DEVELOPING A CONGESTION MANAGEMENT SCHEME TO REDUCE THE IMPACT OF CONGESTION IN MIXED TRAFFIC LORAWANS

by

Jaco Morné Marais

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

August 2022

SUMMARY

DEVELOPING A CONGESTION MANAGEMENT SCHEME TO REDUCE THE IMPACT OF CONGESTION IN MIXED TRAFFIC LORAWANS

by

Jaco Morné Marais

Promoter:	Prof. Adnan M. Abu-Mahfouz
Co-promotor:	Prof. Gerhard P. Hancke
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Philosophiae Doctor (Computer Engineering)
Keywords:	Long Range, Long Range Wide Area Network, Low-power Wide Area Network, Internet of Things

The development of wireless technologies such as the Long Range Wide Area Network (LoRaWAN) protocol has enabled connectivity for a wide range of devices for optimisation and automation purposes. These types of technologies support the rapid expansion of the Internet of Things (IoT) deployments which frequently consist of large quantities of low-cost connected devices which require a network protocol suitable to their needs. These devices are frequently energy-constrained, and communication is a major energy consumer. These connected devices typically generate a lot of uplink traffic, i.e., packets sent from a device to a Gateway (GW). Additionally, minimal downlink traffic, i.e., packets sent from the GW to a device would also be required. A significant source of downlink traffic is the transmission of Acknowledgements (ACKs) to confirm the reception of packets sent by devices.

The advantages of LoRaWANs are excellent scalability if the network is uplink-dominated (no or minimal packets require downlink traffic in response). Traffic in these networks can be defined as being one of two types: unconfirmed (no ACK required) and confirmed (ACK is required). A network's scalability is quickly reduced by the presence of even low amounts of downlink traffic as this causes congestion. The impact of supporting mixed-traffic (unconfirmed and confirmed) on scalability is

minimal in small networks but increases quickly in large networks, especially if they only contain one gateway.

The contributions of this research consist of three objectives. The first objective was to study the cause and impact of congestion in mixed-traffic LoRaWANs operating with only one gateway. By studying the root causes of congestion, new solutions were developed to improve network performance (in the form of improved packet delivery). These new solutions were the focus of the second objective and led to a congestion management scheme and an algorithm that reduces congestion. The third objective was to study how congestion should be managed in battery-less capacitor-based devices that obtain energy through energy harvesting.

The current research literature revealed that the two main reasons confirmed traffic leads to congestion are that LoRaWAN gateways are half-duplex and that Duty Cycle (DC) restrictions limit the number of sent ACKs. Thanks to their half-duplex design, a gateway cannot receive any transmissions whilst transmitting downlink messages, causing it to miss packets belonging to both traffic types. DC restrictions may prevent the transmission of an ACK, resulting in the retransmission of packets a GW has successfully received. Furthermore, an ACK must be sent during one of only two timing windows (called receive windows), with the second window using a very low data rate (increasing transmission time).

The literature survey revealed that the LoRaWAN protocol does not have a congestion monitoring and response mechanism. In response to this, a novel method to monitor congestion as well as a response mechanism was developed for LoRaWANs. The response mechanism is a novel algorithm tasked with reducing congestion, which is done through requests for the aggregation of confirmed packets by nodes. A commonly used open-source ns-3 based LoRaWAN simulator (“lorawan") served as a suitable research platform. Network performance was studied via packet reliability, by examining the success probability of unconfirmed traffic (ULPDR) and confirmed traffic (CPSR). The developed algorithm was then evaluated using a series of simulations to determine its impact on network performance across congestion levels and how several LoRaWAN parameters influence this impact. Congestion was created by increasing the packet arrival rate (traffic volume) in the network by adjusting the transmission interval of nodes.

Congestion was monitored through the newly developed Adaptive Congestion Scheme (ACS), which

is similar in design to the protocol's Adaptive Data Rate (ADR) scheme. The developed scheme is split into two parts, with one operating as part of the Network Server (NS) and another on the device, capable of communicating with each other. When congestion is detected, the developed algorithm called groupedPackets is activated to improve packet reliability.

The groupedPackets algorithm was designed to reduce the traffic volumes generated by confirmed nodes by requesting that they aggregate their application packets. This causes nodes to send larger but less frequent confirmed packets. A major cause of congestion was the number of ACKs required, which is now reduced due to fewer confirmed packets being sent. This change enabled GWs to be able to send more ACKs before DC restrictions were met.

This improvement in the ratio between required and transmitted ACKs caused the number of retransmissions of confirmed packets to reduce, which resulted in less interference thanks to fewer packet collisions. Finally, this also mitigated the half-duplex nature of GWs, as the number of packets lost during GW transmissions was reduced as fewer transmissions were required.

When groupedPackets was evaluated, it enabled improved CPSR and ULPDR values when compared to identical networks operating using the standard protocol. The algorithm showcased considerable improvements in CPSR in congested networks (high packet arrival rates) with only minimal impact at lower rates as CPSR was already above 98 %. Performance was further increased when the number of retransmissions was set to eight, allowing aggregated payloads multiple chances at success. Increasing the base packet size from 10 B to 20 B slightly reduced groupedPackets' performance as this hampered the number of packets it could aggregate. Using groupedPackets to aggregate unconfirmed packets led to marginal improvements in a best-case scenario and no improvements in less favourable scenarios.

In battery-less LoRaWANs, the radio is the primary energy user. The standard groupedPackets algorithm underperforms in these networks as it is unaware of energy constraints. An improved version was developed, in which the node informs the NS side of the algorithm of energy constraints after optimising the amount of sent payloads. This version allowed confirmed nodes whose capacitance is in a "transition" range of values to increase the number of sent payloads, which resulted in an increase in CPSR for this range.

Enabling retransmissions did result in improved packet delivery in both standard LoRaWANs and the improved groupedPackets networks. The effectiveness of this is dependent on sufficient energy being available for retransmissions. Confirmed nodes require more energy than unconfirmed nodes, and as a result, the minimum capacitance to support multiple transmissions differed between the two types. Matching the transmission settings used by receive window two to those used by window one resulted in a significant increase in CPSR in standard networks and also offered a slight improvement in groupedPackets networks. By matching the settings, more ACKs can be sent, which reduces the number of retransmissions. Energy consumption was also reduced due to the higher data rate used.

Compared with a standard network, combining groupedPackets, enabling retransmissions and matching the window settings were effective. The combination improved ULPDR by $\approx$ ten percentage points, increased CPSR by $\approx$ 40 percentage points and can be achieved by increasing the node's capacitance from 6 mF to 7.5 mF.

This work contributed to an improved version of the protocol in which the viability of the use of confirmed traffic was increased in mixed-traffic LoRaWANs. The developed congestion monitoring scheme proved to be effective in detecting congestion and taking action. This research found that the developed algorithm can significantly improve packet reliability, at the cost of an increased delay and node capacitance in the case of battery-less devices.

LIST OF ABBREVIATIONS

ACK Acknowledgement

ACS Adaptive Congestion Scheme

ADR Adaptive Data Rate

BW Bandwidth

CMSA Carrier Sense Multiple Access

CPSR Confirmed Packet Success Ratio

CRC Cyclic Redundancy Check

DC Duty Cycle

DL Downlink

EIRP Equivalent Isotropically Radiated Power

EU European Union

IIoT Industrial Internet of Things

IoT Internet of Things

ISM Industrial, Scientific and Medical

LIFO Last In First Out

LoRa Long Range

LoRaWAN Long Range Wide Area Network

LPWANs Low-power Wide Area Networks

MAC Medium Access Control

NS Network Server

PDR Packet Delivery Ratio

PER Packet Error Ratio

PHY Physical layer

QoS Quality of Service

RFU Reserved for Future Use

SF Spreading Factor

SNR Signal to Noise Ratio

ToA Time on Air

TP Transmit Power

TPNS Total Packets not Sent

TPS Total Packets Sent

TTN The Things Network

UL Uplink

ULPDR Uplink Packet Delivery Ratio

WSN Wireless Sensor Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

The use of technology to achieve productivity and efficiency improvements is common in several aspects of society. One way this is performed is through a concept referred to as the Internet of Things (IoT), which holds the view that improvements can be achieved through deploying intelligent devices which provide monitoring and control capabilities. Frequently, these improvements only occur when devices are deployed on a large scale, which requires wireless technologies suitable to the task.

IoT deployments are growing globally, and as a result, new types of wireless networks, referred to as Low-power Wide Area Networks (LPWANs), were developed with these requirements in mind [1, 2]. One such technology, namely Long Range Wide Area Network (LoRaWAN), was found to be effective for a range of IoT use cases [3, 4] and therefore worthy of further study. Thanks to its energy efficiency, LoRaWAN is also becoming a popular choice for a network with limited energy resources. This technology consists of a proprietary Physical layer (PHY), namely Long Range (LoRa) and an open-source Medium Access Control (MAC) layer referred to as LoRaWAN.

Researchers found that LoRaWANs are well suited for smart city deployments due to their support for large amounts of devices (scalability). Furthermore, they support different traffic types, allowing for bidirectional communication that can either be unconfirmed or confirmed (requiring an Acknowledgement (ACK) to indicate successful reception). However, the technology has limitations, and scalability is negatively impacted by the presence of downlink traffic (from a gateway to an end-device). This type of traffic has two primary sources, namely ACKs and any data designated for the application running on an end-device, with ACKs being much more common. The impact of downlink traffic on

scalability is minimal in small networks but occurs in large networks even with a low ratio of confirmed to unconfirmed traffic.

1.1.2 Research gap

Since the release of the LoRaWAN protocol in 2015, the conducted research has focused on a few areas such as exploring the PHY, performance evaluations examining range and throughput, and the protocol's Adaptive Data Rate (ADR) scheme. Surveys, such as [5, 6, 7, 8, 9], reveal that only a subset of work touched on the concept of acknowledgements, with even fewer studying their impact on a network. Additional research is required to understand congestion, especially in mixed-traffic (i.e. bidirectional) networks. Some authors have noted the performance impact of ACKs in a network and have given suggestions based on modifying existing network parameters, but these suggestions tend not to be fully evaluated. Additionally, combinations of these parameters or novel methods are rare. Finally, the energy impact of the various methods, be they standard parameters or custom methods, tend not to be studied. The energy impact of design choices is important for energy-constraint nodes, which commonly have to make do with the energy provided by energy harvesting methods.

Another approach commonly taken by researchers to the scalability problems of LoRaWANs is to replace the MAC component with their novel protocol, commonly using scheduling to avoid all packet collisions. Whilst this approach can reduce congestion, this does not assist existing businesses with large LoRaWANs who do not wish to move from an official standard with reputable vendors to an unknown option. An attempt must thus be made to study congestion and seek to improve network performance whilst keeping any changes to the protocol to the minimum.

A standard solution to scalability issues is to increase the number of gateways; whilst this is very effective, it is also quite costly. A research gap thus exists to find ways to improve network performance in single gateway networks to help reduce the costs of rolling out a network.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The main objective is to improve network performance in congested mixed-traffic LoRaWANs operating with only one gateway. By studying the root cause of congestion in these networks, new solutions can be developed to improve packet delivery. Furthermore, how congestion should be managed in battery-less capacitor-based devices that obtain energy through energy harvesting will also be studied.

The following research questions were the main focus.

- What is the cause and impact of congestion in mixed-traffic LoRaWANs?
- How can a congestion management scheme be developed that monitors for congestion and reduces the impact of congestion on network performance?
- How should congestion be managed when a device's energy source is limited, such as in the case of battery-less LoRaWANs?

1.3 APPROACH

The first step that was taken was an in-depth background review of LoRaWANs with a focus on congestion and its underlying causes. An investigation followed this to find a suitable simulator to study these networks and methods to reduce congestion. A method to monitor congestion in a network and a novel algorithm (referred to as groupedPackets) to reduce congestion was then developed and evaluated. Finally, the impact of this algorithm and other methods to reduce congestion was investigated for battery-less LoRaWANs, with findings presented on reducing congestion in networks with limited energy sources.

1.4 RESEARCH GOALS

The primary research goal was to develop a novel method to improve the scalability of one gateway LoRaWANs, focusing on networks containing mixed-traffic. A secondary goal is to investigate how scalability can be improved when energy resources are also constrained, such as in the case of battery-less LoRaWANs.

1.5 RESEARCH CONTRIBUTION

The research technical contributions of the conducted work are the following. Firstly, the root causes of the decrease in network performance present in congested networks were identified and detailed. Secondly, an congestion management scheme and an algorithm were developed and described and their effectiveness was evaluated in terms of reducing congestion in a wide range of network setups. The final contribution was an in-depth investigation into how congestion can be reduced using the developed scheme and algorithm in battery-less LoRaWANs.

Other contributions include additional information on the causes of congestion which will allow network operators to manage their networks better. Furthermore, improvements and fixes were also made to the simulator, which benefits all researchers using this tool.

1.6 RESEARCH OUTPUTS

The following outputs were generated.

Journal articles.

- J.M.Marais, A.M. Abu-Mahfouz and G.P. Hancke, "A Survey on the Viability of Confirmed Traffic in a LoRaWAN", IEEE Access, 2020 ([5]).
- J.M.Marais, A.M. Abu-Mahfouz and G.P. Hancke, "Improving the sustainability of confirmed traffic in LoRaWANs through an adaptive congestion scheme", IEEE Sensors Journal, 2023 ([10]).
- J.M.Marais, A.M. Abu-Mahfouz and G.P. Hancke, "A comparison of the groupedPackets algorithm and other methods for sustainable confirmed traffic in Battery-less LoRaWAN Devices", under review at IEEE/ACM Transactions on Networking.

Conference articles

- J.M.Marais, A.M. Abu-Mahfouz and G.P. Hancke, "The Impact of Application-based Downlink Traffic in LoRaWANs", Southern Africa Telecommunication Networks and Applications Conference (SATNAC), 2021 ([11]).
- J.M.Marais, A.M. Abu-Mahfouz and G.P. Hancke, "A Review of LoRaWAN Simulators: Design Requirements and Limitations", International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 2019 ([12]).

1.7 OVERVIEW OF STUDY

Chapter 2 describes the review of relevant LoRaWAN literature. Chapter 3 discusses the research methodology and inner workings of the chosen simulator. Chapter 4 outlines the groupedPackets algorithm and its results are presented in Chapter 5. Managing congestion in battery-less networks is discussed in Chapter 6. Chapter 7 concludes by summarising what was done, giving final remarks on each research question and highlighting what can be done in future work.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

This chapter discusses the literature dealing with the presence of confirmed traffic in a LoRaWAN. In Section 2.2 some context is first provided on why confirmed traffic would be required in the first place before the LoRaWAN protocol is discussed in Section 2.3. Section 2.4 discusses the impact of downlink traffic on network performance and an overall comparison and analysis of this impact are presented in Section 2.5. Section 2.6 contains a comparison of proposed methods to reduce the resultant congestion. Open challenges and potential solutions to the congestion caused by confirmed traffic are given in Section 2.7. The chapter finishes off with Section 2.8 discussing the considerations required when dealing with congestion in energy-constraint networks. An overall summary of this chapter is presented in Section 2.9.

2.2 INTRODUCING THE INTERNET OF THINGS

The concept of the IoT stems from the belief that devices can be transformed into “smart" devices through the addition of sensing and communication [13]. Through advancements in wireless networking, computing and battery technologies, it has become much more viable to create a worldwide network of interconnected devices. These networks have resulted in several domain-specific applications that seek to improve all sectors of society.

The IoT industry is proliferating with more and more devices being deployed. These devices are frequently battery-powered and are scattered over large areas, thus requiring communication technologies designed to meet their needs [7]. This need has been met through the development of LPWAN technologies such as Sigfox, NB-IoT and LoRaWAN.

2.2.1 Use cases that require ACKs

At a glance, most IoT applications appear similar and can be considered as a large number of battery-powered devices that infrequently send small amounts of data. The assumption is that the value lies in collecting a large amount of data, so the delivery of specific packets becomes less critical than collecting sufficient data. However, when IoT use cases are compared with one other, their unique Quality of Service (QoS) requirements start to be revealed [3].

The QoS requirements for a moisture measuring device part of a smart agriculture IoT project are minimal. The device's task is simply to periodically deliver a moisture reading, of which a few can get lost without causing any problems. For an IoT smoke detector, however, a confirmation that its alert has been received is critical.

There are potentially multiple QoS metrics that must be met when IoT use cases are examined; the chosen focus is on one QoS criterion, namely reliability and how this can be achieved through acknowledgements. As pointed out in [3], focusing on one QoS metric frequently comes at the cost of other metrics. In their network simulations, it was found that maximum capacity was achieved only at very low packet success rates as there exist tradeoffs between the metrics. There are several use cases where confirmation that a data packet or a command has been received is critical. These have been classified into a few broad categories, namely Smart City, Smart Health, Industrial and Critical Situations. This is not meant to be an extensive list but rather to highlight that there are use cases in these popular IoT deployment domains that require confirmed traffic.

2.2.1.1 Smart City

An area in which LoRaWAN deployments are highly suitable is Smart City operations, as these involve a high amount of devices that transmit only a few times a day [14, 15]. A Smart City's network would include several thousands of sensors monitoring aspects of the city such as air and water quality, road conditions and traffic congestion [16]. These aspects could all use unconfirmed traffic, but others such as smart lighting and smart metering require acknowledgements for commands transmitted to the devices.

Smart metering involves the remote monitoring and control of electricity, water or gas consumption and would involve thousands of devices in urban cities [17]. Smart meters are part of the large resource networks responsible for keeping the city functional. Worldwide, the electricity sector has increasingly

been aiming to optimize and automate the generation, transmission and distribution of electricity through the creation of smart grids, with smart meters being involved in this process [18]. LoRaWAN is a strong contender to enable the installation of smart meters due to the urban ranges possible and the number of devices a single gateway can support [18]. Acknowledgements are essential in allowing providers to use their smart meters to switch off a non-paying customer's electricity.

LoRa's low power consumption makes it an attractive choice for Smart City deployments, as frequent battery changing would be difficult in several applications. The technology has also shown the potential to be combined with energy harvesters to, e.g. not only monitor traffic flow but be powered from the vibrations generated by vehicles [19].

2.2.1.2 Smart Health

Smart Health is an area in which tremendous growth is expected to occur due to technological advancements and the rise in global life expectancy [20]. This area is quite broad as it contains the detection of an illness and the long-term monitoring of health conditions/illnesses. Smart Health applications frequently have strict QoS requirements, but this is also application-specific. Real-time systems monitoring glucose levels or heart rhythms require low communication delay, whilst other systems such as fall detection are sensitive to packet loss [20, 21]. These systems demonstrate potential differences as some systems simply transmit sensor data whilst others process the data and only send updates.

One example of a system where the data is processed locally and only alerts are sent is fall detection [22]. A device is attached to a person and continually examines accelerometer data to determine when an alert to indicate a fall has occurred must be sent. Another example is the use of LoRaWAN in an IoT bio-fluid analyzer testing for urinary tract infections [23]. The analyzer sends the test result to a secure remote server, and the use of acknowledgements here can ensure that the patient's test results were captured.

LoRaWAN was evaluated as a potential technology for Smart Health in [24]. It was deemed suited in most cases due to its range, latency and network capacity capabilities. The use of Class A was deemed a good compromise between battery efficiency and receiving an acknowledgement for transmitting health data. In many cases, not all data is required to be received successfully. Only critical data such as an alert of a potential heart attack or fall would require acknowledgement.

Another aspect of Smart Health is ensuring the cold chain is kept intact for products such as insulin and vaccines [25]. These products have strict acceptable temperature ranges that must be adhered to during transportation, storage and handling. If this acceptable range is breached, it is critical that any sent alerts are received, as this item should no longer be considered safe to use.

2.2.1.3 Industrial

One of the significant areas in which the IoT is expected to play a major role is in industrial settings; hence the term Industrial Internet of Things (IIoT) was coined. The deployment of low-cost connected devices is seen to pave the way to productivity improvements, and cost savings across manufacturing and supply chains [26]. Industrial settings can have harsher requirements than other IoT deployment areas, and deployment must be economically viable [20].

The use of LoRaWAN in an industrial setting is up for debate as whilst the technology offers good scalability, range and security, it might not meet a specific use case's jitter, delay or bandwidth requirements [27]. The protocol was not designed with delay in mind, and even for small packets of 10 B sent using the fastest LoRaWAN data rate available, the transmission time required would be 20 ms. This delay would be too high for real-time operations such as closed-loop control but suitable for most other operations [14, 20]. The protocol supports different device classes and depending on the power sources available, the power-intensive but lower delay class C could be used.

The protocol's support for confirmed traffic is beneficial as devices would commonly be used to monitor equipment [28]. During normal operations, non-critical data such as temperature would be transmitted for use in fault prevention strategies. Critical faults, such as a breakdown or measured values outside of acceptable operating ranges, can be sent as confirmed traffic to ensure a manufacturing line can be halted.

Alternatives to using LoRaWAN as the MAC layer for industrial settings with real-time requirements has been developed. Examples of this are RT-LoRa ([29]) and Industrial LoRa ([30]), which aim to optimize message latency. Alternatively, works such as [31] aimed to maintain compliance with the standard protocol when a low-latency solution for factories with toxic, inflammable gasses was designed. This was achieved through coordinating transmissions through a downlink control packet to indicate which sub-bands to use for urgent packets sent using confirmed traffic.

Industrial settings vary significantly from crowded factories in cities with lots of radio interference to wide-open areas such as open-pit mining. The number of devices would also differ from thousands of devices down to potentially only one hundred. The network might also be split between a majority of devices that periodically send measurements and a few whose packets are critical to be received [20]. In such a scenario, it would be beneficial to only deploy one network servicing both needs rather than deploying two networks.

Smart Agriculture is an industry in which the IoT can play a significant role. This sector traditionally had limited monitoring and automation options, but as land and water become scarce better ways to farm are needed [26]. Animal tracking is a use case where LoRaWAN is starting to feature strongly, with commercial applications such as mOOvement's cattle tracker already available [32]. The technology is highly suited for this type of use case due to its excellent coverage capabilities in rural areas [3]. Most of the animal's movement can be sent without requiring acknowledgements; however, once an animal breach geo-fences due to, e.g. theft, confirmation of the animal's location becomes essential.

Similarly, most sensor measurements can be sent as unconfirmed messages when LoRaWAN is used to enable precision agriculture. Only the devices that enable the automation elements on a smart farm, such as irrigation pumps, need to use acknowledgements to ensure proper operation.

2.2.1.4 Critical Situations

One area in which the IoT can be of great benefit is during emergency scenarios or natural disasters. Due to its scaling and long-range capabilities, LoRaWAN has started to attract the attention of researchers building networks for these critical situations [33, 34]. IoT deployments in these situations will have to share the limited available resources whilst ensuring QoS requirements are met.

For the reaction to an event to be effective, information about the event and the environment in which it occurs is vital [35]. As the situation unfolds, data from different nodes will become more critical than others, and here the use of priority-based acknowledgements can be put to use [34].

The emergency services could also use a LoRaWAN to enable basic communication [36]. During disasters, traditional networks are frequently inactive or overwhelmed. Whilst a LoRaWAN does not have the bandwidth capabilities of these networks, it can be used to send text information and provide a reliable network.

2.2.1.5 Common requirements

All use cases mentioned previously require that the network supports downlink traffic. The Downlink (DL) traffic volume is much lower than the Uplink (UL) volume but still plays a vital part in making the network viable. Other use cases usually considered uplink only, such as collecting sensor readings, can still benefit from downlink traffic capability. The use of groupcasting and geocasting, [37], can effectively reduce traffic by specifying which nodes should transmit data whilst the rest can remain in sleep modes. In this manner, the use of downlink traffic can improve network performance, and only the high-value data will be transmitted and stored.

A common theme across all of these use cases is the need for not only acknowledgements but also that traffic should be spliced so critical applications can be assured of delivery. One method, suggested in [14], is that new channel hopping methods are used with dedicated uplink and downlink channels for certain (critical) packets.

2.3 THE LORAWAN STANDARD

Unlike some of its competitors, the LoRaWAN standard is an open standard developed by the LoRa Alliance. Due to its openness, it has proven to be very popular amongst academics, industry and the maker community. The PHY of a LoRaWAN network is a modulation technique called LoRa and was developed by Cycleo before being acquired by Semtech and remains proprietary. LoRaWANs operate in the Industrial, Scientific and Medical (ISM) bands, and their operation thus differs depending on the network's region. Note that this thesis was written from the perspective of European Union (EU) 868 networks with the regional parameters specified in [38].

A LoRaWAN-compliant radio offers several data rates, and packets have maximum packet lengths depending on the data rate used for transmission. When the data rate is set, either statically or dynamically through LoRaWAN's ADR scheme, this is mapped into specific values for two other settings, namely Spreading Factor (SF) and Bandwidth (BW). In the EU region, a LoRaWAN compliant radio has two BW settings namely 125 kHz and 250 kHz and offers six different SF values namely 7-12. The choice of SF involves a tradeoff between the achievable distance and throughput; higher SFs offer more range but also have higher time-on-air durations causing Duty Cycle (DC) restrictions to be reached earlier [17]. These DC restrictions are region-specific and must be adhered to in LoRaWAN networks.

The LoRaWAN standard adds the required MAC functionality and specifies that the network has a star-of-stars topology which supports uplink and downlink messages. The network is managed by a central Network Server (NS) which is responsible for managing packet duplicates, scheduling acknowledgements, the ADR scheme and will route any packets to the Application Server they are destined for [39, 40].

Uplink messages are sent by a device to the NS and will be received by one or more gateways which all forward the packet to the NS [41]. Downlink messages are sent from the NS to a specific device and will be transmitted by only one gateway chosen by the NS. LoRaWAN gateways are based on Semtech's SX1301 chip, capable of demodulating 8 LoRaWAN frames simultaneously [42]. Historically, LoRaWAN gateways are half-duplex; thus, a gateway cannot receive any transmissions whilst transmitting downlink messages. Semtech is however continuously working on the technology and has recently released a reference design that allows for full-duplex operations [43].

Since the release of version 1.0.0 in January 2015, the LoRaWAN specification has continually been improved. The latest minor update, revision 1.0.4, was released in October 2020. A significant update, 1.1.0, was released in October 2017 and is backwards compatible with 1.0.x end devices and networks. The updates have, amongst other things, added new MAC commands and deprecated old ones, increased security and added frequency plans.

The protocol specifies three possible device classes, all allowing bi-directional communication and are referred to as classes A, B or C. With class A each uplink from a device to the gateway is followed by two short downlink receive windows as shown in Figure 2.1. The device can only receive communication from the gateway during these two windows.

This optimizes power consumption but causes devices to be unreachable until they send a message. Class B adds additional scheduled receive windows to make devices more reachable and downlink communication frequency more predictable. This is done through the gateway transmitting a time-synchronized beacon, and as devices must remain synchronized, it increases a device's power consumption. The final class, C, is best used by devices without power concerns as these devices have near-continuous receive windows. This allows downlink communication to class C devices to have lower latency than the other classes but has no impact on uplink latency.

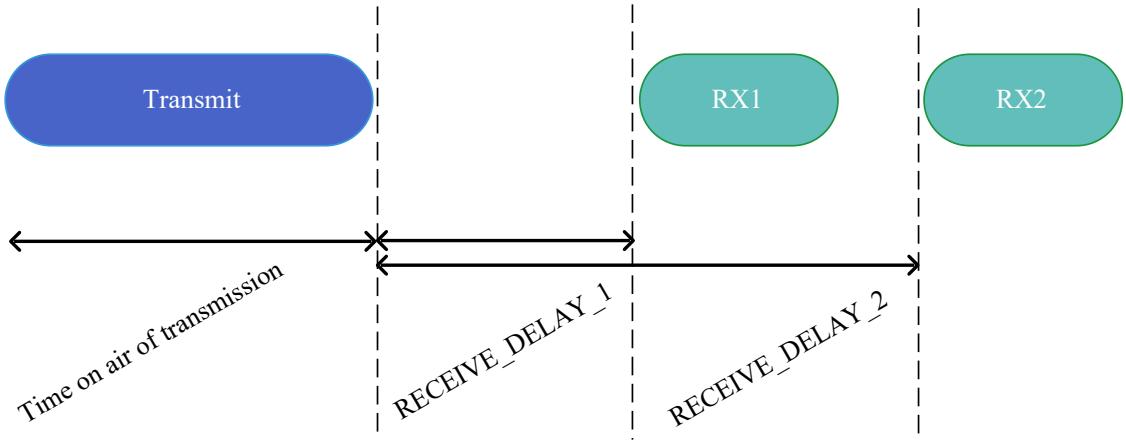


Figure 2.1. Timing diagram for receive windows for Class A devices. Taken from [5], © 2020 IEEE.

The most commonly deployed class in a LoRaWAN is envisioned to be class A. In terms of receiving acknowledgements, this class has the strictest requirements, as class A devices will only be able to receive an ACK during their two downlink receive windows. Whenever an ACK is requested from the device, as required as a response to several of the MAC commands, the device should transmit an ACK as soon as possible [40, 44]. The device may immediately transmit a potentially empty packet with the ACK bit set or simply set the ACK bit of its next message.

The protocol specifies three default channels in the EU 868.0-868.6 MHz h1.5 sub-band that all devices on the EU 868 channel plan must support. For these channels, the maximum Transmit Power (TP) is +14 dBi Equivalent Isotropically Radiated Power (EIRP) and a sub-band DC limitation of < 1 % must be adhered to. By default, a device's second receive window is configured to operate in sub-band h1.7 and transmits on 869.525 MHz using the maximum spreading factor (SF12), since this sub-band allows a DC of 10 % [45, 46]. Depending on the region, more channels can be added; for example, the popular The Things Network (TTN) uses ten channels in its EU 863-870 channel plan [47].

Figure 2.1 also shows the timing used in opening the receive windows after a device's uplink transmission. The first window opens after the uplink's modulation finishes (exact time specified by RECEIVE_DELAY1) and will, by default, use the same channel and data rate as the uplink. The second window opens RECEIVE_DELAY2 seconds after the uplink, and its frequency and data rate are configurable, and region-specific [40, 41]. Transmitting an ACK using the first window could

result in a lower transmission time as transmitting an ACK using any SF other than SF12 would be faster. However, the first window may not be available as it will have a tighter DC restriction than the second one. The default timing values for these delays and the delay between retransmission attempts when an ACK was not received are given in Table 2.1.

Table 2.1. Default parameter settings in EU 868 Networks. Adapted from [5], © 2020 IEEE. RETRANSMIT_TIMEOUT used to be referred to as ACK_TIMEOUT in versions prior to 1.04.

Parameter	**Default setting**
RECEIVE_DELAY1	1 second.
RECEIVE_DELAY2	RECEIVE_DELAY1 + 1 second.
RETRANSMIT_TIMEOUT	random delay between 1 and 3 seconds.

2.3.0.1 LoRaWAN packet structure

LoRaWAN packets consist of a Preamble used to synchronize the receiver, an optional header and its Cyclic Redundancy Check (CRC), the payload, and then an optional payload CRC [40]. The CRC is always excluded in downlink packets to keep their size to a minimum. LoRaWAN packets also have two format options, explicit and implicit, of which explicit is the default [48]. The explicit mode adds the header mentioned above, which specifies the payload length (in bytes), the coding rate and if a CRC is provided for the payload. When the implicit mode is used, this information is excluded, thereby reducing transmission time and manual configuration is required on either side.

The LoRaWAN protocol supports unconfirmed uplink and downlink transmissions and a confirmed version of both [41]. Whilst confirmed traffic is supported, it is assumed that in most deployments nearly all traffic is sent as unconfirmed messages [49]. The MType bit field specifies the message type in the MAC header part of the packet and the protocol allows a single packet to contain application data and MAC commands [41].

The MAC payload of a data message contains a frame header (FHDR), and acknowledgements are managed by this header's frame control octet (FCtrl) part [41]. It is through this octet that a device can request an ACK or an ADR acknowledgement request, a crucial part of the ADR scheme. As the ADR scheme aims to optimize a device's data rate and TP, it could potentially dictate settings that result in a loss of network connectivity. As most data would be sent as unconfirmed messages, the device needs a method of confirming that the gateway is receiving the sent data. Should the ADR scheme be used,

the protocol requires the end device to track how many messages it has sent since it last received a downlink transmission. Once this exceeds a threshold, the ADRACKReq bit is set in the following message [41]. Upon detecting this bit, the gateway must transmit a downlink message, confirming that the device is still connected. If no such confirmation is received, the device adjusts its data rate and TP until it is reconnected.

2.3.0.2 Procedure when an ACK is not received

There are several causes for failure to receive an ACK: the confirmed message was not received by the other party, DC limitations prevented the transmission of the ACK, and finally, the ACK was transmitted but not received [7]. Another potential cause could be the delay between the NS receiving the request, selecting a gateway from which to acknowledge and the chosen gateway transmitting the ACK [50].

The retransmission procedure when an ACK is not received is one part of the LoRaWAN protocol where versions can differ. LoRaWAN versions can be split between 1.0.x, where x equals 0, 1, 2, 3 and 4 and the new versions 1.1.x, for which x is currently 0. In versions 1.0.4 and 1.1 [41, 40], it is specified that a device must wait for RETRANSMIT_TIMEOUT seconds after RECEIVE_DELAY2 seconds before sending a new uplink. Timing diagrams showing examples are also provided in Chapter 16 of the protocol ([41]). Additionally, a retransmission back-off strategy is provided in Chapter 7 for retransmissions triggered by external events causing synchronisation between devices ([41]).

This retransmission back-off strategy spaces retransmissions RETRANSMIT_TIMEOUT (a random value between one and three) seconds after the second receive window, and opts for a different channel than was used for the uplink [41]. An earlier version of the protocol (1.0.3) also suggested lowering the data rate of retransmission attempts, but this was omitted in subsequent versions. It is left to the application and NS developers to decide what should happen if transmission remains unsuccessful: either continue attempting to send the same data or simply discard it and send the next message.

Versions 1.1 and 1.04, unlike versions prior to 1.0.4, differentiate between the response to either downlink and uplink frames and specify that the NbTrans field's value now also applies to confirmed uplink frames and no longer to only unconfirmed frames [41].

The NbTrans value improves redundancy by specifying the number of transmissions for each uplink

message. For a confirmed uplink, this specifies how many times retransmission may occur. The default value is one, with a maximum of fifteen [41]. With the default value of one, an ACK will be requested, but no retransmission will occur if the ACK is not received. This field forms part of the LinkADRReq command, which is also responsible for the modifications requested by the ADR scheme [41].

For downlink retransmissions, a new frame counter value must be used; thus, the application server must be notified of the failure [41]. This server must decide if a new confirmed downlink frame must be sent to the device. Version 1.1 also specifies that a device will only acknowledge a downlink confirmed frame once. Uplink frames are to be retransmitted NbTrans times, and the device must apply frequency hopping between the repeated transmissions [41]. The delay between attempts is left to the end device's developer, and any retransmissions should stop once a valid ACK is received.

Retransmissions are not limited to confirmed traffic and can also occur due to specific MAC commands [41]. One such command is the join-request command which the device will retransmit if a response in the form of a join-accept message is not received. The LoRaWAN protocol specifies a retransmission back-off mechanism to prevent network overload during events such as power outages that cause many devices to simultaneously require an acknowledgement or a response to their uplink frames [41]. The mechanism requires that the retransmit interval between frames be random and differ between devices and also adds additional duty-cycle limitations.

2.4 IMPACT OF DOWNLINK TRAFFIC ON NETWORK PERFORMANCE

Downlink traffic in a LoRaWAN can transmit an ACK for a confirmed uplink message, send MAC commands, application layer downlink data, or a combination of all of these [51]. Accurate network models are key when determining the impact of downlink traffic on network performance, and modelling should include the PHY and MAC layers. Due to their Aloha-like MAC configuration and the DC limits imposed by operating in ISM bands, LoRaWANs suffer from scalability issues [52]. They do, however, perform better than pure Aloha networks, in part due to the capture effect [53, 54], and SF pseudo-orthogonality [55, 17].

Network models evolve, and earlier models become outdated as new information on the capture effect and SF orthogonality come to light. Earlier work such as [56], modelled LoRa with the assumption that simultaneous transmissions sent on the same frequency but at different SFs are orthogonal and can thus be simultaneously decoded by the gateway. Research has, however, shown that the orthogonality

is not perfect and should instead be considered pseudo-orthogonal. As a result, models should instead consider the interference caused by transmissions using the same SF (co-SF) and different SFs (inter-SF) [17, 42]. Another aspect that models should include is the capture effect. This effect describes the possibility that a packet can successfully be received even if it collides with other packets if its power is sufficiently high [57]. This is problematic for packets from remote devices, as any collisions with packets sent from nearby devices will result in the far away device's packet not being received.

Even when these factors are considered, authors interpret the protocol differently, as seen in the case of [54]. In their theoretical model, gateways always transmit two acknowledgements (one for each receive window), but this behaviour is not specified in the protocol [58]. This approach to acknowledgements was investigated in [39], which found that transmitting in both windows hurts performance.

In [39], the point is made that all three of the main approaches to evaluating LoRaWAN performance have limitations: experimental work is limited by a large number of variations in cost, size and explorable setup parameters. Simulation-based approaches are limited by computational power, assumptions made, and achievable level of detail. Mathematical models use ideal assumptions which impact the accuracy, but can be used to focus the efforts of the other options [39].

LoRaWAN simulators have been developed for various existing Wireless Sensor Network (WSN) platforms such as ns-3, OMNET++ and SimPy. Publicly available LoRaWAN simulators that support downlink traffic are LoRaSim [59], the expanded version of LoRaSim used in [52], the ns-3 modules presented in [49] and [58], LoRaEnergySim [60] and FLoRa [61].

LoRaWAN is very popular among European researchers, and as a result, most simulators are built assuming operation in the EU863-870 MHz band. One exception is [62], which simulates 915 MHz North American networks but does not support downlink traffic. However, this simulator is open-source and thus can be modified to investigate ACKs. Simulators also need to remain updated with changes to the LoRaWAN protocol and relevant research findings such as pseudo-orthogonality and the impact of the Doppler effect on LoRa transmissions as described in [63].

The current LoRaWAN literature focuses almost exclusively on ACKs during downlink performance evaluations. Notable exceptions to this are the work presented in [64] and [49]. The simulations performed in [64] showed that adding downlink data traffic increases the competition experienced by ACK

transmissions. As the DC restrictions are tight, this further increases the number of retransmissions performed by nodes. When downlink data traffic was investigated in [49], a small decrease in the Packet Delivery Ratio (PDR) for uplink transmissions was noted due to more packets being missed. This is due to the increased time a gateway now spends in transmission mode. Their analysis showed that adding additional gateways can help reduce this negative impact.

The current LoRaWAN literature is also dominated by works examining networks containing only class A nodes. This is logical as class A is expected to be the most common, but does leave the performance of the other classes and networks containing a mix of classes unexplored. One exception is the work on class B networks in [65]. They found that the data rate and the number of channels significantly impact the delay of downlink traffic.

2.4.1 The impact of confirmed traffic

The impact of confirmed traffic arises from different aspects, and thus the works examining confirmed traffic were broadly split into several categories. Some works examined more than one aspect and were thus referred to multiple times.

2.4.1.1 The impact of NbTrans

To determine why confirmed traffic impacts network performance so severely, the authors of [66] build a MATLAB-based simulator. They found that as the ratio between confirmed and unconfirmed messages increases, the throughput for both types of traffic decreases.

Increasing the NbTrans value (number of transmissions per uplink) was effective in increasing the packet delivery for confirmed traffic, but came at the expense of unconfirmed traffic's delivery success. Additionally, higher numbers are not necessarily the best as the maximum number of 15 performed the worst as this causes too many collisions. Simulations determined that a value of nine performed best for confirmed traffic. These high numbers may not be usable in public networks as resources must be shared fairly, and the cost to unconfirmed traffic might not be acceptable.

