LoRaWAN Class B Multicast Scalability

Yonatan Shiferaw
Delft University of Technology
Delft, the Netherlands

Apoorva Arora
Delft University of Technology
Delft, the Netherlands

Fernando Kuipers
Delft University of Technology
Delft, the Netherlands

***Abstract*—LoRaWAN has emerged as a popular IoT communications technology. It comes with three classes of operation: A, B, and C. Although many IoT use-cases, like Firmware-over-the-Air updates, require multicast, Class A cannot be used for that purpose. Class C can, but consumes a lot of energy. This leaves Class B. In this paper, we investigate Class B multicast and its scalability properties. Issues like multicast member capacity, beacon blocking, and beacon collisions are highlighted, and several approaches to mitigate them are proposed: (1) "Ping-Slot Relaying," to allow for more multicast members, (2) a scheduling approach indicating when to best send multicast packets, and (3) "Dynamic Region Formation" to coordinate the sending of beacons over multiple gateways. The proposed solutions do not require any modifications to the LoRaWAN protocol.**

***Index Terms*—LoRaWAN Class B, Multicast, IoT.**

I. Introduction

Typical Internet-of-Things (IoT) applications involve the deployment of and data collection from many IoT devices. These IoT devices might receive the same control instructions or Firmware-over-the-Air (FotA) updates from a controlling entity. Transmitting identical information separately (via unicast) to each individual IoT device would be highly inefficient in terms of energy/load and can be done more efficiently, via multicast, in a single transmission. LoRaWAN [1] has emerged as a popular IoT communications technology. It is a Medium Access Control (MAC) protocol built on top of LoRa, a long-range radio technology, and it offers three modes of operation – Class A, Class B, and Class C – of which only classes B & C support multicast. Class C, however, is quite power hungry, as devices continuously listen for possible messages. In Class B they only do so periodically, making that class better suited for battery-operated or energy-harvesting devices, which IoT end-devices typically are. Although LoRaWAN Class B does support multicast transmissions, that particular feature is fairly novel and its performance has not yet been carefully studied. In this paper, we fill that gap and also offer several improvements.

II. LoRaWAN Class B

In Class B, in addition to the two receive windows opened after an uplink (as with Class A), the end-device opens periodic reception windows called ping-slots [1]. These ping-slots allow the network server to send downlinks called pings to the end-device without having to wait for an uplink. Such a communication requires time synchronization between the network server and the end-devices, which is achieved via downlink packets called beacons.

1) Beacon: Beacons are transmitted every beacon-period of 128 seconds by gateways. A node is allowed to switch from Class A to Class B only after it locks onto a beacon. In order to avoid a conflict at the end-device between a ping downlink reception and a beacon reception, a beacon is preceded by a guard time of 3 seconds, during which a ping-slot cannot be placed [1]. Further, 2.12 seconds are reserved for beacon transmission. Thus 122.88 seconds are available for ping-slots in a beacon-period. This is referred to as the beacon-window.

2) Ping-slots: Once a node has successfully locked onto a beacon, it switches to Class B and starts opening periodic reception windows – ping-slots – for 8 symbols[1] for each locked beacon and goes back to sleep if no preamble is detected. If a preamble is detected in a ping-slot, the end-device(s) will continue to receive the entire downlink packet.

The duration between two consecutive ping-slots – the **pingPeriod** $:= 2^{12}/$**pingNb** – is computed as the beacon-window in terms of 2^{12} slots (where each slot is **slotLen** = 30ms long) divided by the number of pings (**pingNb**) in a beacon period. **pingNb** $:= 2^{7-Periodicity}$ in turn is derived from the ping-slot periodicity (*Periodicity*).

3) Slot-randomization: The LoRaWAN specification proposes a technique called **slot-randomization**: a random offset, called **pingOffset**, is calculated as a function of both the beacon time and the end-device devAddr address and added to the start of the first ping for each beacon period.

A. LoRaWAN Class B Multicast

All the nodes in a multicast group should be able to listen and decode the multicast transmissions for that group. This is realized by configuring the nodes in a multicast group with the same SF[2], channel, bandwidth, and ping-slot periodicity. In addition, the nodes should have a common multicast device address (McAddr) [3]) and security keys [3].

Compared to Class B unicast, Class B multicast has some restrictions. To avoid collisions, the specification [1] restricts the use of MAC commands and acknowledgments that require all the receiving nodes in a multicast group to respond at the same time. Further, [1] does not offer support for multicast groups creation and securely distributing keys and addresses.

