A LoRaWAN Module for ns-3: Implementation and Evaluation

Brecht Reynders
KU Leuven
Leuven, Belgium
brecht.reynders@kuleuven.be

Qing Wang
KU Leuven
Leuven, Belgium
qing.wang@kuleuven.be

Sofie Pollin
KU Leuven
Leuven, Belgium
sofie.pollin@kuleuven.be

ABSTRACT

Long Range (LoRa) communication has been proposed to connect massive numbers of devices in large areas. LoRa itself is a physical layer technique. Together with the MAC layer solution LoRaWAN to realize IoT, they have attracted increasing attention from both industry and academia. In this work, we present our implemented LoRaWAN module for ns-3, to help boost the research in this rising area. Our implementation is compliant with the class A of the LoRaWAN 1.0 specification. It is highly configurable and thus can be easily used to exploit the impact of different parameters on LoRaWAN's performance. Our implemented flexible backbone architecture also allows for the easy integration of new protocols to improve the network performance. In the past two years, we have used this module to develop new protocols and algorithms to improve LoRaWAN's performance, in terms of reliability, capture effect, scalability, power consumption, among others. Our research outcomes have demonstrated the usefulness, flexibility, and configurability of the proposed LoRaWAN ns-3 module. We have made the source code of our module publicly available[1].

CCS CONCEPTS

• **Networks → Network simulations**;

KEYWORDS

LoRa, LoRaWAN, ns-3 model, flexible, implementation, evaluation

ACM Reference Format:
Brecht Reynders, Qing Wang, and Sofie Pollin. 2018. A LoRaWAN Module for ns-3: Implementation and Evaluation. In *Proceedings of the 2018 Workshop on ns-3 (WNS3 2018), June 2018, Surathkal, India.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3199902.3199913

1 INTRODUCTION

Within the Internet of Things, Long Range (LoRa) communication has received a lot of attention. Due to its capability of supporting long communication range, LoRa can fill the gap between mobility and low power. With this paradigm, it is possible to track cars and people with coin cell batteries and monitor power consumption on the electric grid in real-time, even in remote places like basements. This all thanks to the powerful physical layer technology of LoRa that trades off range with throughput.

Together with the growing interest from industry, there is also an increasing interest from academia not only to exploit the performance and scalability of LoRa networks, but also to improve LoRa's reliability. *LoRa itself is a physical layer technique.* To build a complete communication system, *the MAC layer protocol LoRaWAN* has been proposed which can work well with LoRa. However, due to the complex network topology and the ALOHA nature of the LoRaWAN MAC protocol, closed-form equations, e.g. the packet error rate for ALOHA access, are insufficient to analyze the network's performance in a wide range of scenarios. Simulation is an elegant tool and can provide an easy way of demonstrating the impact of certain parameters on the network performance. In LoRa, these interesting parameters include acknowledgment response time, spreading factors and bandwidths. The impact of these parameters could be easily done with a network simulator.

With this in mind, we propose a new long range, low power module for ns-3. Our model is compliant with the LoRaWAN v1.0 class A specification [20]. It is highly configurable to study the effect of different parameters on the network performance. The flexible backbone architecture allows for easy integration of new protocols not only to improve the random access performance of these networks, but also to study what will happen if a backbone server is congested. One of our main contributions in the model is that it supports distributed gateways. These gateways are connected over an IP network to the network server that controls the whole network. We also provide base classes that allow for an easy implementation of new applications in the network server.

The remainder of this work is as follows. In Section 2, we introduce the LoRaWAN ecosystem and the state of the art. We also highlight the differences between our model and other solutions. In Section 3, we describe the implementation in details. Then, in Section 4 we present some simulation results to show the performance of our model. Finally, we conclude the work in Section 5.

2 BACKGROUND AND RELATED WORK

2.1 The LoRaWAN Ecosystem

LoRaWAN is an open source MAC protocol developed around LoRa. The LoRaWAN architecture consists of three important elements: nodes, gateways and the backbone, as illustrated in Figure 1. Similar to cellular networks, gateways connect nodes within their coverage to the backbone. The difference is that gateways in LoRaWAN use non-licensed ISM bands and target extremely at low power and long range communications. In LoRaWAN, nodes typically only have a few messages to transmit per day.

Depending on the application scenario, LoRaWAN nodes can be configured into three different types:

[1] https://github.com/networkedsystems/lora-ns3.git

WNS3 2018, June 2018, Surathkal, India

ACM ISBN 978-1-4503-6413-3/18/06…$15.00
https://doi.org/10.1145/3199902.3199913

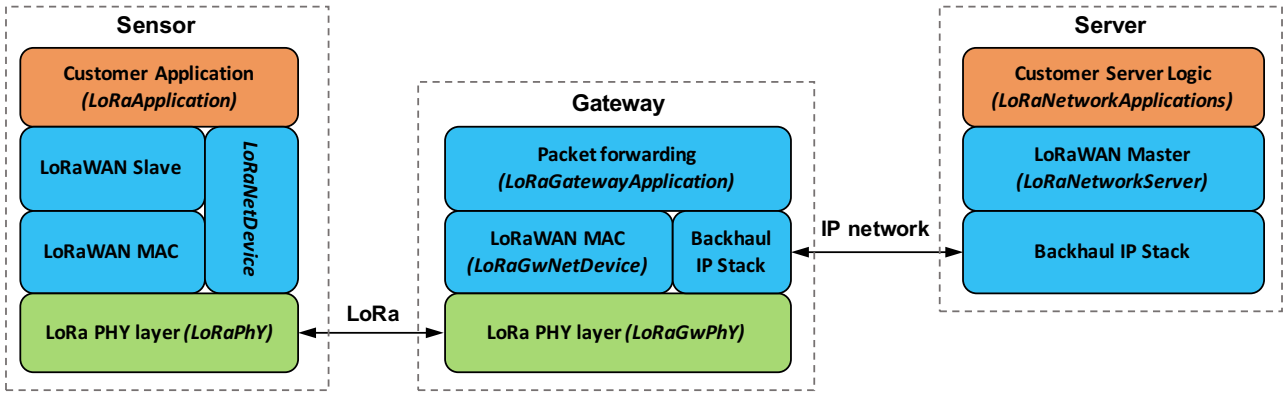

Figure 1: The structure of a LoRaWAN network according to the LoRa Alliance [1]. The sensor on the left is connected to the network server on the right. It sends the sensing data to the Gateway using the LoRa technique over long distance. At the Gateway, a higher-throughput IP network takes over and forwards all the data to the network server. In this work, we present implementation of the above structure in ns-3 as a LoRaWAN module, to help boost the research in this rising area. *The corresponding classes we implement are highlighted in italics in the above structure*

- *Class A*: Sensor nodes only send small number of data packets to the gateway and sleep for most of the time.
- *Class B*: Except for the actions in Class A, sensor nodes also wake up at scheduled slots to receive downlink messages.
- *Class C*: Sensor nodes always listen to the channel.

To enable long range communication, LoRa uses chirp spread spectrum with different *Spreading Factors (SFs)* to trade off data rate for extra range. These spreading factors specify the chirp rate. In the specification of LoRa, the SF can be selected from SF7 to SF12.[2] A lower SF chirps faster over the bandwidth and, as a result, symbols are shorter. This leads to a higher transmission rate and shorter transmission time, but requires a higher SNR. It should be noted that a lower SF results in a shorter communication range.

2.1.1 MAC Protocol (for Class A). In LoRaWAN, next to the long range, power efficiency is very important and the MAC protocol plays an important role in this aspect. In addition, the MAC layer also sets the boundaries of network scalability and mobility. In LoRaWAN, the most energy efficient method has been selected to send data to the gateway, i.e. random access. Whenever a node has data, it sends its data immediately without sensing the channel first. The transmission parameters are selected by the node itself, and from the node's point of view, sufficient to communicate with the gateway.

After each transmission, two downlink slots are available for the gateway at fixed time instances: the first slot at least one second after the transmission and the second slot exactly one second after the first. The gateway picks one of the two slots for transmission. The communication parameters of this slot, like spreading factor and bandwidth, depend on the agreement between the node and the gateway. The first slot uses lower spreading factors most of the time and requires less time on air, while the second slot uses more reliable parameters.

[2]In this work, the definition of spreading factor from the LoRa community is used rather than the typical definition of spreading factor: chip rate over symbol rate.

1-persistent MAC. The LoRaWAN MAC layer also provides a way for reliable communication: with acknowledgments or confirmations. In the case that a confirmed message has been transmitted by a node, the gateway is obliged to acknowledge this message. However, in the unconfirmed case, a gateway sends acknowledgments at will. These acknowledgments are not compulsory, but confirm a proper working connection between a node and the rest of the network. If a confirmation is not received, the MAC layer can decide to switch to more robust modulation schemes by selecting a higher spreading factor or decreasing the bandwidth.

2.1.2 Network Server. An important part of the network which is often neglected is the backbone, as visualized in Figure 1. The left part of the figure shows the long range low power network and sensor discussed in the first part of this section. The right part shows the backbone structure. The gateways collect all information, control and data from the LoRa network and forward this to the network server. The network server on its turn, selects the correct parameters to use, performs localization and forwards data to remote backends of customers. To date, the most well-known network server is the network server from The Things Network [2] that provides a free and open source network to connect LoRa sensors. Many other vendors also exist. Note that although a gateway has a powerful receiving chip, it is not capable of adding intelligence to the network. This capability is provided at the network server.

2.2 Related Work

Many work has been done in estimating the performance of LoRA networks, such as [3, 4, 11]. Although their conclusions are interesting, they only use a simplified MAC protocol. This calls for a powerful network simulator that is useful to study the real network performance.

Several simulation tools have been proposed for LoRaWAN. The most well-known LoRaWAN simulator is the LoRaSim built with python [5, 21]. It is open source and gives great insights in the LoRaWAN performance. However, LoRaSim does not implement