When NbTrans was examined in [64], it was found that this number should depend on network size. Simulations using LoRaWANSim found that only a small improvement is seen in small networks. For a 5000 node network, switching from transmitting once to using five attempts improved the number of successfully acknowledged frames from 40 % to $\approx$ 75 %. However, this increase comes at a cost, and

energy consumption rose by nearly 10 % due to the extra energy required when allowing for up to 5 retransmission attempts.

The evaluation presented in [58], also concluded that a dynamic mechanism is needed and that it should be based on the traffic load. Their ns-3 simulator, which added downlink support to their original work ([67]), was first compared with the analytical model presented in [54]. Unlike the model, the simulator accounts for SF pseudo-orthogonality and the DC restrictions imposed on transmissions.

For small networks (500 devices), increasing the number of transmission attempts was effective in improving the PDR as the DC restrictions are not yet exceeded. Increasing the number of transmission attempts from 2 to 8 allowed a PDR of 95 % to be maintained as the ratio of confirmed traffic increased from 10 % to 70 %. In large networks (2000 devices), increasing this number no longer allowed a significantly larger percentage of traffic to be confirmed as it also increased the probability of collisions. In large networks, it was found that receiver saturation starts to become the leading cause of packet loss and thus increasing the number of transmissions is no longer beneficial.

A mathematical model was presented in [39] and used to study the performance of single gateway networks with 100 % confirmed traffic. Their validation with the ns-3 simulations ([67, 58]) showed that their model is slightly optimistic when predicting the PDR but remains valid. Their further comparison with [54] highlighted that DC restrictions are a key parameter for models to include as without it the achievable performance will seem much higher.

An updated version of the model was presented in [68], which now also considers aspects such as the intricacies of the receive windows and the number of demodulators available at a gateway. The updated version is more accurate in predicting PDR and found that more devices need multiple retransmissions as traffic load increases to receive an ACK correctly. At high enough traffic levels, using all attempts no longer guarantees that an ACK will be received.

One of the assumptions made by [39] and [68] is that the probability of receiving an ACK sent in the second receive window is one, the reasoning for this is that the frequency channel chosen for window two transmissions is only used for downlink and thus no collisions can occur. This is true for networks in remote areas where the current network is the only traffic on channel 869.525 MHz. Other LoRaWANs will likely be present in urban areas, and this assumption can no longer be made.

The simulator developed in [49] was also used in the LoRaWAN evaluation presented in [69]. Their evaluation focused on the impact of the number of transmissions per packet and the number of available channels. Only two options for the maximum number of transmission attempts are considered (one or eight) and only two for the number of channels available (one or seven). The comparison evaluated 100 % unconfirmed against 100 % confirmed traffic. In a network with seven available channels and under maximum load, unconfirmed traffic's PDR slowly reduces from 100 % to 80 % as the network scales from 1 to 150 devices.

In contrast, confirmed traffic's PDR has an aggressive exponential decay and reaches 40 % at 20 devices. This decline is very aggressive, however, it should be noted that this decay did not start from 100 % for a single device network; the results of their simulations showed that only a PDR of 80 % was achievable. Increasing the number of retransmissions enables a high PDR of nearly 100 % for unconfirmed traffic, but the authors claimed it had no impact on the performance of confirmed traffic in their scenario. The lack of impact was likely caused by the GW having already reached its DC limits before multiple retransmissions were allowed.

When a reduced load is considered through a traffic intensity of 10 %, the network's performance differs from above. In small one-channel networks (less than 40 nodes) confirmed traffic which allowed up to 8 transmissions performed best. Over this 40 device threshold, however, unconfirmed traffic with only one transmission remains the best performer. As only one channel is available, increasing the number of retransmissions for unconfirmed traffic drastically reduces the PDR due to the increased collisions it causes. In seven-channel networks, unconfirmed traffic maintained a near 100 % PDR for both maximum transmission options with a less than 5 % difference between them. Confirmed traffic with up to eight transmissions outperformed only one transmission significantly and maintained a near 100 % PDR up to 60 devices, after which it declines sharply. The presented results show cases in which confirmed traffic outperforms unconfirmed traffic. However, only when the number of channels is limited, the number of devices is low and in networks with moderate loads. Increasing the number of retransmissions can increase the PDR for confirmed traffic but does not guarantee performance similar to that of unconfirmed traffic.

As part of investigating the impact of several parameters, the authors of [70] examined NbTrans. They found that increasing this parameter improves performance for confirmed traffic, but the increases have diminishing returns. Adjusting this parameter hurts unconfirmed traffic's performance, and

thus choosing the lowest possible value that meets confirmed traffic's performance requirement is best.

2.4.1.2 Confirmed traffic ratios

A MATLAB simulator, presented in [71], was used to compare scenarios in which nodes require no ACKs, all nodes require an ACK, 25 % require an ACK and 33 % of nodes require an ACK. This simulator only utilized the first receive window when calculating the viability of downlink responses, and networks consisted of either 1000 or 2000 end devices. The simulator considered the Packet Error Ratio (PER) per node and an average QoS value calculated as the network's total PDR. For the 1000 node network, ACKs did not significantly decrease the average QoS, but requiring an ACK for each uplink increased the percentage of nodes with a PER of between 10 % and 50 % from 1.2 % to 23.9 %. In a 2000 node network, the decrease in QoS and negative impact on packet delivery was much more pronounced. The percentage of nodes with a PER between 10 % and 50 % increased from 6.9 % for no ACKs to 24.5 % when 25 % of traffic require an ACK and when all traffic required an ACK it increased to 70 %.

To investigate bidirectional traffic, the authors of [64] created LoRaWANSim, a closed-source LoRaWAN simulator based on LoRaSim. Several downlink configurations were tested, namely 5 % downlink application layer data (no ACKs), 5 % of uplink traffic that require ACKs and finally, the combination of the previous two options. As with most evaluations, some simplifications were made. The first was that each device was set to use the maximum data rate allowing it to reach the gateway (similar to the ADR scheme but not dynamic).

Their investigation found that the leading cause of retransmissions was the DC restrictions imposed on gateways [64]. Even when only 5 % of transmissions required an ACK and there was an additional 5 % downlink data, the DC restrictions caused the success of downlink transmissions to drop below 80 % once a network reaches 1000 devices. Any downlink data transmissions aggravate the problem as they cause the gateway to reach its limitations earlier, increasing the number of ACKs it can not transmit.

The network's goodput (application layer throughput) also drops significantly as the percentage of traffic requiring an ACK increases [64]. When 100 % of the traffic required acknowledgement, the goodput drops to 15 % of the level achievable by a network with no acknowledgements. The impact is

minimal for low volumes of requests ($\leq$ 5 % require ACKs) but increases to nearly 20 % when 20 % of the traffic requires confirmation. Beyond 20 % of traffic requiring ACKs, the goodput is increasingly negatively impacted.

The impact of traffic ratios was also examined in [58]. In a network where the number of devices is fixed, the PDR was quickly reduced when the percentage of devices using confirmed traffic was increased. In the inverse case, where the percentage of devices remains fixed, increasing the number of devices did not significantly decrease the PDR.

To better understand why the PDR is so negatively affected, a scenario in which all devices send confirmed traffic, up to 8 transmissions are allowed, and each device sends a packet every 30 minutes was examined closely. It was found that channel impairments had a negligible impact on the PDR and that there is a threshold where interference no longer dominates, and the unavailability of receiving paths becomes the leading cause. It is noted that adding more gateways will help alleviate this problem, but this must be done whilst also optimizing the SF allocations within the network. Uniformly spreading gateways will allow more nodes to use the higher data rates, decreasing transmission times and collision probabilities.

A simulator was fed with recorded real traffic traces collected from up to 4 Semtech PicoCell gateways in [51]. When the ratio is below 20 %, the packet loss remains below 10 % and is mainly caused by uplink transmissions being missed due to the half-duplex nature of the gateways. Above this point, DC restrictions become the leading cause of packet loss. In the case of 100 % confirmed traffic, the network loses 86 % of packets, with 76 % of this as a result of their ACKs being unable to be transmitted.

LoRaSim also served as the basis for an investigation into confirmed traffic conducted in [72]. The authors investigated several confirmed ratios and found that as the ratio increases, more uplink packets are lost due to the half-duplex nature of gateways. The simulations show that GW transmissions account for the majority of lost packets compared to those lost due to collisions.

2.4.1.3 The impact of SFs

The MATLAB simulator presented in [66] also examined the impact of SFs. Their evaluation of the different SFs found that performance is impacted negatively, due to the additional collisions caused by the increase in transmission time from one SF to the next. Their simulator has flaws as

it assumes perfect SF orthogonality, does not randomize the time interval between retransmissions (ACK_TIMEOUT), and handles ACK transmissions on the second windows differently. In their setup, an ACK is always transmitted with the same SF used by their corresponding uplink regardless of the used window. However, the second window typically uses a fixed value, and the suggested is SF12.

The work presented in [58] was followed up with the development of a mathematical model for single gateway networks in [68] and used to study LoRaWAN in [70]. This model was compared with the simulator described in [58, 67] and was found to match favourably. The impact of using SF12 versus matching the uplink packet's SF was investigated, and it was found that more packets could be successfully acknowledged by not restricting the gateway. This effect depends, however, on the traffic volumes. When SF12 is used to transmit ACKs, the gateway would quickly reach its DC restrictions, keeping it in receive mode. When other SFs are used, the gateway spends more time in transmit mode. Whilst it is thus able to transmit more ACKs, it also missed more uplinks due to the half-duplex nature of the gateway [70].

The mathematical model presented in [39] also examined the impact of SFs. Their model assumes perfect SF orthogonality and is for a single gateway scenario. Restricting acknowledgements sent using window two to SF12 was considered sub-optimum, as allowing these ACKs to be sent with the same SF as the corresponding uplink improved network performance as this eases meeting the DC restrictions on the gateway [39]. During the presentation of the updated model in [68], it was found that a network should not necessarily use all available SFs, as high SFs occupy a gateway's receivers for a long time, increasing the number of packets dropped due to no more receivers being available.

A further problem highlighted in [64] and [58] is the congestion caused by the retransmission mechanism of version 1.0.3 and earlier networks, detailed in Section 2.3.0.2. The strategy continually increases the SF used by successive retransmission attempts as it assumes no ACK was received because of distance and does not consider congestion as a cause. Protocol versions 1.1 and 1.0.4 removes this auto-reduction of the used data rate and also allows the number of retransmission to be configurable.

The ns-3 simulator presented in [58] also examined the ADR scheme and found that it tended to assign

the same SF to several users and does not consider the resultant loss of SF orthogonality between these nodes. The ACK_TIMEOUT parameter was also evaluated with the simulations performed showing no benefit to this random delay of between one and three seconds. Upon investigation, it was found that collisions occurred amongst nodes using the same SF and that the back-off time is too short for the higher SFs. During retransmissions, the packets would simply collide again as a maximum length SF12 packets take 3.15 s to transmit, which is more than the ACK_TIMEOUT delay.

In [3], several IoT use cases were grouped into one of three groups based on traffic considerations. Out of each group, one use case was selected, and network throughput and packet success rate were analyzed via simulation for confirmed versus unconfirmed SF7 and SF12. In terms of network throughput, unconfirmed outperforms confirmed traffic, but the authors note that this result should not be viewed in isolation, instead the PDR should also be examined. The authors examined two scenarios: a small area with a single gateway and a large area covered using multiple gateways. In the first scenario, SF7 confirmed mode outperformed SF7 unconfirmed mode up to a point, whilst in SF12's case; unconfirmed mode remained consistently better than confirmed. As throughput and congestion increase, confirmed traffic would start to result in more collisions, causing confirmed mode to no longer outperform unconfirmed mode.

2.4.1.4 Receive windows

Gateways are required to respond in either of the two receive windows and have to consider the DC restrictions and potential interference problems of each window. The first window uses, by default, the same channel as the original uplink. The use of this window will cause interference with uplink messages from nodes in the network [73].

Most works examining confirmed traffic use mathematical models or simulation tools; however, an empirical approach was followed in [74]. A gateway and several nodes were placed in an anechoic chamber with the duty cycle restrictions for the nodes selectively disabled. This allowed a single device to emulate 88 to 700 nodes (depending on the SF used). Their experiments found that downlink traffic caused a significant increase in uplink traffic's PER. This is due to the uplink-downlink interference when receive window one is used. Additionally, it was found that the gateway tended to respond using receive window two for high SF traffic and window one for low SF traffic. This was attributed to the duty cycle limitations that a gateway must adhere to. The time-on-air of high SF ACKs will cause a gateway to exhaust window one quickly, and will thus force the use of window two, which has a more

relaxed DC requirement.

The impacts of raising the DC restriction used in window one to match that of window two were explored [68]. Significant performance gains can be obtained thanks to more ACKs being sent at lower SFs. Unfortunately, raising the DC restriction is not allowed as this would violate operation in the ISM bands.

The second window is on a dedicated downlink-only channel, which would be used only by gateways. A network will contain significantly fewer gateways than nodes, making this window a better choice from an interference perspective. However, the DC restrictions applied to each window's channels and the SF used for the response must also be considered. The use of the second channel tends to occur with higher SFs, the default is SF12, resulting in longer on-air time for transmissions [74]. This will also increase the interference one gateway's transmissions have on another; thus, the lowest possible SF should be used [39].

The impact of swapping the sub-bands used by each window was investigated in [70].In a mixed-traffic network swapping the sub-bands showed a marginal improvement in the confirmed packet success ratio, implying that the uplink-downlink interference normally experienced in window one had minimal impact.

A potential solution to eliminate uplink-downlink interference is to disable the use of receive window one and rely only on window two. This idea is explored in [73], which is discussed in more detail in Section 2.6.

2.4.1.5 Multi-gateway networks

In a multi-gateway LoRaWAN, the NS is responsible for choosing which gateway must transmit a downlink frame to a node. A gateway selection algorithm is thus required, and two approaches are explored in [51]. The first is a simple Signal to Noise Ratio (SNR) based approach currently used by the open-source LoRaWAN NS ChirpStack (previously called LoRaServer) [75]. The recorded SNR values of a node's previous packets are used to select the best gateway with which it is best to respond. If one gateway is preferred over others for many nodes, it could quickly become overworked by the NS. As the algorithm only uses the best gateway and does not consider DC restrictions, other gateways

with slightly lower SNR values end up being underused [51]. The second approach corrects this flaw by considering gateways' DC restrictions.

The addition of more gateways was found to result in a significant improvement, even when the SNR-based approach was used. In a quad versus single gateway scenario, the additional gateways halved the packet loss experienced in the network. The leading cause was that downlink traffic was now split amongst multiple gateways, each with its own DC restrictions, allowing more ACKs to be successfully sent.

The results revealed that this can be improved upon, as with the SNR-based approach the first gateway was assigned to transmit 60 % of the ACKs. This gateway was only able to send 24 % of these requests due to DC restrictions, whilst two of the four gateways receive nearly no requests. The second approach, which attempts to balance the load between the gateways, reduced the packet loss by 25 % due to its better load distribution.

An open-source LoRaWAN ns-3 module developed for a scalability analysis is presented in [49]. This analysis investigated confirmed traffic in a single uplink channel network with a fixed NbTrans of four with either one, two or four gateways. In a one gateway network, confirmed traffic under-performed unconfirmed traffic, with the under-performance increasing as the number of devices or the traffic volume increases. The decrease in network PDR is attributed to the number of receive windows missed by the gateway due to DC restrictions. Increasing the number of gateways did improve the PDR, but had a more significant impact on unconfirmed traffic than confirmed traffic.

Adding gateways is very effective for unconfirmed traffic as it decreases the distance between nodes and their closest gateway, allowing them to use faster data rates and thus decreasing congestion. For confirmed traffic, traffic volumes also play a role as adding gateways does allow more ACKs to be sent, but it also allows more packets to be successfully received that now require an ACK. This increases the number of required ACKs, causing DC restrictions to remain a problem. In networks with low traffic volume, additional gateways are very effective, as the volume is low enough that the only bottleneck is DC restrictions.

Multi-gateway networks were studied in the context of industrial scenarios in [76]. Increasing the number of gateways from one to three resulted in significant improvements in CPSR at higher arrival

rates, with the returns diminishing as more than three gateways were added. A combination of multiple gateways and increasing retransmissions was a viable strategy to obtain reliability levels above 99 %. The authors note that simply retransmitting a packet several times could be better than transmitting the data as a confirmed packet, as the latter is limited by the ability of a GW to respond with an ACK in one of the two receive windows.

2.5 DISCUSSION

The first focus of the discussion is comparing the work discussed in the four categories dealing with the impact of confirmed traffic in LoRaWANs. The second focus is a more general analysis of the important aspects identified.

2.5.1 Comparison

Table 2.2 compares the papers discussed earlier. A common theme between evaluations is to only consider the uplink and downlink frequency channels considered mandatory by the LoRaWAN protocol. These are the three channels used for both downlink and uplink and an additional channel for downlink only. The maximum is 16 channels, [38], and Section 7.2.3 of the ETSI EN300.220 standard ([45]) dictates several sub-bands that a LoRaWAN may use. Adding channels in a different sub-band will assist with scalability as DC restrictions are calculated per sub-band. An analysis using, e.g. the 10 channels used currently by the TTN or the 12 channels proposed in [73] would provide a result closer to the current performance in deployed large networks.

The table shows several instances in which either 0 % or 100 % confirmed traffic was investigated. Section 2.2.1 highlighted several scenarios in which confirmed traffic was not required by all nodes. Instead, only a small minority of devices in a network will require the use of acknowledgements. Furthermore, acknowledgements are frequently linked to an event, such as cattle breaching a geo-fence. The feasibility of using acknowledgements in a LoRaWAN should thus be seen in the contents of a ratio between confirmed and unconfirmed traffic, with the confirmed traffic potentially occurring in bursts. How well a network scales is thus dependent on this ratio between confirmed and unconfirmed traffic as well as the number of devices, packet size and the transmission frequencies of confirmed and unconfirmed traffic.

Nearly all evaluations did consider the DC restrictions imposed on end devices and gateways, and those that purposefully simulated the impact of NbTrans examined multiple values for this parameter.

Table 2.2. Comparison between works examining confirmed traffic. Adapted from [5], © 2020 IEEE.

Reference	Uplink channels	% confirmed traffic present	DC restrictions	NbTrans
[3]	3	0,100	-	8
[39]	3	100	Yes	1
[51]	-	0-100	Yes	1
[58]	3	0-100	Yes	1, 2, 4, 8
[49]	1	1, 10	Yes	4
[64]	3	5	Yes	1-8
[66]	3	5, 10, 30	Yes	1, 3, 9, 15
[69]	1,7	0, 100	Yes	1, 8
[70]	3	0, 50, 100	Yes	1,4 8
[71]	-	0, 25, 33, 100	No	1
[68]	3	0-100	Yes	1, 2,4, 8
[74]	8	0, 100	Yes	1
[72]	3	0, 20, 60, 100, 100	Yes	1

2.5.2 Analysis

The currently published work examines downlink viability by almost exclusively focusing on the impact of confirmed traffic, i.e. sending ACKs to end devices. However, downlink traffic consists of more than just ACK transmissions, and there are MAC command transmissions and application layer data also to consider. Even if confirmed traffic is not allowed by a network operator, downlink traffic would still occur [51]. Improvements in those areas will reduce the total downlink traffic, easing DC restrictions and thus also improving the performance of confirmed traffic.

A potential improvement for downlink traffic is offered in [58, 70], who notes that when end devices open their receive windows, there is the potential that they accidentally lock onto the uplink transmission of another node instead of the gateway's response. This is caused by the identical preamble of downlink and uplink transmissions, so the receiver is not immediately aware that it is locking onto an uplink transmission.

As the issue of viability only comes into play in large networks, analysis of confirmed traffic has

predominately been done using LoRaWAN simulators. The currently available simulators have differences in how the LoRaWAN protocol is implemented. As an example, [66] shows in their Table 1 that ACK_TIMEOUT has a value of 1 s; the LoRaWAN protocol specifies that this should be a random delay between 1 and 3 seconds [38]. In general, simulators are used to investigate a specific topic; thus, assumptions/decisions are made on other areas deemed irrelevant. Direct comparisons between results are also complicated as different payload lengths, sending intervals, number of nodes, channels, and so forth are used.

The LoRaWAN protocol could be optimized by modifying the current receive window behaviour. The published work has shown that gateways struggle with the current implementation. Currently, the second window is opened in a downlink only 10 % DC restriction channel. If this channel is instead used for the first receive window, more devices would be able to receive an ACK as not only are DC restrictions reduced in this channel, but there would be no collisions with uplink traffic [58, 74]. Matching the downlink's SF with the original SF of the uplink packet would further improve how many responses can be sent before the 10 % DC restriction is reached.

Whilst the issues raised above must be considered, the existing literature can be summarised to capture what is currently known about the state of confirmed traffic in LoRaWANs and what parameters influence its viability.

Researchers found that sending two ACKs instead of one decreases network performance, receive window two's use of SF12 is not ideal, and the impact of prioritizing reception over transmission at gateways depends on NbTrans and traffic volume [39, 68, 70]. Adjusting the NbTrans parameter can be effective in increasing the PDR, but this improvement is limited to small networks [58, 64]. Consensus on the exact threshold after which it becomes harmful and the optimum number of transmissions was not achieved amongst researchers. In large networks, the leading cause of the performance reduction was the DC restrictions imposed on gateways [64]. This is problematic, as nodes are creating a retransmission snowball effect for successfully received data. Adding additional gateways can be an effective solution to easing the DC restrictions on gateways, but load balancing between the gateways must be considered [51]. The RETRANSMIT_TIMEOUT random back-off interval is too short for packets sent using high SFs, they will simply collide again due to their long transmission times [58, 77]. It is good that versions 1.1 and 1.0.4 of the protocol no longer increase the SF between retransmission attempts, as this only made congestion worst in 1.0.3 networks [58, 77].

2.6 NEW PROPOSED ACK METHODS

An alternative approach to data collection and acknowledgement is suggested and compared with standard LoRaWAN in [52]. A scheduling scheme called FREE is proposed, which requires that nodes store data in bulk and collection takes place periodically. The scheme also specifies the SF, TP, and channel to be used. This approach allows devices to send an order of magnitude fewer packets simply by bulking data and sending longer packets. This system is thus not designed for priority data as collection could potentially only occur once every few hours. The authors note that collecting every 12 hours or when devices have gathered roughly 2.5 KB of data is a good trade-off point between fast data collection and device lifespan. With FREE, data reception is acknowledged by the gateway using a compressed group ACK scheme.

During their evaluations, nodes generated 20 bytes every five minutes, could potentially retransmit a packet up to eight times and simulations considered one day's worth of data. For a standard LoRaWAN using confirmed traffic, the device lifetime drops quickly below two years due to the cost of retransmissions. For a network of 1000 devices, the number of transmissions increased 7.4 times the level of a similar network using unconfirmed instead [52]. LoRaWAN's overall Data Delivery Ratio (DRR) plummets sharply, falling below 50 % at $\approx$ 200 devices as the number of collisions rises sharply. The researchers note that collisions are not the primary cause of the drop in DRR. Instead, the gateway's DC restrictions prevent it from responding with ACKs. The loss of either the data packet or the ACK due to channel fading also had an impact, but this was much smaller. Their proposed solution outperformed LoRaWAN significantly as the precise scheduling prevented collisions, and the DC restrictions were kept in mind.

A method to perform acknowledgement aggregation is proposed in [78]. In this method, the NS periodically sends out an "AggACK" containing ACKs for several devices and received packets. The proposed method was evaluated in a modified version of the ns-3 simulator presented in [67]. The results show that this method improved the normalized throughput and found that transmitting an aggregated ACK every 20 seconds was the optimum waiting time for their evaluated scenario. It is unclear which SF the aggregated ACK is sent with or how nodes know on what channel it will be transmitted. The presented throughput model assumes the number of nodes is large, 1000 nodes, but the ns-3 only examines up to 200 nodes.

An alternative radio-resource management solution is proposed in [73]. By assigning 869.525 MHz

as the designed channel for both downlink windows, collisions can be prevented between uplink and downlink traffic. An operational mode called time-power multiplexing is presented, which relies on new hardware designs allowing a gateway to transmit more than one packet at a time. Additionally, new power amplifier subsystems are required to allow for variable TP based on the number of simultaneous transmissions [73].

The authors suggest a final modification, namely that only one receive window is used to allow a gateway to maximize the number of downlinks it can group. A comparison, using a MATLAB simulator, between the proposed system and the default protocol showed that higher network capacity is now supported. The improvements not only improved the capacity for confirmed traffic but also slightly improved unconfirmed traffic's capacity. Whilst the scheme offered improvements, simply reducing the percentage of confirmed traffic would still have a bigger impact on the scalability of unconfirmed traffic.

In addition to a transmission scheduling algorithm, the use of group acknowledgements is proposed in [79]. The scheduling algorithm dictates the SF, channel and timeslot for all transmissions, thereby allowing a group acknowledgement to be sent for all packets received after each timeslot. The format of this acknowledgement is not described, and the proposed solution was compared with ALOHA using simulations. The results showed that the solution can improve packet delivery but is dependent on the number of channels.

An extension to the LoRaWAN protocol named A2S2-LoRaWAN is proposed in [80]. This extension uses a time-slotted periodic frame structure combined with aggregated ACKs to improve the scalability of LoRaWANs. The proposed extension is compared with standard LoRaWAN in simulations, although the gateway is limited to only using one bidirectional channel with a DC restriction of 1 %. This does not quite match the regular operation of an EU-based LoRaWAN gateway, which would also have access to the downlink only 10 % DC restricted channel used by receive window two.

The use of time slots is also proposed in [81], combining it with a network architecture called edge acknowledging (EACK). In this architecture, uplink packets are acknowledged by a nearby gateway rather than waiting for the NS to first process the uplink and request an ACK. This architecture was combined with a new MAC protocol called contention-constrained p-CSMA (CCP). This is a modified version of persistence-based Carrier Sense Multiple Access (CMSA) as several back-off periods are

added before transmissions are allowed, and nodes wait a random number of these periods before proceeding with the conventional p-CSMA.

A modified version of LoRaWAN using p-CMSA was compared with the proposed combination of EACK and CCP. The proposed method offered higher packet reliability at larger loads than the p-CMSA based approach, and retransmissions further increased packet reliability. The proposed method's energy consumption was also evaluated, and it was found that it offers lower consumption than p-CMSA thanks to a reduction in the amount of carrier sensing performed. In terms of packet delay, EACK and CCP resulted in delays between 20 and 30 times experienced with p-CMSA due to the time slotting required to offer higher packet reliability. The authors note that a much lower packet delay would be possible in very small networks (10 to 50 devices).

A method to improve the redundancy of confirmed transmissions is proposed in [34]. A replication scheme (LoRa-REP) using different chirp durations is combined with a repeater to offer a fully LoRaWAN-compliant solution for emergency transmissions. Emergency data is replicated several times and sent using different SFs, allowing multiple chances of reception and receiving an ACK. A range extender device serves as a repeater and retransmits uplinks sent using SF7 and will forward an ACK should the GW respond to the repeater's transmission and not the original one.

The proposed method is envisioned for industrial locations such as factories, where alarm events generating confirmed traffic are sporadic but time-sensitive. Furthermore, the factory must be fairly small as it is envisioned that using SF7 is viable. The method was evaluated in a heavy industry indoor area with one end-node producing traffic and another node serving as the range extender. Just adding the range extender was able to increase the reliability for packets sent using SF7 from 84 % to 99.5%. When a mobile version of the end-node was evaluated, which moved along a 700 m path, enabling LoRa-REP increasing reliability from 62.5 % to 72.5 % when SF9 was used which was increased to 85.5 % when the range extender was added.

Most LoRaWAN research focuses on stationary nodes, but [82] investigated how reliability can be improved when devices are mobile such as in the case of pet-tracking applications. In these environments, channel conditions change as devices move around, causing the transmission parameters set by the standard ADR scheme to become unfeasible. To solve this, Semtech proposed blind ADR (BADR), which was evaluated alongside several other ADR schemes and a proposed alternative called

R-ARM. The authors showed that their method significantly improved packet reliability in mobile 100 % confirmed networks versus other approaches. The improvement is attributed to their algorithm selecting better SFs and TPs due to considering the number of retransmissions left and GW sensitivity. The algorithm also offers lower energy consumption and converges faster than other approaches.

2.6.1 Comparison

Table 2.3 compares the alternative methods to implement ACKs discussed earlier. Only one of the methods was released as open-source, allowing other researchers to examine the proposed method further. Only one of the methods was implemented on currently available LoRaWAN hardware, with [73] unable to do so until a gateway capable of time-power multiplexing is developed.

Table 2.3. Comparison between proposed ACK methods. Adapted from [5], © 2020 IEEE.

Reference	**Simulation platform**	**Open-source**	**Energy studied?**	**Implemented**
[52] (FREE)	based of Simpy and LoRaSim	Yes	Yes	Future work
[73] (time-power multiplexing)	MATLAB	No	Yes	Requires new hardware
[78] (AggACK)	ns-3 ([67])	No	Future work	No
[79] (GACK)	-	No	No	No
[80] (A2S2-LoRaWAN)	Python	No	Future work	No
[34] (LoRa-REP + repeater)	-	No	No	Yes (small scale)
[81] (EACK + CCP)	Matlab	No	Yes	No
[82] (R-ARM)	ns-3 ([67])	No	Yes	No

One of LoRaWAN's strengths is its low power consumption, as IoT deployments are frequently in remote and inaccessible areas. The impact of the proposed method on energy consumption is a crucial aspect not all works examined.

2.7 OPEN CHALLENGES & POTENTIAL SOLUTIONS

Several challenges must be overcome to improve the viability of confirmed traffic in LoRaWANs. The main challenges found in the literature have been identified and discussed in the section. Aspects of the protocol that are currently statically configured may have to be modified to rather optimally respond to network conditions.

2.7.1 Indicating network congestion and reply urgency

Gateways currently have no method of indicating network congestion to nodes. End devices can delay transmitting an ACK until their following data message, but the gateway does not have a similar option. Nodes are unaware of network congestion so are not instructed to perhaps open a third transmission window, after a much longer interval, or attempt to send less confirmed traffic. This indication of network congestion can also be used by nodes to adjust their transmission frequency or aggregate their data.

Currently, the LoRaWAN protocol has no method of indicating the urgency of receiving a response for confirmed traffic. The combination of a class A device's capability of only receiving an ACK during two short downlink windows and a gateway's DC restrictions is a significant bottleneck as the ACK has to be sent immediately. Should the protocol be adapted to allow end devices to indicate the urgency of an expected acknowledgement, this would ease a gateway's DC burden. This indication does not necessarily have to be binary. A value between zero and one can allow gateways to selectively decide which nodes will be answered using their first window versus those acknowledged in their second window.

2.7.2 Grouping traffic and ACK aggregation

Should the confirmation that data is received be necessary, but latency is not a concern, grouping several measurements and transmitting them as one large payload will have several benefits. Doing so not only has a power consumption benefit [52, 83] but also reduces the number of ACKs sent when compared to sending the data as several smaller packets. The simulations conducted in [52], showed how their scheme's approach of sending fewer but larger packets helped reduce the total number of sent packets significantly (an order of magnitude) which helped reduce the number of collisions.

Acknowledging several devices using one packet can be a method to reduce the number of transmissions required by gateways. However, this would increase the time-on-air of the transmissions and potentially increase the number of packets a gateway did not receive as it is now transmitting for longer periods. This approach would differ from the existing design as the protocol currently aims to minimize the time-on-air of downlink transmissions through steps such as excluding a CRC for the PHYPayload. Should a packet containing a grouped ACK become corrupted, it would no longer impact only one node but all of the nodes expecting an ACK in this message. How often this would occur and the impact of the resulting retransmissions, as well as the impact of the increasing time-on-air, needs to

be examined further. This is a growing area of LoRaWAN research, with some suggested methods already proposed.

An adaptation of the LoRaWAN protocol to enable broadcasts via a cooperative downlink listening algorithm was proposed in [84]. This method can also be used to broadcasts ACKs to several devices at once, thereby reducing the gateway's transmission times. This approach also introduces different methods of reducing traffic volume in a network via groupcasting (requesting data from only specific devices) and geocasting (request that only devices in a particular area transmit). Experiments showed that this method does lead to slightly higher energy consumption in networks where devices do not often transmit, e.g. one packet per hour. However, the consumption soon matches LoRaWAN when the transmission frequency increases.

2.7.3 Competitive space

The use of Acknowledgements is one way to improve packet delivery in a network, but a big aspect that influences deployments is that an urban area might have multiple LoRaWAN networks competing with one another. Unlike Sigfox, where an operator provides the network, anyone can deploy a LoRaWAN network. Creating one network, similar to the Sigfox approach, is, however, viable with a project such as The Things Network [85] currently boasting approximately 8 000 gateways forming a global network.

Several networks in one place will frequently receive packets from each other's nodes. If these networks work together, they can reduce the overall load on gateways but this must be done securely. Received packets could be shared with the destined network through services such as the recently announced Packet Broker [86] but sharing resources remains a challenge.

2.7.4 Other challenges

A challenge when determining the viability of confirmed traffic is how accurately a specific use case's requirements were captured. More information is required than estimating how many nodes, a target PDR and expected packet length. Another challenge is that as the protocol is still reasonably new, LoRaWAN simulators are not very advanced. They are also not necessarily regularly updated once new research comes to light. Using these tools to examine a complicated project involving a ratio between confirmed and unconfirmed traffic, multiple gateways, and the radio environment of a planned deployment site are currently not very feasible.

LoRaWAN gateways have limited scalability as only eight simultaneous signals can be received [57]. This limitation is not always considered by researchers; for example, [66] states their gateways have infinite receive paths. The use of ACKs in large networks causes further scalability issues as LoRaWAN gateways are half-duplex and hence miss uplink messages when transmitting ACKs [51]. In small networks, the traffic volume is low, and thus the probability that devices transmit at the exact time interval that an ACK is sent is relatively small. This is not the case for large networks, and due to the eight parallel demodulation paths of the gateway, multiple transmissions could be lost for each ACK sent. The use of multiple gateways can help alleviate this problem, as the messages can be received by another gateway, assuming it is in receive mode. The coverage areas of these gateways should be carefully considered to ensure adequate overlapping zones to assist with sharing the load of downlink traffic. The potential performance of full-duplex gateways was investigated in [70], which found them to be very effective in improving uplink traffic's PDR.

LoRaWAN nodes support multiple channels, although most evaluations stuck to the default channel allocations. The protocol already supports informing nodes of additional channels supported by the gateway. As suggested in [14], some of these additional channels could be dedicated as retransmission-only traffic channels for critical packets.

The maximum number of transmission attempts has been shown to improve the PDR for confirmed and unconfirmed traffic. This increase is seen in small networks and disappears in larger networks. It is, however, dependent on the traffic volume of a network and not simply on the number of devices. Evaluations examining this configuration parameter typically statically examined this, i.e. fixing the traffic volume and the number of transmission attempts. As suggested in [58, 64], a dynamic system is required to adjust the maximum number of transmission attempts as the network's traffic volume changes.

There exists an asymmetry in the receiver sensitivity between gateways and end nodes, and this could be problematic [58, 70]. A situation can occur where an end device can receive a gateway using a certain SF, but the gateway would be unable to successfully transmit an ACK using the same SF, as the end device's receiver is not sensitive enough. The gateway is then forced to transmit the ACK using the second receiver window, which would use the maximum spreading factor. This problem can be alleviated by either informing an end device that its first window cannot be used to receive an ACK or informing it to expect a reply using a different SF than it initially used [58].

A challenge raised by [87] is that most research assumes that uplink traffic's distribution is uniform and tends to be periodic. An examination of a deployed network revealed significant and periodic fluctuations in the number of transmissions with higher packet loss experienced in specific periods. Through a MATLAB-based model, it was shown that if a large number of devices join a network at the same time, they will be admitted periodically. Their transmissions will thus not be evenly spread out over time but bunched up with the number of transmissions varying over time.

2.8 THE USE OF LORAWAN IN ENERGY-CONSTRAINT NETWORKS

An important consideration when deploying an IoT network is the energy consumption of the chosen wireless communication solution. These networks are frequently spread over wide geographical locations or in hostile environments, which would make regular battery replacement undesirable [88]. The desire for frequent data gathering or timely reporting of incidents often needs to be balanced with the realization that energy resources are finite and energy-efficient operation is required. One way to reduce the reliance on battery replacement or the use of batteries is to turn to energy harvesting methods which can be combined with options such as super-capacitors to eliminate their use.

The energy consumption of a LoRaWAN end-device can be captured by a basic model which considers all of the operational states that a device cycles through, its duration in each and the energy consumption of each state. IoT devices typically have a sleep mode in which they remain for a long duration before proceeding with a cycle of taking a measurement, transmitting that measurement, performing data reception and reverting to the sleep mode [89]. LoRaWAN transmissions have several parameters, and in particular, the chosen SF has a significant impact on energy consumption due to its increase in the Time on Air (ToA) of a packet. Packets are also sent with a specified TP, and modelling has shown that energy could be saved by sending a packet using a lower SF at a higher TP than using a lower transmission power and a higher SF [89].

In general, confirmed traffic causes the same performance decreases in energy-constraint networks as in other networks. Similar solutions to improve performance can thus be considered, but the energy impact of a solution must now also be studied.

Several energy efficiency protocols were surveyed in [88] with authors proposing alternatives to the standard ADR scheme's approach to optimizing devices' SF, BW and TP. Approached varied from optimizing resource allocation, devices states or proposing hardware/software improvements such

as lowering the sampling rate used for packet reception. Other options to improve energy efficiency are eliminating collisions (which result in retransmissions) by opting for MAC protocols using time slotting or carrier sensing. Changing a network's topology from a star-of-stars to multi-hop has also been proposed, enabling transmissions to require less power [88]. It was noted that retransmissions and using ACKs would increase energy consumption while severely degrading network performance in large networks.

When the impact of ACK on energy consumption was studied in [90] the increase in power consumption was attributed to the collisions caused by ACKs. Their work found that NbTrans is a critical design parameter to maintain reliability when networks contain confirmed traffic. Simulations revealed that higher NbTrans values would be required in networks with higher arrival rates if high reliability is required. At lower arrival rates, fewer retransmissions are required, with the presented analytical framework able to calculate the average amount of retransmissions that would be required.

A battery-free design for structural health monitoring was presented in [91] with a 22 mF super-capacitor chosen for energy storage. By disabling the downlink capabilities of a class A LoRaWAN node, energy use was minimized but required that only unconfirmed traffic be sent. Another structural health monitoring application was presented in [19], with the vibration energy caused by traffic on a bridge stored in a 100 mF super-capacitor. A LoRa radio (MAC disabled) was used for communication. However, energy harvesting depended on the number of vehicles using the bridge and the estimated time to charge the super-capacitor entirely was 3.5 hours.

A major contributor to the study of LoRaWAN usage in battery-less devices is the work of [92]. A Markov model was developed for UL and DL transmissions, and simulations found that battery-less communication is feasible. They noted that the second receive window highly affects performance and recommend that only ACKs are sent as downlinks. Furthermore, the turn-on voltage should be configured to the lowest value that still allows a full transmission-reception cycle [92]. This model for battery-less communication was implemented in [93] and extended the popular ns-3 *lorawan* module's features.

Simulations confirmed that the implemented model matches the mathematical one and that packet delivery strongly depends on data rates. Whilst lower data rates enable packets to still be successfully received in the presence of channel impairments; they have considerable energy costs. This can be

mitigated by using larger capacitors, which take longer to charge. Simulations also showed that sending an ACK in receive window one to prevent the opening of window two can effectively reduce a node's energy consumption due to the high cost of using SF12 in window two [93].

The benefits of making a node energy-aware were explored in [94], where several transmission scheduling approaches were evaluated. It was found that a conservative sender (assuming no energy harvesting during the transmission cycle) outperformed all other approaches as it ensured the node remained operational. Harvested power was obtained via a solar panel with an average value of 7.2 mW but exhibiting high variability (8.2 mW) due to reduced sunlight in the afternoon. Predicting the harvesting voltage was not accurate, and this caused the other schedules to turn off the node frequently.