Part of this work has been supported by SURFnet.

Annex to ISBN 978-3-903176-28-7 ©2020 IFIP

[1]This is not stated in [1], but extracted from the LoRaMac firmware [2].

[2]Spreading Factor is a LoRa parameter. The higher SF, the longer a transmission takes, but the more resilient the signal is to noise and interference.

Input: Received packet in ping-slot i
  if Packet hop-count $<$ hop-count limit **then**
    Configure ping-slot $i+1$ for transmission;
    Randomly select transmission power T_x in range $[0, 14]$ dB based on no. of end-devices N as $T_x = \frac{14}{\text{a random integer from } [1, \lfloor \frac{N}{2} \rfloor]}$ dB;
    Transmit packet in ping-slot $i+1$;
  end if

Fig. 1: Ping-slot relaying algorithm.

This is deferred to the application layer, for which two specifications were released: [3] and [4].

III. Multicast member scalability

In a LoRaWAN multicast session, after the transmission of all data fragments, the network server sends an application-layer command to the member end-devices asking them to report their reception status within a BlockAckDelay time [4]. The end-devices, with a too low Packet Reception Ratio (PRR), randomly select a time between 0 and BlockAckDelay time to send their response (asking for a retransmission). The more responses, the more potential collisions at the gateway, which affects the multicast capacity (i.e., the number of end-devices that can be accommodated in a multicast group).

That multicast capacity can be improved in two ways: by relaxing the PRR requirement and/or by improving the PRR distribution. The first is accomplished by using coding techniques, such as the one provided in [4]. The second can be accomplished by using any technique that improves the reception of multicast packets for all member end-devices. We propose one such technique, called *Ping-slot relaying*.

A. Ping-slot relaying

The main idea behind ping-slot relaying is to use the end-devices themselves, instead of the duty-cycle limited gateways, in the *re-transmission* of multicast packets.

Fig. 1 presents our ping-slot relaying algorithm. The number of member end-devices (N) is provided via the application layer to those end-devices during the creation of their multicast group. A hop-count is set in the payload of the packet to limit the number of ping-slots that could be used for ping-slot relaying after a multicast packet is received. Else, if left unlimited, it might interfere with subsequent ping-slot transmissions from the gateway. Furthermore, the transmission power is selected randomly based on the number of multicast members to allow (with higher probability) the capture effect[3] [5] to take place. This also allows higher energy savings as the number of members can increase.

IV. Multicast Group Scalability

This section presents a discussion and simulative analysis of beacon blocking in LoRaWAN Class B multicast as the number of groups increases. Because beacon blocking – as explained in Sec. IV-A – happens independently for each gateway that transmits both Class B downlinks (unicast and/or multicast) and beacons, we take the perspective of a single gateway. Beacon planning over multiple gateways is discussed in Sec. V.

A. Beacon Blocking

Beacons, as well as the downlinks on ping-slots (pings), are by default sent on the 869.525 MHz channel falling in the G3 sub-band having a 10% duty-cycle restriction. Thus, the gateways enter an "off-period" for 9×(Time-on-air[4]) after every beacon or ping transmission. Hence, ping transmissions have the capacity to block a scheduled beacon transmission by pushing the gateway into the off-period.

One of the contributions of this work has been the development of an ns-3 module to enable a detailed analysis of LoRaWAN Class B [6]. For each combination of periodicity, number of multicast groups, data-rate and packet size we simulated for 2 hours during which we transmitted the maximum number of Class B multicast downlink packets possible (which differs per setting). This resulted in a heatmap, see Fig. 2, for the number of beacons blocked out of 56 beacons generated, where the maximum packet size was used for all the data-rates (DR0 to DR5, where low data-rates have lower throughput compared to high data-rates) for Class B ping downlink transmissions. The heatmap shows that there is no clear relationship between the number of multicast groups and beacon blocking. This is because *beacon blocking happens if two conditions are fulfilled*: (1) if the combination of ping-slot periodicity, ping offset, packet size, and data-rate results in ping-slots whose transmission will follow an off-period that includes the beacon reserved time, and (2) if the gateway did not have a previous transmission that blocked it from using these ping-slots.

It is interesting to note that the second condition is also affected by the combination of ping-slot periodicity, ping offset, packet size, and data-rate. Thus, at the outlook, a random beacon blocking behaviour is expected, which is indeed what our simulations show.