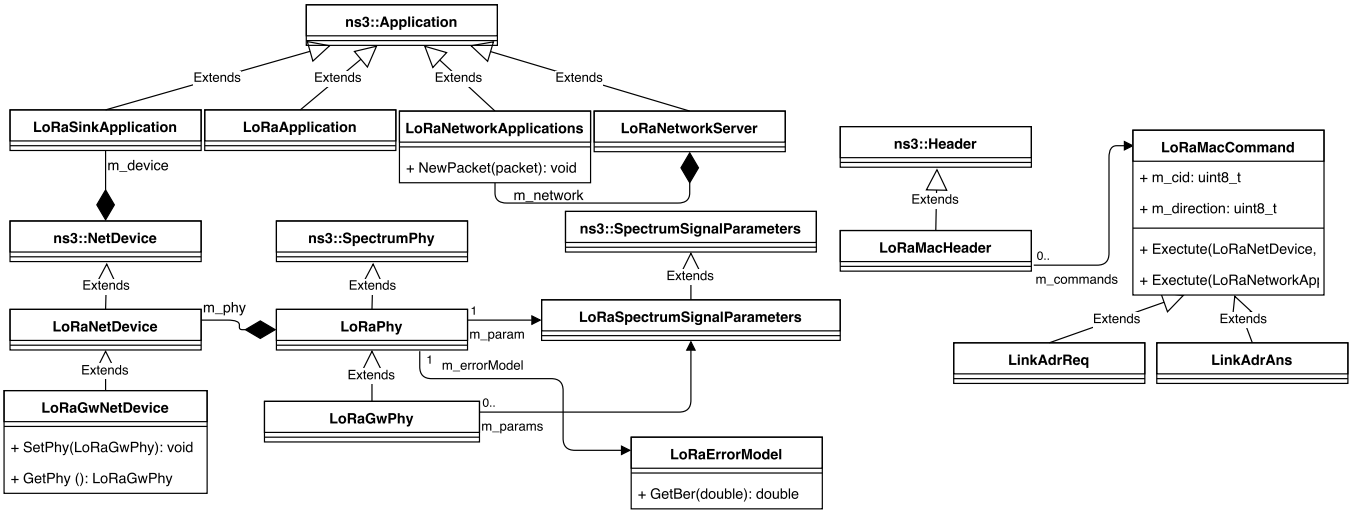

Figure 2: Our implemented LoRaWAN model in a unified modeling language (UML). We present all the relevant classes for the LoRaWAN implementation. The highlighted functions and variables are the important parts for the hierarchy

acknowledgments. Thus, it cannot be used to study the network performance where nodes switch their spreading factor based on the feedback or absence of feedback from the gateway. Similarly, an Omnet++ implementation has been proposed in [19]. It implements an Automatic Data Rate (ADR) scheme where nodes can update their spreading factor and power at runtime.

For ns-3, two different modules have been proposed in literature. The authors in [10] proposed a complete LoRa module. It features MAC commands, different overlapping networks and multi-gateway support. Besides these interesting features, this module has some drawbacks. First, it can only send LoRa messages, so it is impossible to simulate the effect of interference. Next, similar chirp rates do not have effect on each other. Due to the chirp spread spectrum technique, spreading factor 9 with 125 kHz bandwidth has a similar chirp rate compared to spreading factor 11 with a 250 kHz bandwidth. Another small drawback is that all the gateways in this model are virtually directly connected to the network server, so the packets cannot be routed over IP.

Independently from the previous implementation, the authors in [6] have proposed their solution. Their proposal also supports multiple gateways and overlapping networks. However, they did not include MAC commands. With their solution, interfering networks are possible as they accept interference from any network working on the same channel and frequency. Also in this implementation, the network is not connected to the IP layer, but directly to gateways.

Compared to the above models, our implementation is totally compliant with the LoRaWAN v1.0 class A specification. It is highly configurable. Its flexible backbone architecture allows for easy integration of new protocols. Our model supports distributed gateways that are connected over an IP network to the network server that controls the whole network. We also provide base classes for the easy implementation of new applications on the network server and new MAC commands. With this model, we have investigated many aspects of LoRa networks, such as the effect of different spreading factors [15], the effect of interference [14], the reliability and scalability [17], etc. Our model can also be used to study the effect of downlink messages [13] and multiple gateways [6, 21].

3 IMPLEMENTATION

In this section, we present the implementation details of our LoRaWAN ns-3 module. We start from the physical layer, then the MAC layer and finally the application layer[3]. Our implemented PHY and the MAC layers are connected to the standard ns-3 callbacks to communicate with the remainder of the network stack. It allows us to reuse a significant amount of the code within the ns-3 framework. A helper class is provided to automatically connect all required callback functions. As a result, the LoRaWAN NetDevice can easily be configured with a default set of parameters, but these parameters can also be altered using the corresponding interfaces.

3.1 Physical Layer

Our physical layer is similar to *lr-wpan* [8] and is built upon the *SpectrumPhy* code. The standard interfaces for *SpectrumPhy* allow us to easily create a *SpectrumChannel* simulating the 868 MHz band for LoRa traffic. To allow fast simulations, we limit the spectrum of this channel by the default LoRa Channels specified in LoRa in the 868 MHz ISM band, being 868.1 MHz, 868.3 MHz and 868.5 MHz. Packets on this channel are modeled as a subclass of *SpectrumSignalParameters*, where the spreading factor and bandwidth of this signal are added as extra features. These parameters are required during reception of this message.

The physical layer consists of two classes: *LoRaPhy*, the physical layer for a node, and *LoRaGwPhy*, the physical layer for a gateway.

3.1.1 Phy for Nodes. At each node, *LoRaPhy* keeps track of the noise on all the channels and records which spreading factors have been used. It also checks if similar LoRa patterns are being used,

[3]For better readability, the names of classes and functions are highlighted in italics.