2.9 CHAPTER SUMMARY

The findings in literature dealing with the presence of confirmed traffic in a LoRaWAN were summarised. Context on why confirmed traffic would be required was provided before a discussion on the LoRaWAN protocol. This was followed by an investigation into the impact of downlink traffic on network performance and a review of the proposed methods to reduce the resultant congestion. Finally, a discussion on the additional considerations required when dealing with congestion in energy-constraint networks was given.

Section 2.2.1 showed that by supporting downlink traffic, a more diverse selection of IoT use cases could be supported by the LoRaWAN protocol. This includes cases where confirmation is required that critical information was received by the other party and this confirmation is done through ACKs. The details of how the LoRaWAN protocol supports ACKs were presented in Section 2.3. ACKs from the GW to a node can only be received during two receive windows that are opened at specific time intervals. Furthermore, GWs transmitting ACKs must comply with DC restrictions in the relevant sub-bands and their half-duplex design causes them to be unable to receive any uplink frames whilst transmitting an ACK.

The impact of downlink traffic was studied in Section 2.4 and can be summarised as a minimal impact in small networks or ones with low packet arrival rates and a severe impact in congested networks. The negative impact of packet reliability can be reduced by enabling retransmissions, with some consideration required to limit their use. Network performance is susceptible to the ratio of confirmed

to unconfirmed traffic and will be negatively impacted even if only 5 % of traffic is sent as confirmed packets. The SF chosen by devices to transmit a confirmed packet also has an impact on performance as higher SFs increase the ToA of the resultant ACK by the GW. By default, receive window two uses SF12 for ACK reception, which has a long transmission time. This results in many uplink packets being missed when a GW is forced to use this SF. Increasing the number of gateways is an effective but costly method to improve performance as this allows for more opportunities to transmit an ACK (reducing retransmissions). Furthermore, multi-gateway networks allow a different GW to still receive uplink transmissions when another GW is transmitting ACKs.

Methods to reduce the impact of confirmed traffic were discussed in Section 2.6. These include methods focusing on ACK aggregation to reduce the quantity sent or changing the sub-bands used for their transmission to eliminate interference between downlink and uplink transmissions. Other approaches include using time slots to eliminate packet collisions or the repeated transmission of confirmed packets using different SFs to increase the probability of successful reception by a GW. A comparison of these methods revealed that few are open-source and that the energy impact of a proposed method is not always considered.

Open challenges to the viability of confirmed traffic were documented in Section 2.7. Examples are that there is no method to indicate congestion to nodes or that a GW is close to DC restrictions, so devices simply retransmit already received packets. Using multiple gateways is an effective solution, but urban deployment could consist of several private networks deployed within reach of one another. Methods to share packets and resources must be considered in these areas instead of simply deploying more gateways.

Finally, the use of LoRaWAN in energy-constraint networks was the focus of Section 2.8. It was found that choosing the proper transmission parameters is crucial when LoRaWAN is to be used in devices where frequent battery replacements are infeasible. Several works studied the use of LoRaWAN with battery-less devices, which offered promising results. The use of confirmed traffic in these devices requires additional design considerations, due to the increase in energy consumption caused by the reception of ACKs and potential retransmissions.

CHAPTER 3 METHODS

3.1 CHAPTER OVERVIEW

This chapter details the chosen research methodology and the reasoning behind it. A discussion on the overall approach used to study congestion in LoRaWANs is given in Section 3.2 and is followed in Section 3.3 with information on the chosen research instruments. Section 3.4 contains several methodology considerations which include the outputs of the chosen simulator, the chosen performance metrics, how congestion was generated and how valid results were gathered. Finally, the limitations of the chosen methodology are discussed in Section 3.5 and an overall summary of the chapter is provided in Section 3.6.

3.2 RESEARCH DESIGN

The literature review presented in Chapter 2 revealed that congestion is best studied in large networks. One way to achieve this would be to deploy an outdoor LoRaWAN testbed, but this comes with some challenges as deploying many nodes would be quite costly and require constant care. The number of nodes required could be reduced by disabling the DC restrictions of nodes, but this approach would not be friendly to other LoRaWANs in the area. The idea of a physical network was thus abandoned, and the use of simulators was adopted instead. The choice of a simulator is essential, and several factors must be considered.

Accurately modelling of the PHY layer is crucial in the development of any simulator. Whilst the channel access mechanism is pure ALOHA, research has shown that LoRa outperforms a pure ALOHA approach, and thus a simulator should not assume pure ALOHA performance [62, 95]. Published lab experiments showed that when concurrently transmitted packets collide with similar or weaker RSSI values, the interfered packet will still be received as long as the last six symbols of the preamble and the interferer's header did not collide [55].

Simulators should also consider that LoRa is susceptible to self-interference [49, 96, 17] and exhibits the capture effect [49, 55, 97]. Self-interference can either be co-SF interference (the same SF) or inter-SF interference (between different SFs). LoRa transmissions are also subject to the capture effect: a stronger signal can suppress a weaker signal at the receiver [53], causing the stronger signal to be received successfully [64].

As LoRa is proprietary, the development of simulators was initially slow as experimental work must first be done to understand its design and performance fully. Thus, simulators must be updated as new research refining our understanding of the PHY layer is released. Whilst the PHY layer is essential, a simulator's implementation of the LoRAWAN protocol must also be considered. Features such as frequency plans, retransmissions and acknowledgements are required to study congestion. Another requirement is the ability to monitor the energy usage of nodes, with simulators able to cater for situations where the nodes turn off due to insufficient energy.

Another consideration is how actively the research community is using a simulator. A vibrant community is critical to ensure errors are found and new features are added over time. As final considerations, a convenient method to examine what happens in extensive simulations would be beneficial, as are detailed output values from which to calculate performance metrics.

3.3 RESEARCH INSTRUMENTS

3.3.1 LoRaWAN simulators

A literature survey was conducted with the previously mentioned requirements to find suitable LoRaWAN simulators. Table 3.1 compares the different LoRaWAN simulators that are available and open-source. Where mentioned, a simulator's name was used for the description column. The table reveals that three simulators meet the listed requirements and are worthy of further study.

After investigations, the "lorawan" module developed for ns-3 by the SIGNET lab of the University of Padova was chosen. This simulator can be found at `https://github.com/signetlabdei/lorawan` and is a discrete-event network simulator with simulations consisting of a series of events, each with a corresponding time stamp. This approach allows for faster simulations as the simulator can execute events one after the other even if they are far away in time, as no state changes would have occurred between these two events [100]. The simulator is capable of determining the outcome of each packet based on the conditions that were present during its transmission and stores the outcomes of

Table 3.1. Comparison between open-source simulators. Adapted from [12], © 2019 IEEE.

Ref.	**Description**	**Imperfect SF**	**Capture effect**	**DL traffic**	**DC limitations**	**Tracks energy**
[53, 59]	LoRaSim	No	Yes	No	No	Yes
[52]	FREE	Yes	Yes	Yes	Yes	Yes
[62]	Java based, 915 MHz	No	Yes	No	Implements North America requirements	No
[98]	From drakkar-lig	No	Yes	No	Yes	Yes
[49]	From imec-idlab	Yes	Yes	Yes	Yes	No
[67, 58]	**lorawan**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
[60]	LoRaEnergySim	No	Presumably, is based on [53]	Yes	Yes	Yes
[61]	FLoRa	No	Yes	Yes	Yes	Yes
[99]	From network-systems	Yes	Yes	Yes	Yes	Yes

each transmission. The “lorawan" module does not assume perfect orthogonality between different SFs and accounts for this, and the capture effect [68] through the link-level model developed in [101]. Additionally, it supports all required features and has the most active community of the three.

The version of “lorawan" used was a combination of the develop branch, specifically commit 159cc5 of 4 May 2021, and additional corrections. Two corrections have been made to this version, namely ensuring that the NbTrans implementation is compliant with the protocol (documented in `https://github.com/signetlabdei/lorawan/pull/114`) and that the closing of the receive windows occurs as specified in the protocol (documented in `https://github.com/signetlabdei/lorawan/issues/124`).

As a part of exploring this simulator, an attempt at replicating the published results from the authors was made. After some communication via e-mails with the authors, Figure 2 of [102] was successfully replicated. This replication was made possible through the sharing of a more advanced private version of the LoraPacketTracker class, which gathered more detailed statistics than the publicly available version. This private version was permanently adopted, as it allowed for more significant insights into the behaviour of confirmed traffic.

An updated version of the “lorawan" module was published in 2021, which supported simulating battery-less capacitor-based devices that obtain energy through energy harvesting. This version can be found at `https://github.com/signetlabdei/capacitor-ns3`, and was used for investigating congestion on these types of devices. For battery-less simulations, commit 581167c of the branch tmp-master was used and the NbTrans implementation was made compliant (receive windows were already fixed).

3.3.2 Other software used

In addition to the modules mentioned above, version control and graphing software were used. Version control consisted of using GitHub and GitLab, with repositories managed through the GitKraken application. Simulations were run and managed through the sem Python package [103] with graphs generated using the seaborn graphing package [104] in JupyterLab notebooks.

Simulations conducted with “lorawan" are defined and configured in a single C++ script which specifies the simulation scenario, configures all of the devices, performs the simulation and saves any outputs to files. Configuration options can be specified in the file or entered as command-line arguments and parsed to set local and global variables based on the arguments. This parsing ability is how the sem Package can perform parameter sweeps of variables by executing a simulation for each value. Keeping track of all simulations is done by sem by creating a simulation campaign, which contains a database of all conducted simulations with a script, as well as the values used for all parameters. Thanks to free credits through the Google research credit program, simulations were conducted locally or on the Google Cloud Platform.

GitLab was used to keep a copy of ns-3 at version 3.30 to ensure the same version is used for all simulations. A forked version of “lorawan" was kept on GitHub to enable pull requests to be sent to developers, whilst private repositories were used for expanded versions used for publications.

3.4 CONSIDERATIONS WHEN USING “LORAWAN"

3.4.1 Simulation output

The verbosity of simulation output can easily be controlled thanks to ns-3’s logging system. This system allows for logging to be enabled on a per-component basis and allows for selectable verbosity levels. By enabling all levels of output for specific components, as shown in Listing 3.1, a detailed understanding of a network’s behaviour can be gained. The operation of each device in the network can be monitored, and the logging indicates a timestamp, device, the relevant component and executing function of each logging output.

```
LogComponentEnable ("LoraPacketTracker", LOG_LEVEL_ALL);
LogComponentEnable ("ClassAEndDeviceLorawanMac", LOG_LEVEL_ALL);
```

Listing 3.1. Enabling logging.

Through the logging system, statements such as the one shown in Listing 3.2 make it easy to provide additional information during script execution. Enabling logging on several components can quickly overwhelm the terminal output, so the output of a simulation is usually redirected into a file for examination.

```

NS_LOG_INFO ("PHY packet " << packet
                     << " was successfully received at gateway "
                     << gwId);


//Example of output
+484.868052375s 24 LoraPacketTracker:PacketReceptionCallback(): PHY packet 0
    x55fade4ebbe0 was successfully received at gateway 24
```

Listing 3.2. Example of logging from LoraPacketTracker.

Whilst the logging system is handy when validating a network’s behaviour, it was disabled when simulations were run for graphing purposes. Instead, only summaries of the outcomes of events captured by LoraPacketTracker were saved to text files. These summaries require a time interval, so a starting and finishing time must be given. This flexibility allows one to, for example, exclude all events of the first five minutes in a simulation or to simulate for a longer period than is to be included in the results.

All conducted simulations were simulated for longer than the period used to gather results. The reason for this is that a packet sent in the last period of interest might only conclude several periods later. This was frequently the case for simulations conducted with high NbTrans values, as these packets are allowed to be retransmitted several times. If a packet still has retransmissions left, it is recorded as sent but does not yet have all of its corresponding outcomes recorded yet. The packet is still considered *in progress* and would be missing from the gathered results. To ensure that this does not happen, simulations were run for a few more application periods and the number of *in progress* packets were counted in LoraPacketTracker and included in the simulation output so that they can be confirmed to be zero.

It is customary for simulations conducted with “lorawan" to only have one output at the end of a simulation: a summary of the outcomes of events for a specified time interval. The development of the groupedPackets algorithm did, however, introduce the need to better understand what is happening throughout a simulation. Two files were thus generated per simulation, one containing a summary of the entire period of interest and another with several lines, each containing a summary for a specific period. These periods match a simulation’s application period (the period that specifies how often nodes generate a packet). For example, if a simulation’s duration was set to ten minutes and nodes were configured to transmit one packet per minute, the file would contain ten lines.

These summaries were extensive and included items such as the number of generated application packets (confirmed and unconfirmed), the number of these successfully transmitted, and the PHY outcomes of these (received/interfered/no available receivers/lost due to gateway TX). From these, performance metrics were calculated, and any other items of interest were plotted.

3.4.2 Used Metrics in the study

A similar approach to the ones used by the authors of “lorawan" was taken to define packet outcomes and performance metrics. For unconfirmed traffic, a packet is considered successful if it is received by a GW who forwarded it to the NS. The case of confirmed traffic is more complicated and can be split into two cases. In the first case, transmission is only considered successful when both the UL and the corresponding DL (ACK) were successfully received. A second, more relaxed case, is also possible where success only requires that at least one of the generated UL packets was successfully delivered to the NS. Due to the focus on confirmed traffic, the second case for confirmed traffic is not considered.

Based on the breakdown above, the two performance metrics of interest are:

- Confirmed Packet Success Ratio (CPSR): the probability that a confirmed UL packet was received by a GW and the corresponding ACK was received by the node.
- Uplink Packet Delivery Ratio (ULPDR): the probability that a UL unconfirmed packet was correctly received.

Equations 3.1 and 3.2 indicate how these metrics were calculated. The term unique in the equations refer to the fact that a node will generate an application packet, and depending on the type and NbTrans settings, it will be transmitted several times. For unconfirmed packets, a packet may have been received eight times by the gateway, but will count as only one packet, as the other transmissions were merely repeats containing the same data. For confirmed packets, a packet may have been retransmitted several times until an ACK was received. The LoraPacketTracker component was responsible for determining the outcome of each generated packet in a simulation with the final calculation of the metrics performed in the JupyterLab notebooks.

$$ULPDR = \frac{successfully\ received\ unique\ unconfirmed\ data\ packets}{total\ unique\ unconfirmed\ data\ packets\ sent} * 100 \tag{3.1}$$

$$CPSR = \frac{successfully\ ACKed\ unique\ confirmed\ data\ packets}{total\ unique\ confirmed\ data\ packets\ sent} * 100 \tag{3.2}$$

3.4.3 Creation of congestion

It is typical for simulations conducted with “lorawan" to be periodical in nature. The simulator was designed for scenarios in which data transmissions are periodic and occur per device; this setup is typical in IoT sensing applications such as a farm covered by numerous soil moisture sensors. The simulator thus requires the number of devices involved, whether each device is sending confirmed or unconfirmed traffic and a desired sending interval. This sending interval is specified in seconds, and each device will continue to generate a packet every $< x >$ seconds until the simulation is stopped. This interval is referred to as the application period, which refers to how frequently the “application" running on the node will wake up, sample a value of interest and transmit the results. As all of this is simulated, a function is periodically called, generating a random number of bytes of the desired length. Whether a generated packet can be sent is dependent on the device’s energy level (if energy-aware simulations are being run) and if a frequency band is available (DC restrictions as per the frequency plan are enforced).

To create congestion, a sufficient traffic volume must be present in the network. One way this can be created is by specifying the desired number of packets being generated per second, also known as the packet arrival rate (λ). This rate can be calculated as shown in Equation 3.3:

$$\lambda = \frac{Number\ of\ devices}{Application\ period}\ (packets/second) \tag{3.3}$$

For example, a network with 1200 devices, each configured to transmit a packet every 591 seconds, will result in an arrival rate of 2.031 pkt/s. A low packet arrival rate will reflect a network where there are very few nodes or many nodes that infrequently transmit with a high arrival rate indicating the opposites. The congestion in a network can thus be changed by keeping the number of devices the same and varying the application periods. Similar to the example above, a network with 1200 devices configured to transmit only every 120 000 seconds will result in an arrival rate of only 0.01 pkt/s.

The arrival rate calculated above can be considered to be the theoretical arrival rate, as the actual arrival rate will depend on factors such as any DC restrictions preventing a node from transmitting, if sufficient energy is available for transmission and if multiple copies of packets are being sent (NbTrans thus has an impact). This actual arrival rate is only known after a simulation is completed. The theoretical arrival rate, in the form of the number of devices and their application periods, was thus used as an input when performing simulations.

With the arrival rate now being defined, it is worthwhile to study how it leads to congestion. When one considers a network consisting of 1200 devices, each transmitting a single packet every 591 seconds and one gateway, the following occurs. Firstly, each device has a starting delay from a uniform random variable over [0, 591). As all 1200 devices must transmit a packet within 591 seconds, this delay allows these transmissions to be spread out whilst some overlap will still occur. Each device will then transmit its generated packet at the specified delay value and repeat this procedure 591 s later. For example, if node 10's initial delay value was 11 s, it will transmit its first packet at 11 s and its second packet at $11\ s + 591\ s = 601\ s$. The transmission at 11 s might overlap with another node's transmission at 9.85 s if the duration was long enough. For example, a 12 B packet sent using SF12 with a BW of 125 kHz will take 1482.8 ms (EU868 frequency plan). This overlap might result in the packets being lost due to interference. The probability of overlaps reduces as the application period increases, and the period increases also reduces the probability that a gateway will be unable to transmit an ACK due to DC restrictions.

3.4.4 Ensuring valid results

Simulations conducted with “lorawan" are highly configurable, and some considerations were thus required to ensure that reputable and repeatable results were generated. When conducting simulations, an important parameter is that of simulation length. Short simulations will result in the output not capturing the actual behaviour of a network. Excessively long simulations could either hide a significant but infrequent event or be an unnecessary waste of time and computing resources. Simulation length is influenced by the number of dynamic events that occur during a simulation; such an event will influence the behaviour of the following events. For example, a gateway not receiving a confirmed packet will result in retransmission by the node.

Due to the focus on congestion, many dynamic events, such as packet collisions, can be expected. Furthermore, the simulation of confirmed traffic also introduces more dynamic events caused by the transmission of ACKs and the potential retransmission of confirmed packets. Additionally, the value chosen for NbTrans also creates more events as this specifies how many repeats of unconfirmed packets must be sent and also the number of retransmissions allowed for confirmed traffic. Finally, the groupedPackets algorithm also influences nodes’ behaviour and does so repeatedly. Outside of this, though, the simulation setup was such to keep other aspects as static as possible. Nodes do not move from their randomised starting positions, spreading factors are assigned at the start of a simulation and are not changed, and the number of nodes stays the same. Depending on the simulations, packet lengths are either fixed or varied within known sizes.

Another parameter to consider is the number of times simulations are repeated, referred to as the number of runs, with the final results being the average over these simulations. By running multiple simulations, the results are more generalised and do not reflect the performance of a specific network with a specific chain of events. Each random variable in a simulation should have a corresponding independent stream of values from a Pseudo Random Number Generator to ensure no correlation. A system to ensure repeated versions of the same simulation use distinct streams also exists. Thankfully, ns-3 and the sem Python package make this process easy, and only a suitable value for the number of runs must be found.

To determine suitable values for simulation length and the number of runs, exploratory simulations were performed on the Google Cloud infrastructure. The desire to calculate accurate ULPDR and CPSR metrics must be balanced by realising that computing resources are finite. Figure 3.1 shows the

impact of simulation length, the number of runs and a simulation's NbTrans value on the average time to run a simulation. This figure reflects the time to simulate a single network, and parameter sweeps will contain several, potentially hundreds of simulations. Simulation length was calculated based on a multiple of a simulation's application period. Thirty simulation periods in a 591 s application period simulation thus refer to a simulated time of $591\ s * 30 = 4.925\ hours$. During this period, each node had thus generated 30 packets for which the relevant performance metric was calculated. As can be expected, simulating more periods increases the time per simulation.

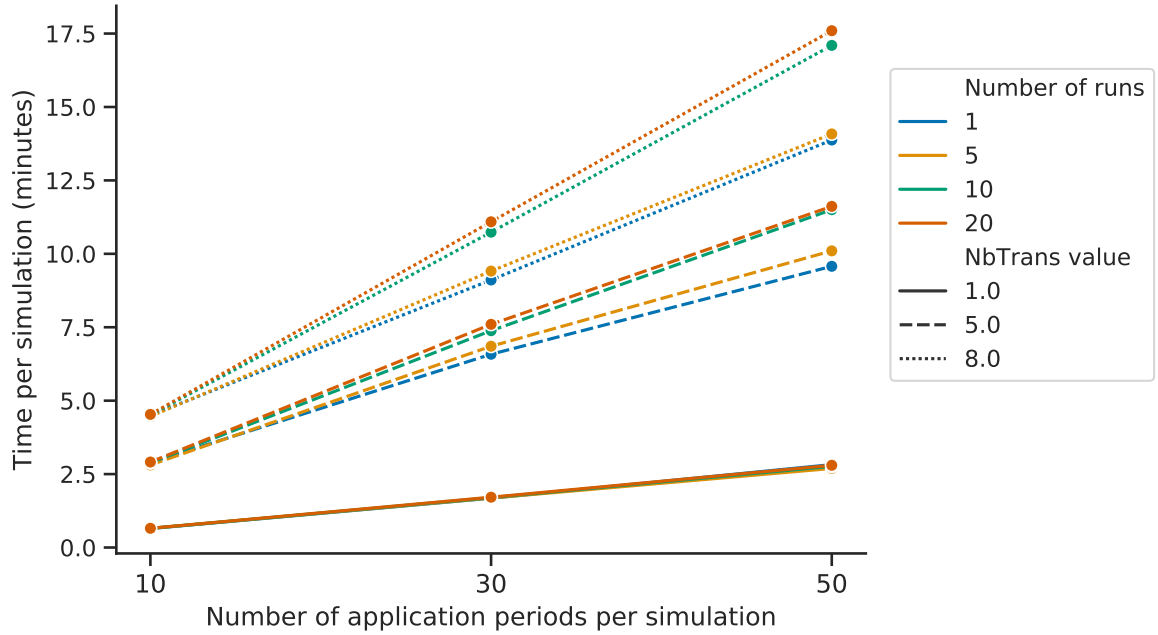


Figure 3.1. Average time it takes to complete one simulation given a scenario of 1200 devices, 15 % sending confirmed traffic and all generating packets of between 12 B and 18 B every 591 s.

Another simulation input that had a significant impact on simulation duration was NbTrans. Higher values of NbTrans increase the number of transmitted packets significantly, which leads to more computations that must be completed. The number of times a simulation is repeated (runs) also impacted the average simulation duration. When 20 simulations are performed, some will finish earlier than others due to fewer calculations required, as they might have fewer congestion events. Each time a simulation is conducted, a unique network is generated with a different split between the SF used by nodes and differing starting delays. Simulations will thus have a different amount of transmission overlaps, which impacts the performance metrics and the number of computations required. A more realistic average time per simulation can thus be calculated when more runs are used.

Figure 3.1 thus has to be kept in mind when deciding the target values for simulation length and the number of runs. Table 3.2 shows how the mean of both performance metrics change as both simulation length and the number of runs increase. The data captured in the table is shown in graph form in Figures 3.2 and 3.3.

Table 3.2. The arithmetic means of ULPDR and CPSR values as percentage points when calculated over different simulations lengths and number of runs. The standard deviation is given in brackets. All values are for the case of NbTrans=8.

Number of runs		**Simulation length**		
		10	30	50
	1	ULPDR: 91.41 CPSR: 45.98	ULPDR: 88.81 CPSR: 44.92	ULPDR: 91.78 CPSR: 46.31
	5	ULPDR: 91.75 (0.74) CPSR: 46.88 (0.82)	ULPDR: 90.67 (1.3) CPSR: 44.78 (0.34)	ULPDR: 91.19 (0.68) CPSR: 45.22 (0.98)
	10	ULPDR: 91.37 (0.8) CPSR: 46.78 (0.99)	ULPDR: 90.21 (1.12) CPSR: 44.87 (0.84)	ULPDR: 91.08 (0.8) CPSR: 45.09 (0.95)
	20	ULPDR: 91.48 (0.78) CPSR: 46.71 (0.91)	ULPDR: 90.57 (1.02) CPSR: 44.98 (1.05)	ULPDR: 91.16 (0.92) CPSR: 45.42 (1)

The changes in ULPDR and CPSR must be considered in the context of wireless networks. In these networks, performance often fluctuates as many internal and external factors influence network conditions. As a result, the impact of, e.g. an algorithm is only considered significant if it changes performance by a large enough margin, such as 5 percentage points. With this context in mind, an examination of the graphs shows that increasing the number of application periods would not significantly change either performance metric; the impact is too small. The graphs also reveal that whilst running more than one simulation is a good idea (especially for CPSR), the difference between five and twenty simulations is minimal.

An approach using 30 application periods and at least ten runs would thus be an acceptable trade-off between the number of computations required and ensure that the results are still representative. The results for NbTrans=8 were presented here, and NbTrans=1 or NbTrans=5 would not change the conclusion. The tables and graphs for these NbTrans values can be found in Section A.1.

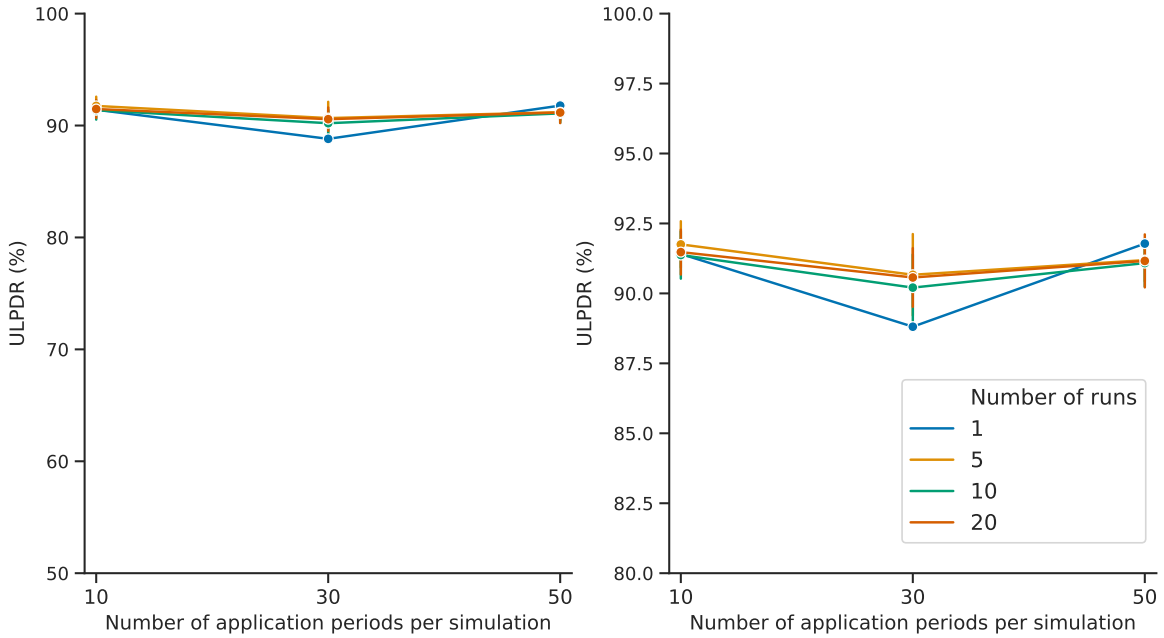


Figure 3.2. The mean and standard deviation of the ULPDR values in Table 3.2.

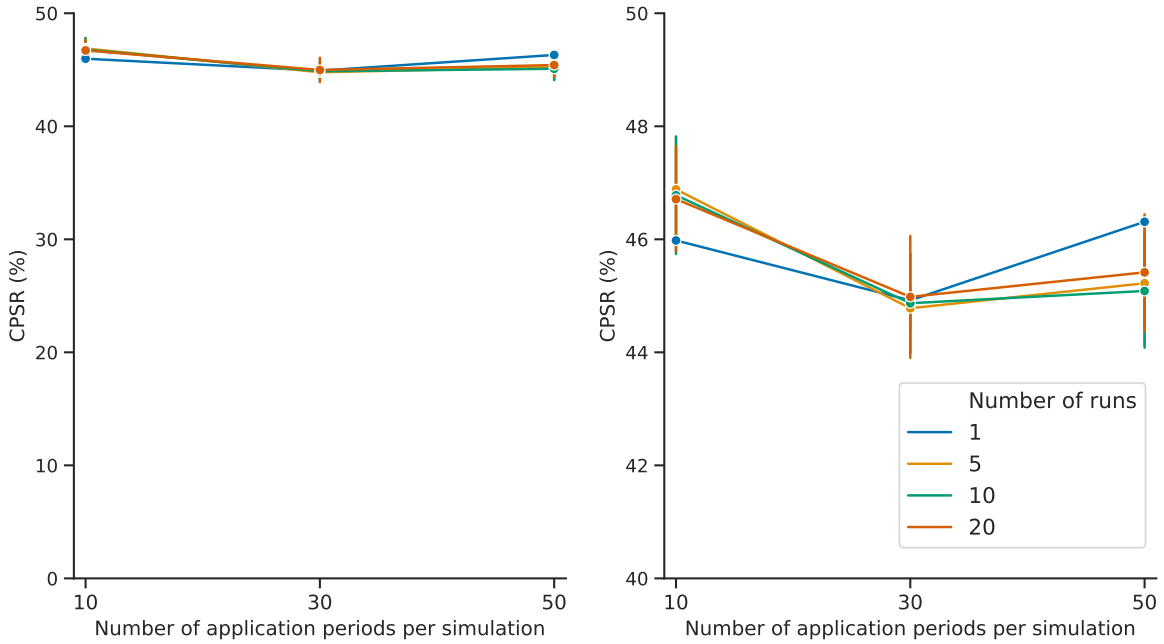


Figure 3.3. The mean and standard deviation of the CPSR values in Table 3.2.

This approach is valid for simulations conducted with standard (stock) “lorawan". The simulations conducted with the modified and battery-less versions were subjected to similar considerations. However, as different aspects were investigated, simulation length and the number of runs differed for these simulations. The exact values and their reasoning are thus given in the relevant chapters.

3.5 LIMITATIONS

While simulators have many benefits, they are not without faults. At a fundamental level, they only have a representation of the physical world and thus do not capture all aspects thoroughly. The conducted simulations considered an open-air environment with only path loss and thus did not consider the impact of streets and buildings. Furthermore, simulations assumed the existence of a standalone LoRaWAN, free from interference from other LoRaWANs in the area or from other transmissions in the same frequency bands. The presented results should thus be considered a baseline and may change in urban deployments.

If a specific deployment scenario is envisioned, the differences between the simulation and a deployed network can be minimised by first matching all of the expected LoRaWAN parameters between the simulation and the deployed network. Secondly, measurements of the SNR achievable at node locations could be performed and the simulator's PHY model updated to better reflect the conditions experienced at the target location. The "lorawan" simulator has support for mobile nodes and the specifications of buildings, so these aspects could be looked at as required by the deployment scenario.

All simulation scenarios contained one LoRaWAN gateway, as not all deployed networks have multiple gateways due to their cost. Furthermore, a network may start with only a few nodes and one gateway and grow over time. Multiple gateways significantly change a LoRaWANs performance, and any research done with one gateway networks should not be generalised to multiple gateway networks or vice versa.

Finally, whilst "lorawan" has an active community that regularly provides updates, research on all aspects of LoRa and LoRaWAN is still ongoing. The presented results were thus obtained with a version of this simulator, and later versions might produce different results. For example, whilst the simulator accounts for the capture effect, research on this is still ongoing with [95] indicating that its performance in small-scale lab experiments might not scale to larger networks.

3.6 CHAPTER SUMMARY

This chapter presented an overview of why simulation is the preferred method to study congestion and also reviewed the available simulators. The rationale for the simulator choice and other implementation aspects were provided. This was followed by a discussion of the other software elements and the approach taken to data analysis. Finally, the limitations of the chosen approach were indicated.

In Section 3.2, the research design was presented, which justified why a simulator-based approach was chosen over a more costly testbed-based approach. The considerations and requirements for such a simulator were also given, which indicated that an accurate model of the PHY layer and support for several aspects of the LoRaWAN protocol was required.

Various aspects of the research instruments were discussed in Section 3.3, with the chosen simulator being the ns-3 component “lorawan" from SIGNET lab. Two versions of this simulator were used, as the authors created a different version with support for battery-less nodes. This simulator met all of the stated requirements and has an active research community. Simulations were conducted utilising the sem Python package with graphs created by seaborn in JupyterLab notebooks.

The nature and analysis of simulation outputs were given in Section 3.4, which indicated how the network’s behaviour could be studied through ns-3’s logging system. Additionally, a study on how simulation length and the number of simulations impacted results was presented. Furthermore, how congestion was created by manipulating the application period (the number of seconds between periodic packet generations by a device) was indicated. It was found that running simulations for 30 application periods (30 packets per device) and averaging over ten simulations was a good choice. This section also indicated that the two main performance metrics used would be CPSR and ULPDR. Finally, the limitations of this work were given in Section 3.5, which indicated that care should be taken to generalise the findings to all types of LoRaWANs as the deployment environment and the number of gateways both have an impact on performance. Additionally, active research is still being done on LoRa and LoRaWANs, so future simulator versions might present different findings.

CHAPTER 4 ACS AND GROUPEDPACKETS

4.1 CHAPTER OVERVIEW

This chapter details the rationale behind and the implementation details of the Adaptive Congestion Scheme (ACS) and the groupedPackets algorithm. A discussion on how congestion can be studied is given in Section 4.2 to indicate the simulation setup used during development. It is followed by an analysis of the impact and causes of congestion. An overview and the purpose, of the ACS, are provided in Section 4.3 and information on its implementation is also provided. Finally, a novel congestion reduction algorithm, referred to as groupedPackets, is discussed in Section 4.4. Finally, the chapter is summarised in Section 4.5.

4.2 A STUDY OF CONGESTION

4.2.1 Simulation setup and scenario

Before an algorithm can be developed to improve congestion, congestion must first be studied and reliably generated. To this end, a simulation scenario must first be defined, which causes congestion and the resultant congestion studied. The chosen simulation scenario is a single GW serving multiple nodes, with nodes periodically generating packets as described in Section 3.4.3. The network is configured to follow the EU868 channel plan with nodes transmitting using one of three channels (sharing a 1% DC restriction). Per the specifications, receive window two was fixed to DR0 and 869.525 MHz which allows a 10 % DC.

A set of 1200 nodes were uniformly distributed in a circular space around the GW with radius $r = 6300\ m$. This value is near the maximum distance a node using SF12 can still communicate when only propagation loss is considered[1]. SF assignment was performed with the SetSpreadingFactorsUp

[1] Older publications using this simulator used 7500 m, see `https://github.com/signetlabdei/lorawan/issues/101` on why this is no longer valid.

function of “lorawan", which calculates the lowest SF a device should use to still ensure connectivity (ADR is disabled). This setup thus ensures that all nodes are within range of the GW and are using the optimum spreading factor, which minimises transmission time whilst still ensuring connectivity.

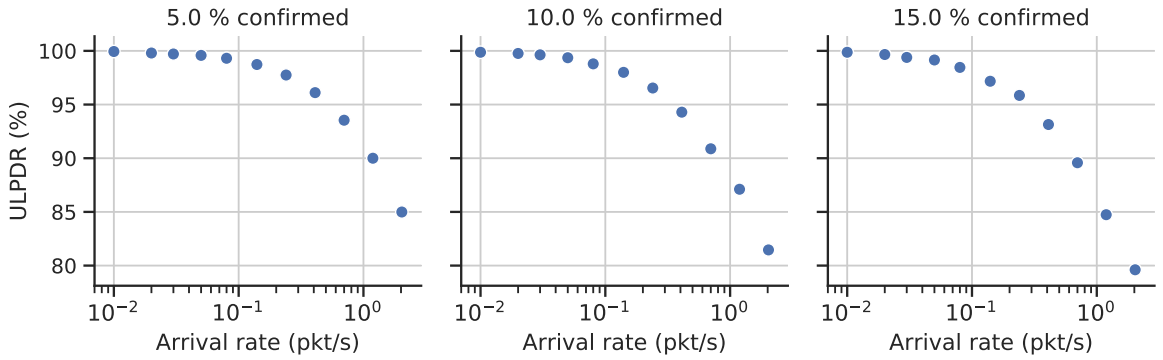
Nodes generate application packets with a starting size of 10 bytes to which a random additional i, $i \in \{2,...,8\}$ bytes are added. The payload analysis conducted in [105] of devices in The Things Network (a crowd-sourced LoRaWAN) found that 50 % of payloads are less than 19 bytes, with the average payload size being 18 bytes. A random element was added as not all devices in such a large network can be assumed to be transmitting the same data in every period.

The literature study revealed that congestion is especially prevalent in networks containing confirmed traffic (packets which require ACKs). Specifically, uplink traffic that requires an ACK from the gateway to indicate that it was received. The same approach as used in [102] was adopted to obtain confirmed traffic ratios; a percentage of devices is selected to send only confirmed traffic whilst the rest send unconfirmed traffic only.

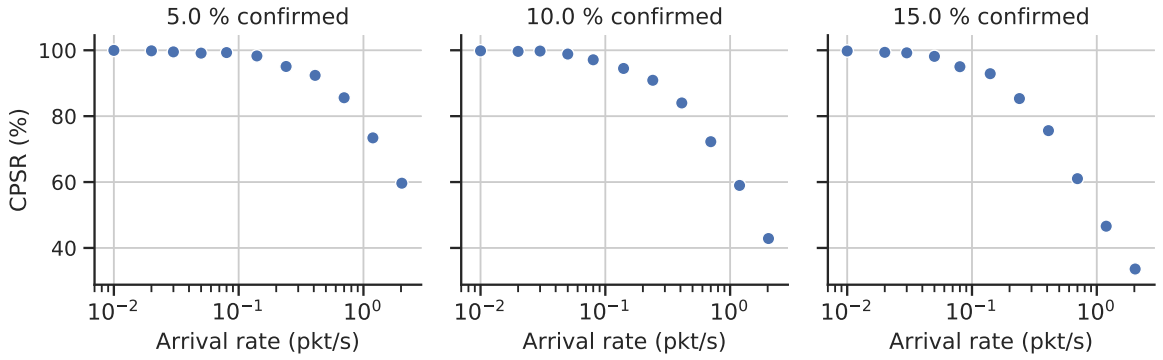
The presented setup served as the basis of all conducted work, and by varying the packet arrival rate, the congestion level in this simulation scenario can be varied. A practical method of generating several packet arrival rates is using the logspace function of Python’s NumPy package. Using ten as the base, arrival rates spaced evenly on a log scale can be generated for 0.01 pkt/s up to 2.031 pkt/s. In the following section, the impact of these arrival rates can be seen on network performance, which also shows that one does not need to simulate higher arrival rates (performance is already severely impacted).

4.2.2 Impact of generated congestion

Figures 4.1(a) and 4.1(b) show how both performance metrics are impacted when the arrival rate is increased. The figures were generated from a 30 application period network, averaged over ten runs and where NbTrans was equal to one, implying that no repeated transmissions occurred for either traffic type. At low arrival rates, such as those less than 10^{-1} pkt/s, reliability remains above 95 % as networks are not congested. After this threshold, a steep decline is experienced for both traffic types, with higher ratios of confirmed traffic worsening the decline. Up to a ratio of 15 % was examined because higher ratios are unlikely as LoRaWANs were designed for mainly unconfirmed traffic.



(a) Unconfirmed traffic.



(b) Confirmed traffic.

Figure 4.1. Performance of an NbTrans=1 network, examined over several arrival rates and confirmed ratios.