Fortunately, beacon-blocking behavior is not completely random and several observations can be made from Fig. 2:

1) Even low ping-slot periodicity gives rise to systematic beacon blocking. This ranges to 90% beacon blocking for DR0, DR1, DR2, DR3 and DR4. Even for DR5, the fastest data-rate, more than 50% of the beacons could be blocked.
2) As the number of groups increases, there is a high probability of having at least one group that opens a ping-slot on each slot throughout the beacon-window. This starts a pattern in the network server where it starts sending out ping downlinks always as soon as the off-period ends. This continues regardless of the number of

[3]The capture effect is usually seen in frequency modulated signals, where the strongest signal can still be demodulated successfully from colliding packets.

[4]Time-on-air or airtime of a transmission is the time needed for transmitting a packet from node to gateway, or vice versa.

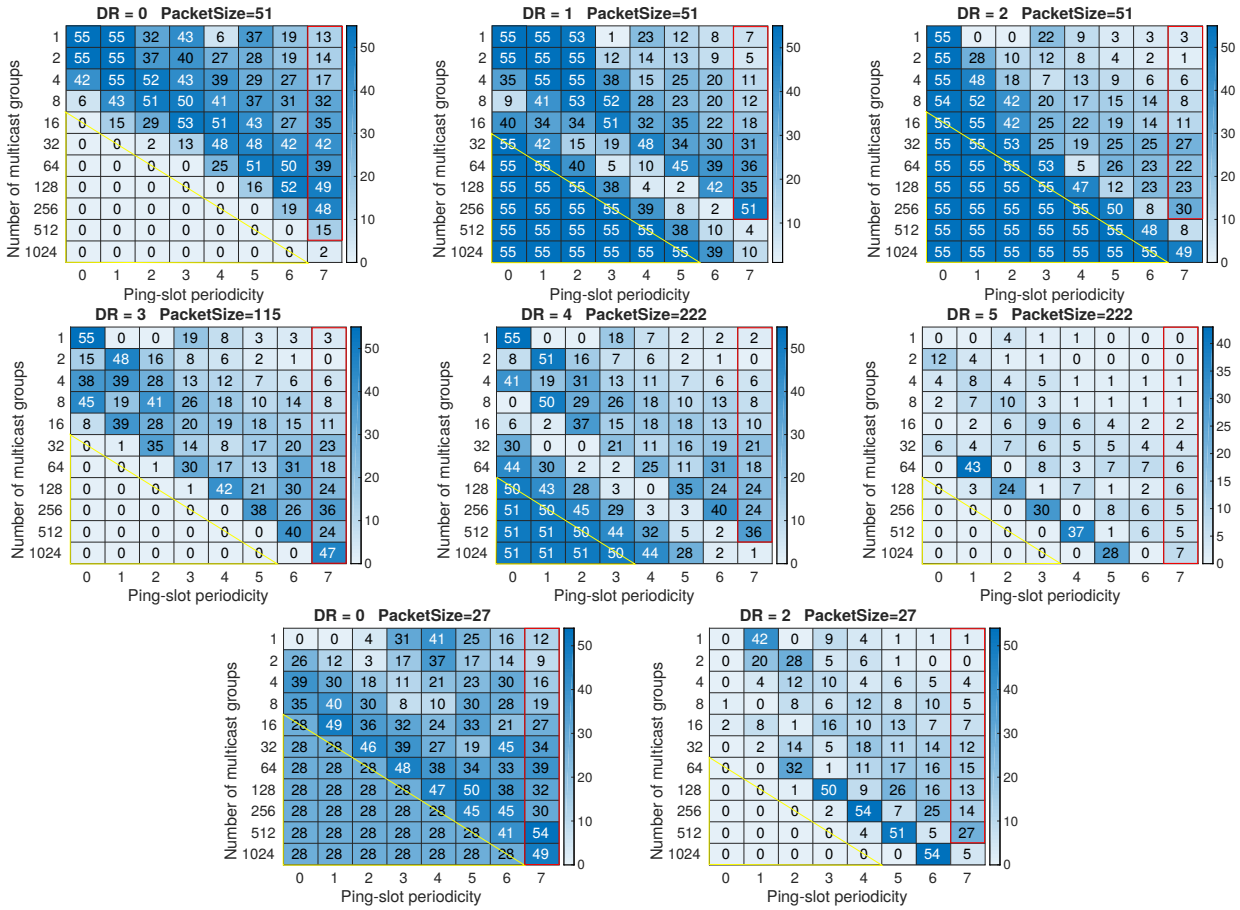

Fig. 2: Heatmaps for the number of blocked beacons out of 56 beacons generated.

groups added further, hence saturation. Because lower ping-slot periodicities have more ping-slots per beacon period, saturation for them begins with fewer groups compared to higher ping-slot periodicities. The yellow triangle in Fig. 2 illustrates the above.