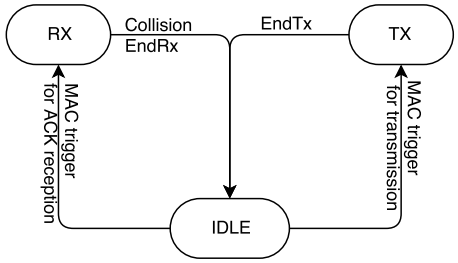

Figure 3: The finite state machine for the physical layer

for example, the chirp rate of a signal with SF9 and bandwidth 125 kHz is identical to the chirp rate of a signal with SF11 and bandwidth 250 kHz. This similarity is a result of the chirp spread spectrum modulation in LoRa. Whenever a new packet arrives, *LoRaPhy* keeps track of the instantaneous signal-to-noise ratio of this packet. This is different from the *LrWpanPhy* where only the average SNR over the whole packet is considered. Keeping track of this instantaneous SNR allows short high powered messages to destroy a colliding message. So, with each new interfering signal that arrives at this PHY layer, the bit errors are recalculated up to that point using the formulas presented in [16]. When a message is finally successfully received, it is forwarded to the MAC layer for further processing. In the other direction, when a packet is ready for transmission (forwarded from the data link layer to *LoRaPhy*), *LoRaPhy* will immediately try to send this packet. It is the MAC layer's duty to agree with the timings in LoRa. When the transmission fails for any reason, the data link layer is notified.

Figure 3 shows the finite state machine for the physical layer of the node. LoRa messages are only decoded when the physical layer is in RX mode. Transmission of packets is only possible when the physical layer is in TX mode. Whenever the physical layer is done with the transmitting or receiving, it returns to the IDLE mode.

3.1.2 Phy for Gateways. The implementation of the physical layer for gateways is slightly different from the physical layer for nodes: a gateway is able to receive multiple packets simultaneously. Thus, a different PHY layer is required for gateways. However, since most parameters are identical to those used in the PHY layer for nodes, the physical layer class for gateways *LoRaGwPhy* is implemented as a derived class of *LoRaPhy*, shown in Figure 2. The main difference between them falls into the receiving part. *LoRaGwPhy* keeps a list of all incoming LoRa signals instead of only adding those to the noise. For each of these incoming LoRa signals, the bit errors are calculated. Notice that each incoming signal contributes to the noise floor of the other incoming signals, even when the signals use different spreading factors. In our implementation, a gateway can track up to eight LoRa signals simultaneously due to the restrictions in hardware [18]. Transmitted messages are also handled as incoming messages as well. Uplink and downlink in LoRaWAN occur on the same channels and use the same modulation. Therefore, the self-interference of a gateway will destroy all messages on that channel of that gateway [13]. That is, the gateway does not have any full-duplex capabilities.

The finite state machine for the physical layer of the gateway is omitted due to its simplicity and its similarity to the one for the physical layer of the node.

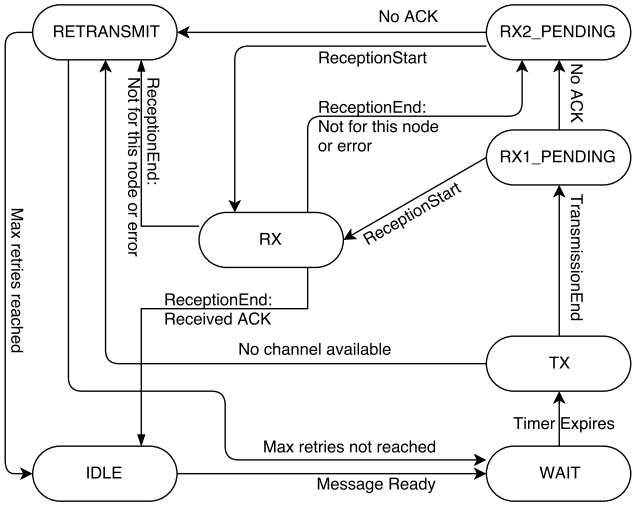

Figure 4: The finite state machine for the MAC layer

3.2 MAC Layer

Upon the physical layer is the LoRaWAN MAC protocol. The implementation of this layer is also divided into two different classes due to the difference between uplink and downlink. *LoRaNetDevice* contains the implementation of the MAC layer of a node, while *LoRaGwNetDevice* implements the MAC layer for the gateway. Similary to the case in the physical layer, the class *LoRaGwNetDevice* for the gateway is a derived class from the class *LoRaNetDevice* of the node, as visualized in Figure 2.

3.2.1 MAC for Nodes. The MAC layer of a node is simple. Whenever a message comes in, the node checks if there is a free channel on which it can send out the message. This check needs to be done to comply with the duty cycle regulation from both the regulation point of view [7] and the LoRaWAN restrictions [20]. If there is a channel available, the node waits for a random delay and sends its message. If there is no channel available, the node waits for a random delay and tries again. After each successful transmission, two receiving slots are opened for potential downlink messages. If the first slot is used, the second downlink is automatically disabled.

Nodes can send two types of traffic: confirmed or unconfirmed traffic. The former traffic retransmits the packet up to 8 times until the gateway responds with an acknowledgment, while the latter traffic transmits only once. In both cases, the two receiving slots are opened for downlink messages. If an acknowledgment is expected, but not received, the MAC layer automatically retransmits the message. For confirmed messages, the MAC layer also increases the spreading factor every two lost acknowledgments to increase reliability. For unconfirmed messages, nodes will switch to a higher spreading factor as well. This happens after 96 unacknowledged messages. After 96 messages, the node assumes that communication is not possible with the current spreading factor and switches to a higher one. To warn the gateway that this deadline is approaching, a node transmits a flag after 64 non-confirmed messages.