The graphs also show that the reliability of unconfirmed traffic differs significantly from that of confirmed traffic. ULPDR remains above 80 % in nearly all of the cases whilst CPSR drops to below 40 % in 15 % networks. At such a low level, confirmed traffic is too unreliable to be considered viable for most if not all IoT use cases.

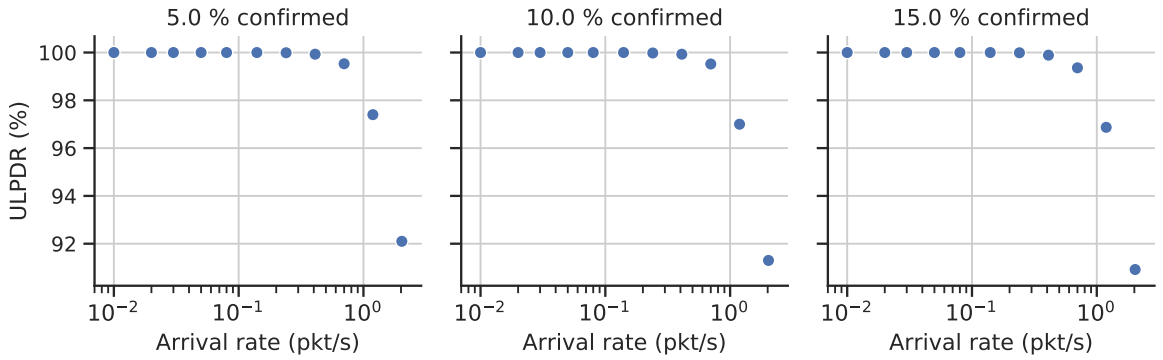
A significant cause of the reduction in CPSR performance is that the DC restrictions imposed on gateways cause the performance to drop significantly [105, 51]. The gateway successfully receives many confirmed packets but is not permitted to transmit an ACK. As this is an NbTrans=1 network, the nodes do not retransmit the packet, so no additional opportunities are given to the gateway to respond. The half-duplex nature of LoRaWAN gateways also increases missed UL transmissions as a gateway cannot receive any transmissions whilst transmitting an ACK. Unconfirmed packets are also not retransmitted, so these packets are lost.

Limiting NbTrans to only one is the default in a LoRaWAN. However, increasing this value is common to enable retransmissions. A balance must be found in the chosen value, as the wrong choice can result in nodes depleting their energy sources prematurely. The wrong value will also unnecessarily strain the gateway due to it being overloaded with packets it had already received. This overload can occur with unconfirmed traffic, as each packet is simply sent *NbTrans* times, and the gateway could thus receive several copies of the same packet and miss other packets it had not yet received.

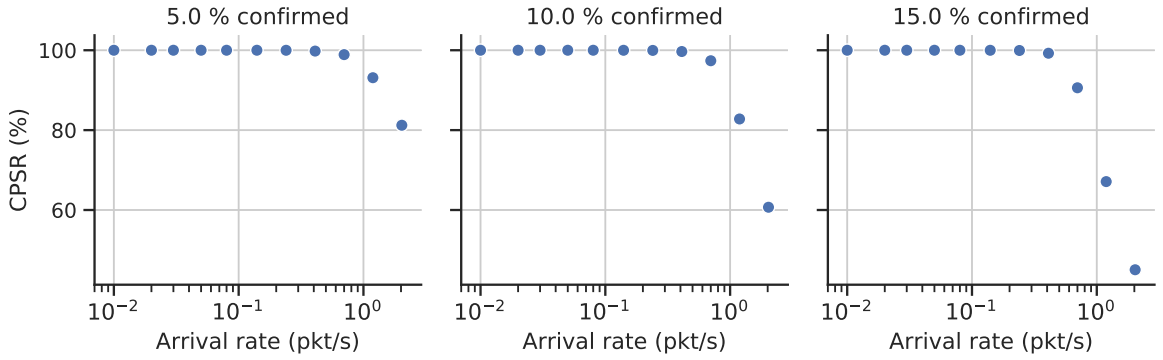
Figures 4.2(a) and 4.2(b) show how effective changing NbTrans to eight can be in improving network performance. ULPDR remains above 90 % in all cases and CPSR performs much better at higher arrival rates. A value higher than one allows packets missed due to gateway transmissions/interference or no more available receivers other chances at reception. Furthermore, confirmed packets will be retransmitted up to *NbTrans* times should an ACK not be received. By configuring NbTrans to be eight, a reliability target of 95% can be achieved up to arrival rates of 1 pkt/s, with the only exception being the 15 % confirmed traffic network failing to do so.

Whilst NbTrans can positively impact the network, it is costly. This cost is especially noticeable at higher arrival rates, as high NbTrans values cause more packet transmissions to occur in increasingly shorter time periods, resulting in a higher effective arrival rate. Figure 4.3 shows how the number of packets lost due to interference, lost due to all gateway receivers being utilised or lost due to GW transmissions increase at higher arrival rates. The stated values are for a 15 % confirmed network and are based on sent packets in the network, unlike CPSR and ULPDR, which looked at unique packets. If a uniquely generated packet was thus sent eight times, all eight outcomes were recorded.

The graphs reveal that whilst NbTrans is effective and a packet can be received as long as one of the eight attempts survived; it has its limitations in higher arrival rate networks. A LoRaWAN gateway only has eight receive paths; thus, the number of packets lost due to this limitation can only be solved by adding more gateways or hardware changes. Similarly, downlink traffic will result in packets lost due to GW transmissions, so these packets can be successfully received only by minimising occurrences of this or by adding more gateways. Losing packets due to self-interference occurs at all arrival rates but becomes a bigger problem at high arrival rates. Optimising the SF used by nodes is one way to reduce the number of packets lost to this type of interference. Finally, at an arrival rate such as 2.031 pkt/s, nodes would not be able to support higher NbTrans values as they must consider the DC restrictions of their transmissions.



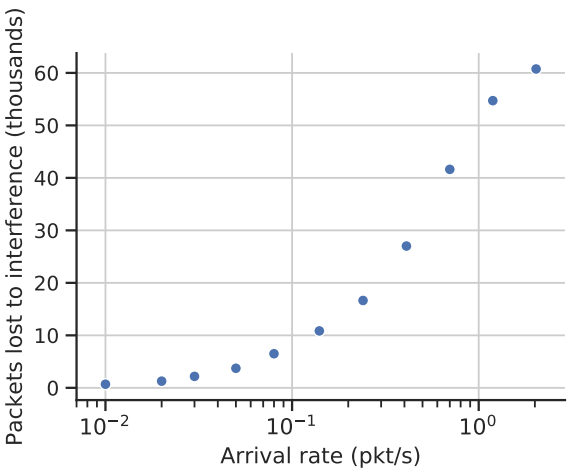
(a) Unconfirmed traffic.



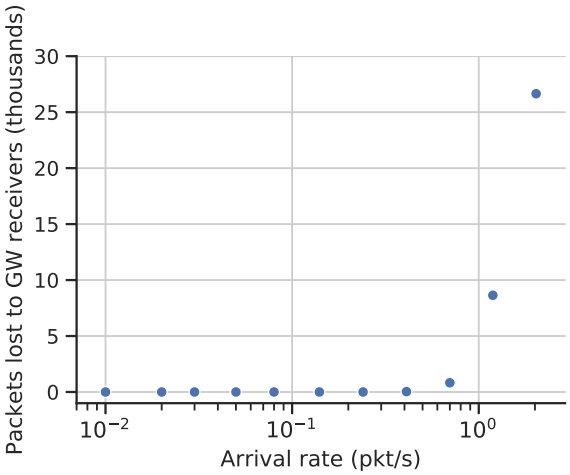
(b) Confirmed traffic.

Figure 4.2. Performance of an NbTrans=8 network, examined over several arrival rates and confirmed ratios.

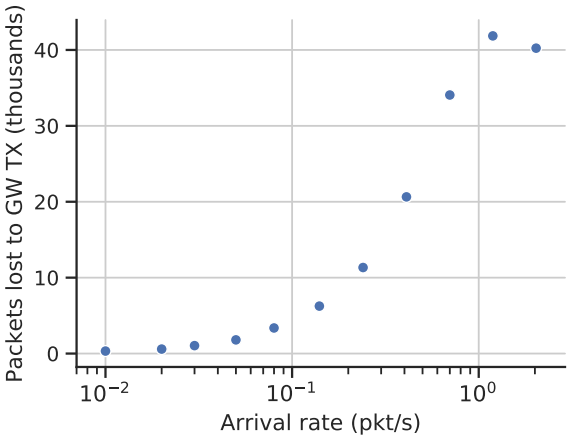
The graphs also indicate that there is a need to improve the LoRaWAN protocol to better deal with congestion, especially that caused by confirmed traffic. Increasing the presence of this traffic from 5 % to 15 % was sufficient to drop the CPSR from 81.22 % to 45.04 % in an NbTrans=8 network. The literature survey in Chapter 2 revealed that the LoRaWAN protocol does not have a congestion monitoring and response mechanism. This work aims to solve that and has resulted in the creation of the Adaptive Congestion Scheme (ACS).



(a) Packets lost due to interference.



(b) Packets lost because all receivers are used.



(c) Packets lost due to GW transmissions.

Figure 4.3. Causes of packet loss in a 15 % confirmed network configured to use NbTrans=8.

4.3 ADAPTIVE CONGESTION SCHEME (ACS)

4.3.1 Purpose and overview of the ACS

The ACS was born out of the realisation that the LoRaWAN protocol needs a mechanism that continuously monitors a network for congestion and which is capable of taking steps to reduce this congestion. The protocol already has a mechanism that continuously monitors networks: the ADR scheme responsible for optimising the data rates of nodes. The ADR scheme could thus be studied to learn how a similar scheme can be built. By studying the ADR scheme, the ACS can be designed to integrate well into the existing protocol and make implementation as straightforward as possible.

The ADR scheme can be classified into two parts, one operating on the NS and the other on the node [106]. The NS part monitors uplink frames and calculates an SNR “margin" to determine if any changes can be made. A change to the data rate or the transmission power is then made if a sufficient margin exists through a LinkADRReq MAC command. The node acknowledges the changes through a LinkADRAns MAC command in its next uplink frame. The node part of the algorithm acts as a guard against the NS part requesting changes that result in a lack of connectivity. Should the node part detect no requested downlink feedback, it decreases the data rate, thus improving the range and restoring connectivity.

A similar two-part configuration was chosen for the ACS. A NS part will monitor the network and, upon reaching preset congestion levels, take action through a custom MAC command (LinkACSReq). In the same fashion as the ADR scheme, nodes acknowledge the requested changes via a LinkACSAns command. Similarly, a node part will ensure that transmissions still occur if the NS part has requested settings that turn out to be unachievable.

One major difference between the two schemes is that the ACS was designed to be more flexible and allow future solutions to be incorporated. The ADR scheme is fixed because it will only change data rates or transmit power. The ACS, on the other hand, was designed to allow for more changes by supporting different congestion-reducing algorithms. One such algorithm, referred to as groupedPackets, is described in Section 4.4.

One of the things that the ACS and ADR scheme share is the ability to opt-out. Participation in the ADR scheme is voluntary, and this ability keeps the developer in control as they will understand a specific network’s operating conditions better than a protocol designer and may wish to use fixed SF assignments. The ACS also follows this convention, and nodes have a method to indicate that they do not wish to receive any change requests.

Another aspect that both schemes share is the fact that they operate continuously. In both cases, the scheme will send a series of change requests the first time a node joins the network as its current settings are likely not the optimised settings desired by the scheme. Both schemes will continue to monitor a node’s performance and send adjustments as required during the entire duration of a node’s connection to the network. In the case of the ADR scheme, this is driven by the fact that SNR levels do not remain constant over time whilst in the ACS’s case congestion could also differ over time.

In networks with low traffic volumes, congestion control via the ACS is not needed. The ACS can remain dormant until it detects traffic volumes greater than a set threshold in combination with the presence of significant confirmed traffic. For example, an examination of Figure 4.2 revealed that 10^{-1} packets per second and the presence of more than 5 % confirmed traffic are reasonable thresholds for this particular scenario. In general, these values should be chosen by the network operator after monitoring the network and deciding on acceptable ULPDR and CPSR levels. Monitoring should be performed at the network level for multiple gateway networks, but the load balancing between gateways should be considered. The activation of the ACS might be unnecessary if more optimum gateway placements can allow for even utilisation of SFs or better overlap of coverage zones.

4.3.2 Implementation details

To enable the implementation of the ACS, the LoRaWAN protocol had to be modified. The MAC payload part of a message contains the Frame Header (FHDR) that has a one-byte long FCtrl element. To add ACS support, this element was expanded to two bytes, as shown in Fig. 4.4. For downlink packets, one bit is used to indicate that the network server can send ACS commands. For upload packets, one bit is used to indicate a device's participation in ACS (nodes may opt-out). Another three bits are used in uplink packets by the groupedPackets algorithm to indicate how many application packets were aggregated.

The remaining bits are classified as Reserved for Future Use (RFU). At this stage, the ACS has only one congestion reduction algorithm (groupedPackets). More algorithms can be added later and activated on their own or in conjunction with groupedPackets; thus, space was reserved in the header for these.

Bit#	[15..9]	8	7	6	5	4	[3..0]
	RFU	ACS	ADR	RFU	ACK	FPending	FOptsLen

Downlink FCtrl fields

Bit#	[15..12]	[11...9]	8	7	6	5	4	[3..0]
	RFU	GroupedP	ACS	ADR	ADRACKReq	ACK	ClassB	FOptsLen

Uplink FCtrl fields

Figure 4.4. New contents for FCtrl of the frame header. Taken from [10], © 2022 IEEE.

To compensate for a larger header, the maximum application payload size was reduced by one byte to keep the maximum ToA for LoRaWAN packets unchanged. The maximums are SF dependent, and this change would reduce the maximum allowed payload for packets sent using, e.g. SF12, from 51 to 50 bytes. The number of IoT applications this would impact would be minimal, as analyses performed in [105] showed that 93.7 % of TTN's payloads were below 50 bytes. The larger header size will cause transmissions to take slightly longer and also slightly increase power consumption.

In terms of implementation, the ACS can be added as another element to the NS in a similar fashion to ADR algorithms. The meta-data of all packets received by the NS can be analysed to determine the network traffic volumes and the ratio of confirmed traffic to unconfirmed traffic. In "lorawan" the network-controller component queries its list of downlink traffic generating components to see if any component(s) wishes to transmit to a particular device. Such traffic would include ACKs or the ADR scheme component making requests. An additional component was thus created to contain the NS part of the ACS and added to this list.

Inside the ACS component, each uplink packet's uplink FCtrl fields are inspected to see if this device opted in for ACS control. Should this be the case, the node's packet history is retrieved and presented to the activated algorithm(s). This process is performed for every uplink packet received whilst the network is operational. Currently, only groupedPackets exist, so it is the only algorithm that inspects the history. Once the algorithm has been executed, the ACS examines the results to see if a LinkACSReq must be sent to the node.

While the impact of each congestion reduction algorithm available to the ACS can be tested individually, combinations of these and seeing the impact of an algorithm and other changes to LoRaWAN parameters such as NbTrans are also worthy of study. Several possibilities are explored in Chapter 6.

4.4 GROUPEDPACKETS ALGORITHM

The study into the causes of congestion (Figure 4.3) revealed that confirmed traffic is the primary cause of packet loss in congested networks. Not only does a confirmed packet result in two opportunities for self-interference (the uplink and the ACK), but the half-duplex nature of GWs results in several packets being potentially lost whenever a GW must transmit an ACK.

It was thus decided that congestion can be reduced by targeting the behaviour of confirmed nodes to

reduce the number of ACKs transmitted. A survey of the literature found that existing approaches tended to focus on aggregating ACKs, and so the alternative approach (aggregating application payloads) was decided to be worthy of further study. Upon reception at the application server, the aggregated payloads can then be disaggregated and processed as individual data messages.

A new algorithm, groupedPackets, was thus developed and sought to reduce the traffic volumes generated by confirmed nodes by requesting that they aggregate their application packets. This causes nodes to send larger but less frequent confirmed packets. DC restrictions limit the sending of ACKs by a GW, and by reducing the number of confirmed packets, more ACKs can be sent. This, in turn, reduces the number of retransmissions of confirmed packets, causing fewer packet collisions due to interference. Finally, this also combats the half-duplex nature of GWs, as the number of packets lost due to GW transmissions is reduced as fewer transmissions are required.

Similar to the ADR scheme, groupedPackets will require the developer to decide if their application will allow its use. A developer can decide to opt-out entirely or use the algorithm and opt-out only for specific events. Whilst the ADR scheme changes the data rates, groupedPackets will aggregate application payloads. This can occur on the MAC layer of the node (thus not visible to the application running on the device) but it is preferred that the application is aware of this. This will enable applications to indicate that the aggregation target is to be ignored and the latest payload is to be sent immediately should certain events occur for which the time delay would be undesirable.

The algorithm contains two parts; one is executed on the NS side to calculate an ideal value for application payload aggregation for each device. The second is executed by each node and ensures that if the aggregation target is too aggressive, as many payloads as possible are sent instead.

All nodes start with no aggregation (only sending the current application payload) whilst the scheme builds up a history of sent packets and their sizes. Once sufficient history has been gathered, currently three packets, the NS side algorithm is enabled and starts making adjustments. Adjustment can be expected to potentially occur over the entire duration that a node is connected to the network. A node's sent packet history is continuously monitored by the algorithm and updated on each received uplink from the node. The node's history, by default three packets long, is thus the latest three received packets and not the initial packets sent upon first joining the network.

Algorithm 1 Pseudocode for NS side groupedPackets. Taken from [10], © 2022 IEEE.

INPUT: *curNum*: Current number of payloads aggregated by device;
device: a Pointer to a specific end device;

OUTPUT: *newNum*: new recommended aggregation value;

1: $AppPSize \leftarrow 0$ ▷ Max application packet size
2: $TotalAppSize \leftarrow 0$ ▷ Max total application packet size
3: $historyRange \leftarrow 3$ ▷ Num of packets to consider
4: maxGrouping $\leftarrow 5$ ▷ Maximum aggregation limit
5: Update $AppPSize$ and $TotalAppSize$ by calling GetMaximums($device, historyRange$)
6: **if** $TotalAppSize + AppPSize \leq 50$ **then**
7: **if** $(curNum + 1) < maxGrouping$ **then** ▷ Aggregation limit not yet reached
8: $newNum \leftarrow curNum + 1$
9: **else**
10: $newNum \leftarrow curNum$ ▷ Limit hit, no changes
11: **end if**
12: **end if**
13: return *newNum*

The pseudocode for the NS side algorithm is provided in Algorithm 1. Adjustments to payload aggregation are in increments of one payload to prevent large changes between adjustments (Algorithm 1, lines 8). The current setting used by a device is extracted from its latest packet's FCtrl element and is used as input *curNum*. There are two limiting factors on how many payloads can be aggregated: the maximum allowed payload size as dictated by the chosen SF, and the time delay caused by aggregation may become unacceptable.

The first factor's impact can be limited by estimating the total packet size, and should the algorithm get it wrong, sending as many packets as possible (node side algorithm). This task is made more difficult because application payload sizes may not be static but vary. Additionally, aggregation requires the addition of a delimiter between payloads to allow the application server to separate them, and each is assumed to be one byte long.

Total size estimation first estimates the largest application payload size in a node's sent history (Algorithm 1, line 5). A test is then done to see if adding this largest size and the current payload

size would exceed the maximum allowed (Algorithm 1, line 6). The AppPSize value returned by GetMaximums() not only determined the estimated maximum application payload sent in the specified history but already added one byte to the total. This was done to compensate for the additional delimiter that would be required should the number of packets to be aggregated be increased.

The maximum application payload length returned by GetMaximums() is an estimation of the maximum as the correct value is unknown due to the end-to-end encryption applied to all application payloads. The maximum is estimated by dividing the total application payload length by the number of aggregated packets (obtainable from the header). For example, an aggregated payload containing three payloads of 10 B, 12 B and 14 B respectively would result in a total payload length of 38 B (2 B of delimiters were added). The estimated maximum payload length would then be 12.66, which when rounded down and another delimiter added would result in a reported length of 13 B. The addition of 13 B to 38 B would result in an application payload that exceeds the maximum of 50 B, so aggregation would not be increased.

If the combined total remains below the maximum, the node is sent a request to increase the number of payloads it currently groups. Currently, this test uses 50 bytes as the maximum. LoRaWAN does support larger packet sizes, but the maximum depends on the SF used to send a packet. To keep things simple, 50 bytes were chosen to ensure compliance with all SFs. Payloads are likely much smaller than this, and this limit would still allow several to be aggregated.

As application payloads vary, a node may not be able to successfully aggregate a newly received payload with the aggregated payloads without exceeding the maximum size. In these cases, the node side algorithm given in Algorithm 2 is activated. To minimise losing payloads, the node side algorithm dictates that a node re-aggregates as many payloads as possible by first sorting them in a Last In First Out (LIFO) queue (not shown). This ensures the latest packet is always sent with the oldest packet(s) being discarded. For example, if three aggregated payloads are 32 bytes with a newly created payload of 20 bytes, a node cannot transmit all of them. For this example, assume that each of the aggregated payloads was 10 bytes, with the delimiters each requiring one byte. The node will send the latest payload (20 bytes) as well as two of the previous payloads for a total of $20+1+10+1+10=42$ *bytes*. In this case, only the oldest payload was lost as the maximum of 50 bytes must be adhered to. Lines 6 to 8 of Algorithm 2 show how the latest packet is selected before the stored packets are examined and added in lines 10 to 13. A payload and its delimiter are stored together in *storedData*.

Algorithm 2 Pseudocode for node side groupedPackets. Adapted from [10], © 2022 IEEE.

INPUT: *totalPSize*: Total application packet size due to aggregation + latest payload;
numGrouped: The number of payloads aggregated;
latestPSize: The size of the latest payload;
latestP: The latest payload;
storedData: Older payloads stored with delimiters;

OUTPUT: *sentData*: The data to send;

if $totalPSize \leq 50$ **then**
  Call SetSentGroupedPackets($numGrouped + 1$) ▷ Update MAC header
  $sentData \leftarrow (storedData + latestP)$ ▷ Send packet
else ▷ max size exceeded
  $newnumPackets \leftarrow 0$ ▷ New number of payloads.
  $newTotalSize \leftarrow latestPSize$
  $newData \leftarrow latestP$ ▷ newnumPackets is 0 as 0 here refers to 1 packet being added so far (the latest one).
  for all p in $storedData$ **do**
    if $newTotalSize + size(p) < 50$ **then**
      $newnumPackets \leftarrow newnumPackets + 1$
      $newTotalSize = newTotalSize + size(p)$
      $newData = newData + p$
    end if
  end for
  Call SetSentGroupedPackets($newnumPackets$)
  $sentData \leftarrow newData$ ▷ Send packet
end if

The aggregation can be too aggressive if payloads are less than, e.g. 5 bytes, as it will take much aggregation before the maximum size is reached. To prevent long delays, the maximum number of packets to be aggregated can be set by the application developer/network administrator, which is five in this case (Algorithm 1, line 4). The nodes remain in control, and the application running on a node can decide to ignore any aggregation targets and transmit the latest payload immediately (should aggregation cause an unacceptable delay).

Depending on the application responsible for generating payloads, their length could remain constant for the entire duration of a node's connection to the network or change over time. In the case of fixed sizes, a node would receive an initial series of requests to increase aggregation but would from there on only be monitored with no further requests (maximum aggregation has been reached). Alternatively, should payload sizes vary the algorithm would send aggregation requests as dictated by the changes in size, always seeking maximum aggregation based on its payload history for a node.

The algorithm detailed above was implemented in a modified version of "lorawan" to allow for comparisons to be completed with the standard network and networks utilising this algorithm. An analysis of the impact of groupedPackets can be found in Chapter 5.

4.5 CHAPTER SUMMARY

This chapter introduced the ACS and the groupedPackets congestion reduction algorithm. As both of these elements involve congestion, a detailed discussion was presented on the simulation scenario and setup used to create congestion. An analysis of the created congestion was performed, followed by details of the implementation aspects of the ACS. Finally, the purpose and implementation details of groupedPackets were provided.

Section 4.2 discussed the chosen simulation scenario of a single GW serving 1200 nodes. The traffic generation pattern and ratios between confirmed and unconfirmed nodes were given, and simulations of this scenario were presented, which shows how ULPDR and CPSR were affected by congestion. An analysis of the congestion revealed that packet loss could be attributed to self-interference, a lack of available GW receivers, and the half-duplex nature of GWs. The benefits and limitations of simply increasing the NbTrans value used by a network were also examined, and found that whilst effective at low arrival rates, it does not scale well to high arrival rates.

An overview and implementation details of the ACS were given in Section 4.3. This novel scheme aims to continuously monitor LoRaWANs for congestion, and once specified performance indicators were sufficiently affected, action was taken through congestion reduction algorithm(s). The ACS theoretically supports multiple algorithms, with only one developed and evaluated in this work. The modifications necessary to the LoRaWAN protocol were presented, including how this scheme would fit into the overall architecture of LoRaWANs.

The ACS only monitors for congestion and thus requires a congestion reduction algorithm. Once congestion is detected, an algorithm is activated, given its inputs, and the algorithm's response is forwarded to the node by the ACS through a MAC command. A novel algorithm, called groupedPackets, was developed and discussed in Section 4.4. This algorithm reduces congestion by requesting that confirmed nodes start aggregating their application payloads. The algorithm calculates an aggregation target for each node and informs nodes of this through the ACS. Through aggregation, confirmed nodes send larger packets but do so less frequently, easing the strain on gateway resources. The algorithm contains a node part to ensure that aggregation does not become too aggressive, and nodes are allowed to opt-out should their applications be unable to tolerate any delays in receiving ACKs. This algorithm is executed every time an uplink is received from a node, as that specific node's packet history is updated and examined to determine if optimal aggregation still occurs.

CHAPTER 5 GROUPEDPACKETS RESULTS

5.1 CHAPTER OVERVIEW

This chapter presents the results obtained when standard LoRaWAN networks were compared with those using groupedPackets through the ACS. Section 5.2 first explains the procedure that was followed to ensure valid results before findings are presented in Section 5.3. The impact of NbTrans, arrival rate and base packet sizes on groupedPackets were studied. Section 5.3 is finished with an investigation on how effective groupedPackets would be if it were tasked with grouping unconfirmed packets instead of confirmed packets. The chapter is summarised in Section 5.4.

5.2 ENSURING VALID RESULTS

5.2.1 Activation of groupedPackets

Before a suitable value for simulation length and the number of runs can be found, the behaviour of groupedPackets must first be understood. A study of the first couple of application periods is thus required to understand how the algorithm operates before it reaches what can be considered a steady state.

As discussed in Chapter 4, the algorithm must first gather some packet history which will allow it to determine if aggregation is possible. Whilst this is in progress, the algorithm is not requesting any changes. Once sufficient history has been gathered, the algorithm will send requests (with aggregation step sizes limited to one). The groupedPackets algorithm will thus take several periods before nodes are configured to transmit their optimum aggregation number. For example, a node could be able to aggregate up to four packets, but the algorithm will first request it to group two, then three and finally four. It will then monitor a node's packet history and may request adjustments (increasing or decreasing aggregation) depending on the history.

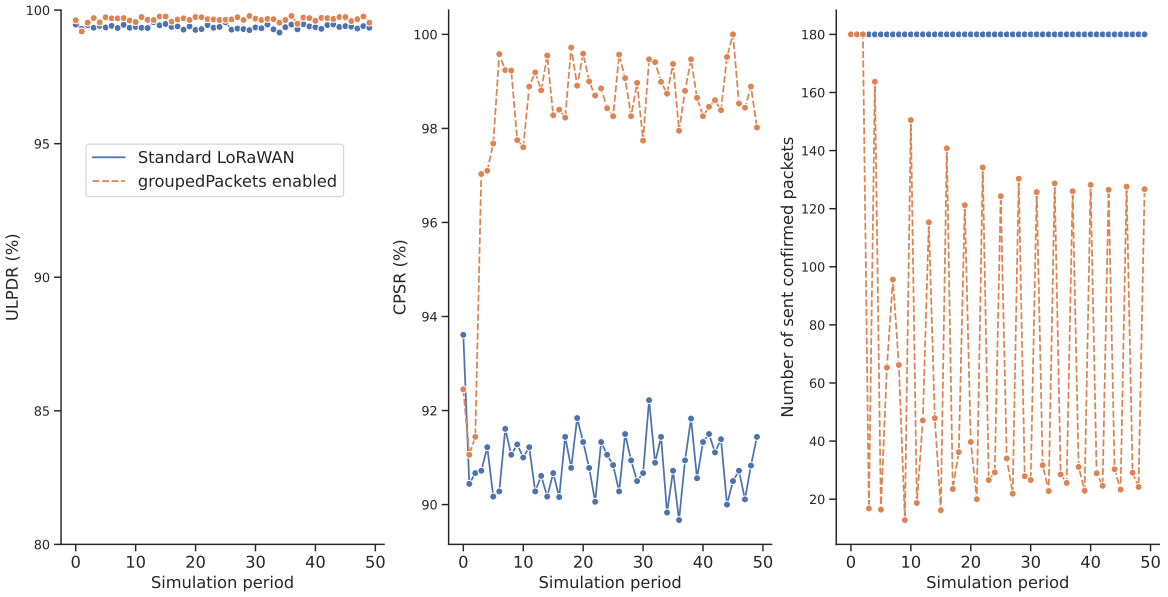


Figure 5.1. Performance impact of groupedPackets for a 15 % confirmed network with $\lambda = 0.702$ pkt/s when examined over a long duration (NbTrans=8).

The algorithm's behaviour can be seen in Figure 5.1, which shows how CPSR performance improves over time. The x-axis indicates the current simulation period, and can be considered as time. The third graph, showing the total number of sent confirmed packets in a simulation period, shows how the algorithm reduces this total from 180 (15 % of a 1200 device network) to a range in which it various (approximately) between 130 and 20 sent confirmed packets per period. The history-gathering phase can be seen in the first few periods (the number of sent packets stays at 180), followed by a decrease in the maximum amount of sent packets before the algorithm settles into this range.

At this point, it is logical to question why is there such a wide range in the number of sent confirmed packets? The answer lies in the CPSR of the standard LoRaWAN, which should be noted to be less than 100 %. This implies that some nodes never receive an ACK, even when retransmitting a packet eight times. When the groupedPackets algorithm requests aggregation, it will only occur if the node received an ACK as the LinkACSReq is piggybacked on this ACK. After sufficient history has been gathered, most nodes would have received an ACK in the next simulation period, but some did not. These nodes are now "out of sync" with the others, as they only receive a request to aggregate in the period thereafter (assuming an ACK is received). This process can happen repeatedly, and several groups of nodes can thus evolve that transmit at different times. As a result, a cyclical behaviour occurs in which some periods will contain nearly all of the 180 confirmed nodes' aggregated packets, with the

rest of the 180 nodes transmitting in the periods thereafter.

The algorithm can be considered to be in a stable state once all nodes have reached their maximum aggregation target. This target can change over time should nodes start consistently sending smaller or larger payloads. The stable state does thus not refer to groupedPackets no longer requesting changes but rather that the initial set of changes have occurred.

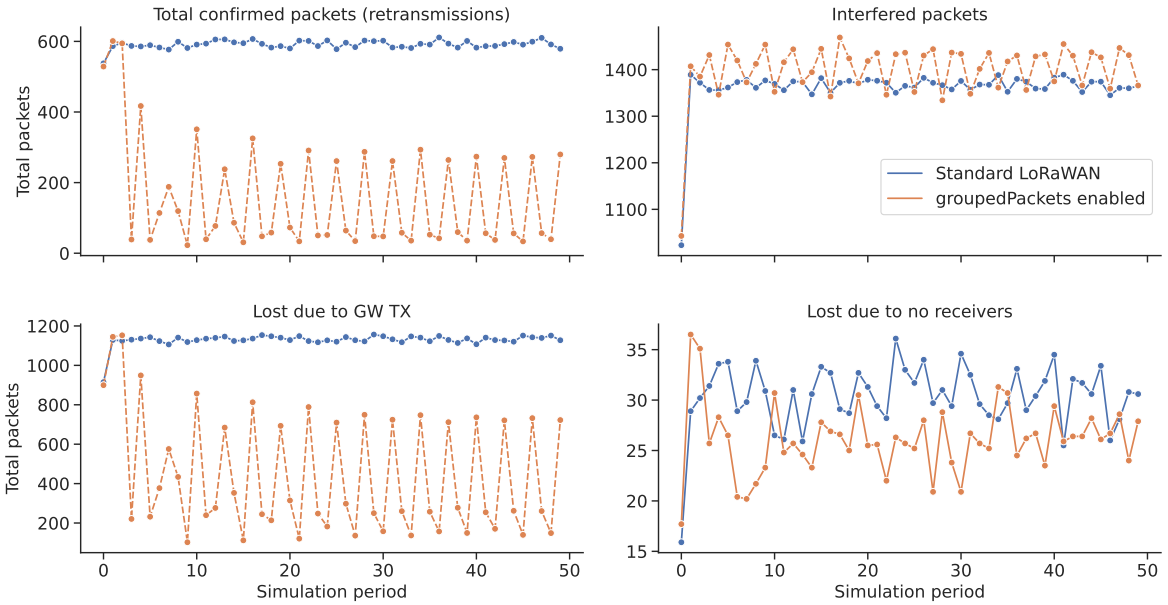


Figure 5.2. Analysing the reasons behind the performance improvement shown by groupedPackets.

Figure 5.2 further explores the impact of the algorithm. The total number of sent confirmed packets reduces (Figure 5.1), which causes the total number of retransmissions also to reduce, as seen in Figure 5.2. This figure also shows that the number of packets lost due to interference (collisions) increases slightly and that fewer packets are now lost to ACK transmissions. The increase in interfered packets can be attributed to the now larger confirmed packet payloads which have a longer ToA. Little change can be seen in the number of packets lost due to no more receivers being available. Unconfirmed packets primarily drive this outcome, and no changes were made to the behaviour of unconfirmed nodes.

Using the algorithm reduces the effective arrival rate and subsequently increases network performance. The study of the behaviour shows that including the earlier intervals in calculations will skew any results as a stable state has not yet been reached. Therefore, the first 20 periods should not be included in the period given to LoraPacketTracker to use when calculating performance metrics. For example,

should 30 periods worth of data be of interest, a simulation of 50 application periods should be run with the final 30 periods used for calculations.

5.2.2 Simulation length and number of runs

As was the case with simulations run with standard “lorawan", experimental simulations were run to ensure that suitable values were chosen for simulation length and the number of runs. Simulations were conducted using NbTrans=8, as this value proved effective in improving packet outcomes and the use of groupedPackets in conjunction with this was chosen to be studied. The same methodology was followed as discussed in Section 3.4.4, with Figure 5.3 showing the impact of simulation length and the number of runs on the average time to run groupedPackets-based simulations. The simulation length values used here are after the initial 20 application periods as per the reasoning of the previous section. Simulations conducted with groupedPackets enabled took longer than standard “lorawan" simulations. This is caused by having to execute the initial 20 periods first, and more calls are made to the computationally expensive LoraPacketTracker class. The additional calls are part of creating the second output file containing the outcomes of each application period.

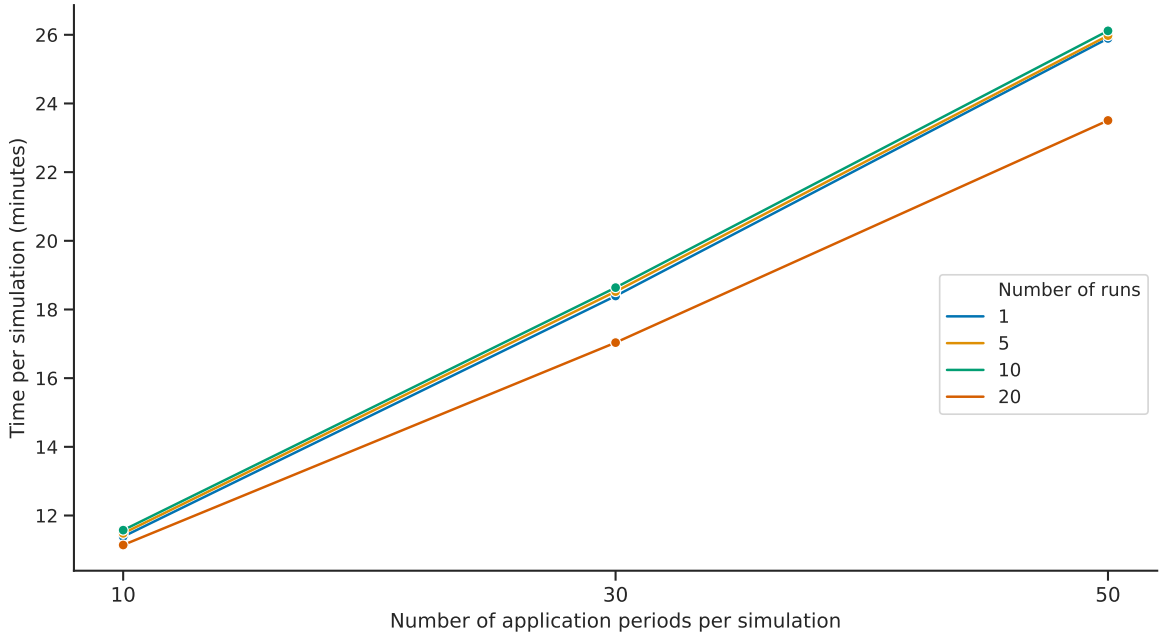


Figure 5.3. Average time it takes to complete one simulation of a groupedPackets enabled network given a scenario of 1200 devices, 15 % sending confirmed traffic, an NbTrans value of eight and all generating packets of between 12 B and 18 B every 591 s.

Table 5.1 shows how the mean of both metrics changes as both simulation length and the number of runs increase. The data captured in the table is shown in graph form in Figures 5.4 and 5.5.

Table 5.1. The arithmetic means of ULPDR and CPSR values as percentage points when calculated over different simulations lengths and number of runs. The standard deviation is given in brackets. All values are for the case of NbTrans=8.

Number of runs		**Simulation length (after initial 20)**		
		10	30	50
	1	ULPDR: 93.76 CPSR: 81.75	ULPDR: 91.14 CPSR: 80.49	ULPDR: 93.69 CPSR: 80.61
	5	ULPDR: 92.64 (1.41) CPSR: 78.44 (3.27)	ULPDR: 92.05 (0.93) CPSR: 79.24 (0.85)	ULPDR: 93 (0.38) CPSR: 79.85 (1.14)
	10	ULPDR: 92.67 (1.02) CPSR: 79.1 (2.47)	ULPDR: 92.82 (1.28) CPSR: 79.75 (1.31)	ULPDR: 93.18 (0.55) CPSR: 79.72 (1.18)
	20	ULPDR: 92.93 (0.83) CPSR: 79.08 (2.62)	ULPDR: 92.99 (1.11) CPSR: 80.32 (1.53)	ULPDR: 93.03 (0.98) CPSR: 80.14 (1.4)

The same conclusions can be reached from these figures (Figures 5.4 and 5.5) as those of standard “lorawan" simulations. The number of application periods does not significantly change either metric and whilst running more than one simulation is wise, the difference between using five and twenty simulations is small. Similar to standard simulations, an approach using 30 application periods (after the initial 20) and at least ten runs would thus be an acceptable trade-off between the number of computations required and ensure that the results are still a good representation. To keep simulations straightforward when comparing standard and groupedPackets networks, standard simulations were also run for 50 application periods, with the final 30 periods used to calculate metrics.