3) DR0 has the highest airtime, but the lowest beacon blocking at saturation. This is because DR0, due to longer airtime, prevents the use of beacon blocking ping-slots. This behaviour changes if the packet size is decreased.
   When the packet size is reduced to 27 bytes, DR0 has increased beacon blocking at saturation, while DR2 has decreased beacon blocking at saturation. Therefore, the saturation value of beacon blocking depends not only on the data-rate and the ping-slot periodicity, but also on the length of the packets.
4) For higher ping-slot periodicities, the number of beacons blocked increases with the number of groups (irrespective of their sizes), before saturation is reached. This pattern is highlighted in red in Fig. 2.

Hence, a more sophisticated beacon guard mechanism, other than the beacon guard from the specification, is needed. In Sec. IV-B, *we propose our solution to prevent beacon blocking*.

B. Prevention of Beacon Blocking

First, we propose an intuitive approach to prevent beacon blocking and then refine it to address it drawbacks.

1) Limiting the maximum payload size for each data-rate: The beacon guard is fixed to 3 seconds and the maximum payload size given in the specification [7] is capped. The maximum packet size that will not lead to beacon blocking satisfies:

$$T_p + 9T_p \leqslant B_G \tag{1}$$

where T_p stands for the airtime of packet p, $9T_p$ is the resulting off-period, and B_G stands for the beacon guard duration. This can be solved using Eq. 2, where BW stands for bandwidth. Eq. 2 can then be solved for maximum PL (payload length) that would not cause beacon blocking. The LoRaWAN application payload can then be calculated by subtracting the LoRaWAN header (13 bytes) from the PL [1].

$$\max\left(5\left\lceil\frac{8PL-4SF+44}{4SF}\right\rceil,0\right) \leqslant \frac{B_G \times BW}{10\times 2^{SF}} - 20.5 \tag{2}$$

Unfortunately, the equation reveals that, for DR0 and DR1, it is impossible to get the combined airtime and off-period below the beacon guard time (of 3 seconds). Hence, this technique needs to be refined for low data-rates as follows.

2) Limiting the maximum payload size for each ping-slot and ping-offset: By also considering the ping-offset in which a packet will be transmitted, we allow any transmission at any slot to be transmitted as long as it will not lead to beacon blocking. To accommodate this modification, Eq. 1 is extended to calculate the maximum packet size based on the location (ping-slot index and **pingOffset** combination) of the ping-slot on top of the beacon guard (B_G):

$$T_p + 9T_p \leqslant B_G + 0.03\left(P_P\left(P_N - 1 - P_I\right) + \left(P_P - 1 - P_O\right)\right) \tag{3}$$

where P_P is the **pingPeriod**, P_N is **pingNb**, P_I is the ping-slot index, BW is bandwidth, P_O is the **pingOffset**, and the 0.03 is the **slotLen** in seconds. Hence, Eq. 2 will also be modified as follows:

$$\begin{aligned} &\max\left(5\left\lceil\frac{8PL - 4SF + 44}{4SF}\right\rceil, 0\right) \leqslant \\ &\frac{BW\left(B_G + 0.03\left(P_P\left(P_N - P_I\right) - \left(P_O + 1\right)\right)\right)}{10 \times 2^{SF}} \\ &- 20.5 \end{aligned} \tag{4}$$

The network server solves Eq. 4 before every Class B ping transmission, to make sure that the transmission will not lead to beacon blocking. If such a transmission would lead to beacon blocking, the network server postpones it to a ping-slot after beacon transmission.

Effectively, we have created a packet scheduler that makes sure that no beacon is blocked and all packets that do not result in beacon blocking are transmitted. The apparent disadvantage of this algorithm is that the network server has to solve Eq. 4 before every transmission. Fortunately, the values can be pre-computed and stored in a lookup table, so that the network server can simply resolve the ping-slot periodicity, ping-offset and data-rate into the maximum packet size allowed in a particular slot.