The above process is summarized with the finite state machine shown in Figure 4. The device starts in the IDLE state. Whenever

a new packet arrives, it waits at least a second before going to TX. After the TX mode, there are 2 states waiting for a downlink message from the gateway. When the sequence is done, the device goes to the RETRANSMIT state to check if the packet needs to be retransmitted or not or goes immediately to the IDLE state. If no messages are in the queue, the MAC layer returns to the IDLE state.

3.2.2 MAC for Gateways. While a node uses a random access approach to transmit its message to the intended receiver, the gateway makes use of a synchronized downlink. Whenever a node transmits a message, at least 1 second later, the gateway can send its reply or message. Whenever a new message is received, each gateway schedules these downlink slots. It is important that each gateway does this for the synchronization of the downlink. The received message then gets forwarded as a whole to the application layer (explained in Section 3.3.2). In the meanwhile, the gateway waits for a response from the network server. If it replies with a message to discard the transmission, the scheduled slots are discarded and the gateway continues its operation. In the other case, it checks if there are any ongoing transmission and if not, it sends its downlink message. If there is a reception going on, it tries to preserve this message and schedules the downlink transmission in the next slot. It is important to note that messages arriving from the physical layer are forwarded as a whole to higher layers. This means that the payload as well as the MAC header are forwarded to the transport layer and up. This MAC header is needed for later processing, e.g. in the *LoRaNetworkApplications* to see whether a MAC command was accepted or not. The only difference between the packet coming in and the packet forwarded to higher layers is an RSSI value that is added to the end of a packet to assess the quality of the link at the remote servers.

3.2.3 MAC Commands. An important aspect in LoRaWAN are the MAC commands. These MAC commands are embedded in the MAC header. MAC commands in LoRa provide a way to remotely configure the transmission parameters of nodes in the network. For example, a gateway can use a MAC command to request a lower duty cycle. To allow easy integration of all MAC commands in the LoRa MAC header, an abstract base class *LoRaMacCommand* has been implemented in our model that provides all basic functionality, see as well in Figure 2. In that regard, the most important function is the *Execute* function. There are two versions of this function: one for nodes and the other for the network side. The execute function for nodes acts directly upon the MAC layer of the nodes, and changes parameters there. The execute function for the network side is different in the sense that a gateway does not have a complete overview over the network and cannot provide all answers to the requests of nodes. Therefore, the network server or rather a network application will provide the answer for all requests, so the *Execute* function will be called in *LoRaNetworkApplications*.

3.3 Transport and Application Layer

The transport and application layer resides above the MAC layer. The networking layer is skipped because there is no routing involved in LoRaWAN. As has been presented in Section 2, a gateway translates the messages on the network to an IP packet. This will be described in Section 3.3.2.

3.3.1 Application Layer on the Node Side. The application on the node side is a simple constant packet rate module that sends a message at regular intervals. The time of these intervals is configurable as well as the length of the messages on the channel. To skip the networking layer, the *LoRaNetDevice* is bounded with the socket of this application.

3.3.2 Application Layer on the Gateway and Backbone. The application layer of LoRaWAN has been divided over multiple nodes. This is similar to the real implementation of LoRaWAN. In this split, *LoRaSinkApplication* runs on each gateway, while *LoRaNetwork* and *LoRaNetworkApplications* run on a different single node somewhere in the network. The latter two applications run over an IP network.

LoRaSinkApplication. It is a translation application. It collects all the messages received by the LoRa gateway, encapsulates it in IP packets, and sends them to a remote server, and vice versa. The remote server in our model is the *LoRaNetwork*. Besides, the packet content from the *LoRaGwNetDevice* is kept intact, including the MAC header and RSSI trailer, as has been mentioned before, although a standard interface between the forwarder and the network backend has been provided by the LoRa alliance. In our model, this standard interface has not been implemented.

LoRaNetwork. This application is the brain of the network. It is the orchestrator of the network that determines which gateway is responsible for talking to a node in its network. It also decides which nodes are part of the network and which nodes are not. It is therefore important to whitelist the MAC address of each allowed node in the network with the *WhitelistDevice* command. This function makes sure that whenever a message is received from one of these devices, a proper answer is created. In addition, this class keeps track of the unique messages that arrive at all gateways. Potentially, multiple copies are received at different gateways, but the remainder of the network is only interested in the unique messages. This application therefore discards any received copy without passing it to different applications and only signals these applications with a trace source on unique messages. The *LoRaNetwork* also provides an interface to send data to a specific node. When data needs to be transmitted from any *LoRaNetworkApplication*, it is sufficient to send it to the *LoRaNetwork* that decides which LoRa gateway will respond a specific node.

LoRaNetworkApplication. This application layer is the application layer for implementing algorithms in the LoRaWAN network. They could do power control, store data and so on. The only requirement when installing them is to connect it as a trace sink to the trace source of the *LoRaNetwork*. This will trigger when a new packet has been received, to which these applications can react. Some important *LoRaNetworkApplications* that we have already implemented include a power application that resets the internal power and an application that controls the spreading factor of nodes, as used in our previous work [15]. Notice that *LoRaNetworkApplication* can also translate LoRaWAN messages to IPV6 messages, to communicate with a different server.

3.4 Power Consumption

An extra class is added to track power consumption. The *LoRaRadioEnergyModel* tracks the energy consumption of the physical layer. The energy model tracks the state of the physical layer: RX, TX and IDLE state. All these states have a corresponding current that gets drawn from the energy source connected to the node. Multiplying this current with the supply voltage of the energy source and the time of the state, the used energy is calculated and subtracted from the energy source attached to the node.