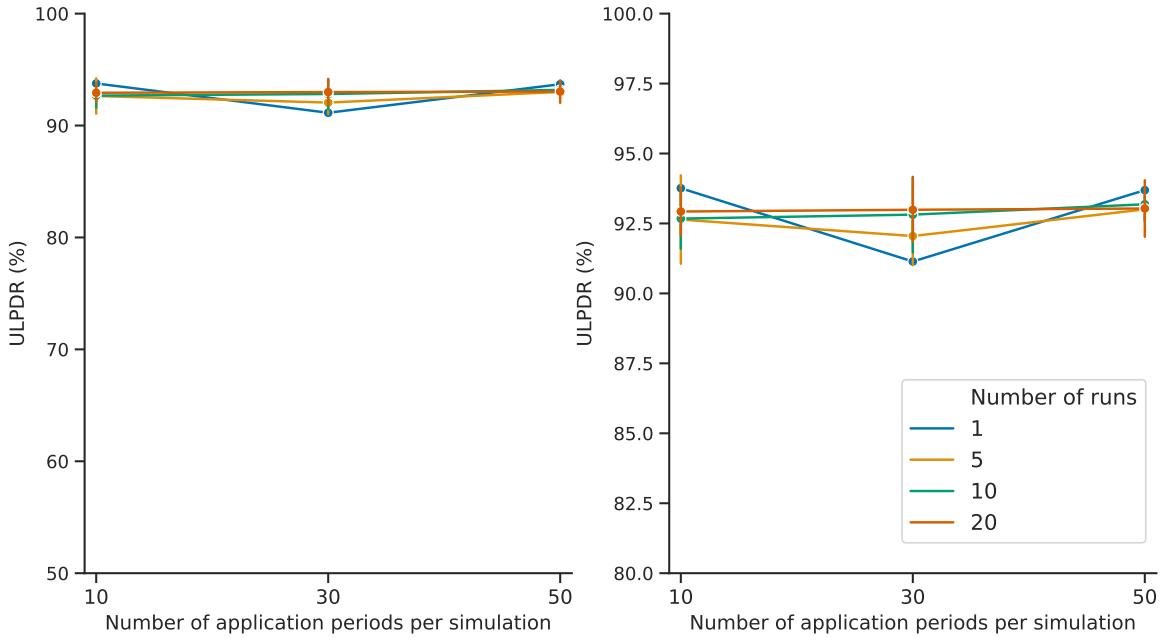


Figure 5.4. The mean and standard deviation of the ULPDR values in Table 5.1.

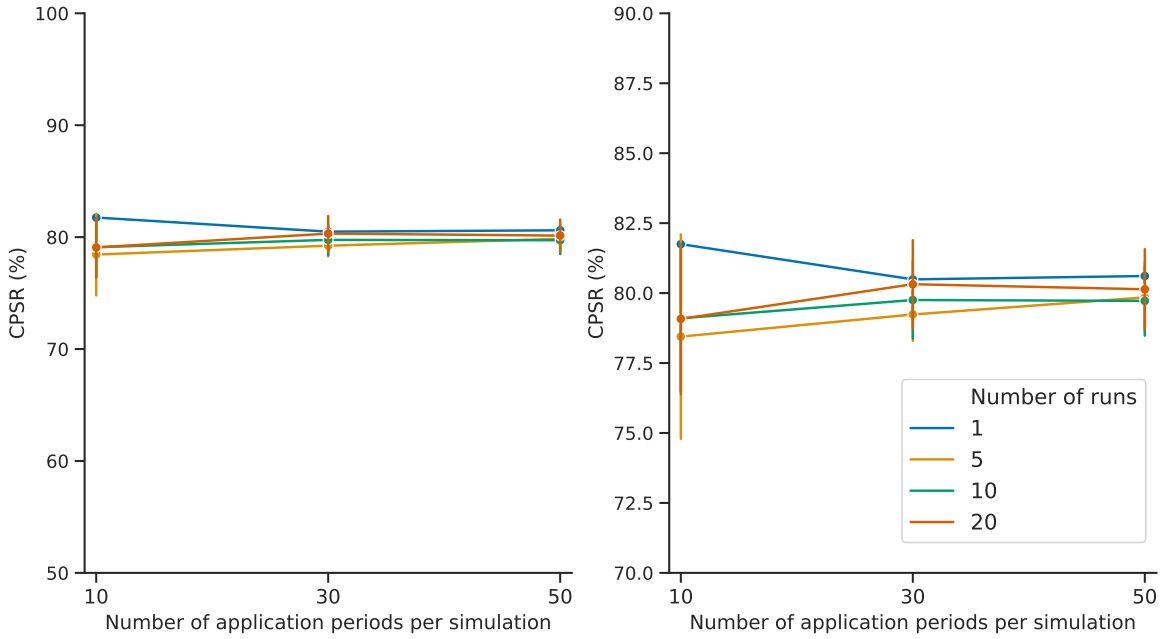


Figure 5.5. The mean and standard deviation of the CPSR values in Table 5.1.

5.3 RESULTS

Once the process to ensure valid results were finalised, several parameter sweeps were performed to investigate groupedPacket's performance. All of the results presented here were taken from networks

in which the ACS and groupedPackets were left enabled throughout the simulation. In other words, the algorithm was not turned off if a performance metric hit a certain threshold, nor were any limitations placed on aggregation (other than the defaults). Only application periods higher than 10^{-1} pkt/s are shown, as Section 4.2.2 revealed that little congestion occurs before this.

5.3.1 Impact of NbTrans

Section 4.2.2 revealed how effective increasing NbTrans can be on both performance metrics in standard LoRaWANs. Using a value higher than one should also be beneficial for groupedPackets enabled networks as it would allow the aggregated packets multiple chances at success. Losing an aggregated packet would result in the loss of multiple data payloads and should thus be minimised if possible.

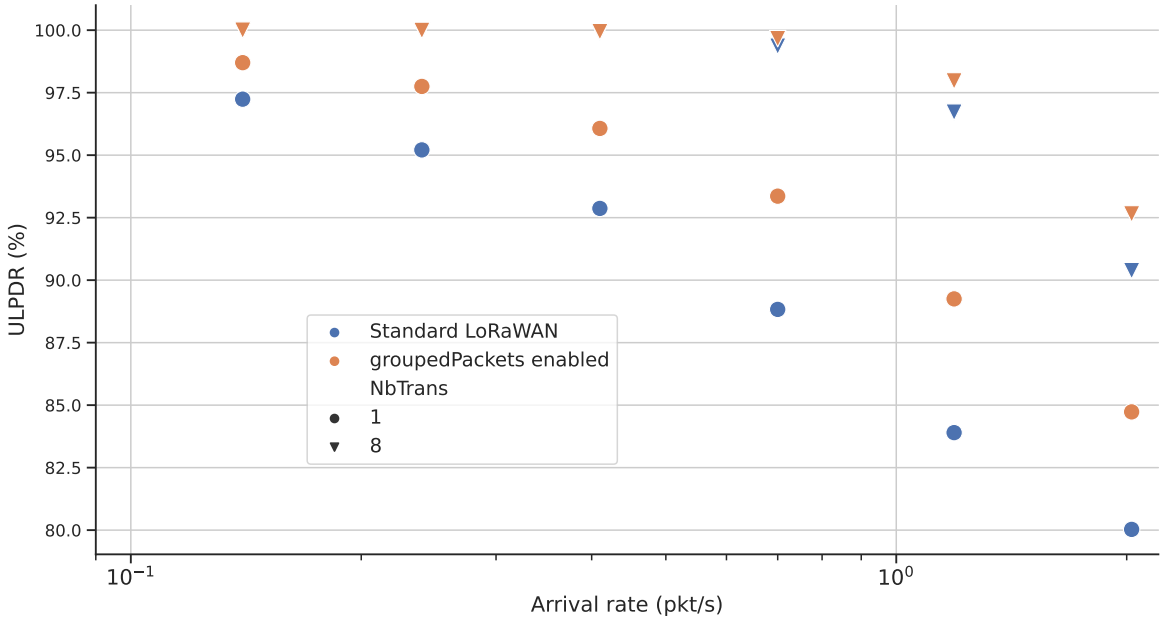


Figure 5.6. Impact of NbTrans on the performance of unconfirmed traffic in a 15 % confirmed network, examined over several arrival rates (12 B - 18 B payloads).

Figures 5.6 and 5.7 show how a higher NbTrans value improves the performance of standard networks and those using groupedPackets. In the case of lower arrival rates, 100 % ULPDR is achieved for both network types and their icons thus overlap in the figure.

The use of groupedPackets is beneficial to ULPDR across all arrival rates and can be improved further when NbTrans is set to eight. Similarly, groupedPackets improved CPSR for all arrival rates and was further increased in NbTrans=8 networks. The impact of a higher NbTrans value on CPSR in

groupedPackets networks is especially noticeable at higher arrival rates and resulted in increasing the CPSR from 62.78 % to 79.94 % for the case of $\lambda = 2.031\ pkt/s$.

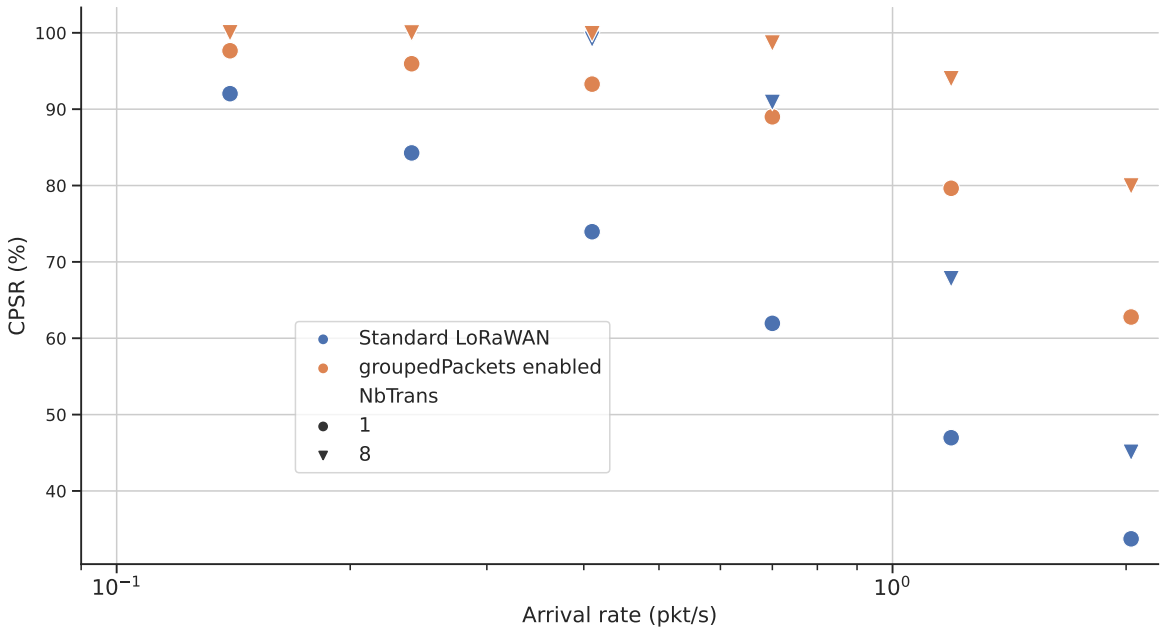


Figure 5.7. Impact of NbTrans on the performance of confirmed traffic in a 15 % confirmed network, examined over several arrival rates (12 B - 18 B payloads).

Section 4.2.2 also noted that NbTrans has limitations at higher arrival rates. The use of groupedPackets reduces the effective arrival rate in a network, allowing for NbTrans to remain effective at higher arrival rates. Due to the aggregation, fewer packets are lost to interference or gateway transmissions.

5.3.2 Impact of arrival rate on groupedPackets

The algorithm's effectiveness across a range of arrival rates is shown in Figure 5.8 and Figure 5.9, which contain error bars showing the standard deviation. The error bars were added to show how the respective metric can deviate across time as the data was collected over 30 simulation periods. Nodes transmitted packets between 12 B and 18 B in an NbTrans=8 network. The figures show the improvements achieved when groupedPackets were activated. Using groupedPackets allows a network to maintain high reliability at faster arrival rates than standard networks. However, the increase in CPSR does come at the cost of an increased delay in ACK reception by nodes.

The algorithm had a minimal impact on unconfirmed traffic except for networks with 15 % confirmed traffic at higher arrival rates. Confirmed packets did not majorly impact unconfirmed packets, thanks

to the simulation's high NbTrans value and the fact that LoRaWAN gateways have multiple receivers. Nevertheless, the reduction in transmitted ACKs allows fewer unconfirmed packets to be lost.

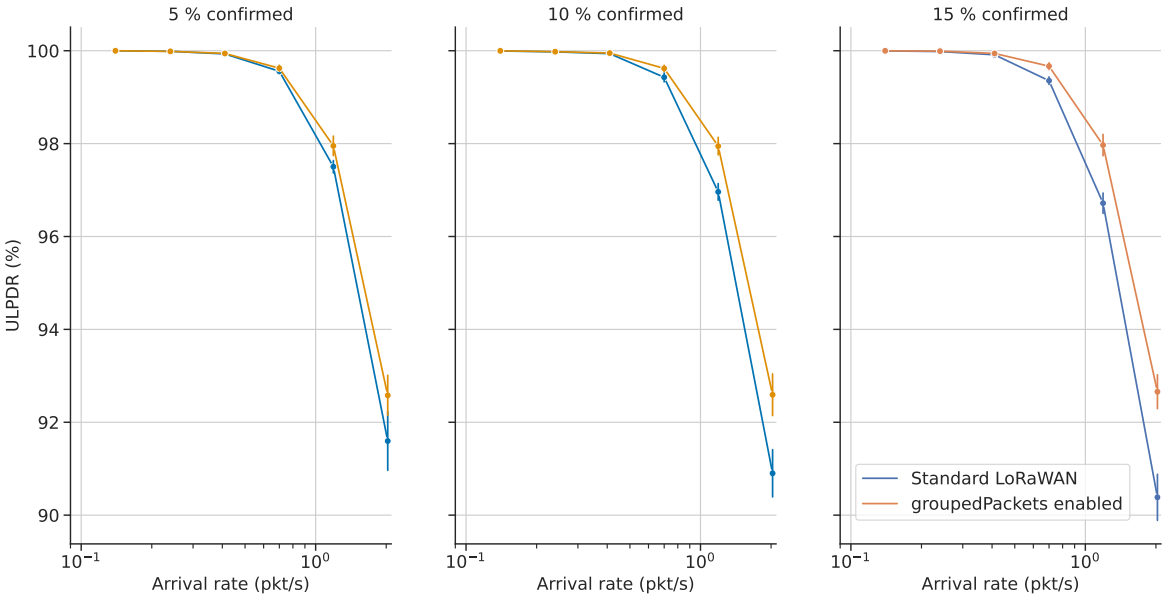


Figure 5.8. Performance of unconfirmed traffic over various arrival rates and confirmation ratios (NbTrans=8).

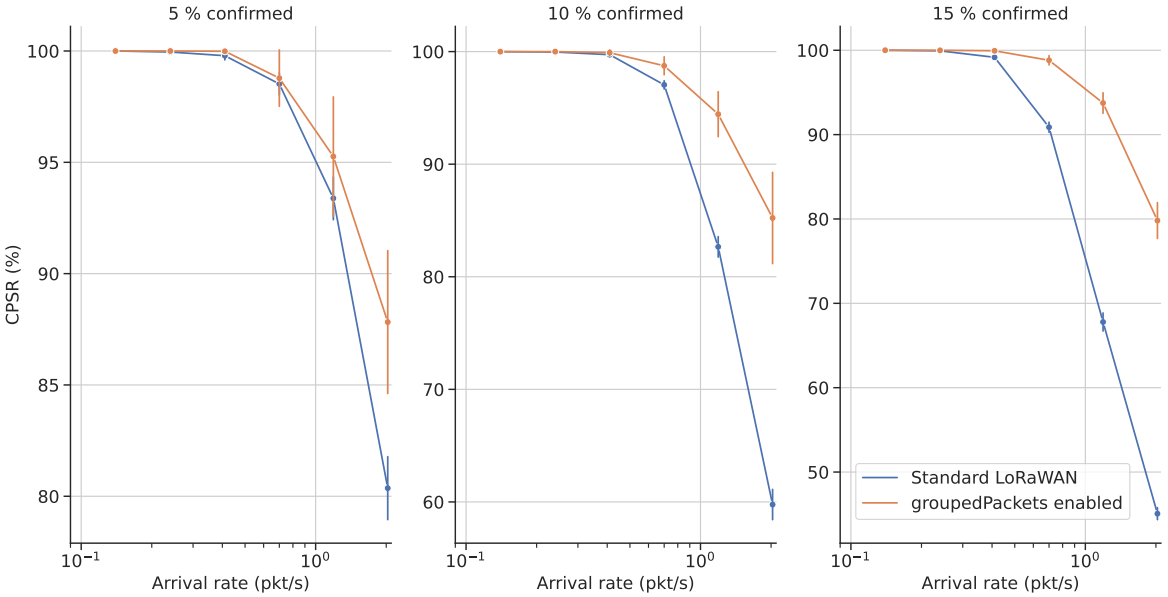


Figure 5.9. Performance of confirmed traffic over various arrival rates and confirmation ratios (NbTrans=8).

In the confirmed traffic case, the algorithm provided considerable improvements for high arrival rates with minimal impact at low arrival rates. As expected, the algorithm offered more significant improvements at higher confirmation ratios. When comparing standard and groupedPackets, a larger standard deviation can be seen when the algorithm is enabled. This increase in the standard deviation was caused by the confirmed nodes being “out of sync" with one another, causing groups which transmit at different times (see Section 5.2.1). The CPSR thus had an increase in its variance as the periods with fewer sent confirmed packets tend to have higher packet delivery compared to intervals with a higher number of sent confirmed packets. This phenomenon can also be seen visually in Figure 5.1, which shows how the number of sent confirmed packets differed between intervals.

5.3.3 Interaction of groupedPackets and various base packet sizes.

A packet’s size impacts its transmission time, and larger packets will require longer ToA, increasing the probability of collisions. Figures 5.10 and 5.11 show the impact of three base packet sizes on the performance in an NbTrans=8 network. As with previous graphs, 30 simulation periods were used, and an additional random element between two and eight bytes was added in all cases. Performance improves for all ratios and base payloads, especially at higher arrival rates and confirmed ratios.

Figure 5.10 continues the trend that groupedPackets do not significantly improve or reduce ULPDR in NbTrans=8 networks. The influence of an increase in base application payload sizes can be noted in both standard and groupedPackets enabled networks, with larger packet sizes resulting in lower performance. The algorithm’s ability to aggregate application packets depends on the maximum packet size (set to 50 bytes). Therefore, smaller base sizes allow for a higher impact on ULPDR due to more packets being aggregated before reaching the 50-byte limit.

When examining standard LoRaWAN in Figure 5.11, it can be noted that base application payload size had little impact on CPSR. Whilst the uplink confirmed packets might have been larger, performance was primarily hampered by the lack of ACKs and not by the slightly higher interference experienced by these packets.

The impact of base application payload sizes is much more prominent in the case of groupedPackets; however, it resulted in performance improvements regardless of size. As was the case for unconfirmed traffic, the algorithm is more effective with smaller payloads due to the ability to aggregate more payloads before reaching the 50-byte limit.

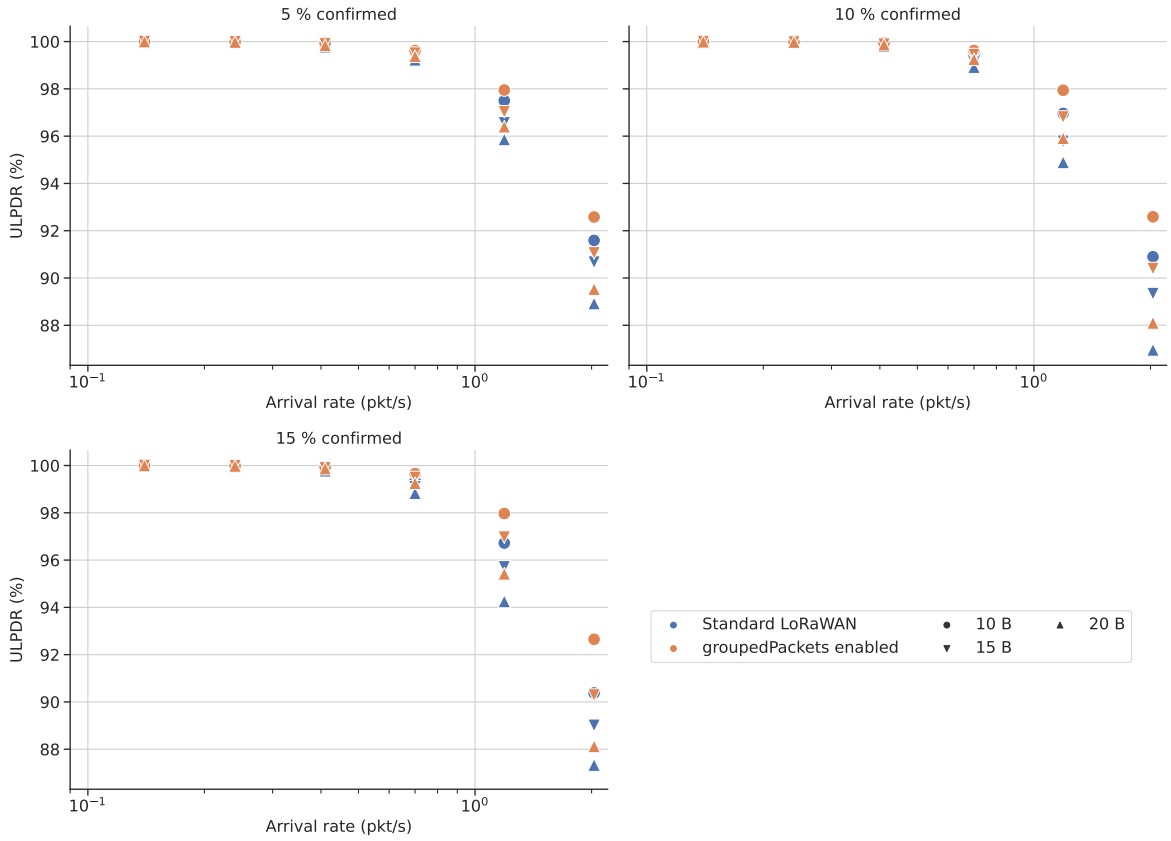


Figure 5.10. Performance of unconfirmed traffic, examined over several arrival rates, base application payload sizes and confirmed ratios. The y-axis of the 5% and 10 % networks are shared.

For an arrival rate of 2.031 pkt/s, the algorithm improved the ULPDR of a 15 % confirmed network by 2.3, 1.3 and 0.8 percentage points for the three base sizes respectively. This improvement is slight, but the CPSR improved by 34.9, 21.9 and 11.8 percentage points for the same case.

The groupedPackets algorithm will continue to provide an improvement provided that it can aggregate at least two payloads. Should payload sizes cause this to be unfeasible, the algorithm will perform the same as standard LoRaWAN. The groupedPackets algorithm works on a node-by-node basis, so this would only occur for the individual nodes with too large payloads.

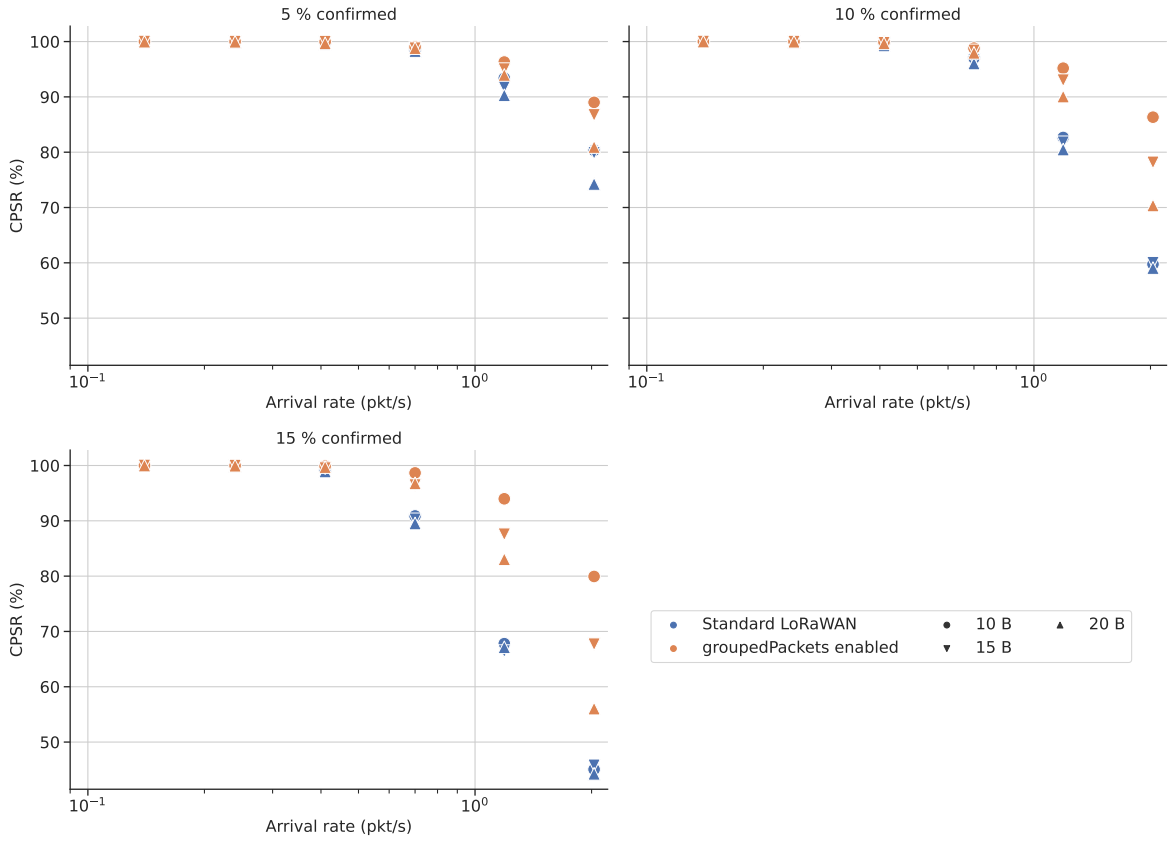


Figure 5.11. Performance of confirmed traffic, examined over several arrival rates, base application payload sizes and confirmed ratios. The y-axis of the 5% and 10 % networks are shared.

5.3.4 groupedPackets with unconfirmed only traffic

Up to this point, groupedPackets was configured to aggregate confirmed traffic. However, it could potentially instead aggregate unconfirmed traffic. Aggregating unconfirmed packets would also reduce a network's effective arrival rate and reduce the number of packets lost due to either interference or no more receivers available at a gateway. Its use would, however, increase the number of packets lost due to GW transmissions, as previously, no such transmissions were made for unconfirmed nodes.

A best-case scenario for an unconfirmed version would thus be a network with no confirmed traffic and one in which packet sizes do not vary. This would allocate all of the available downlink transmissions to groupedPackets, and it would not be required to update nodes once they have reached their maximum allowable aggregation. The algorithm's behaviour in such a best-case scenario is given in Figure 5.12, in which nodes consistently sent 10 B payloads in an NbTrans=8 network (averaged over 10 runs).

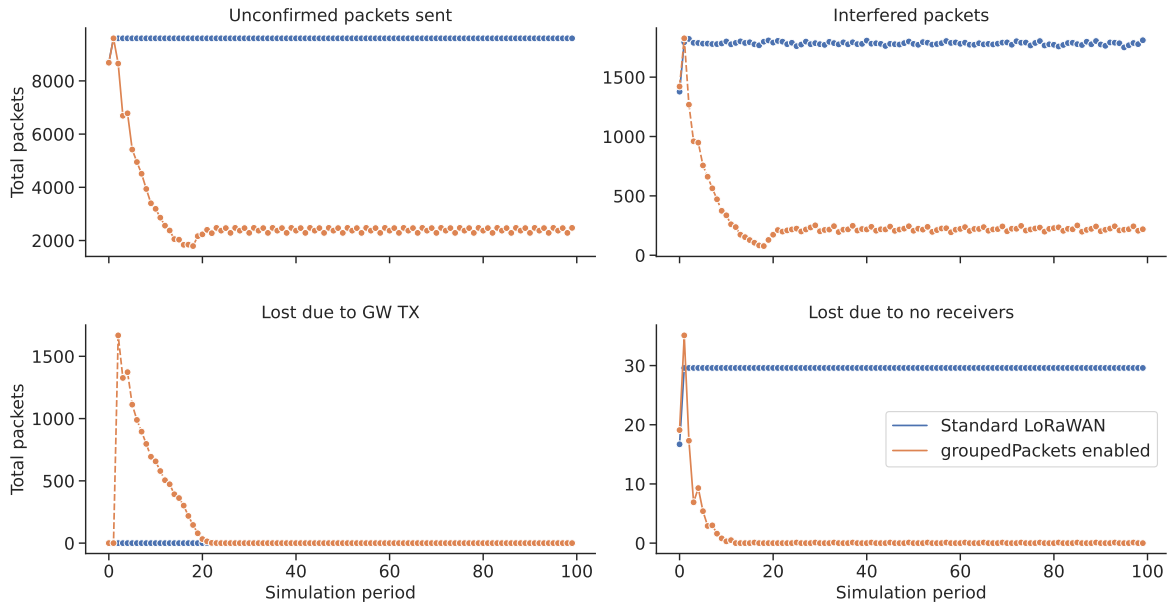


Figure 5.12. Analysing the behaviour of an unconfirmed aggregating groupedPackets in an NbTrans=8 network with $\lambda = 0.702\ pkt/s$.

The graphs show how the algorithm gradually decreases the number of sent unconfirmed packets from $1200 * 8 = 9600$, down to $300 * 8 = 2400$. Thanks to fixed packet sizes, all nodes can be configured to aggregate four payloads, resulting in nodes only sending 43 B payloads (due to the delimiters) every fourth simulation period. Similar to the case for confirmed aggregation, nodes are "out of sync" with their peers, and thus groups of nodes evolve that transmit at different times.

Whilst grouping unconfirmed packets does reduce the number of interfered packets, each of these packets now contains multiple application payloads. All of these would be lost if not for a high NbTrans value, allowing these payloads multiple chances at reception. The number of packets lost due to no more available receivers was also reduced thanks to the aggregation reducing the effective arrival rate in the network.

In the simulation's early phases, many packets are lost due to the gateway transmitting LinkACSReq MAC commands. Thanks to the best-case scenario, no more requests are sent once all nodes are instructed to aggregate four application payloads. Due to the higher number of nodes which require these MAC commands (the entire 1200 versus 180 nodes in a 15 % network performing confirmed aggregation), the algorithm takes much longer to reach a stable state than previously. Figure 5.12 shows how for an arrival rate of 0.702 pkt/s, it takes up to simulation period 20 before less than 50 packets are

lost due to the half-duplex nature of gateways. This duration is caused by the DC restrictions for the gateway limiting the number of aggregation requests it may send in an hour. Each node must receive three aggregation requests, and in a network with an arrival rate of 2.031 pkt/s, the algorithm takes up to period 51 before less than 50 packets are lost due to GW transmissions.

Higher arrival rates cause nodes to send packets more frequently, resulting in faster opportunities for downlink transmissions. Unfortunately, the GW is still hampered by the same DC restrictions as those of lower arrival rates, so sending all of the requests cannot complete any faster. This can be partially alleviated with a frequency plan containing more channels in different sub-bands.

The ULPDR performance achieved for this best case can be seen in Figure 5.13 which contains error bars showing the standard deviation. The first 50 simulation periods were excluded to allow groupedPackets to reach a stable state with metrics calculated over the following 30 periods. From the graph, any improvements offered by groupedPackets are minor at lower arrival rates. A significant standard deviation can be noticed in ULPDR when groupedPackets is used in an NbTrans=1 network at the highest arrival rate. Aggregated payloads are only transmitted once, so any packet lost due to interference contained multiple payloads, impacting the ULPDR calculation proportionally. This can be mitigated by using NbTrans=8 instead, which resulted in the ULPDR being above 98 % due to the multiple transmissions.

Figure 5.13 shows that even where the algorithm outperforms standard LoRaWAN, the margin remains below 5 percentage points and is not enough to validate its use. Any out-performance must be weighed against the long duration to reach the stable state, during which the standard network would perform better.

When one considers using groupedPackets in a more standard network containing 15 % confirmed nodes with nodes transmitting between 12 B and 18 B, a different outcome is achieved compared to the best case. Figure 5.14 shows how groupedPackets unperformed or offered little benefit over standard LoRaWANs for NbTrans=1 and NbTrans=8, respectively. These graphs were generated from simulation periods 50 to 80, so groupedPackets already have had sufficient time to settle. For the NbTrans=1 networks, the presence of confirmed traffic, and their resulting ACKs, causes packets to be continuously lost to due to GW transmissions. As a result, aggregated payloads are lost due to this as well as interference.

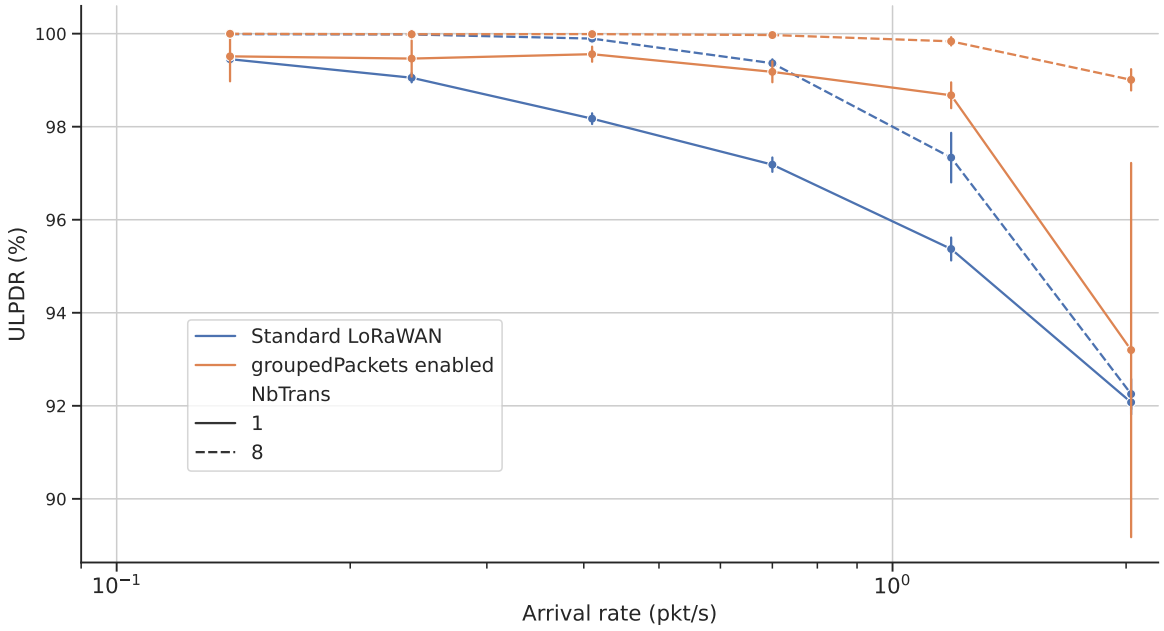


Figure 5.13. Performance of unconfirmed traffic, examined over several arrival rates, in an unconfirmed only network aggregating unconfirmed packets using groupedPackets. Results were gathered from simulation periods 50 to 80.

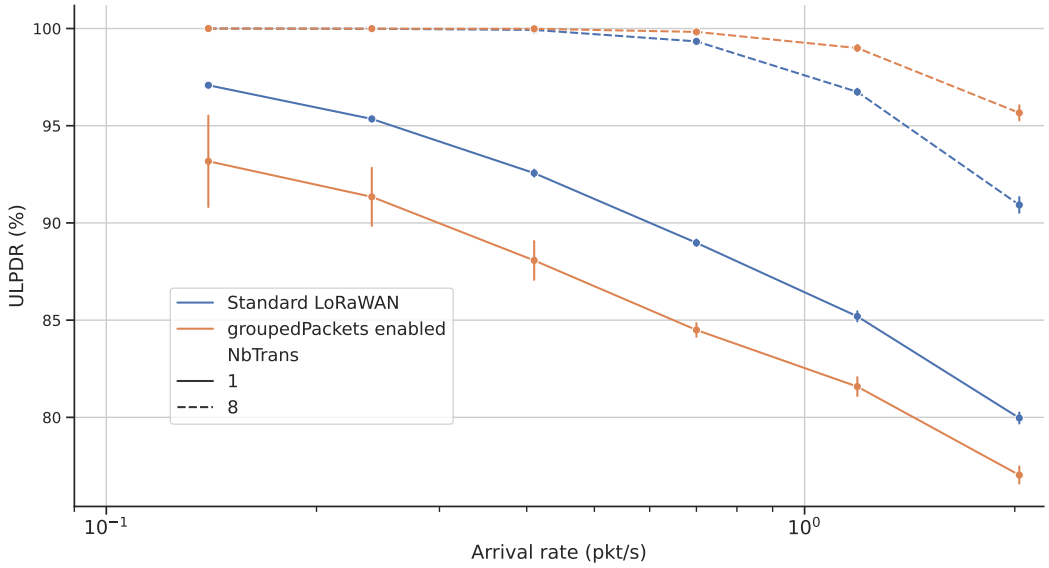


Figure 5.14. Performance of unconfirmed traffic, examined over several arrival rates, in a 15 % confirmed network aggregating unconfirmed packets using groupedPackets. Results were gathered from simulation periods 50 to 80.

Figure 5.15 indicates what occurs to CPSR in the first 30 application periods. During these periods, groupedPackets has not settled yet, and as a result, fewer ACKs can be sent as LinkACSReqs were sent instead. This causes a considerable variation in the CPSR between simulation periods and also causes the standard network to offer better reliability. Once groupedPackets has settled by period 30, it would only send the occasional LinkACSReq (due to the variance in packet sizes). Therefore, the influence of groupedPackets on the transmission of ACKs becomes nearly zero and the CPSR difference between the two networks become minimal.

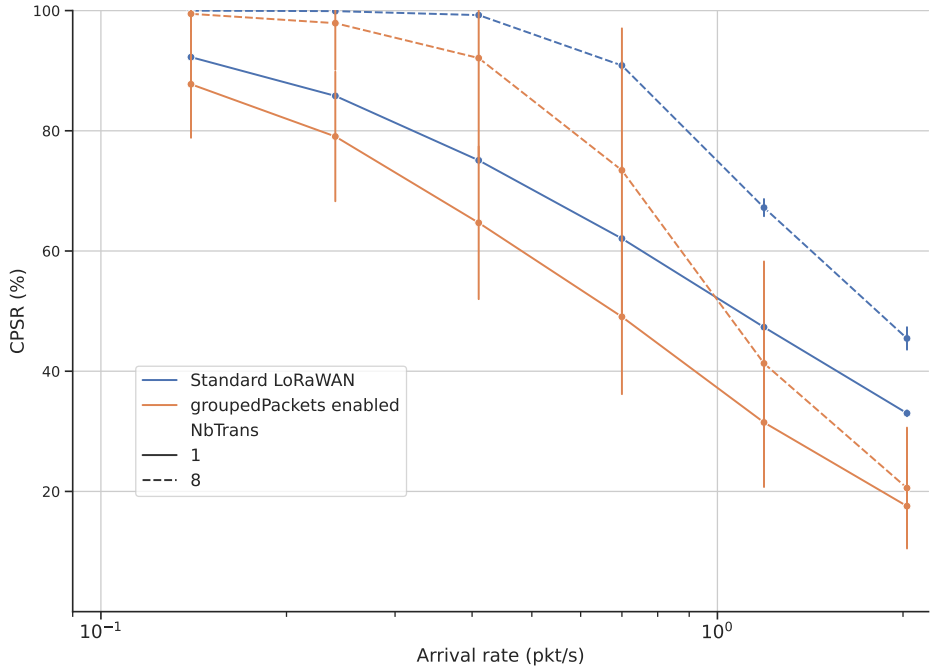


Figure 5.15. Performance of confirmed traffic, examined over several arrival rates, in a 15 % confirmed network aggregating unconfirmed packets using groupedPackets. Results were gathered from simulation periods 0 to 30. X-axis values were shifted slightly to allow the error bars to be seen clearly at each data point.