V. Beacon Planning for Capacity Expansion

In Sec. IV-A, we have discussed how the duty-cycle restriction on gateways can result in beacon blocking. Following the same argument, beacon transmissions may result in delayed ping transmissions by pushing the gateway in the off-period, which results in increased downlink data transmission latency. As the network scales in terms of multicast groups or unicast end-devices, an increase in the number of pings from the network server is expected, demanding more airtime of the gateway. This means, to minimize the transmission latency of the *increased* number of pings, beacon transmissions need to be controlled more efficiently.

Note that not all gateways are required to participate in a beacon broadcast round and their participation to a given round can be randomized, as mentioned in [1]. Hence, the gateways transmit in a given beacon round only with a certain probability. This overcomes the problem of systematic beacon collisions. But the random nature of the algorithm may still result in sub-optimal beacon planning, where there are occasional collisions and no beacon signals reaching an end-device. Hence, there is a need for a more intelligent solution for beaconing to guarantee beacon reception and enable capacity expansion. We propose, in Sec. V-A, the *Dynamic Region Formation (DRF)* algorithm to realize those objectives.

A. Dynamic Region Formation(DRF) Algorithm

The fundamental idea behind DRF is the classification or clustering of end-devices into groups. Each group is formed around a single unique gateway. Only the gateways around which a group is formed are activated for beacon transmission, while others do not participate. The process of group formation is dynamic to ensure that all end-devices are covered for beacon reception. For that purpose, DRF utilizes the same information as collected for the Adaptive Data Rate (ADR) scheme. ADR is used by the network to optimize the data-rates of each end-device according to the changing network conditions to minimize the airtime and energy consumption [1]. The ADR algorithm itself is not part of the specification and can be modified at will, but Semtech has provided an example ADR algorithm, where the network uses 20 most recent uplinks to measure the signal-to-noise ratio (SNR) and number of gateways that received each uplink. The network then uses the SNR of the best gateway to optimize the data-rate and transmit power of the end-device [8].

In DRF, the (average) SNR information is used to associate an end-device to its best gateways: the SNR should exceed a certain threshold. In the absence of (recent) uplinks, the gateway(s) might solicit uplinks. Multiple end-devices associated to a gateway together form a group. This step, when executed for all end-devices, results in group formation around *most* gateways and also indicates the coverage overlap that exists. Hence, the next step is to find the smallest possible subset among the gateways that collectively still cover all the end-devices. Essentially, we need to solve a minimum set cover problem, which unfortunately is NP-hard [9], where the gateway groups form the sets and the end-devices form the elements that can be in (multiple of) those sets. We need to select the minimum amount of sets that cover all end-devices. There is a simple, but effective, polynomial-time greedy algorithm for the minimum set cover problem [9] that at each iteration chooses the set that contains the largest number of uncovered elements. Its approximation ratio is $H(g) = \sum_{k=1}^{g} \frac{1}{k}$, which is the g-th harmonic number, where g is the number of gateways. We have adopted this greedy approach.

The final step of DRF is to set a random probability: $Pbeacon$ for the activated gateways to eliminate **systematic collision** scenarios.

The above computations (1: associate end-devices to gateways based on a minimum SNR threshold, 2: run a greedy algorithm, and 3: set a random beacon probability) are done periodically at the network server. By using the SNR infor-

mation, DRF takes into account any changes in the network such as gateway down scenarios, or new obstacles blocking end-devices to update the beacon transmitting gateway set.

To test DRF, we did not need the detail of our ns-3 simulator and speed was more important, which is why we developed a high-level SimPy-based simulator [10]. Fig. 3 shows a comparative performance study of our Dynamic Region Formation algorithm (DRF), Randomization with all gateways transmitting beacons 50% ($Pbeacon = 0.5$) and 10% ($Pbeacon = 0.1$) of the times, and No Planning (All).

The simulations were run in different experiments for 20, 50 and 100 stationary end-devices, uniformly distributed in a region with 60 gateways for 235 beacon broadcast rounds. The numbers are averaged over the number of end-devices to illustrate average performance per end-device. For the simulations, we assumed that each gateway had a coverage ranging 7.5 km in all directions.