3.5 Potential Interface with 6LoWPAN

An interesting direction is to interface LoRaWAN with the 6LoWPAN, or the *sixlowpan* module. This enables IPv6 addresses for all LoRa devices and creates extra flexibility for end-to-end solutions and simulations. At the moment, this is not developed in our model yet, although little effort need to be done to enable this. The only requirement is to create an extra interface that forwards all the received data, but strip the *LoRaMacHeader*. In this case, the devices could also make use of IPv6 addresses. It is important to note that all the messages received from the LoRa nodes should also be forwarded as a whole to the network server, to allow for further exploitation of the messages in order to optimize the whole network performance.

The only drawback of the above approach will be the multiple copies of a message transmitted by a sensor node. Every receiving gateway will forward the packet and send it to the intended destination. It is then up to the application protocol to filter out these messages. Also, due to the timing constraints of TCP, only UDP can be used in LoRa networks.

4 EVALUATION

To showcase the performance of our implementation, we have run simulations in three different scenarios:

- *Scenario 1:* the first scenario is a simple network topology where nodes are spread in a circle and send data to a central gateway. All the nodes transmit unconfirmed data to the network server, and hence do not wait for any acknowledgments. Therefore, the gateway only responds within every 96 messages to prevent the nodes switching to higher spreading factors.
- *Scenario 2:* the second scenario is a variation of the first scenario. Also, a circular set-up is considered, but with more gateways. These gateways are located as shown in Figure 5. This scenario is used to show the space diversity of LoRaWAN and that different gateways can connect to the networks and that multiple gateways can respond to different nodes in the network.
- *Scenario 3:* the final scenario is again a variation of the first scenario. Instead of using unconfirmed traffic, all nodes will transmit confirmed messages. Through this scenario we will show that it is impossible to acknowledge every message in the network because the low power network can not provide such throughput.

A tremendous amount of work has also been carried out to measure the path loss of GSM signals in the 900 MHz band. The most well known model for this band, and available in ns-3 is the Okunura-Hata model [9]. This model together with Rayleigh fast fading (Nakagami-m with default parameters) [12] has been used as the path loss model. The packet generation is deterministic, no variation is added. And the nodes are uniformly distributed over the circle.

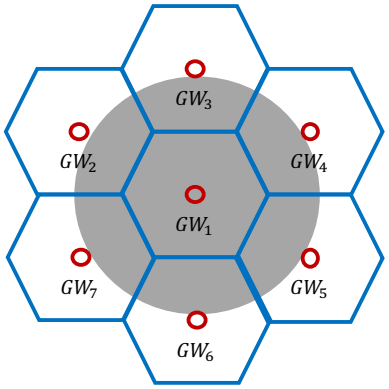

Figure 5: The network topology used in the simulation. This figure shows both the gateway used in the simulated single gateway scenario (GW_1), like the Scenario 1 and Scenario 3, as well as the multi gateway scenario where all the seven gateways are used (Scenario 2)

Table 1: Parameters used in the simulations

Parameter	Value	Unit
Number of nodes	100, 500, 1000	/
Packet length (excluding header)	51	byte
Maximal distance to the GW_1	1000	meter
Distance between GWs	1000	meter
Average inter-packet interval	120	second

Before diving into results, we summarize the simulation parameters in Table 1. In the simulation of each of the three scenarios, we use the same parameters. Two metrics are used to evaluate the network performance: the total network throughput (the number of received packets are the gateway(s)) in function of the number of nodes, and the reliability (the packet error rate) in function of the distance to the gateway. In the last subsection, we also present some statistics of our model in the simulation, such as memory usage and simulation time.

4.1 Single Gateway

The first simulation is just a single cell scenario. All nodes try to communicate with the gateway GW_1. Due to the ALOHA nature of the protocol, all nodes have to contend for sending their data to GW_1. The simulation results are shown in Figure 6 and Figure 7. The former figure presents the reliability as a function of the distance. Clearly, the closer to the gateway, the higher the received power and thus the lower the packet error ratio. This is a result of the capture effect. This effect lets high powered packets survive collisions, while low powered packets are discarded. As expected, more devices results in a higher packet error ratio. Figure 7 shows the

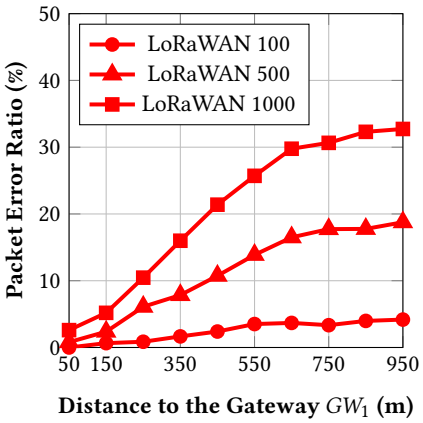

Figure 6: Average packet error ratio versus distances to the gateway *(Scenario 1)*. Close to the gateway, the packet error ratio is very low, while far from the gateway, the performance goes down. Also, the more contending nodes, the higher the packet error ratio

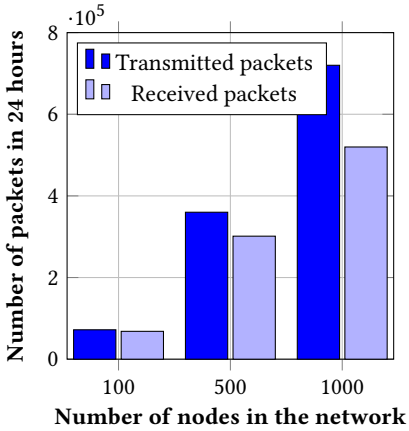

Figure 7: The packets transmitted by all the nodes *(Scenario 1)*. In dark blue, all transmitted messages are counted. The light blue bars show the packets that are successfully decoded at the gateway. The difference between the two bars is the amount of packets that got lost