Overall, the simulations revealed that an unconfirmed packet version of groupedPackets is only beneficial in specific scenarios. It could offer minor ULPDR improvements in networks with only unconfirmed traffic which also have favourable traffic patterns such as fixed small packet sizes.

5.4 CHAPTER SUMMARY

This chapter detailed the performance impact of using groupedPackets in a congested LoRaWAN. The procedure that was followed to ensure valid results was stated before several investigations into the impact of the algorithm were provided. The algorithm was evaluated across a range of confirmed to

unconfirmed node ratios and the impact of NbTrans, various arrival rates and base packet sizes on the algorithm were presented. Finally, the chapter indicated how groupedPackets performs when tasked with aggregating unconfirmed payloads instead of confirmed ones.

To ensure valid simulation results, the algorithm's behaviour was plotted against time in Section 5.2, indicating that the first 20 simulation periods should be excluded from the final results. The algorithm must first gather packet history before requiring several time periods to request adjustments. The inclusion of these periods would skew the results in comparison with standard networks. Thus, simulations should be executed for 50 application periods with results calculated over the final 30 periods and combined with averaging over ten runs. This approach ensures that the stated performance is not dependent on one randomly generated network setup but is rather a generalised representation.

Overall, Section 5.3 showed that the algorithm improves CPSR and ULPR compared to standard networks, with more significant increases in CPSR than ULPDR. The algorithm's performance was further increased when NBtrans was set to eight, allowing aggregated payloads multiple chances at success. The algorithm showcased considerable improvements in CPSR at arrival rates higher than one pkt/s with only minimal impact at lower arrival rates (CPSR at these rates was already above 98 %). The algorithm offered more significant improvements in CPSR at higher confirmation ratios.

Increasing the base packet size from 10 B did reduce the performance improvement offered by groupedPackets as this hampered the number of packets it could aggregate before the 50 B limit was reached. The algorithm will offer a benefit as long as at least two packets can be aggregated, and because it operates on a node-by-node basis, nodes could still benefit depending on their traffic pattern.

The chapter concluded with an investigation into what would occur if groupedPackets were re-tasked to aggregate unconfirmed traffic instead. Simulations found that the algorithm offered a marginal improvement in the base case scenario (fixed payload lengths and no confirmed traffic). In less favourable scenarios which contain confirmed traffic, the algorithm should not be used due to its negative impact on CPSR and the increased time required to reach a stable state.

CHAPTER 6 CONGESTION IN BATTERY-LESS LORAWANS

6.1 CHAPTER OVERVIEW

This chapter details how congestion in battery-less LoRaWANs was studied as well as the outcome of several methods to reduce congestion. Section 6.2 first states the simulation setup and chosen scenario and Section 6.3 provide the procedure followed to ensure valid results. The rationale behind an energy-aware version of groupedPackets is then provided in Section 6.4. Section 6.5 contains the results and indicates the impact of NbTrans and changing the SF of receive window two on the performance of standard and groupedPackets networks. A final summary of the chapter is provided in Section 6.6.

6.2 SIMULATION SETUP AND SCENARIO

The literature survey (Section 2.8) found that whilst LoRaWAN can be used in Green IoT deployments, confirmed traffic increases the transmitting node's energy consumption, and large deployments will thus have to deal with limited energy budgets in addition to the problems caused by congestion. To better understand the problems faced by such nodes, the use of groupedPackets was explored as a possible solution.

The updated version of "lorawan" with support for battery-less capacitor-based nodes ([93]) was used to investigate these types of networks. In this version, nodes no longer have an infinite battery source but are modelled as a battery-less device consisting of a microcontroller, radio, sensors, a supercapacitor and a harvesting mechanism based on the approach followed in [92]. Full support for the groupedPackets algorithm and NbTrans was then added to this version.

As defined in [93], nodes had several operational states, each with a corresponding current consumption

as given in Table 6.1. Except for turning on, the radio consumes the bulk of all current compared to other circuit elements. The energy consumed by any sensors was considered negligible compared to the radio and microcontroller. The radio transmitting a packet requires the most current, followed by receiving data and the standby state, in which the radio is listing to the idle channel during the receive windows.

Table 6.1. Current consumption in the different states. The current consumption due to MCU is 11 μA in the active state, 5.5 μA in the standby state. Taken from [93], with permission.

State	MCU	Radio current	Total current
Off	Standby	0	5.5 μA
Turn on	Active	-	15 mA
Sleep	Active	1 μA	5.6 μA
Tx	Active	28 mA	20.011 mA
Idle	Standby	1.5 μA	7 μA
Standby	Standby	10.5 mA	10.5055 mA
Rx	Active	11 mA	11.011 mA

For the scenario, a similar approach was taken as that of [92]. A generic harvesting system model was used rather than a specific one based on the type of energy source. To provide stable output, it is assumed that the energy source is followed by a voltage regulator to supply a constant voltage and is thus modelled as shown in Figure 6.1. The series resistance r_i limits the output power of the source, and an ideal supercapacitor with no internal resistances was used. The impact of leakage current was studied in [92], which found that an ideal capacitor model can be used as the impact of capacitor self-discharge on results was minimal for the small capacitor values used for IoT devices. Simulations were thus conducted with an ideal harvester delivering a constant ten mW.

The chosen simulation scenario is similar to the one described in Section 4.2.1. It consists of a single gateway supporting 1200 nodes uniformly spread out in a circle with a radius $r = 500\ m$. The EU868 channel plan was also followed with its three default channels, and nodes generated fixed application payloads of 10 bytes every 591 seconds. All nodes were configured to use data rate five (SF7 with a bandwidth of 125 kHz); this results in a congested network with a traffic volume of $\lambda = 2.031$ packets per second.

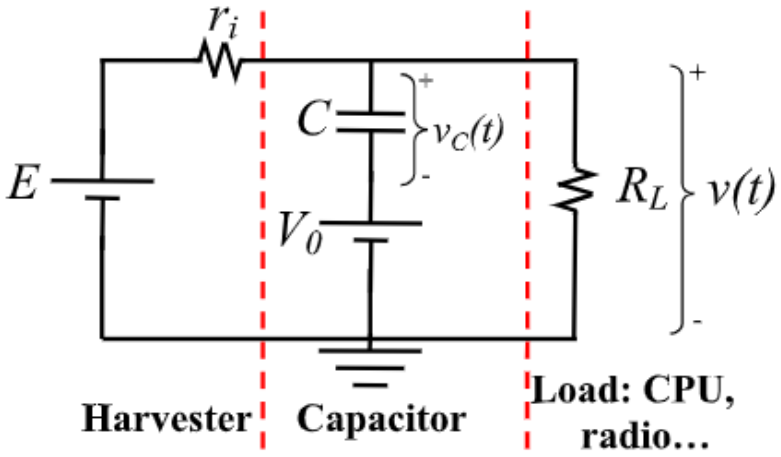


Figure 6.1. Electrical circuit model of a battery-less IoT device using a voltage source energy harvester and an ideal capacitor. Taken from [92], © 2021 IEEE.

Unlike the previous simulations, all nodes use the same high data rate to minimise the energy consumed by transmissions and the radius was also reduced to ensure all nodes are within communication range.

To ensure congestion, 15 % of devices were selected to send only confirmed traffic whilst the rest sent unconfirmed traffic only. Nodes sleep until they are awakened periodically with a new application data packet to send. Nodes will attempt to send this data NbTrans times unless this process is interrupted by a new application data packet or they received an ACK or ran out of energy. Furthermore, nodes have a “smart" sender (as detailed in [93]), which will only transmit a packet if the predicted energy cost of that packet will not cause the operating voltage to drop below minimum levels. This is to ensure that the node does not switch off mid-transmission and will predict the cost based on the worst-case scenario (e.g. ACK only being received in the second receive window with SF12). Should the first attempt to send a packet not have enough energy, the node will not attempt to transmit it again but instead returns to a sleep state until the next period. For groupedPackets enabled simulations, the “smart" sender was also used, with nodes opting to transmit only the latest payload should the aggregated payload fail the energy cost check.

Four performance metrics were identified to be of interest, with the first two focusing on reliability. The first two are very similar to the ones defined in Section 3.4.2, except that they are now calculated using generated packets and not transmitted packets, as a lack of energy could have prevented packet

transmission. In the previous chapter, this distinction would not have been necessary, but this is no longer necessarily the case.

These two can thus be redefined as the following.

- CPSR: the probability that a generated confirmed UL packet was transmitted, received by a GW and the corresponding ACK was received by the node.
- ULPDR: the probability that a generated UL unconfirmed packet was transmitted and correctly received.

The main reasons a packet would be recorded as a failure with these two metrics are insufficient energy to complete the entire transmission cycle, interference, the gateway reaching its maximum simultaneous receive limit (8) and active gateway transmissions preventing any UL packets from being received. Finally, in CPSR's case, the ACK could have never been received because of, e.g. DC restrictions preventing its transmission.

The final two metrics focused on traffic volume and energy. The first is **Total Packets Sent (TPS)**: the total amount of UL packets that were successfully sent in the network and does not distinguish between confirmed and unconfirmed traffic. Finally, **Total Packets not Sent (TPNS)**: the total amount of times an UL packet could not be sent due to a lack of energy preventing its transmission. TPNS will be calculated separately for unconfirmed and confirmed packets. The ideal value for this metric would be zero. Additionally, in networks with NbTrans values higher than one, an unconfirmed packet can be recorded as being successfully received (ULPDR) whilst also being recorded under TPNS if, for example, the third transmission of a packet could not be sent but the first/second transmissions were successfully received. Confirmed packets will be recorded under TPNS if the first attempt did not have sufficient energy or if subsequent retransmissions (due to an ACK not being received) did not have enough sufficient energy. It should be noted that confirmed traffic requires more energy as nodes need to complete the full UL and DL cycles [93].

6.3 ENSURING VALID RESULTS

6.3.1 Choosing appropriate simulation periods

The process to ensure valid results for battery-less simulations can be aided by the lessons learnt from the previous chapters. Section 3.4.4 revealed that the performance of standard LoRaWAN does not

significantly change when the number of application periods is increased and one would expect the same behaviour in standard battery-less simulations assuming that the supercapacitor is fully charged at the end of each application period (it does not slowly decline over time). Section 5.2 indicated that groupedPackets require 20 application periods to reach a stable state and Section 5.3.2 showed that at high arrival rates (such as the 2.031 pkt/s used for battery-less simulations), confirmed nodes become “out of sync" with one another. At much lower arrival rates, all nodes would have received their aggregation requests in the same simulation period and the transmission pattern would have been no transmission for three periods followed by all nodes transmitting in the fourth period.

Figure 6.2 shows an example of what can occur in the battery-less simulations in which transmissions occur in groups rather than all nodes in the same periods. In this simulation, groupedPackets requested that nodes aggregate four payloads and four groups were formed which transmit aggregated packets in different periods. Perfect aggregation would have been zero transmissions for three periods followed by $4\ packets * 180\ nodes = 720\ packets$. Figure 6.2 shows that, instead, most nodes are in group four as this group resulted in ≈ 400 of the 720 packets with the rest divided into the three remaining groups. In the graph, the number of generated confirmed application payloads is shown to be a steady 180 packets (15 % of nodes in a 1200 node simulation were confirmed nodes). What can be learnt from this graph is that should metrics be calculated over one less period than what is shown, group 4’s transmission at period 31 will be missed completely whilst generated packets would simply be reduced by 180. This would result in a large change in the CPSR calculation because packets that have been generated for several periods would only have been sent in the next period, which was now excluded. Care should be taken to ensure that this does not occur!

Valid results for groupedPackets simulations in Section 5.2, examined the algorithm over what was effectively 30, 50 and 70 application periods (the first 20 were excluded). Exploratory simulations revealed that in the case of battery-less simulations, this needs to change to 32, 56 and 80 (the first 20 will still be excluded).

The period used to calculate performance metrics needs to be divisible by two, three and four due to the change in how CPSR is calculated. CPSR now uses generated packets as the denominator (previously it was transmitted packets) and groupedPackets’ aggregation will result in similar behaviour as shown in Figure 6.2. By using a period divisible by all possible aggregation values, the CPSR calculation can be accurate as all generated packets in the period would have been sent (assuming sufficient energy).

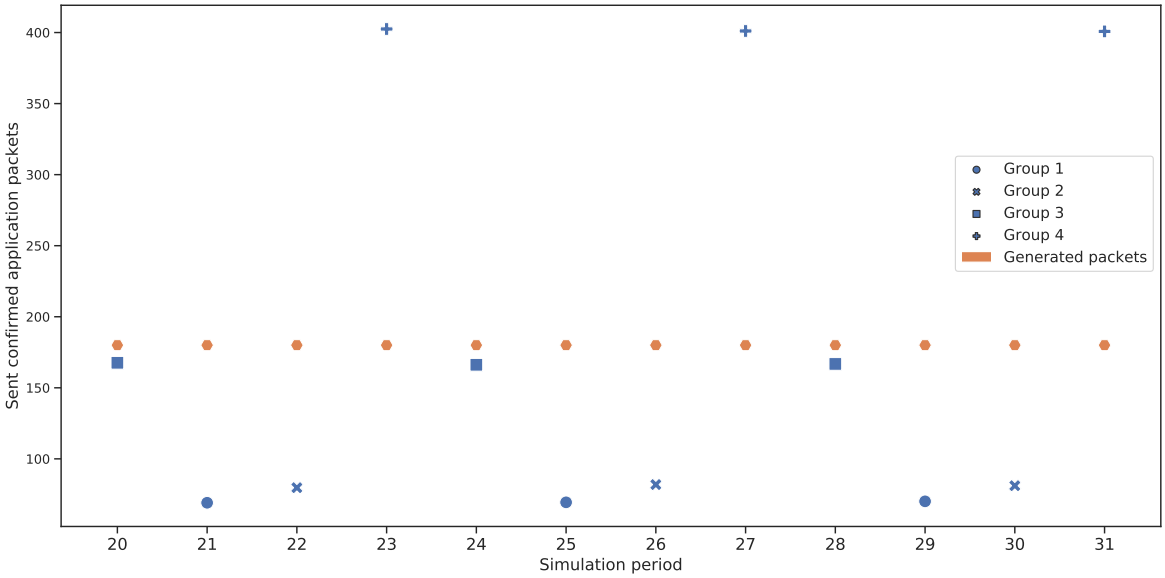


Figure 6.2. Impact of confirmed nodes being “out of sync" when groupedPackets is used.

Using a period resulting in a remainder will cause CPSR values to be calculated incorrectly with results being skewed not because of unwanted packet outcomes but instead because generated packets have not been sent yet. For the previous chapter, transmitted packets were used in the CPSR calculation, so any generated but not yet sent packets in the previous simulations did not influence the calculations.

6.3.2 Simulation length and number of runs

As was the case for the previous chapters, experimental simulations were run to ensure suitable values were chosen from the simulation length and the number of runs. Figure 6.3 shows how once again the average time per simulation increases when longer periods are used (values used are after the initial 20 application periods). Battery-less simulations take a lot longer than previous simulations due to all of the additional computations performed to constantly update the capacitor’s charge.

Figures 6.4 and 6.5 show how the mean of both performance metrics changes as both simulation length and the number of runs increases in groupedPackets enabled battery-less networks. Figures 6.6 and 6.7 show the same but for the case of standard LoRaWAN in battery-less networks. Similar conclusions can be taken from these graphs than previous investigations and an approach averaging over ten simulations with each simulation taking 36 application periods would be acceptable. This 36 would be after the initial 20 to allow groupedPackets to settle and the same approach was taken for standard LoRaWAN battery-less simulations.

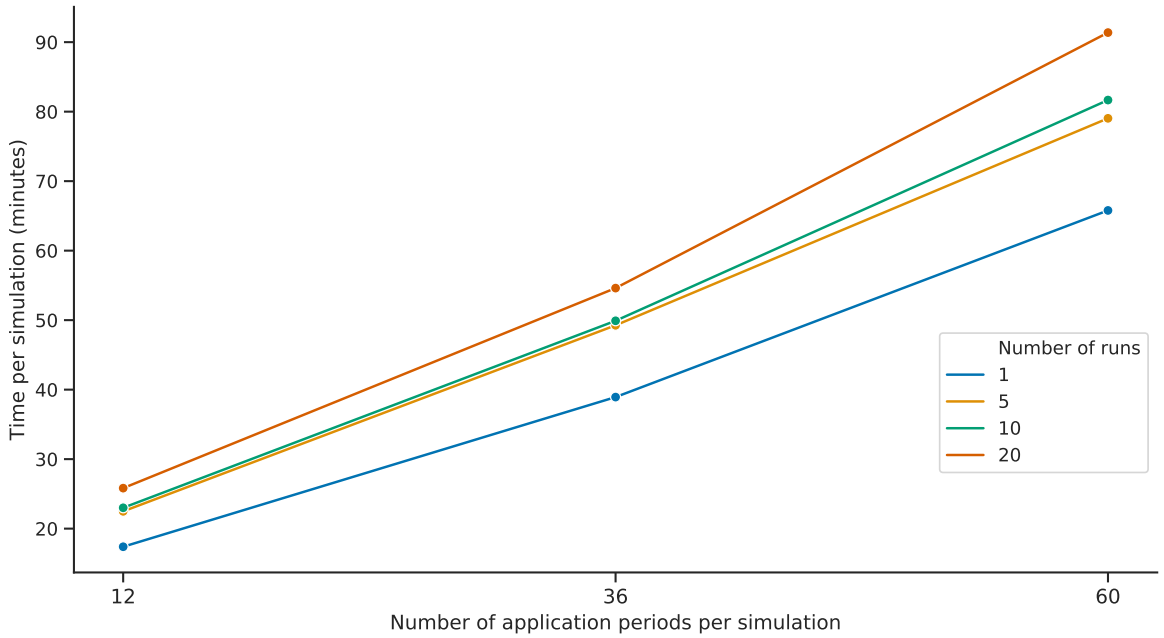


Figure 6.3. Average time it takes to complete one simulation of a groupedPackets enabled battery-less network using a capacitance of 6.5 mF given a scenario of 1200 devices, 15 % sending confirmed traffic, an NbTrans value of five and all nodes generating 10 B packets every 591 s.

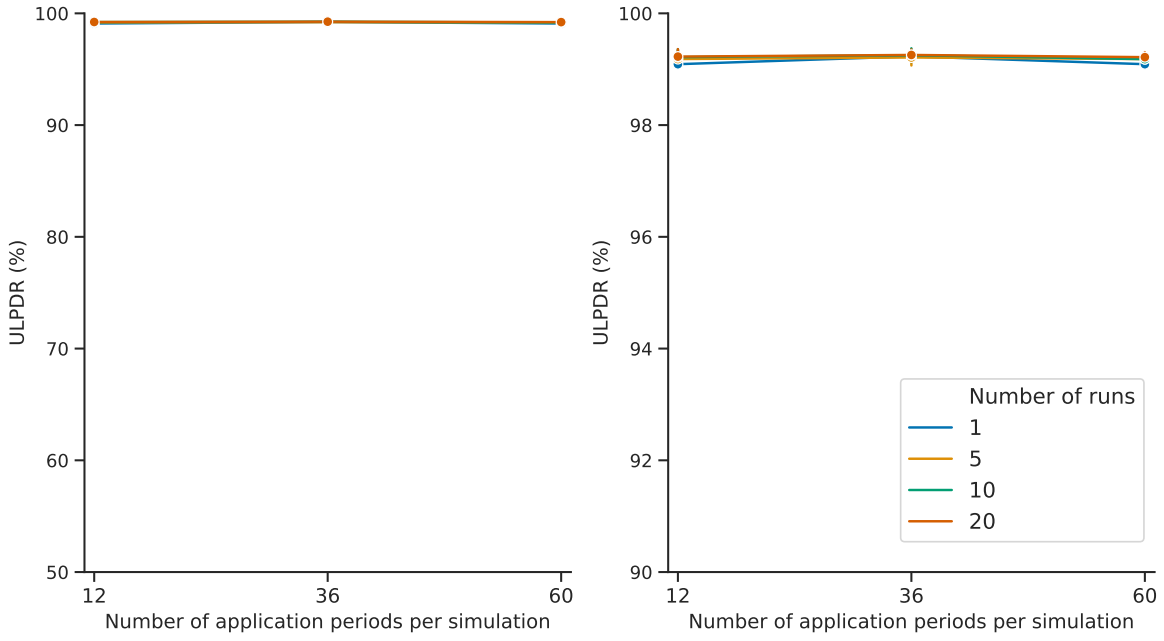


Figure 6.4. The mean and standard deviation of the ULPDR values in groupedPackets battery-less networks.

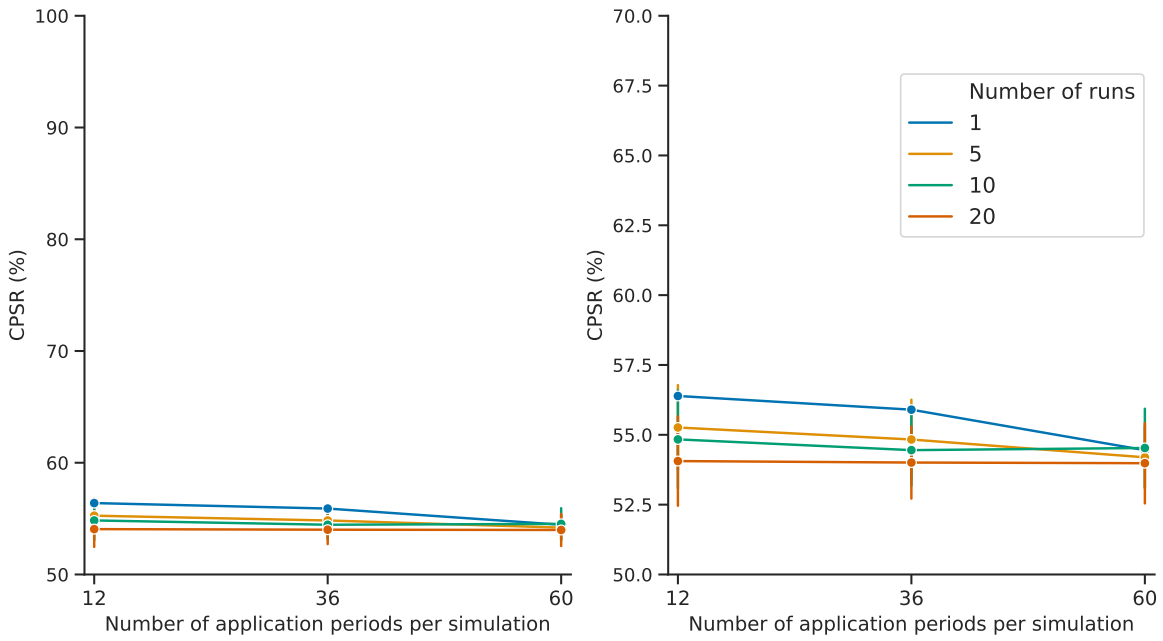


Figure 6.5. The mean and standard deviation of the CPSR values in groupedPackets enabled battery-less networks.

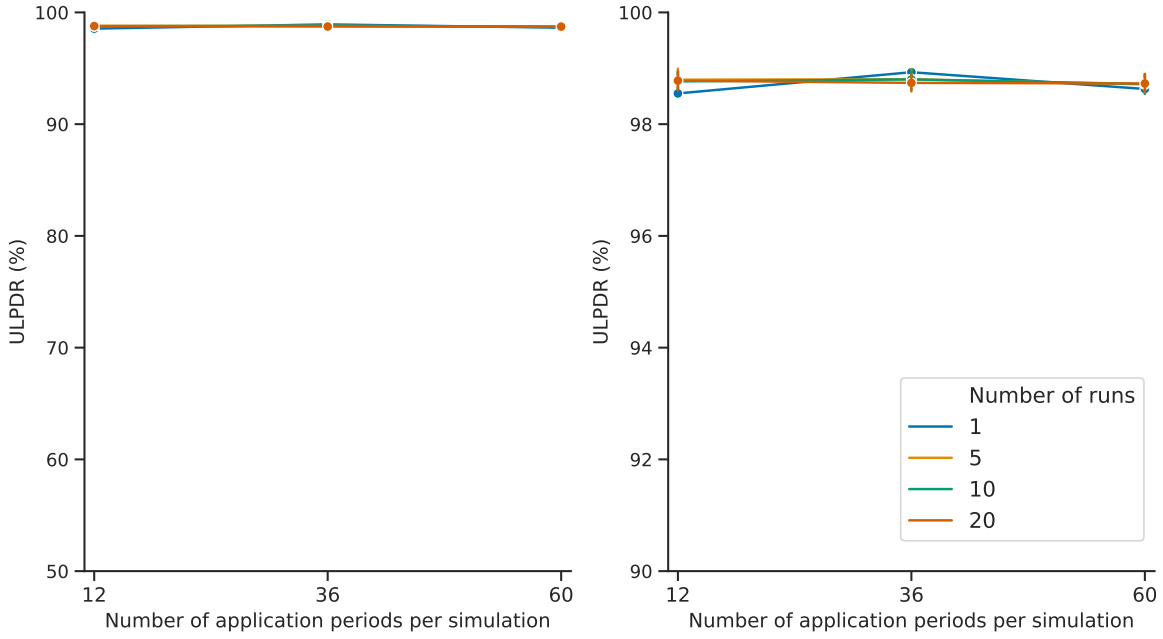


Figure 6.6. The mean and standard deviation of ULPDR in standard battery-less networks.

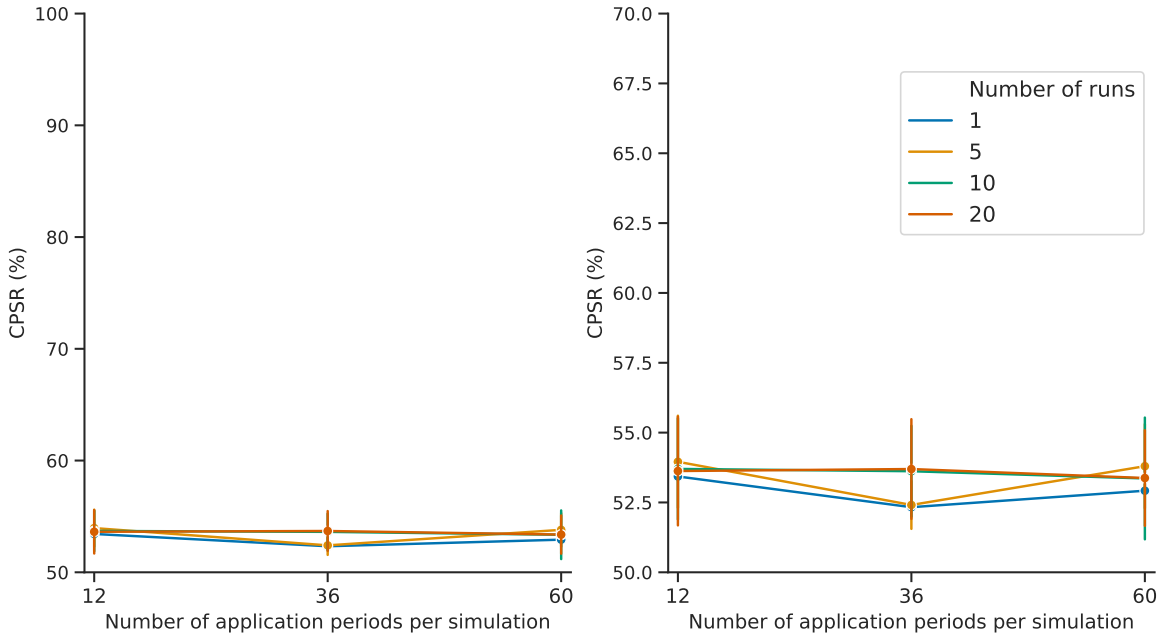


Figure 6.7. The mean and standard deviation of CPSR in standard battery-less networks.

6.4 CREATING AN ENERGY-AWARE GROUPEDPACKETS ALGORITHM

This section starts with the problems that occur when the standard groupedPackets algorithm is used in battery-less networks. This is followed with the technical details of an improved version. The results of using this improved version and other LoRaWAN parameters in battery-less networks are also presented.

6.4.1 The problem with standard groupedPackets

Figures 6.8 and 6.9 show the impact of enabling the standard and improved groupedPackets algorithms in a battery-less network configured with NbTrans=1. The minimum required capacitance for confirmed nodes is shown in Figure 6.10. By aggregating application packets from confirmed devices, both ULPDR and CPSR can be improved. For standard groupedPackets, aggregation requires a one mF increase in the minimum capacitance that nodes must have due to the increase in sent confirmed traffic's packet sizes versus those of standard networks. The improved version of this algorithm eliminates this capacitance increase requirement. When examining these figures, three behaviour zones can be identified. The first is simulations where the capacitance is less than six mF, the second is values between six mF and approximately seven mF and finally, simulations where the capacitance exceeds seven mF.

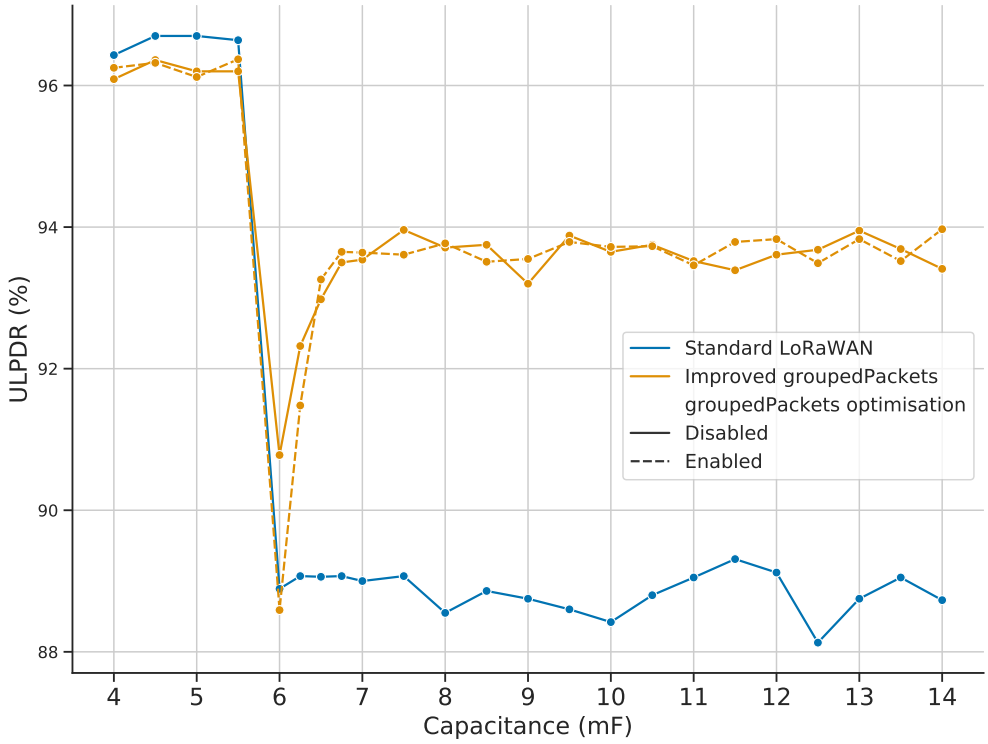


Figure 6.8. Performance of unconfirmed traffic when groupedPackets is used in an NbTrans=1 battery-less network.

In the first zone, the 180 confirmed nodes do not have enough energy to complete the full UL and DL cycle. As a result, no confirmed packets are sent, and ULPDR is high due to the lower traffic, whilst TPNS indicates that no confirmed packets could be sent ($180\ nodes \times 36\ packets = 6\,480$).

For the second zone, a value of six mF or more allows confirmed nodes in the case of standard LoRaWAN to transmit the 10-byte payloads without issue. In the case of standard groupedPackets, the zone represents a “transition" range of capacitance values where there is enough energy to send some of the smaller aggregated payloads but not enough for the optimum aggregation payload. This causes larger aggregated packets to have insufficient energy, and nodes then only send the latest packet (losing the other aggregated payloads in the process). As capacitance increases, larger and larger payloads can be sent, until the algorithm’s maximum is reached just below seven mF. This behaviour can also be seen in Figure 6.10, where TPNS decreases as capacitance increases.

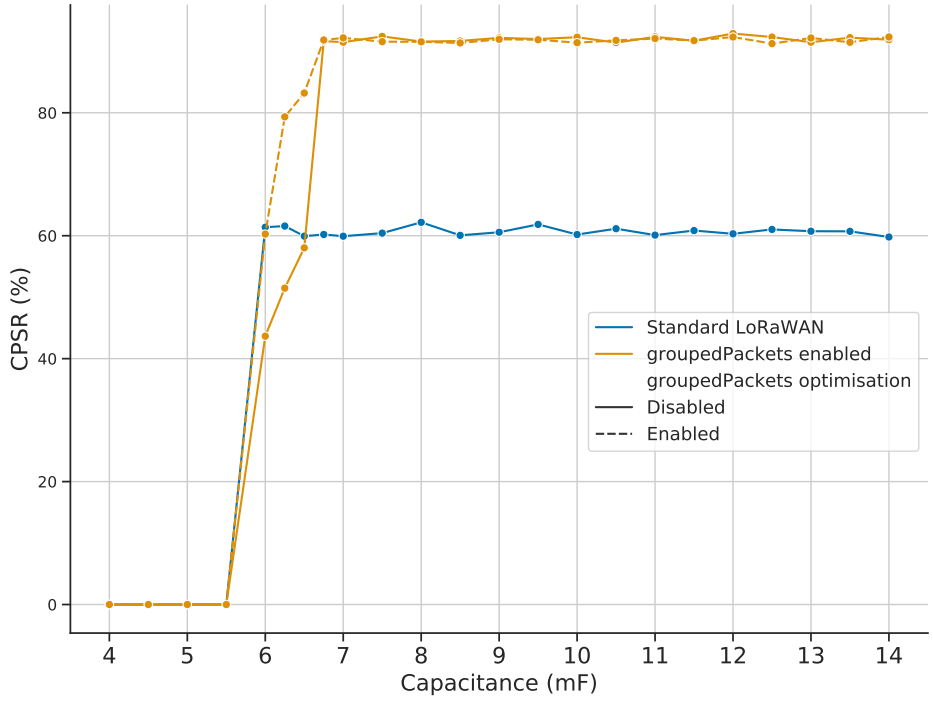


Figure 6.9. Performance of confirmed traffic when groupedPackets is used in an NbTrans=1 battery-less network.

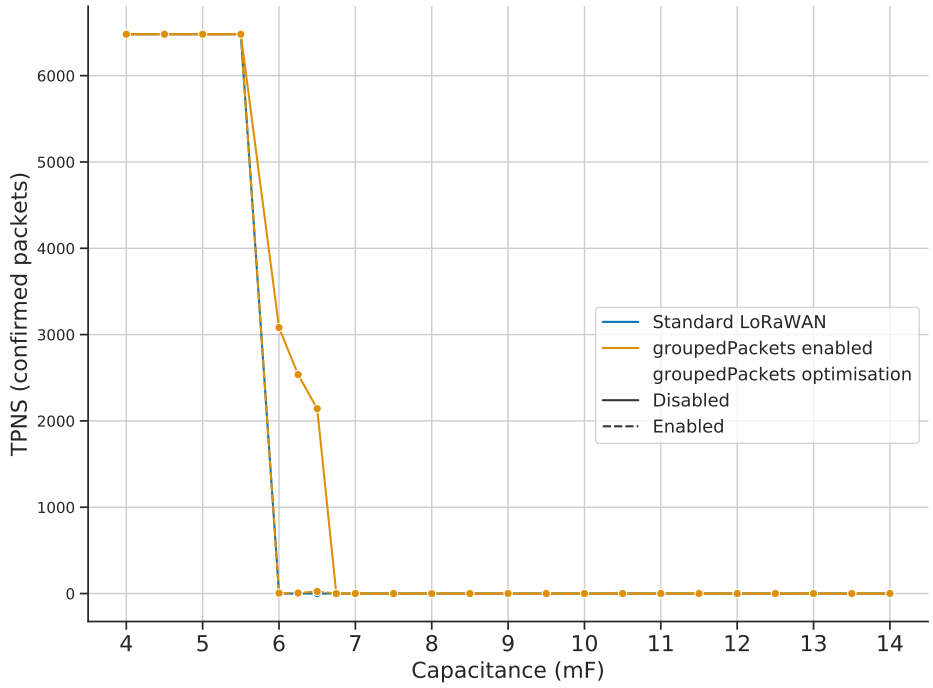


Figure 6.10. Total confirmed packets not sent due to energy constraints in an NbTrans=1 network, examined over several capacitor sizes (all unconfirmed packets could be sent). The case of standard LoRaWAN and the optimised version of groupedPackets overlap visually and both drop to 0 at 6 mF.

This result highlights that the groupedPackets algorithm would benefit from expansion to become energy aware as only sending the latest packet is not ideal. Furthermore, by only sending the latest packet, the NS side of the algorithm will send requests to increase the aggregation, as it is unaware that the optimal value is not possible. This has resulted in the development of the improved version, and Figure 6.9 shows how this version improves CPSR in this transition range.

Finally, in the third zone (just below seven mF and more), all networks have enough energy to transmit their desired payloads and performance is not influenced by capacitor sizing. The minor variations in CPSR and ULPDR are because simulations have a random element. Nodes randomly pick a time to start transmitting in the range of 0 to 591 seconds and as a result, the amount of transmission overlap between devices is slightly different in each simulation, which impacts performance.

6.4.2 Specifications of the improved groupedPackets

In energy-constrained LoRaWANs, the maximum packet size that can be successfully sent is no longer only SF dependent but also depends on the available energy. Thus, the groupedPackets algorithm had to be updated to consider a node's available energy. The algorithm is split into two parts; one part executes on the NS and determines the aggregation target for each node, whilst the other executes on the node and performs the aggregation.

Application payloads are not always fixed in size, and thus the node side component of the algorithm will transmit a re-aggregated packet containing as many as possible aggregated payloads should the latest application payload cause it to be unable to transmit all payloads. As described in Section 4.4, the standard groupedPackets algorithm will aggregate payloads whilst the total payload does not exceed 50 bytes. An updated version, referred to as "improved groupedPackets", also evaluates if there is sufficient energy to transmit the desired payload. This is done by calculating the predicted energy cost of the transmission (UL and DL) which assumes the worst-case scenario (e.g. window two being required to receive an ACK). Should there be insufficient energy, a new payload is formed by re-aggregating payloads and assessing the energy cost after each aggregation. Finally, a LinkACSAns MAC command is created to indicate that the aggregation target supplied by the NS cannot be met due to energy constraints. The pseudocode for this is shown in Algorithm 3.

The NS part of the algorithm was modified to monitor for any received LinkACSAns commands, and should a node indicate that the aggregation target is unfeasible, it is added to a list of such nodes. These

Algorithm 3 Improved node side groupedPackets.