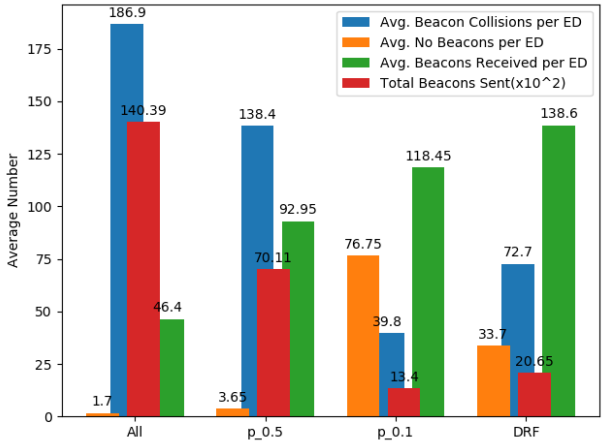

Fig. 3: Comparison of DRF, Randomization with $p = 0.5$, $p = 0.1$, and No beacon planning (all gateways).

It can be inferred from Fig. 3 that our DRF algorithm reduces *both* the number of beacon collisions and the number of "no beacons," hence maximizing the number of received beacons per end-device. Moreover, with DRF, about 85% and 68.86% fewer beacons were transmitted as compared to no beacon planning and randomization ($Pbeacon = 0.5$), respectively. Thus, with DRF, a proportional amount of downlink energy and airtime is saved.

Subsequently, to analyse how DRF performs in terms of the number of delayed pings, as the number of downlink pings increases, the number of ping-slots used by the network server was increased from 10% to 50%. Note that the increase in pings also reflects network scaling.

Fig. 4 illustrates that the percentage of delayed pings increases as the network scales (percentage of ping-slots utilized increases). *DRF performs the best with minimal increase in the delayed pings.*

Hence, DRF allows scaling the number of multicast groups, with the least effect on the downlink latency. The scalability is

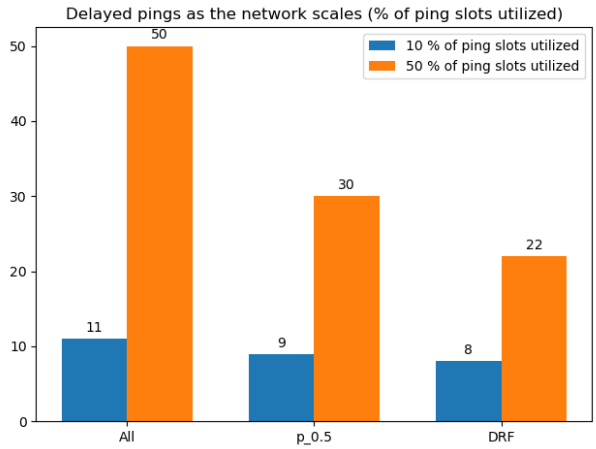

Fig. 4: Comparison of DRF, Randomization with $p = 0.5$ and No beacon planning (all gateways) for the increase in the number of delayed pings as the network scales.

enabled by giving less airtime of gateways to beacons, hence allowing more pings to be accommodated.

VI. Conclusion

In this paper, we have presented a performance evaluation of LoRaWAN Class B multicast. We have proposed a ping-slot relaying technique. Furthermore, we have evaluated *the gateway beacon blocking phenomenon* and proposed a packet scheduler to allow more Class B downlink transmissions without blocking beacons.

Finally, to increase the reliability of beacon reception at the end-devices, we have proposed a novel DRF algorithm for beacon planning over multiple gateways. Our simulations confirmed that this approach yields benefits for both the end-devices and the network operator by *increasing the number of beacons received by 66.52% per end-device on average with 85% less beacon transmissions as compared to no beacon planning.*

References

[1] LoRa Alliance, *LoRaWAN 1.1 Specification*, 2017.
[2] M. Luis, G. Cristian, D. Jaeckle, and J. Bruder, "Lora-net/loramac-node," https://github.com/Lora-net/LoRaMac-node, 2019.
[3] LoRa Alliance, *LoRaWAN Remote Multicast Setup Specification*, 2018.
[4] N. Sornin, A. Yegin, J. Catalano, J.-P. Coupigny, and J. Stokking, *LoRaWAN Fragmented Data Block Transport Specification v1.0.0*, 2017.
[5] A. Rahmadhani and F. Kuipers, "When lorawan frames collide," in *Proc. of WiNTECH'18*, 2018.
[6] Y. Shiferaw, "LoRaWAN Class B ns-3 simulator," https://github.com/yoniwt/lorawan-private, 2020.
[7] LoRa Alliance, *LoRaWAN 1.1 Regional Parameters*, 2017.
[8] The Things Network, "About ADR." [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate.html
[9] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
[10] A. Arora, "LoRaWAN Class B SimPy simulator," https://github.com/Apoorva-ar/LoRaWAN_DRF, 2020.