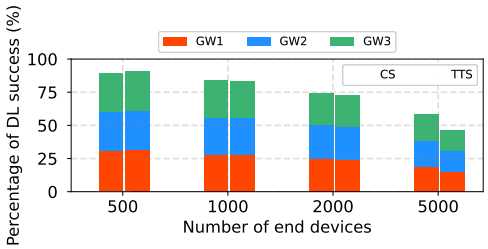


Fig. 6: Efficiency of the DL scheduler, computed as the percentage of successful DL.

saturated (especially at 5000 end devices), CS succeeds in scheduling more DLs than TTS (up to 11.39% of difference). By using pre-delays and transmission margins to avoid overlapping DLs (see Fig. 3), TTS is too conservative, while CS manages to better exploit all the time available at the gateway, hence scheduling more DLs.

C. Reliability Evaluation

Fig. 7 presents the PDR of the DL traffic for each network server, computed for each gateway and receive window, for all simulation scenarios. We can see that the overall PDR is higher when using TTS than CS for all topology sizes (max of 5.24%). The reason behind this is that TTS schedules more packets in the second receive window, which uses SF12 that considerably increases the sensibility in the receiver (as confirmed by the statistics in Table III). However, this comes at the cost of a higher transmission time per packet, which explains why TTS outnumbers CS in the amount of ULs loss due to busy gateway, as seen in Table IV.

The use of DL in class A devices is tied to the UL traffic. We can see here that a higher use of Rx2 for TTS implies a higher PDR for the DL, but at the price of more UL loss. For small networks, the scheduling algorithm of TTS was more efficient than the CS as it considers scheduling conflicts. However, as the amount of end device increases, the amount of DL packets that TTS can schedule saturates.

V. Conclusion

We evaluated the performance of two DL scheduling algorithms implemented by two of the most used LoRaWAN stacks, CS and TTS. To this end, we used the ELoRa emulator that allows to connect a simulation environment to these network servers and injects to them real traffic for the DL generation, which corresponds to the acknowledgement of the confirmed ULs. We studied four different topologies that correspond to four different EDs densities.

We observed that the TTS scheduling is more efficient when the number of EDs is lower: it sends less DL to the GW, but almost all of them are correctly scheduled, whereas CS sends everything to the GW, but with less DL scheduling success. If TTS has a higher DL PDR, CS has a higher UL PDR. When the number of EDs increases, the capability to schedule DL saturates, and the CS performance comes closer to TTS. Clearly, the use of CS for DL is a valid option, specially for

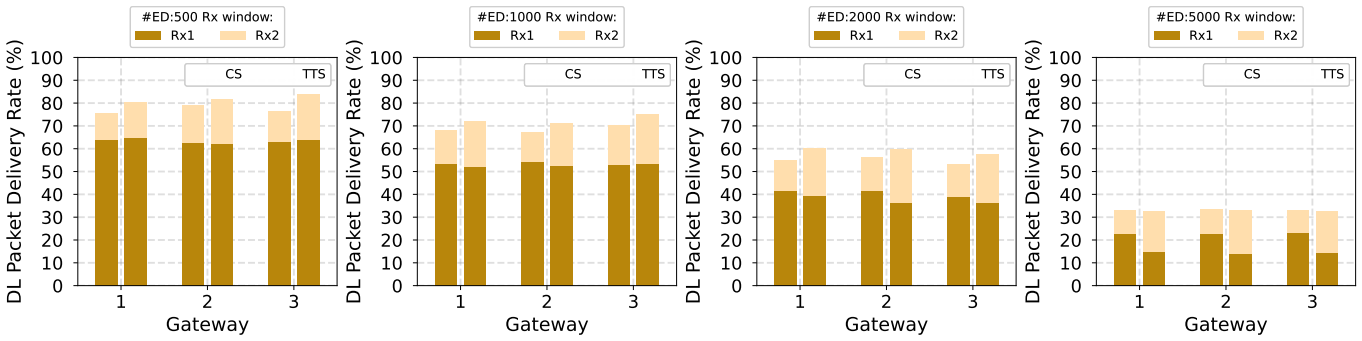


Fig. 7: Downlink PDR detailed for each receive window and each gateway, for the ChirpStack and The Things Stack.

high amount of end devices, also considering it is a simpler deployment. TTS is a better choice for less dense networks, as it was shown by our results.