INPUT: $totalPSize$: Total application packet size due to aggregation + latest payload;
$latestPSize$: The size of the latest payload;
$latestP$: The latest payload;
$storedData$: Older payloads stored with delimiters;

1: $enoughE \leftarrow CheckE(totalPSize)$
2: **if** $totalPSize \leq 50$ and $enoughE = true$ **then**
3: Update MAC header
4: $sentData \leftarrow (storedData + latestP)$ ▷ Send packet
5: **else if** $totalPSize > 50$ and $enoughE = true$ **then**
6: Repeat aggregation process and monitor size
7: Update MAC header with new number of payloads
8: $sentData \leftarrow (newData)$ ▷ Send smaller packet
9: **else if** $totalPSize \leq 50$ and $enoughE = false$ **then**
10: $newnumPackets \leftarrow 0$ ▷ 0 = 1 payload
11: $newTotalSize \leftarrow latestPSize$
12: $eCheck \leftarrow CheckE(latestPSize)$
13: $newData \leftarrow latestP$
14: **for all** p in $storedData$ **do**
15: $newTotalSize = newTotalSize + size(p)$
16: $eCheck \leftarrow CheckE(newTotalSize)$
17: **if** $newTotalSize \leq 50$ and $eCheck = true$ **then**
18: $newnumPackets \leftarrow newnumPackets + 1$
19: $newData = newData + p$
20: **else**
21: $newTotalSize = newTotalSize - size(p)$
22: **end if**
23: **end for**
24: Update MAC header with new number of payloads
25: Create a LinkACSAns command
26: $sentData \leftarrow (newData)$ ▷ Energy conscious packet
27: **end if**

nodes will no longer receive any LinkACSReq MAC commands, and the node is entirely in control of payload aggregation. Should a node's energy situation improve, it transmits a LinkACSAns command indicating that it wishes the NS to resume controlling its aggregation.

6.5 RESULTS

6.5.1 Impact of the NbTrans parameter

Increasing NbTrans from one to five will result in the 1020 unconfirmed nodes sending an expected total of $1020 \times 36 \times 5 = 183\,600$ packets in a simulation (assuming sufficient energy). The 180 confirmed nodes are responsible for a minimum of $180 * 36 = 6\,480$ packets, with the ability to transmit up to five times that amount should ACKs not be received for earlier transmissions.

Figure 6.11 shows that at four mF only 36 720 packets were sent (one per unconfirmed node). Increasing the capacitance to 4.5 mF still prevents confirmed packets from being sent but allows 73 440 packets to be sent in standard networks (this is two packets per unconfirmed node). Unconfirmed nodes do not require large capacitors to accommodate NbTrans=5, and at 5.5 mF all unconfirmed nodes transmit five copies of each packet (183 600 in total, as shown in Figure 6.11). Figure 6.12 confirms this and shows how at four mF all unconfirmed transmissions were recorded that they were unable to transmit the targeted five times. Similarly, at 5.5 mF, no unconfirmed packets indicated that they could not be transmitted five times.

After the minimum capacitance to send confirmed packets has been achieved (six mF), Figure 6.13 shows that in standard networks, the TPNS for confirmed packets drops in NbTrans=5 networks but does not drop to zero as was the case in NbTrans=1 networks. Additionally, the TPNS for unconfirmed packets rises at six mF.

The reason for this is that due to the increase in traffic, caused by the introduction of confirmed packets and unconfirmed devices transmitting multiple copies, the probability increased that a packet is being transmitted whilst a node has one of the two receive windows open. When this occurs, the device increases energy consumption as it is no longer in standby mode (listening to an idle channel) but actively receiving data. This data is likely not meant for the device, as the source is UL transmission or an ACK meant for a specific device, but the node will not know this until it has processed the frame and checked if the address field matches the node's address [58].

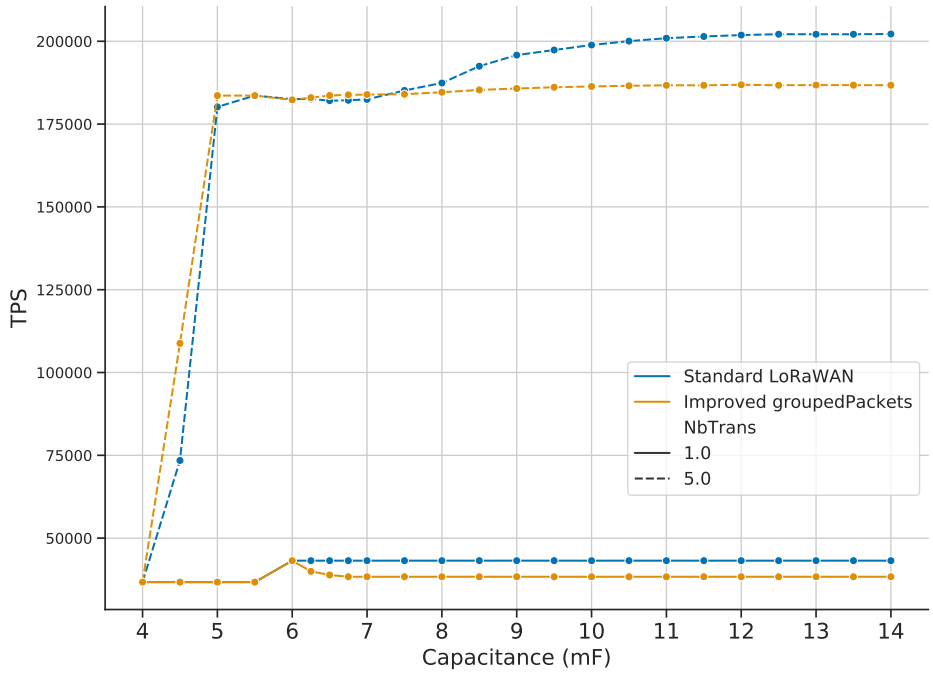


Figure 6.11. Total packets sent in networks with NbTrans values of one and five, examined over several capacitor sizes.

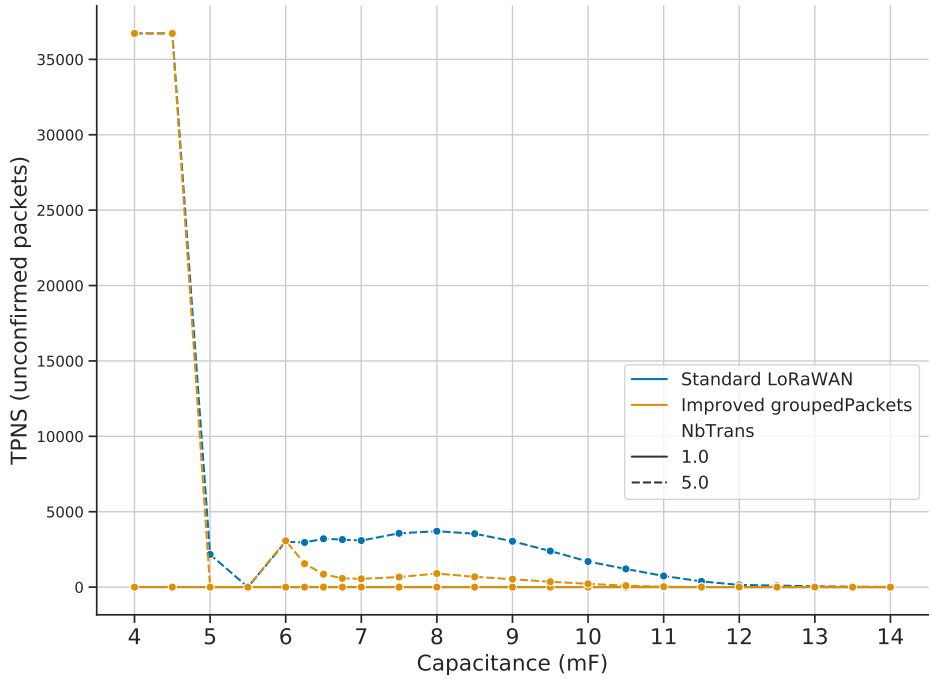


Figure 6.12. Minimum capacitance needed to ensure all unconfirmed packets can be sent, examined over several capacitor sizes.

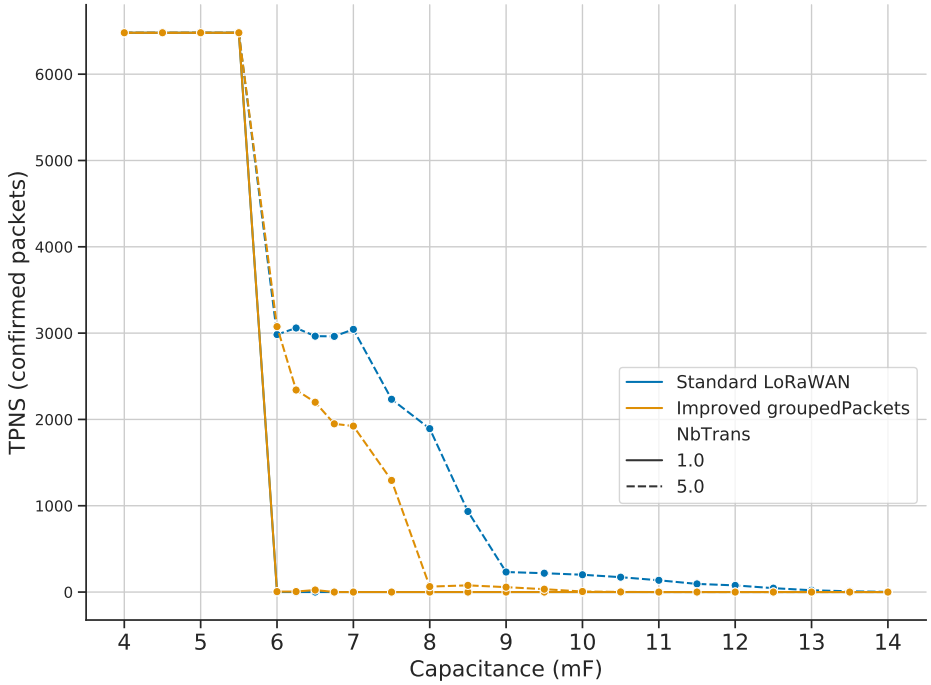


Figure 6.13. Minimum capacitance needed to ensure all confirmed packets can be sent, examined over several capacitor sizes.

As a result, nodes need larger capacitors in NbTrans=5 networks as they need sufficient energy for multiple transmissions of their packets and to also accommodate the “wasted“ energy due to the increase in receiver activity. There is also a snowball effect, in which more capacitance allows nodes to transmit more, increasing the probability of this overlap occurring due to nodes opening more receive windows. Once the capacitance is large enough so that up five transmissions can be done, the performance stabilises, and increasing capacitance no longer changes performance. For standard networks, 99 % of packets can be sent once capacitance reaches 12 mF, with Figure 6.11 showing that the total sent packets stabilise from this value.

A value of 12 mF is not strictly required, as lower capacitance values can still achieve similar ULPDR and CPSR (Figures 6.14 and 6.15). A value of 8.5 mF results in a similar performance for standard networks, and whilst increasing capacitance further will allow for more retransmissions; performance does not improve further as interference and packets lost due to gateway transmissions also increase. Especially in the case of ULPDR, most packets will have been received successfully before five attempts, so not being able to transmit a packet for a fourth/fifth time does not impact reliability.

Examining these graphs within the context of enabling the improved groupedPackets algorithm reveals the following. Firstly, the impact of groupedPackets aggregation can also be seen in the total packets sent in the network (Figure 6.11), with the algorithm decreasing the traffic volume due to fewer confirmed packets being sent. A point of interest would be 4.5 mF, where groupedPackets allowed nearly all unconfirmed nodes to transmit three copies instead of two. This is due to the increase in packet header sizes (to accommodate groupedPackets), causing nodes to sleep longer than in standard networks. The longer sleep duration, caused by the increased ToA of the 1-byte larger packets, allowed nodes to harvest just enough energy for a third transmission.

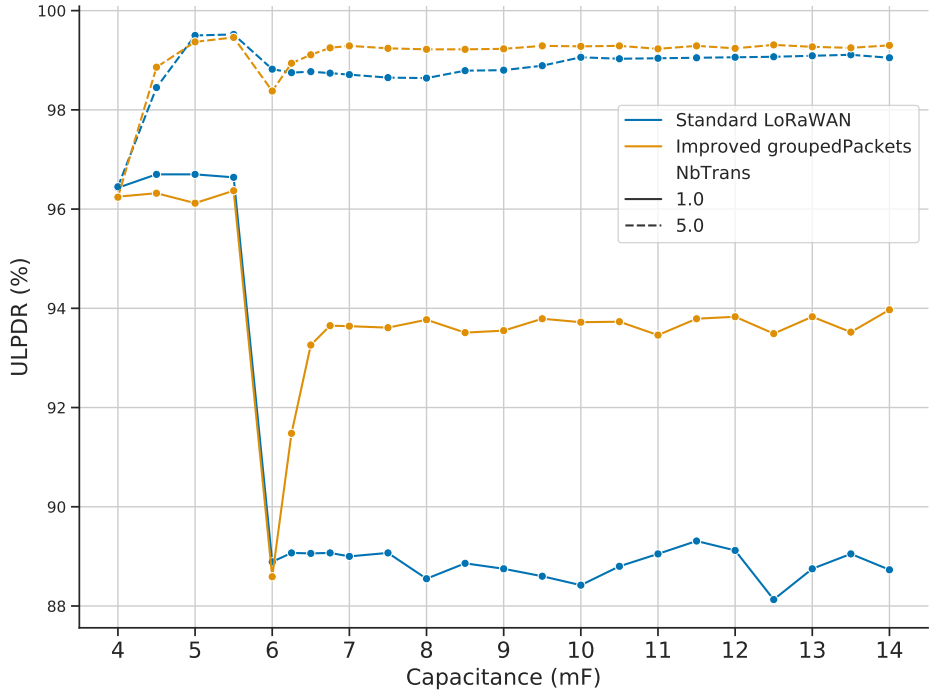


Figure 6.14. Performance impact of NbTrans on ULPDR in the network, examined over several capacitor sizes.

Overall, Figures 6.14 and 6.15 show how effective increasing the number of transmissions from one (the default) to five can be for improving packet delivery. The increase boosts ULPDR and CPSR in standard LoRaWANs and also further improves them in groupedPackets-enabled networks. Examining Figure 6.15 more closely reveals that a “transition" range is visible in CPSR values for standard as well improved groupedPackets networks. In this range, NbTrans=5 underperforms when compared to NbTrans=1. This range starts at six mF and ends at eight mF for groupedPackets and nine mF for standard networks, respectively. Confirmed nodes in this range do not have enough

capacitance for many retransmissions, but their packets suffer from unconfirmed nodes' increased traffic volumes.

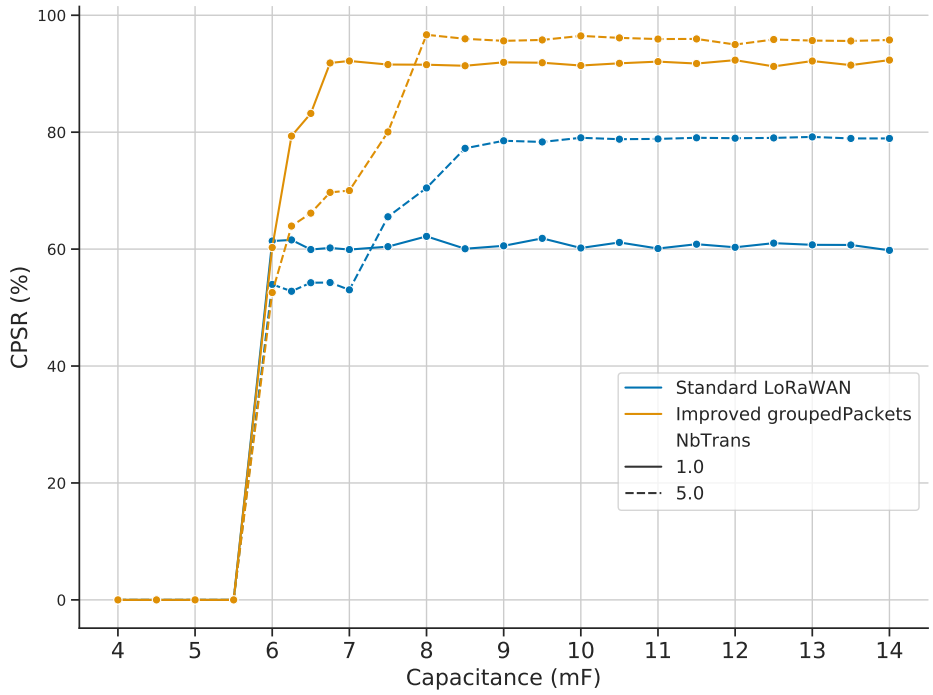


Figure 6.15. Performance impact of NbTrans on CPSR in the network, examined over several capacitor sizes.

In standard networks, once capacitance has reached 7.5 mF, confirmed nodes can do a second transmission, significantly increasing the probability of success. For groupedPackets, at least eight mF is required to beat the performance of NbTrans=1. This increase is because larger packets are being sent due to the aggregation; should one of these not be received (due to interference) and a node cannot retransmit this packet, several application data packets are lost.

The NbTrans setting is configurable by the LoRaWAN network administrator through a MAC command and could thus change over a network's lifetime. Thus, it is essential to not finalise the capacitance value in a design without considering that you might increase a network's NbTrans value in the future.

Regarding a node's behaviour, currently, retransmission attempts are made as soon as DC restrictions allow. For energy constraint nodes, this behaviour is not optimal and will potentially drain the node's capacitor. An alternative method would be to rather wait a period to allow for enough energy harvesting

to occur and then proceed with the retransmission. Waiting too long will, however, jeopardise the energy reserves for the subsequently generated data should transmissions be periodic (as was the case with these simulations). This area can be further explored, with a good starting point being the work conducted in [94].

6.5.2 Impact of changing receive window two's SF

Figures 6.16, 6.17, 6.18 and 6.19 show the performance impact of changing the SF used for the second receive window from the default (SF12) to the same SF used in window one. By matching window two's SF to the one used for window one (SF7 for all simulations), gateways spend the minimum amount of time transmitting compared to using SF12 to transmit ACKs. This allows confirmed nodes to increase their sleep duration and reduces the number of packets lost due to gateway transmissions. For standard LoRaWANs, changing window two's operation results in significant CPSR increases for both NbTrans values.

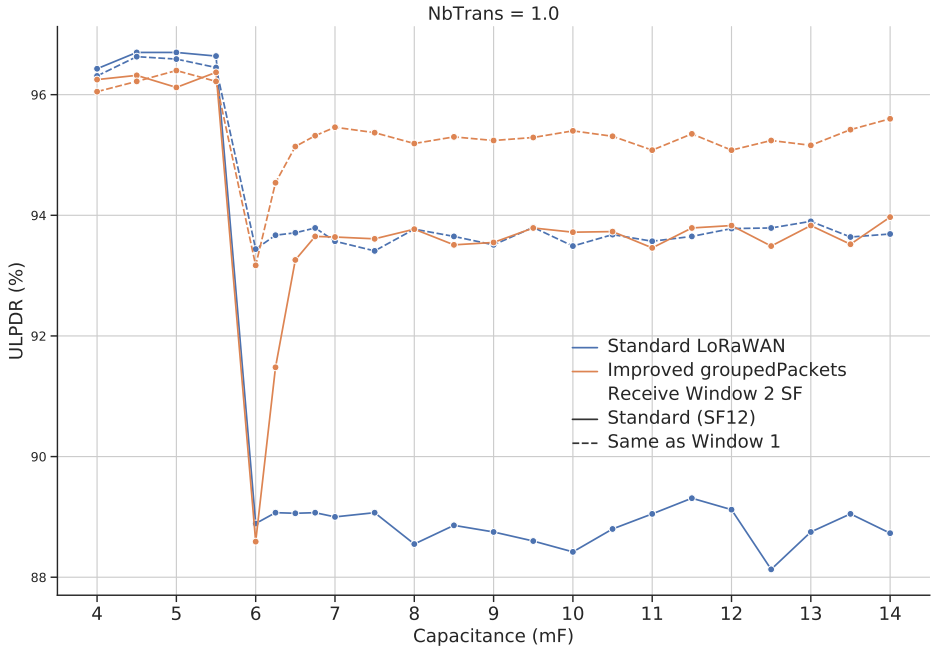


Figure 6.16. Performance impact of using the same SF in receive window two on ULPDR in an NbTrans=1 network, examined over several capacitor sizes.

Modifying the window's operation in conjunction with improved groupedPackets also slightly boosts performance in these networks, especially between six mF and eight mF. The simulations conducted here can be considered a best-case scenario as all nodes used a fast data rate instead of a mix of all possible data rates.

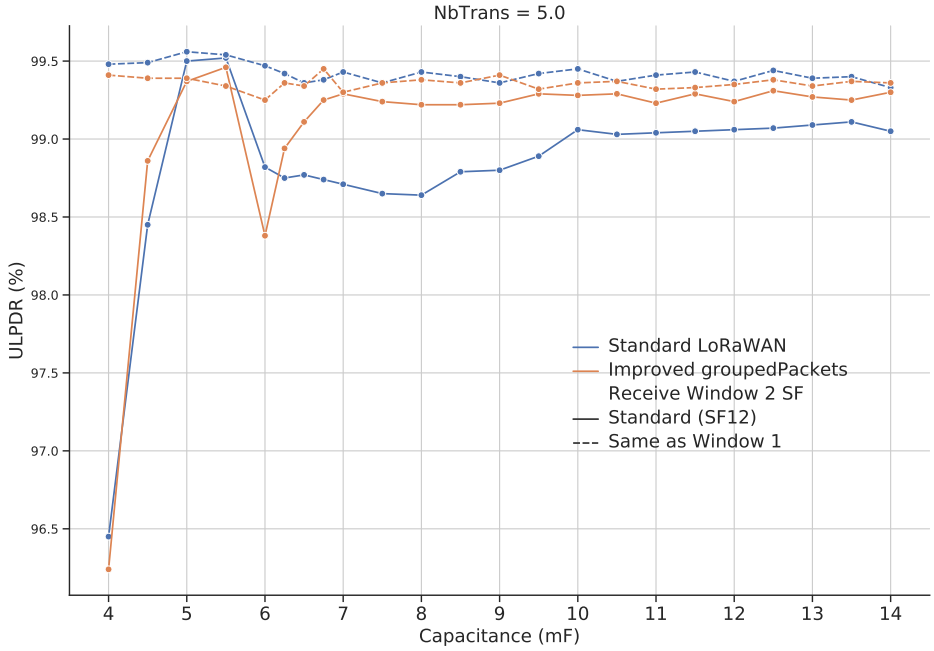


Figure 6.17. Performance impact of using the same SF in receive window two on ULPDR in an NbTrans=5 network, examined over several capacitor sizes.

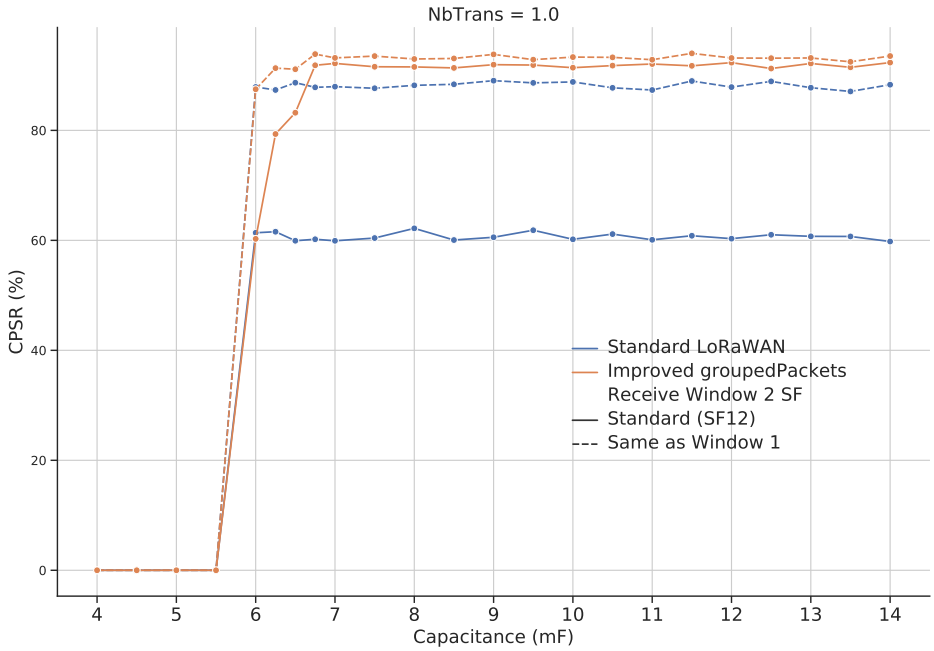


Figure 6.18. Performance impact of using the same SF in receive window two on CPSR in an NbTrans=1 network, examined over several capacitor sizes.

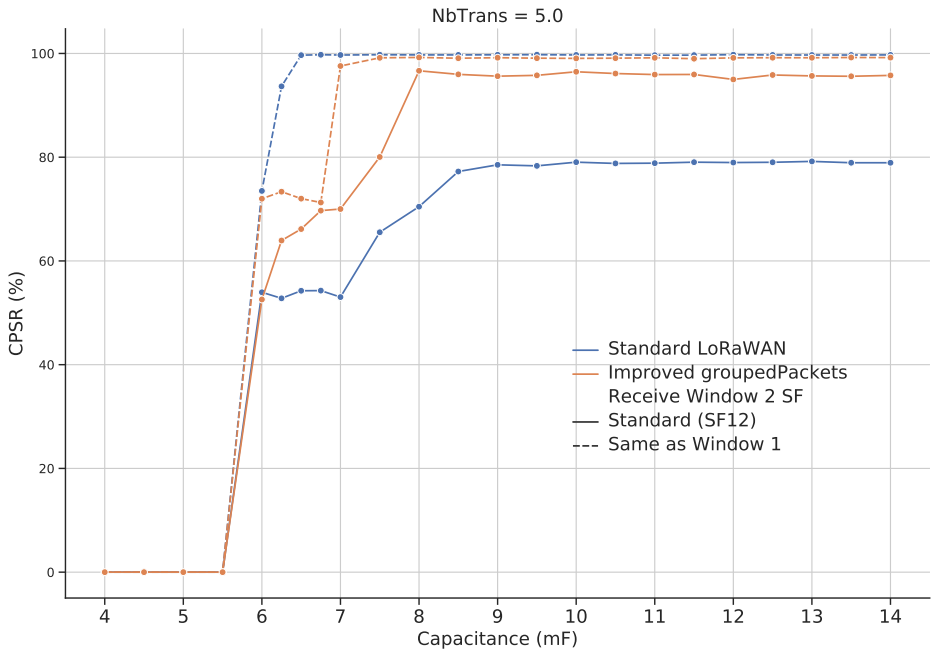


Figure 6.19. Performance impact of using the same SF in receive window two on CPSR in an NbTrans=5 network, examined over several capacitor sizes.

To save space, the following graphs were not plotted as identical copies can be found in other graphs and changing window two's SF had no impact on these graphs. The total packets sent in NbTrans=1 networks were identical to that of NbTrans=1 in Figure 6.11. The TPNS of unconfirmed packets in NbTrans=1 networks remained zero for all capacitance values, and the TPNS for confirmed packets in NbTrans=1 networks were identical to that of standard networks and optimised groupedPackets in Figure 6.10.

Figure 6.20 shows that in NbTrans=5 networks, modifying the second window's receive window reduced TPNS for unconfirmed packets to zero thanks to the decreased duration that unconfirmed nodes must open window two to detect the eight symbols preamble of a downlink frame. Unconfirmed nodes still have to open the receive windows as downlink transmissions containing MAC commands or data meant for the application running on the node are still possible. As shown in Figure 6.21, confirmed nodes matching window one's SF allow the minimum capacitance required to support up to five transmissions to reduce as less energy is required when opening the second receive window. A value of 7.5 mF allows all unconfirmed and confirmed packets to be sent when improved groupedPackets, NbTrans=5 and window two's operation is changed.

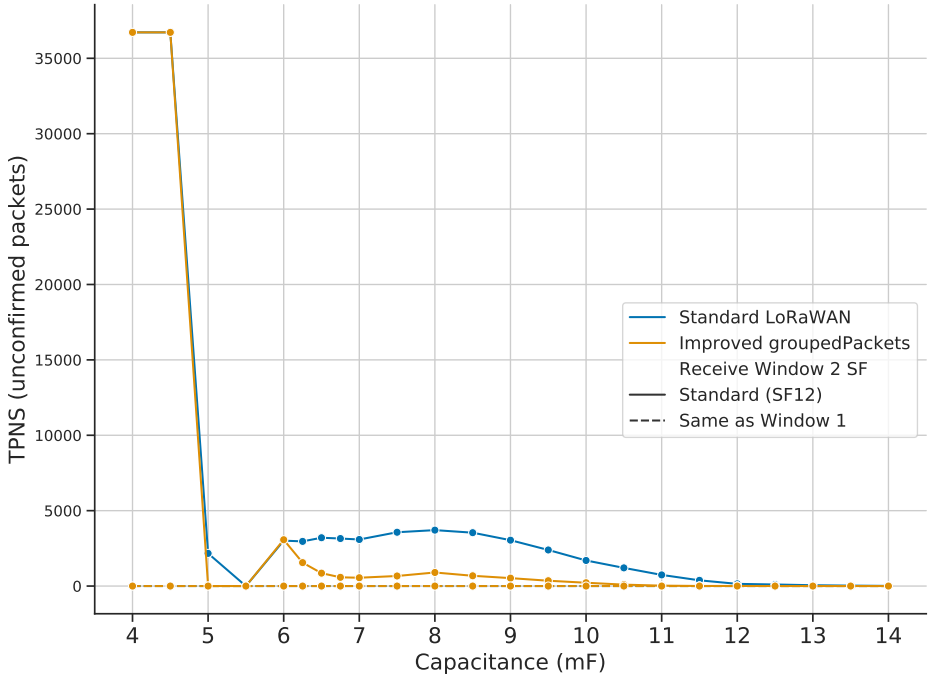


Figure 6.20. Minimum capacitance needed to transmit all unconfirmed packets assuming receive window modification in NbTrans=5 networks.

An additional benefit, as shown in Figure 6.22, in standard networks is that fewer confirmed packets are being retransmitted when window two's SF is matched to window one, thanks to the gateway being able to transmit more ACKs before being DC restricted.

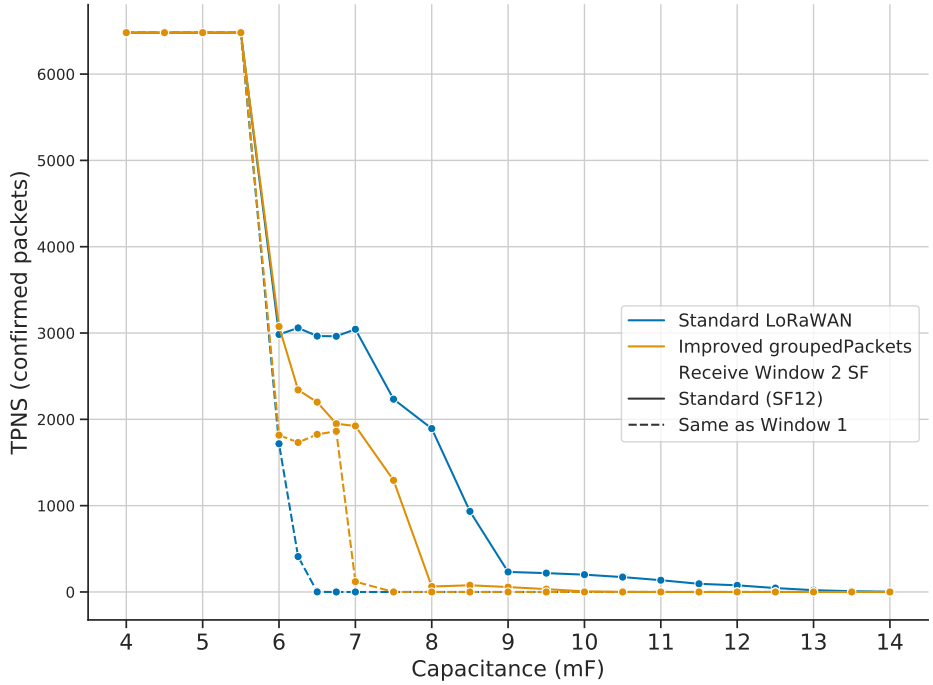


Figure 6.21. Minimum capacitance needed to transmit all confirmed packets assuming receive window modification in NbTrans=5 networks.

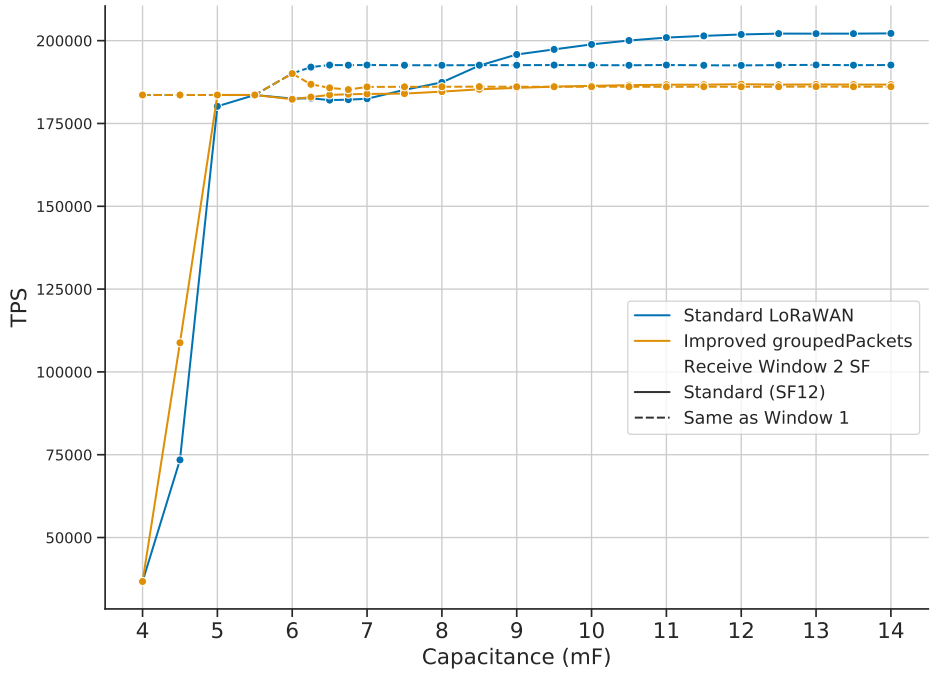


Figure 6.22. Total packets sent in networks with window two modification in NbTrans=5 networks, examined over several capacitor sizes.

6.6 CHAPTER SUMMARY

This chapter detailed the approach followed to investigate congestion in battery-less LoRaWANs as well as methods to reduce its impact. The simulation setup was provided and followed by the procedure used to ensure valid results. The implementation details of an energy-aware version of groupedPackets were then provided. Finally, results showcasing how NbTrans and changing the SF used by receive window two change the performance metrics in both standard and groupedPackets networks, were provided.

In Section 6.2, information was provided on the battery-less capacitor-based nodes, and it was found that the radio dominates current consumption and that its behaviour can be defined into several states, each with its current consumption. For energy harvesting, a generic harvesting system model was used in which the energy source is followed by a voltage regulator with simulations conducted with an ideal harvester delivering a constant ten mW. Four performance metrics were identified to be of interest for battery-less simulations, with the first two being modified versions of CPSR and ULPDR, calculated based on generated packets rather than transmitted packets. The final two metrics were focused on traffic volume (TPS) and energy (TPNS) to gain a better understanding of why packets were not transmitted.

To ensure valid results, exploratory simulations found that the total number of simulation periods used to generate results needs to be divisible by all possible aggregation values. This ensures the CPSR calculation is accurate, as all aggregated payloads would have been sent when the correct value is used. When an incorrect value is used, the CPSR calculation will falsely assume that generated packets were not received when in fact, they have not been transmitted yet. An investigation into simulation length and the number of runs found that results should be averaged over ten runs of 36 periods each (after an initial 20 periods to allow groupedPackets to settle).

Section 6.4 indicates how the standard groupedPackets algorithm underperformed in battery-less simulations as it will send aggregation requests that are infeasible due to energy constraints. When this occurs, nodes opt to transmit only the latest payload, resulting in the loss of aggregated payloads. This underperformance is only for a range of capacitance values in which there is sufficient energy to transmit small packets, but not necessarily enough for the maximum aggregation payload as the energy requirements increase with packet size.

An improved version of the algorithm was then detailed, in which the node uses the LinkACSAns MAC command to inform the NS side of the algorithm of energy constraints after optimising the amount of sent payloads. This is done by re-aggregating the stored payloads whilst also performing a calculation of the predicted energy cost of transmission to ensure that the maximum viable amount of payloads are sent. This approach allowed confirmed nodes whose capacitance is in a "transition" range of values to increase the number of payloads that were sent, which resulted in an increase in CPSR for this range.

In Section 6.5.1 increasing NbTrans did result in improved packet delivery in both standard LoRaWANs and the improved groupedPackets networks. The effectiveness of a higher NbTrans value is dependent on sufficient energy being available for retransmissions. Retransmissions also increase the overall traffic volume in the network, causing nodes to consume more energy by listening to transmissions meant for other devices (a frame must be processed to see if the address field indicates the node's address).

Confirmed nodes require more energy than unconfirmed nodes, and as a result, the minimum capacitance to support five transmissions differs between the two. Therefore, a certain range of capacitance values (six mF to eight mF) exists, which allows unconfirmed nodes five transmissions whilst confirmed nodes can only transmit less than this. In this range, CPSR is lower than what could have been achieved using NbTrans=1. At higher capacitance levels, all nodes can transmit NbTrans times and performance increases. The use of NbTrans should thus be considered when finalising a design's capacitance value to ensure retransmissions can be supported.

Changing the SF used by receive window two to match that of window one was investigated in Section 6.5.2 as this reduces the transmission time of ACKs. Matching the used SF results in a significant increase in CPSR in standard networks and also offers a slight improvement when done in groupedPackets networks. By matching the SF, more ACKs can be sent to confirmed nodes and energy savings occur for unconfirmed nodes due to the decreased duration they have to open receive window two to detect any messages meant for them.

CHAPTER 7 CONCLUSION

7.1 CHAPTER OVERVIEW

This final chapter contains a summary of the work conducted and presents the conclusions and suggestions for future work. Section 7.2 contains a summary of the work conducted and conclusions regarding the research questions presented in Section 7.3. Finally, future work suggestions are discussed in Section 7.4.

7.2 SUMMARY OF WORK CONDUCTED

IoT deployments are rising globally, with LPWANs providing the wireless networks needed for this expansion. One of these technologies, LoRaWAN, has proven to be a prevalent choice. This work set out to improve the understanding of the causes of congestion in mixed-traffic LoRaWANs and how the impact of this congestion can be reduced. Furthermore, the energy impact of methods to improve performance in these networks was studied.

The research questions addressed are What is the cause and impact of congestion in mixed-traffic LoRaWANs? How can a congestion management scheme be developed that monitors for congestion and reduces the impact of congestion on network performance? How should congestion be managed when a device's energy source is limited, as in the case of battery-less LoRaWANs?

A literature study was conducted to answer these questions, and a series of simulations were performed with an open-source LoRaWAN simulator. A novel method to monitor congestion was developed, and a novel algorithm that reduces congestion thanks to the aggregation of confirmed packets by nodes. Network performance was studied in the form of packet reliability, the success probability of unconfirmed traffic (ULPDR), and confirmed traffic (CPSR) using an ACK from the gateway. The

algorithm was then evaluated to determine its impact on network performance, as well as how the interactions between selected key LoRaWAN parameters influenced network performance.

Congestion in the network was monitored through a newly developed Adaptive Congestion Scheme (ACS) which activates the newly developed groupedPackets algorithm. The impact of retransmissions (LoRaWAN's NbTrans parameter), various arrival rates and base packet sizes on the performance of groupedPackets were presented. Finally, how groupedPackets perform when tasked with aggregating unconfirmed application payloads instead of confirmed ones was also examined.

The effectiveness of this algorithm and other parameters to reduce congestion were also investigated in battery-less capacitor-based networks. The implementation details of an energy-aware version of groupedPackets were first developed for battery-less simulations. The results showcased how retransmissions and changing the Spreading Factor (SF) of receive window two improve the performance metrics in both standard and groupedPackets networks.