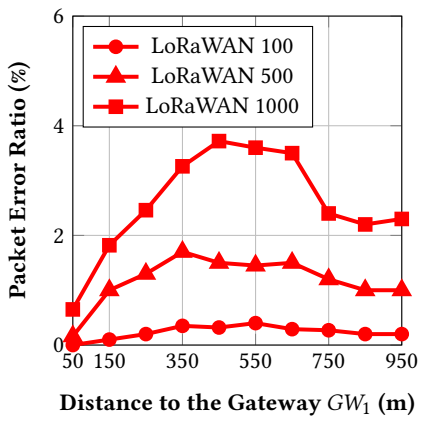

Figure 8: Average packet error ratio versus distances to the gateway *(Scenario 2)*. Close to the gateway, the packet error ratio is very low, and vice verse. However, 500 meters onwards, nodes are closer to the gateways at the edge of the circle and benefit from the capture effect

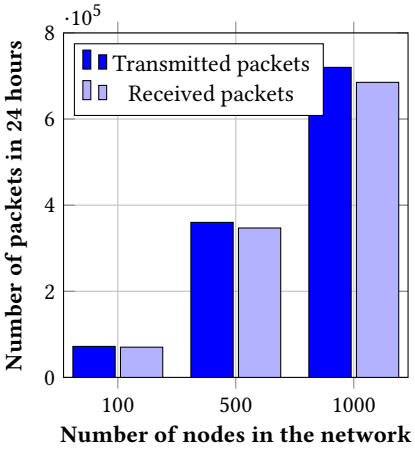

Figure 9: The packets transmitted by all the nodes *(Scenario 2)*. In dark blue, all transmitted messages are counted. The light blue bars show the packets that have been successfully decoded at the gateway. Notice that more gateways increases the reliability significantly

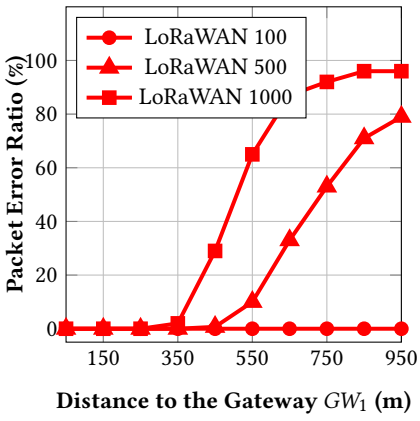

Figure 10: Average packet error ratio versus distances to gateway *(Scenario 3)*. For a small number of nodes, there is nearly no error. But with more nodes, the packet error ratio goes up drastically, deteriorating the channel for nodes far from the gateway

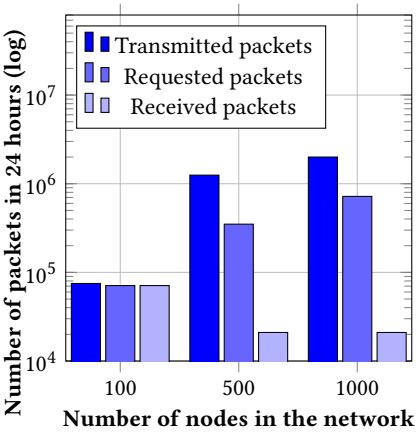

Figure 11: The packets transmitted by all the nodes *(Scenario 3)*. In dark blue, all transmitted messages are counted. The middle bars show the amount of packets sent to the MAC layer from higher layers. Light blue bars denote the packets decoded at the gateway

total amount of packets that are transmitted from all the nodes, and the number of packets that are successfully received by the Gateway GW_1. In other words, the left bar shows the total amount of transmitted messages, while the bar on the right shows the amount of messages that are successfully received. As has been explained earlier, the degradation of the performance is due to the random access of the share channel in the network.

4.2 Multiple Gateways

Next, we evaluate the performance of our model in a multi-gateway scenarios. We use the network topology where seven gateways are deployed as illustrated in Figure 5. The simulation results are shown in Figure 8 and Figure 9. First, we can notice that the reliability in this setup is considerably higher than in the single gateway scenario. Figure 8 shows a 30% lower packet error ratio. Figure 9 also

demonstrates a higher reliability in the scenario of multiple gateways. The reason behind these improvements is as follows: when a collision occurs between two nodes, it is unlikely that both of them are exactly at the same distance from the gateway. Therefore, one of the packets sent from these two nodes will be received by the central gateway GW_1, and the packet from the other node will be discarded. When there are more gateways available in the network, it is possible that the previous discarded packet will be received by a neighboring gateway and therefore will be successfully decoded. As a result, the network performance is improved.

4.3 Acknowledgments

In the final scenario, the first setup was reused again, but instead of sending unacknowledged transmissions, acknowledgments were requested for each transmission. Figure 10 shows that for a small number of devices, an extremely low packet loss can be observed. For a network with 100 nodes, the packet error ratio is 0 in all cases. However, when the number of nodes increases, the reliability drops. The reason for this low reliability can be found in Figure 11. It shows that the amount of packets transmitted by the nodes increases significantly when the number of nodes grows. This value should be compared with the middle bar which shows the actual amount of messages that arrived on the MAC layer. Finally, the third bar shows the amount of message that has actually been received. These figures show that acknowledgments in LoRaWAN do not scale. Extra transmissions, especially transmissions with high spreading factors, cause a lot of collisions. These higher spreading factors are a result of previous collisions. Therefore, nodes far from the gateway get stuck in a vicious circle: while trying to get a more reliable channel, they only increase their probability on extra collisions and quickly hit their duty cycle limitation.