As work for the future, we plan to evaluate these network servers for different types of non-periodic and burst traffic, and to characterize their behavior. Following this, we plan to compare the results with an optimal behavior of the scheduler to check if there is space for improvement and for the design of new scheduling strategies that allow to get close to the optimal performance.

This research has received support from the Project ANR-21-CE25-0002-01.

References

[1] LoRaWAN L2 1.0.4 Specification, TS001-1.0.4, LoRa Alliance, 2020.

[2] F. Van den Abeele, J. Haxhibeqiri, I. Moerman and J. Hoebeke, "Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2186-2198, Dec. 2017.

[3] A. -I. Pop, U. Raza, P. Kulkarni and M. Sooriyabandara, "Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?," IEEE GLOBECOM, Singapore, 2017.

[4] M. Capuzzo, D. Magrin and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," MedHoc-Net, Capri, Italy, 2018.

[5] LoRaWAN Regional Parameters, RP002-1.0.4, LoRa Alliance, 2022

[6] J. M. Marais, A. M. Abu-Mahfouz and G. P. Hancke, "A Survey on the Viability of Confirmed Traffic in a LoRaWAN," in IEEE Access, vol. 8, pp. 9296-9311, 2020

[7] ChirpStack v4. (2022). ChirpStack. [Online]. Accessed: Jun. 20, 2024. Available: https://www.chirpstack.io

[8] The Things Stack, an open source LoRaWAN Network Server. The things industries. Accessed: Jun. 20, 2024. [Online]. Available: https://github.com/TheThingsNetwork/lorawan-stack

[9] A. Aimi, S. Rovedakis, F. Guillemin and S. Secci, "ELoRa: End-to-end Emulation of Massive IoT LoRaWAN Infrastructures," IEEE/IFIP NOMS, USA, 2023.

[10] Ns-3 Network Simulator v3.37. (2008). nsnam. Accessed: Jun. 20, 2024. [Online]. Available: https://www.nsnam.org

[11] "The Things Stack helper for ELoRa emulator". Zenodo, Sep. 30, 2024. doi: 10.5281/zenodo.13863843.

[12] Lora network packet forwarder project v4.0.1. (2017). Semtech. Accessed: Jun. 20, 2024. [Online]. Available: https://github.com/Lora-net/packet_forwarder

[13] LoRa packet forwarder Basics Station v2.0.6. (2019). Semtech. Accessed: Jun. 20, 2024. [Online]. Available: https://github.com/lorabasics/basicstation

[14] S. Abboud, N. el Rachkidy, A. Guitton and H. Safa, "Gateway Selection for Downlink Communication in LoRaWAN," IEEE WCNC, Marrakesh, Morocco, 2019.

[15] M.A. Mojamed, "A Duty Cycle-Based Gateway Selection Algorithm for LoRaWAN Downlink Communication," Comput. Syst. Sci. Eng., vol. 45, no. 3, pp. 2953-2970. 2023.

[16] Silva, J. C. da, Flor, D. de L., Junior, V. A. de S., Bezerra, N. S., & Medeiros, A. A. M. de. (2021). "A Survey of LoRaWAN Simulation Tools in ns-3". Journal of Communication and Information Systems, 36(1), 17–30.

[17] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," IEEE ICC, Paris, France, 2017.

[18] libcurl. (1996). curl. Accessed: Jan. 13, 2023. [Online]. Available: https://curl.se/libcurl

[19] MQTT. OASIS. (2019). Accessed: Jan. 13, 2023. [Online]. Available: https://mqtt.org/

[20] The Things Industries. [Online]. Accessed: Jun. 20, 2024. Available: https://www.thethingsindustries.com/

[21] M. Hata, "Empirical formula for propagation loss in land mobile radio services," in IEEE Trans. on Vehicular Technology, vol. 29, no. 3, pp. 317-325, Aug. 1980

[22] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita and I. Tinnirello, "Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance," in IEEE Comm. Letters, vol. 22, no. 4, pp. 796-799, April 2018.