Evaluations took place by comparing the algorithm's performance against that of the standard LoRaWAN protocol. To ensure accurate comparisons, exploratory simulations were first conducted to ensure that the duration of simulations was suitable and that results were averaged over a sufficient number of simulations. This ensured that the results were more generalised and did not reflect the performance of a specific network with a specific chain of events.

7.3 CONCLUSIONS

The first research question aimed to understand the causes and impact of congestion in mixed-traffic LoRaWANs. The LoRaWAN protocol allows for confirmed traffic from the end device to the gateway (uplink) and the reverse (downlink), therefore increasing the number of IoT use cases that can be supported. However, this comes at a cost as downlink traffic severely impacts scalability due to, in part, a gateway's duty cycle restrictions. Additionally, the following aspects negatively impact the viability of confirmed traffic in large networks: the use of SF12 for receive window two's transmissions, a high maximum number of transmissions and the ACK_TIMEOUT transmission back-off interval.

Due to the strict duty cycle restrictions a LoRaWAN gateway may successfully receive many confirmed packets, without being permitted to transmit an ACK. The half-duplex nature of LoRaWAN gateways also increases missed uplink transmissions as a gateway cannot receive any transmissions whilst

transmitting an ACK. This effect can occur even in networks with low ratios of confirmed traffic to unconfirmed traffic such as 5, 10 or 15 percent. It was found that confirmed traffic is viable in small networks, especially when data transfer is infrequent. Packet arrival rates above 10^{-1} pkt/s result in a steep performance decline for both traffic types, with CPSR dropping below 50 % in the case of only 15 % of nodes being confirmed nodes in standard LoRaWANs at arrival rates higher than one pkt/s. Performance can be improved by allowing multiple retransmissions, but this has its limitations in higher arrival rate networks.

The literature survey revealed that the LoRaWAN protocol does not have a congestion monitoring and response mechanism. To solve this, the ACS was developed. This scheme was inspired by the protocol's Adaptive Data Rate (ADR) scheme and consists of two parts. One part operates on the Network Server (NS) component in a network that will monitor the network and, upon reaching preset congestion levels, take action through a custom MAC command. Similarly to the ADR scheme, a node part will ensure that transmissions still occur if the NS part has requested unachievable settings resulting in payload sizes above allowed limits. The ACS activates in a network once preset levels of traffic volumes and confirmed traffic is detected and proceeds to utilise the groupedPackets algorithm to improve packet reliability.

The groupedPackets algorithm sought to reduce the traffic volumes generated by confirmed nodes by requesting that they aggregate their application packets. This causes nodes to send larger but less frequent confirmed packets. DC restrictions limit the sending of ACKs by a GW, and by reducing the number of confirmed packets, more ACKs can be sent. This, in turn, reduces the number of retransmissions of confirmed packets, causing fewer packet collisions due to interference. Finally, this also reduces the impact of the half-duplex nature of GWs, as the number of packets lost due to GW transmissions is reduced as fewer transmissions are required.

When groupedPackets was evaluated, it improved CPSR and ULPDR compared to standard networks. The algorithm showcased considerable improvements in CPSR at arrival rates higher than one pkt/s with only minimal impact at lower arrival rates (CPSR at these rates was already above 98 %). The algorithm's performance was further increased when NBtrans was set to eight, allowing aggregated payloads multiple chances at success. Increasing the base packet size from 10 B to 20 B slightly reduced the performance improvement offered by groupedPackets as this hampered the number of packets it could aggregate. The use of groupedPackets to aggregate unconfirmed packets was also

examined, with the algorithm offering only marginal improvements in a best-case scenario and no improvements in less favourable scenarios.

In battery-less LoRaWANs, the radio is the primary drain of energy resources. It was found that the standard groupedPackets algorithm underperforms in a congested version of these networks as it is unaware of energy constraints. An improved version was thus developed, in which the node uses the LinkACSAns MAC command to inform the NS side of the algorithm of energy constraints after optimising the amount of sent payloads. This is done by re-aggregating the stored payloads while calculating the predicted transmission energy cost to ensure that the maximum viable amount of payloads is sent. This approach allowed confirmed nodes whose capacitance is in a "transition" range of values to increase the number of sent payloads, which resulted in an increase in CPSR for this range.

Increasing NbTrans did result in improved packet delivery in both standard LoRaWANs and the improved groupedPackets networks. A higher NbTrans value's effectiveness depends on sufficient energy being available for retransmissions. Confirmed nodes require more energy than unconfirmed nodes, and as a result, the minimum capacitance to support multiple transmissions differed between the two types. Changing receive window two's SF to match that of window one (the default is the use of SF12) was also investigated as this reduces the transmission time of ACKs. Matching the used SF significantly increases CPSR in standard networks and offers a slight improvement when done in groupedPackets networks. By matching the SF, more ACKs can be sent to confirmed nodes, and energy savings occur for unconfirmed nodes due to a decrease in the amount of time receive window two must remain open to detect any transmissions meant for them.

Compared with a standard network, the combination of enabling the improved groupedPackets, increasing the number of retransmissions allowed, and matching the SF used by receive window two was found to perform the best. The combination improved ULPDR by $\approx$ ten percentage points, increased CPSR by $\approx$ 40 percentage points and can be achieved by a minor increase in the node's capacitance from six mF to 7.5 mF.

This work contributed to an improved version of the protocol in which the viability of the use of confirmed traffic is increased in mixed-traffic LoRaWANs. A method was also proposed with which congestion can be monitored and acted upon. An evaluation found that the developed algorithm can

significantly improve packet reliability, at the cost of an increased delay. IoT deployments will continue to increase in scale, and the congestion issue must be meticulously studied. Energy-constrained networks require additional considerations, and their deployments must be precisely planned to ensure sufficient energy resources and the management thereof to ensure long-term viability.

7.4 FUTURE WORK

Whilst the presented groupedPackets algorithm did result in CPSR and ULPDR improvements, it is not perfect. Currently, the NS's transmission of LinkACSReq commands is ad hoc, resulting in confirmed nodes forming groups that can be considered "out of sync" with their aggregated transmissions. An improved version could be designed to balance the number of nodes in each transmission group, ensuring that their resultant load is more evenly distributed at the gateway. The algorithm also only aggregated application payloads up to a total size limit of 50 B; this limit could instead be based on the node's SF, allowing for more aggregation.

This work evaluated groupedPackets in the context of single gateway networks. The use of groupedPackets in a network with multiple gateways is thus an unexplored area. The ACS was designed to support multiple algorithms, and other algorithms also aimed at reducing congestion could thus be developed and evaluated alongside groupedPackets.

When performing aggregation, groupedPackets prioritises the latest generated payload(s). This prioritisation will result in the oldest payload(s) being discarded should their addition exceed the 50 B limit. The inverse of this could also be studied, in which the priority is to transmit the oldest payloads instead. An inversed version could be used in IoT scenarios where the requirement is that all sampled data must be received, and a modified groupedPackets could assist with this.

REFERENCES

[1] Y. Song, F. R. Yu, L. Zhou, X. Yang, and Z. He, "Applications of the Internet of Things (IoT) in Smart Logistics: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4250–4274, 2021.

[2] M. A. Albreem, A. M. Sheikh, M. H. Alsharif, M. Jusoh, and M. N. Mohd Yasin, "Green Internet of Things (GIoT): Applications, Practices, Awareness, and Challenges," *IEEE Access*, vol. 9, pp. 38 833–38 858, 2021.

[3] L. Feltrin, C. Buratti, E. Vinciarelli, R. De Bonis, and R. Verdone, "LoRaWAN: Evaluation of link- and system-level performance," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2249–2258, 2018.

[4] E. M. Torroglosa-Garcia, J. M. Calero, J. B. Bernabe, and A. Skarmeta, "Enabling Roaming across Heterogeneous IoT Wireless Networks: LoRaWAN MEETS 5G," *IEEE Access*, vol. 8, pp. 103 164–103 180, 2020.

[5] J. M. Marais, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on the viability of confirmed traffic in a LoRaWAN," *IEEE Access*, vol. 8, pp. 9296–9311, 2020.

[6] J. P. S. Sundaram, W. Du, and Z. Zhao, "A Survey on LoRa Networking: Research Problems, Current Solutions and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 371–388, 2020.

[7] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A Survey of LoRaWAN for IoT: From Technology to Application," *Sensors*, vol. 18, no. 11, pp. 1–38, 2018.

[8] M. A. Ertürk, M. A. Aydin, M. T. Büyükakkaşlar, and H. Evirgen, "A survey on LoRaWAN architecture, protocol and technologies," *Future Internet*, vol. 11, no. 10, pp. 1–34, 2019.

[9] M. Saari, A. Muzaffar Bin Baharudin, P. Sillberg, S. Hyrynsalmi, and W. Yan, "LoRa - A survey of recent research trends," in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018*, Opatija. Croatia, May 2018, pp. 872–877.

[10] J. M. Marais, A. M. Abu-Mahfouz, and G. P. Hancke, "Improving the sustainability of confirmed traffic in LoRaWANs through an adaptive congestion scheme," *IEEE Sensors Journal*, vol. 23, no. 2, pp. 1660–1670, 2023.

[11] ——, "The Impact of Application-based Downlink Traffic in LoRaWANs," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, Central Drakensberg, South Africa, Nov. 2021, pp. 1–5.

[12] ——, "A Review of LoRaWAN Simulators: Design Requirements and Limitations," in *Proceedings - 2019 International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2019*, Vanderbijlpark, South Africa, Nov. 2019, pp. 1–6.

[13] S. Aslam, M. P. Michaelides, and H. Herodotou, "Internet of Ships: A Survey on Architectures, Emerging Applications, and Challenges," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9714–9727, 2020.

[14] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, and J. Melia, "Understanding the limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[15] M. O. Farooq and D. Pesch, "Analyzing LoRa: A use case perspective," in *IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, Singapore, Feb. 2018, pp. 355–360.

[16] F. Yu, Z. Zhu, and Z. Fan, "Study on the feasibility of LoRaWAN for smart city applications," in *IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Rome, Italy, Oct. 2017, pp. 334–340.

[17] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondon, S. A. Hassan, and M. Gidlund, "Scalability Analysis of a LoRa Network under Imperfect Orthogonality," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1425–1436, 2019.

[18] H. G. Schroder Filho, J. Pissolato Filho, and V. L. Moreli, "The Adequacy of LoRaWAN on Smart Grids: A Comparison with RF Mesh Technology," in *IEEE 2nd International Smart Cities Conference (ISC2)*, Trento, Italy, Sep. 2016, pp. 1–6.

[19] F. Orfei, C. B. Mezzetti, and F. Cottone, "Vibrations powered LoRa sensor An electromechanical energy harvester working on a real bridge," in *IEEE SENSORS*, Orlando, FL, USA, Oct. 2016, pp. 1–3.

[20] A. L. I. Nikoukar, S. Raza, A. Poole, M. Güneş, and B. Dezfouli, "Low-Power Wireless for the Internet of Things: Standards and Applications," *IEEE Access*, vol. 6, pp. 67 893–67 926, 2018.

[21] M. B. Mollah, S. Zeadally, and M. Azad, "Emerging Wireless Technologies for Internet of Things Applications: Opportunities and Challenges," in *Encyclopedia of Wireless Networks*. Springer International Publishing, 2019, pp. 1–11.

[22] W. Saadeh, S. A. Butt, and M. A. B. Altaf, "A Patient-Specific Single Sensor IoT-Based Wearable Fall Prediction and Detection System," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 5, pp. 995–1003, 2019.

[23] P. A. Catherwood, D. Steele, M. Little, S. McComb, and J. McLaughlin, "A Community-Based IoT Personalized Wireless Healthcare Solution Trial," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–13, 2018.

[24] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017.

[25] A. Mohsin and S. S. Yellampalli, "IoT based cold chain logistics monitoring," in *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, ICPCSI 2017*, Chennai, India, Sep. 2018, pp. 1971–1974.

[26] R. Sanchez-Iborra and M.-D. Cano, "State of the Art in LP-WAN Solutions for Industrial IoT Services," *Sensors*, vol. 16, no. 5, pp. 1–14, 2016.

[27] G. Margelis, R. Piechocki, D. Kaleshi, and P. Thomas, "Low Throughput Networks for the IoT: Lessons learned From Industrial Implementations," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy, Dec. 2015, pp. 181–186.

[28] R. M. Sandoval, A. J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal Policy Derivation for Transmission Duty-Cycle Constrained LPWAN," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3114–3125, 2018.

[29] L. Leonardi, F. Battaglia, and L. L. Bello, "RT-LoRa: A Medium Access Strategy to support Real-time flows over LoRa-based networks for Industrial IoT applications," *IEEE Internet of Things Journal*, pp. 1–12, 2019.

[30] L. Leonardi, F. Battaglia, G. Patti, and L. L. Bello, "Industrial Lora: A novel medium access strategy for Lora in industry 4.0 applications," in *44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, USA, Oct. 2018, pp. 4141–4146.

[31] D. Tamang, A. Pozzebon, L. Parri, A. Fort, and A. Abrardo, "Designing a Reliable and Low-Latency LoRaWAN Solution for Environmental Monitoring in Factories at Major Accident Risk," *Sensors*, vol. 22, no. 6, 2022.

[32] mOOvement. "GPS ear tags for cattle tracking". [Online]. Available: https://www.moovement.com.au/. [Accessed: 24 Jul. 2022].

[33] J. Navarro-Ortiz, J. J. Ramos-Munoz, J. M. Lopez-Soler, C. Cervello-Pastor, and M. Catalan, "A LoRaWAN Testbed Design for Supporting Critical Situations: Prototype and Evaluation," *Wireless Communications and Mobile Computing*, pp. 1–13, 2019.

[34] E. Sisinni, D. F. Carvalho, P. Ferrari, A. Flammini, and M. Gidlund, "Adding redundancy to lorawan for emergency communications at the factory floor," *IEEE Transactions on Industrial Informatics*, 2021.

[35] N. Maalel, E. Natalizio, A. Bouabdallah, P. Roux, and M. Kellil, "Reliability for emergency applications in Internet of Things," in *IEEE International Conference on Distributed Computing in Sensor Systems, DCoSS 2013*, Cambridge, MA, USA, May 2013, pp. 361–366.

[36] A. Dvornikov, P. Abramov, S. Efremov, and L. Voskov, "QoS Metrics Measurement in Long Range IoT Networks," in *IEEE 19th Conference on Business Informatics, CBI 2017*, vol. 2, Thessaloniki, Greece, Jul. 2017, pp. 15–20.

[37] D. H. Kim, J. B. Park, J. H. Shin, and J. D. Kim, "Design and implementation of object tracking system based on LoRa," in *International Conference on Information Networking (ICOIN)*, Da Nang, Vietnam, Jan. 2017, pp. 463–467.

[38] LoRa Alliance, "LoRaWAN 1.1 Regional Parameters," LoRa Alliance, Specification, Revision B, Jan. 2018.

[39] M. Capuzzo, D. Magrin, and A. Zanella, "Mathematical Modeling of LoRaWAN Performance with Bi-directional Traffic," in *IEEE Global Communications Conference, GLOBECOM 2018*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 206–212.

[40] LoRa Alliance, "LoRaWAN Specification V1.1," LoRa Alliance, Specification, Oct. 2017.

[41] ——, "LoRaWAN L2 1.0.4 Specification," LoRa Alliance, Specification, 2020.

[42] R. B. Sørensen, N. Razmi, J. J. Nielsen, and P. Popovski, "Analysis of LoRaWAN Uplink with Multiple Demodulating Paths and Capture Effect," in *IEEE International Conference on Communications*, Shanghai, China, China, May 2019, pp. 1–6.

[43] S. Corporation.

[44] M. Capuzzo, "Reliable LoRaWAN links: performance analysis," Master's thesis, Università di Padova, Padua, Italy, 2019.

[45] European Telecommunications Standards Institute, "ETSI EN 300 220-1 V2.4.1," ETSI, Standard, Feb. 2012.

[46] CEPT/ECC, "ERC Recommendation 70-03 Relating to the use of Short Range Devices (SRD)," CEPT, Recommendation, Jun. 2019.

[47] The Things Industries, "Frequency Plans," https://www.thethingsnetwork.org/docs/lorawan/frequency-plans.html (Accessed: 27 Jan. 2023).

[48] Semtech Corporation, "SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver," Semtech Corporation, Datasheet, Mar. 2015, Rev. 4.

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

[50] Q. Qian, L. Shu, Y. Leng, and Z. Bao, "LoRaWAN Network Downlink Routing Control Strategy Based on the SDN Framework and Improved ARIMA Model," *Future Internet*, vol. 14, p. 307, 2022.

[51] V. Di Vincenzo, M. Heusse, and B. Tourancheau, "Improving Downlink Scalability in LoRaWAN," in *IEEE International Conference on Communications*, Shanghai, China, China, May 2019, pp. 1–7.

[52] K. Q. Abdelfadeel, D. Zorbas, V. Cionca, and D. Pesch, " *FREE* —Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 669–683, 2020.

[53] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" in *19th ACM International Conference on Modeling, Analysis and Simulation of Wireless*

and Mobile Systems, Malta, Malta, Nov. 2016, pp. 59–67.

[54] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of LoRaWAN channel access with capture effect," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Montreal, QC, Canada, Oct. 2017, pp. 1–5.

[55] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, "LoRa Scalability: A Simulation Model Based on Interference Measurements," *Sensors*, vol. 17, no. 6, pp. 1–25, 2017.

[56] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, pp. 1–18, 2016.

[57] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1830–1842, 2018.

[58] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," in *17th Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2018*, Capri, Italy, Jun. 2018, pp. 1–7.

[59] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating Inter-network Interference in LoRa Networks," in *International Conference on Embedded Wireless Systems and Networks*, Uppsala, Sweden, Feb. 2017, pp. 323–328.

[60] G. Callebaut, G. Ottoy, and L. Van der Perre, "Cross-Layer Framework and Optimization for Efficient Use of the Energy Budget of IoT Nodes," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakesh, Morocco, April 2019, pp. 1–6.

[61] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive Configuration of LoRa Networks for Dense IoT Deployments," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, Taipei, Taiwan, Apr. 2018, pp. 1–9.

[62] A. M. Yousuf, E. M. Rochester, B. Ousat, and M. Ghaderi, "Throughput, Coverage and Scalability of LoRa LPWAN for Internet of Things," in *IEEE/ACM 26th International Symposium on Quality of Service, IWQoS 2018*, Banff, Alberta, Canada, Jun. 2019.

[63] J. Petäjäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage," *International Journal of Distributed Sensor Networks*, vol. 13, no. 3, pp. 1–16, 2017.

[64] A. I. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara, "Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?" in *IEEE Global Communications Conference, GLOBECOM 2017*, Singapore, Singapore, Dec. 2017, pp. 1–6.

[65] F. Delobel, N. El Rachkidy, and A. Guitton, "Analysis of the Delay of Confirmed Downlink Frames in Class B of LoRaWAN," in *IEEE Vehicular Technology Conference*, Sydney, NSW, Australia, Jun. 2017, pp. 1–6.

[66] M. Centenaro, L. Vangelista, and R. Kohno, "On the impact of downlink feedback on LoRa performance," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Montreal, QC, Canada, Oct. 2018, pp. 1–6.

[67] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *IEEE International Conference on Communications (ICC)*, Paris, France, Apr. 2017, pp. 1–7.

[68] D. Magrin, M. Capuzzo, A. Zanella, and M. Zorzi, "A Configurable Mathematical Model for Single-Gateway LoRaWAN Performance Analysis," *IEEE Transactions on Wireless Communications*, vol. 21, no. 7, pp. 5049–5063, 2022.

[69] F. H. Khan and M. Portmann, "Experimental Evaluation of LoRaWAN in NS-3," in *28th International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW, Australia, Nov. 2019, pp. 1–8.

[70] D. Magrin, M. Capuzzo, and A. Zanella, “A Thorough Study of LoRaWAN Performance Under Different Parameter Settings,” *IEEE Internet of Things Journal*, pp. 1–12, 2019.

[71] N. Varsier and J. Schwoerer, “Capacity limits of LoRaWAN technology for smart metering applications,” in *IEEE International Conference on Communications*, Paris, France, May 2017.

[72] Q. L. Hoang, V. N. Q. Bao, and H. Oh, “Performance Evaluation of LoRa Networks for Confirmed Messages,” in *International Conference on Advanced Technologies for Communications (ATC)*, Ho Chi Minh City, Vietnam, Oct. 2021, pp. 162–166.

[73] M. Centenaro and L. Vangelista, “Time-Power Multiplexing for LoRa-Based IoT Networks: An Effective Way to Boost LoRaWAN Network Capacity,” *International Journal of Wireless Information Networks*, pp. 1–11, 2019.

[74] K. Mikhaylov, J. Petajajarvi, and A. Pouttu, “Effect of Downlink Traffic on Performance of LoRaWAN LPWA Networks: Empirical Study,” in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Bologna, Italy, Sep. 2018, pp. 1–6.

[75] O. Brocaar. open-source LoRaWAN Network Server stack". [Online]. Available: https://www.chirpstack.io/. [Accessed: 24 Jul. 2022].

[76] D. Magrin, M. Capuzzo, A. Zanella, L. Vangelista, and M. Zorzi, “Performance Analysis of LoRaWAN in Industrial Scenarios,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6241–6250, 2021.

[77] D. Bankov, E. Khorov, and A. Lyakhov, “On the Limits of LoRaWAN Channel Access,” in *International Conference on Engineering and Telecommunication*, Moscow, Russia, Nov. 2016, pp. 10–14.

[78] Y. Hasegawa and K. Suzuki, “A Multi-User ACK-Aggregation Method for Large-Scale Reliable LoRaWAN Service,” in *IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–7.

[79] J. Lee, W. C. Jeong, and B. C. Choi, "A Scheduling Algorithm for Improving Scalability of LoRaWAN," in *International Conference on Information and Communication Technology Convergence, ICTC 2018*, Oct. 2018, pp. 1383–1388.

[80] G. Yapar, T. Tugcu, and O. Ermis, "Time-Slotted ALOHA-based LoRaWAN Scheduling with Aggregated Acknowledgement Approach," in *FRUCT 2019, 25th Finnish-Russian University Cooperation in Telecommunications Conference*, Helsinki, Finland, Nov. 2019, pp. 1–8.

[81] C. Zhong and A. Springer, "A Novel Network Architecture and MAC Protocol for Confirmed Traffic in LoRaWAN," *IEEE Access*, vol. 9, pp. 165 145–165 153, 2021.

[82] A. Farhad, D. H. Kim, and J. Y. Pyun, "R-ARM: Retransmission-Assisted Resource Management in LoRaWAN for the Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7347–7361, 2022.

[83] L. L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the Energy Performance of LoRaWAN," *Sensors*, vol. 17, no. 10, 2017.

[84] B. Kim and K. I. Hwang, "Cooperative Downlink Listening for Low-Power Long-Range Wide-Area Network," *Sustainability*, vol. 9, no. 4, pp. 1–15, 2017.

[85] The Things Industries, "The Things Network," https://www.thethingsnetwork.org/ (Accessed: 24 Jul. 2022).

[86] Packet Broker, "Packet Broker," https://packetbroker.net/ (Accessed: 24 Jul. 2022).

[87] K. Mikhaylov, "On the Uplink Traffic Distribution in Time for Duty-cycle Constrained LoRaWAN Networks," in *13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, Brno, Czech Republic, Oct. 2021, pp. 16–21.

[88] K. Banti, I. Karampelia, T. Dimakis, A.-A. A. Boulogeorgos, T. Kyriakidis, and M. Louta, "LoRaWAN Communication Protocols: A Comprehensive Survey under an Energy Efficiency Perspective," *Telecom*, vol. 3, no. 2, pp. 322–357, 2022.

[89] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN," *Sensors*, vol. 18, no. 7, 2018.

[90] S. S. Borkotoky, J. F. Schmidt, U. Schilcher, P. Battula, and S. Rathi, "Reliability and Energy Consumption of LoRa with Bidirectional Traffic," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3743–3747, Nov. 2021.

[91] G. Loubet, A. Takacs, E. Gardner, A. De Luca, F. Udrea, and D. Dragomirescu, "LoRaWAN Battery-Free Wireless Sensors Network Designed for Structural Health Monitoring in the Construction Domain," *Sensors*, vol. 19, no. 7, 2019.

[92] C. Delgado, J. M. Sanz, C. Blondia, and J. Famaey, "Batteryless LoRaWAN Communications Using Energy Harvesting: Modeling and Characterization," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2694–2711, 2021.

[93] M. Capuzzo, C. Delgado, J. Famaey, and A. Zanella, "An ns-3 implementation of a battery-less node for energy-harvesting Internet of Things," in *Proceedings of the Workshop on Ns-3*. Association for Computing Machinery, June 2021, pp. 57–64.

[94] M. Capuzzo, C. Delgado, A. K. Sultania, J. Famaey, and A. Zanella, "Enabling Green IoT: Energy-Aware Communication Protocols for Battery-less LoRaWAN Devices," in *Proceedings of the 24th International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Alicante, Spain, Nov. 2021, pp. 95–98.

[95] C. Pham and M. Ehsan, "Dense deployment of LoRa networks: Expectations and limits of channel activity detection and capture effect for radio channel access," *Sensors*, vol. 21, no. 3, pp. 1–21, 2021.

[96] K. Mikhaylov, J. Petäjäjärvi, and J. Janhunen, "On LoRaWAN scalability: Empirical evaluation of susceptibility to inter-network interference," in *2017 European Conference on Networks and Communications (EuCNC)*, Oulu, Finland, June 2017.

[97] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of LoRaWAN channel access

with capture effect," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Montreal, QC, Canada, Oct. 2017, pp. 1–5.

[98] T. H. To and A. Duda, "Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA," in *IEEE International Conference on Communications*, Kansas City, MO, USA, May 2018, pp. 1–7.

[99] B. Reynders, Q. Wang, and S. Pollin, "A LoRaWAN module for ns-3: Implementation and evaluation," in *ACM International Conference Proceeding Series*, Karnataka, Surathkal, India, June 2018, pp. 61–68.

[100] D. Magrin, "Network level performances of a LoRa system," Master's thesis, Università di Padova, Padua, Italy, 2016.

[101] C. Goursad and J. M. Gorce, "Dedicated networks for IoT: PHY/MAC state of the art and challenges," *EAI Endorsed Transactions on the Internet of Things*, vol. 1, no. 1, pp. 1–12, 2015.

[102] D. Magrin, M. Capuzzo, and A. Zanella, "A Thorough Study of LoRaWAN Performance Under Different Parameter Settings," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 116–127, 2020.

[103] D. Magrin, D. Zhou, and M. Zorzi, "A simulation execution manager for ns-3 encouraging reproducibility and simplifying statistical analysis of ns-3 simulations," in *MSWiM 2019 - Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Miami Beach, FL, USA, Nov. 2019, pp. 121–125.

[104] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021

[105] N. Blenn and F. Kuipers, "LoRaWAN in the wild: Measurements from the things network," *arXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03086

[106] J. Finnegan, R. Farrell, and S. Brown, "Analysis and Enhancement of the LoRaWAN Adaptive Data Rate Scheme," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7171–7180, 2020.

ADDENDUM A SUPPLEMENTAL MATERIALS.

A.1 CHAPTER 3: INVESTIGATION INTO IMPACT OF SIMULATION LENGTH AND NUMBER OF RUNS.

Section 3.4.4 contains graphs showing the mean and standard deviation of simulations conducted with standard LoRaWAN for NbTrans=8. For the sake of completeness, the table with its correspondings graphs for NbTrans=1 are given here as Table A.1 and Figures A.1, A.2. The table and corresponding graphs for NbTrans=8 can be found as Table A.2 and Figures A.3, A.4 respectively.

For both values, the same conclusions can be made as those presented in Chapter 3. Increasing the number of periods would not significantly change either performance metric. Similarly, whilst using more than one run would be beneficial, the difference between five and twenty is minimal.

Table A.1. The arithmetic means of ULPDR and CPSR values as percentage points when calculated over different simulations lengths and number of runs. Standard deviation is given in brackets. All values are for the case of NbTrans=1.

		Simulation length		
		10	30	50
Number of runs	1	ULPDR: 79.51 CPSR: 32.39	ULPDR: 79.03 CPSR: 30.8	ULPDR: 80.25 CPSR: 32.49
	5	ULPDR: 79.85 (0.52) CPSR: 32.86 (0.39)	ULPDR: 80.44 (1.08) CPSR: 32.17 (0.93)	ULPDR: 79.74 (0.27) CPSR: 34.51 (2)
	10	ULPDR: 79.81 (0.6) CPSR: 32.92 (0.59)	ULPDR: 80.93 (1.1) CPSR: 32.35 (1.11)	ULPDR: 79.96 (0.99) CPSR: 34.18 (1.57)
	20	ULPDR: 79.68 (0.64) CPSR: 32.81 (0.99)	ULPDR: 80.43 (1.28) CPSR: 33.42 (2.38)	ULPDR: 79.93 (1) CPSR: 33.84 (1.45)

Table A.2. The arithmetic means of ULPDR and CPSR values as percentage points when calculated over different simulations lengths and number of runs. Standard deviation is given in brackets. All values are for the case of NbTrans=5.

		Simulation length		
		10	30	50
Number of runs	1	ULPDR: 91.16 CPSR: 44.18	ULPDR: 91.23 CPSR: 43.07	ULPDR: 92.3 CPSR: 44.71
	5	ULPDR: 92.03 (0.98) CPSR: 43.85 (0.76)	ULPDR: 92.74 (1.01) CPSR: 43.53 (1.26)	ULPDR: 92.14 (0.98) CPSR: 44.06 (1.05)
	10	ULPDR: 92.23 (0.91) CPSR: 43.85 (0.75)	ULPDR: 92.27 (1.18) CPSR: 43.7 (1.18)	ULPDR: 92.16 (0.78) CPSR: 43.69 (1.06)
	20	ULPDR: 92.24 (0.88) CPSR: 44.01 (0.73)	ULPDR: 92.44 (1.04) CPSR: 43.68 (1.23)	ULPDR: 91.93 (0.76) CPSR: 43.66 (0.94)

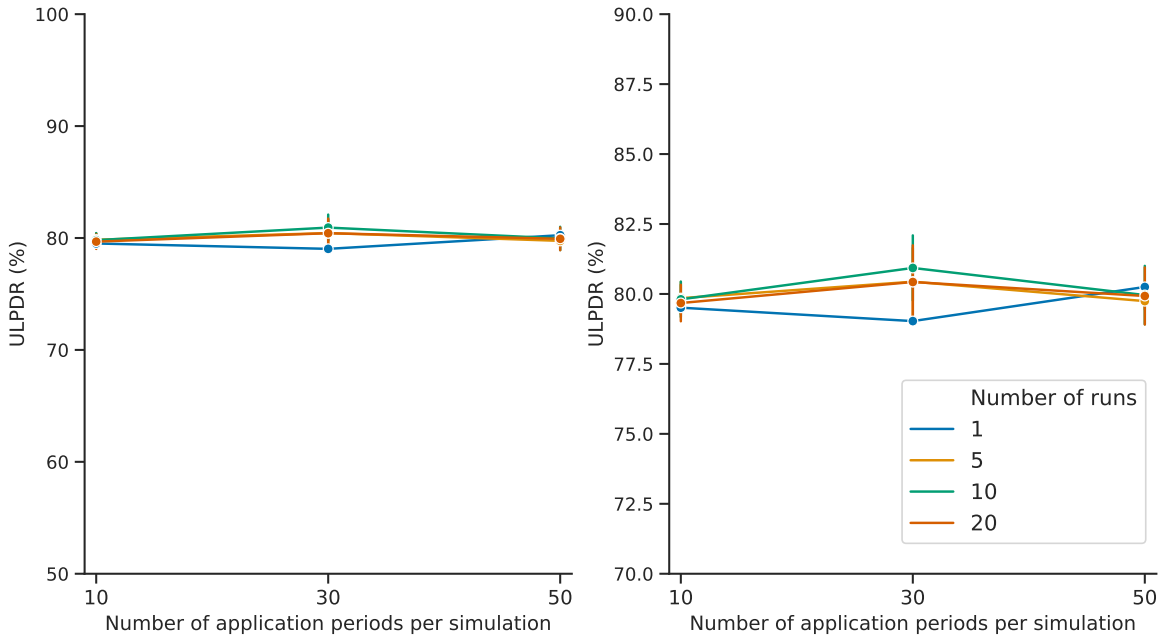


Figure A.1. The mean and standard deviation of ULPDR for an NbTrans=1 setup.

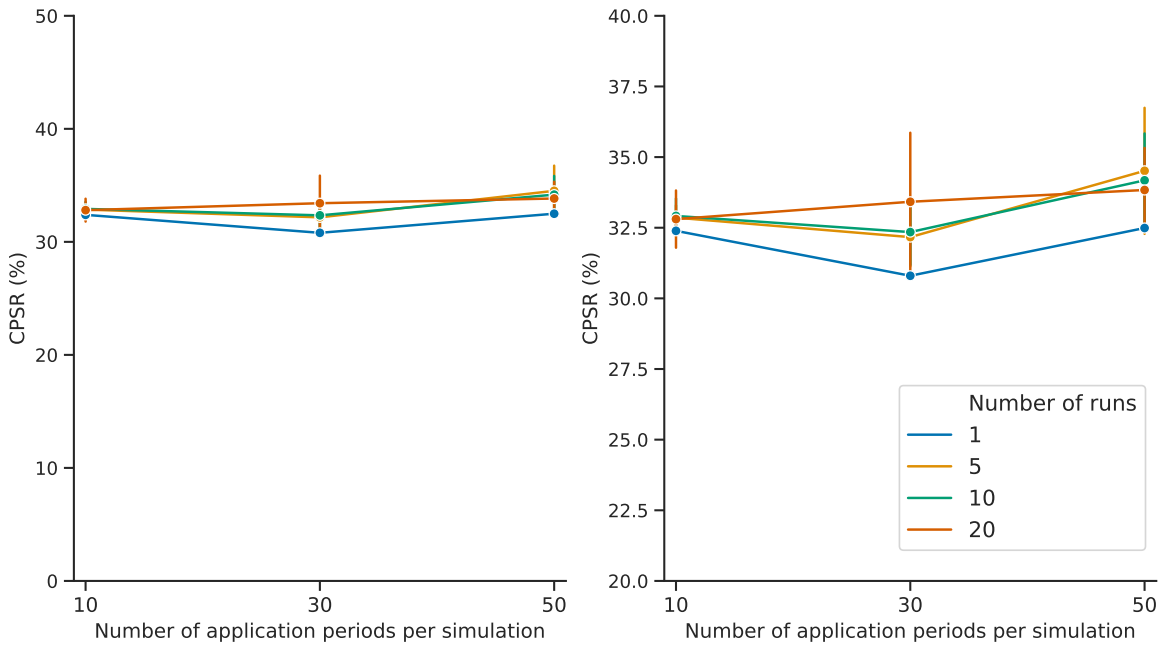


Figure A.2. The mean and standard deviation of CPSR for an NbTrans=1 setup.

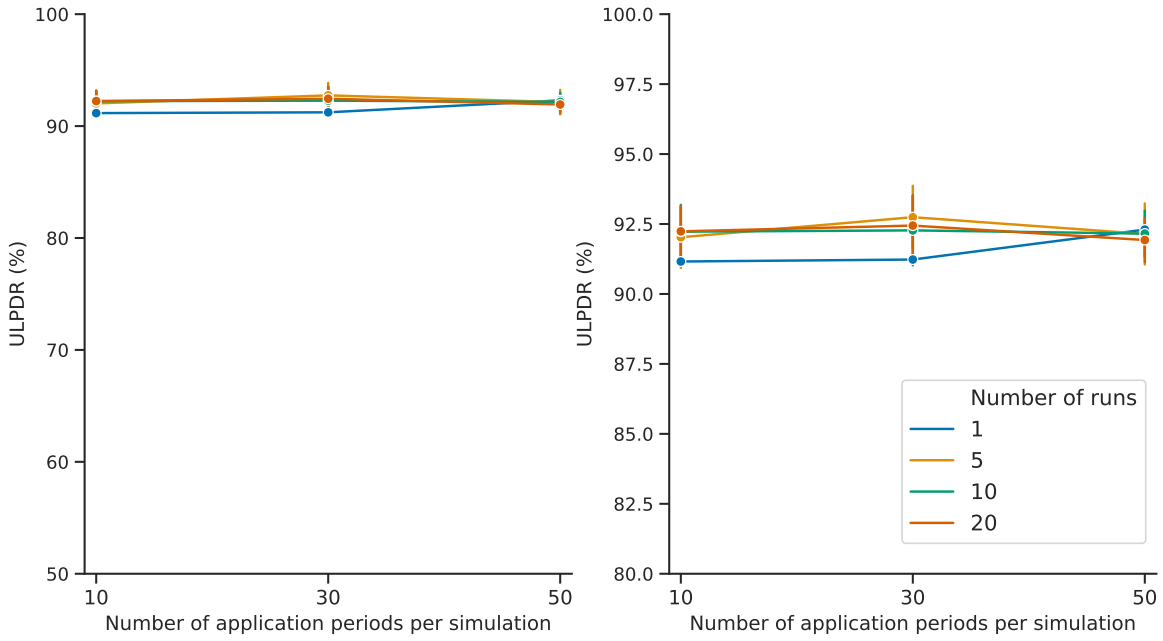


Figure A.3. The mean and standard deviation of ULPDR for an NbTrans=5 setup.

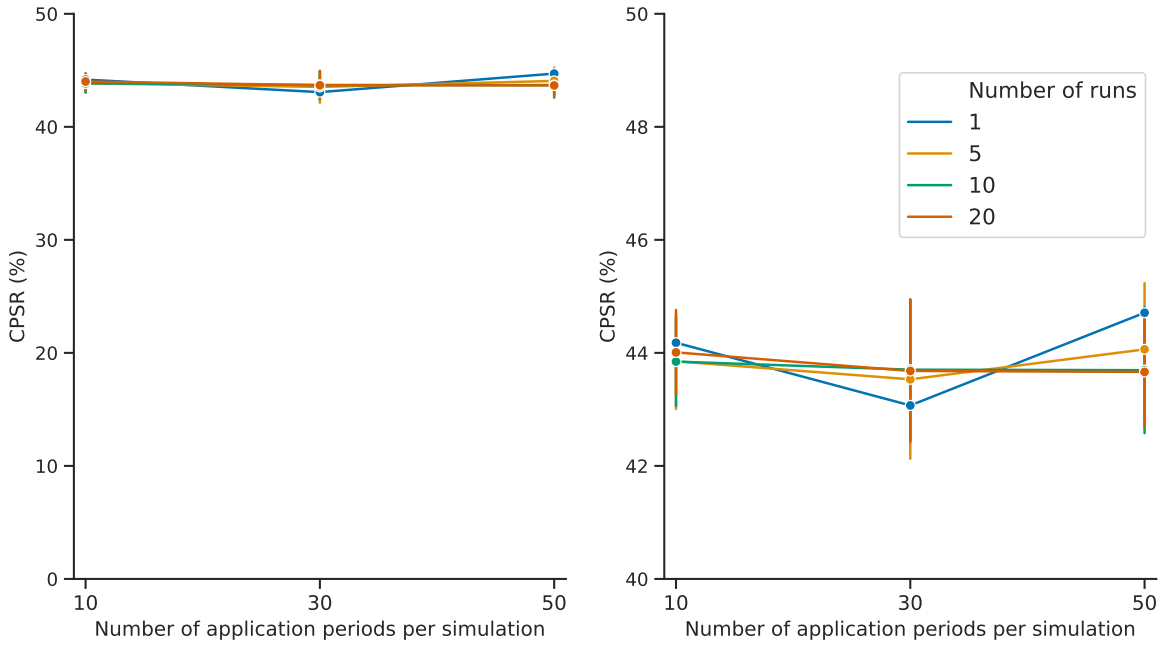


Figure A.4. The mean and standard deviation of CPSR for an NbTrans=5 setup.