4.4 Some Statistics

Finally, we provide some statistics concerning simulation time and memory usage. The results are summarized in Table 2. In the simulation, we use an Intel Xeon X7750 CPU and 128 GB of memory. Each of these values corresponds with a simulation of the provided number of nodes, with the parameters described above. All nodes send messages every 2 minutes for 24 hours. The measurements show that small networks can be simulated pretty fast, and that acknowledgments take a lot of efforts. For the latter case, the requirements increase significantly. This is due to the retransmissions and acknowledgments in the network.

Table 2: Statistics of the resources used in our simulations

Number of nodes		100	500	1000
Scenario 1	Memory (MB)	465	2175	4355
	Time (s)	551	11756	47136
Scenario 2	Memory (MB)	517	2235	4378
	Time (s)	971	13773	48654
Scenario 3	Memory (MB)	1364	3635	10029
	Time (s)	1653	28894	115523

5 CONCLUSION

This work proposed a new LoRaWAN module in ns-3. This module adds long range communication to the ns-3 simulator. In this work, we detailed the LoRaWAN structure and how we implemented this within the ns-3 framework. Our implementation is a flexible approach that can easily integrate new algorithms on the server side to investigate new protocols and the effect of different parameters on the network. We have also run simulations in 3 small scenarios from which we show that acknowledgments are not scalable and that multiple gateways improve the network performance considerably.

ACKNOWLEDGMENTS

This work is supported in part by the Flemish FWO SBO SAMURAI and the IWT CELTIC O&O project ASUA/RoCCS.

REFERENCES

[1] 2018. LoRa Alliance. (2018). http://www.lora-alliance.org
[2] 2018. The Things Network. (2018). http://www.thethingsnetwork.org
[3] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. 2017. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine* (2017).
[4] A. Augustin, J. Yi, T. Clausen, and W. Townsley. 2016. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors* (2016).
[5] M. Bor, U. Roedig, T. Voigt, and J. Alonso. 2016. Do LoRa Low-Power Wide-Area Networks Scale?. In *Proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. Malta, Malta.
[6] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke. 2017. Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3. *IEEE Internet of Things Journal* 4, 6 (2017), 2186–2198.
[7] ETSI. 2010. *Electromagnetic Compatibility and Radio Spectrum Matters (ERM); Short Range Devices (SRD); Radio Equipment to be Used in the 25 MHz to 1000 MHz Frequency Range with Power Levels Ranging up to 500 mW; Part 1: Technical Characteristics and Test Methods.* ETSI EN 300 220-1.
[8] K. Ghomali, N. Elkamoun, K. Hou, Y. Chen, J.P. Chanet, and J. Li. 2013. A new WPAN Model for ns-3 Simulator. In *France-China International Workshop on New Information Communication Science and Technology for Sustainable Development.* Clermont-Ferrand, France.
[9] M. Hata. 1980. Empirical Formula for Propagation Loss in Land Mobile Radio Services. *IEEE Transactions on Vehicular Technology* 29, 3 (Aug 1980), 317–325. https://doi.org/10.1109/T-VT.1980.23859
[10] D. Magrin, M. Centenaro, and L. Vangelista. 2017. Performance Evaluation of LoRa Networks in a Smart City Scenario. In *Proceeding of the IEEE ICC*. Paris, France.
[11] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen. 2016. Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology. In *Proceedings of the European Wireless Conference.* 1–6.
[12] M. Nakagami. 1960. The m-Distribution, a General Formula of Intensity of Rapid Fading, IWC Hoffman, Ed. (1960).
[13] A. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara. 2017. Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?. In *Proceedings of the IEEE GLOBECOM*. Singapore, Singapore, 1–6.
[14] B. Reynders, W. Meert, and S. Pollin. 2016. Range and Coexistence Analysis of Long Range Unlicensed Communication. In *Proceedings of the IEEE International Conference on Telecommunications (ICT)*. Thessaloniki, Greece.
[15] B. Reynders, W. Meert, and S. Pollin. 2017. Power and Spreading Factor Control in Low Power Wide Area Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*. Paris, France.
[16] B. Reynders and S. Pollin. 2016. Chirp Spread Spectrum as a Modulation Technique for Long Range Communication. In *IEEE Symposium on Communications and Vehicular Technologies (SCVT)*. Mons, Belgium.
[17] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin. 2018. Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling. *IEEE Internet of Things Journal* (2018). https://doi.org/10.1109/JIOT.2018.2815150
[18] Semtech Corporation [n. d.]. *SX1301 Datasheet.* Semtech Corporation.
[19] M. Slabicki, G. Premsankar, and M. Di Francesco. 2018. Adaptive Configuration of LoRa Networks for Dense IoT Deployments. (2018).
[20] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent. 2015. *LoRaWAN Specification.* http://www.lora-alliance.org
[21] T. Voigt, M. Bor, U. Roedig, and J. Alonso. 2017. Mitigating Inter-Network Interference in LoRa Networks. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. Uppsala, Sweden, 323–328.