UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SCHC-OVER-SIGFOX: PERFORMANCE MODELING AND EVALUATION OF FRAGMENT DELIVERY

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DIEGO SEBASTIÁN WISTUBA LA TORRE

PROFESORA GUÍA:
SANDRA CÉSPEDES UMAÑA

PROFESOR COGUÍA:
JAVIER BUSTOS JIMÉNEZ

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
DIEGO DUJOVNE HELMAN

This thesis has been financially supported in part by ANID Fondecyt Regular 1201893 and ANID Basal Project FB0008.

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: DIEGO SEBASTIÁN WISTUBA LA TORRE
FECHA: 2023
PROF. GUÍA: DRA. SANDRA CÉSPEDES

SCHC-OVER-SIGFOX: MODELAMIENTO Y EVALUACIÓN DE DESEMPEÑO EN LA ENTREGA DE FRAGMENTOS

El protocolo *Static Context Header Compression and Fragmentation* (SCHC) facilita la comunicación mediante IPv6 en redes de alto alcance y bajo consumo energético (*Low-Power Wide-Area Networks*, LPWANs). Estas redes proveen bajo consumo de batería y un alcance inalámbrico amplio a cambio de tasas de datos bajas y tamaños reducidos de sus datagramas. SCHC comprime y fragmenta paquetes grandes y define el mecanismo de entrega confiable ACK-on-Error para confirmar la recepción de fragmentos y hacer posible su reensamblaje. El mecanismo ACK-on-Error puede usar recursos adicionales de la red, los cuales no siempre están disponibles en redes restringidas como la LPWAN Sigfox.

Muchas veces en despliegues LPWAN, malas condiciones de canal, tales como una alta tasa de pérdida de paquetes, pueden llevar a abortar la comunicación y malgastar recursos de la red. Estos factores son decisivos al determinar si usar SCHC en alguna aplicación o no. Sin embargo, ya que SCHC es un estándar reciente, hay una falta de conocimiento respecto del impacto de la tasa de pérdida de paquetes sobre el número de transmisiones *uplink* requeridas y sobre la tasa de éxito de las transmisiones SCHC.

Esta tesis propone la definición y análisis de las siguientes métricas de desempeño para SCHC: el número promedio de mensajes *uplink* enviados por transmisión SCHC y la tasa de éxito de las transmisiones. Se definen además modelos semiempíricos para ambas métricas. Se demuestra que los modelos proveen una buena aproximación para las métricas, y que dichas métricas son útiles para evaluar el uso de SCHC ACK-on-Error sobre aplicaciones de Sigfox, un análisis que puede extenderse a otras tecnologías LPWAN.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: DIEGO SEBASTIÁN WISTUBA LA TORRE
FECHA: 2023
PROF. GUÍA: DRA. SANDRA CÉSPEDES

SCHC-OVER-SIGFOX: PERFORMANCE MODELING AND EVALUATION OF FRAGMENT DELIVERY

The Static Context Header Compression and Fragmentation (SCHC) framework enables IPv6 communication over Low-Power Wide-Area Networks (LPWANs), which provide low power consumption and wide area coverage in exchange for low data rates and small packet sizes. SCHC compresses and fragments large packets and defines the ACK-on-Error reliable delivery algorithm to confirm the reception of fragments and make reassembly possible. The ACK-on-Error mechanism may use additional network resources, which are not always available in constrained networks such as the Sigfox LPWAN.

Often times in LPWAN deplyoments, poor connectivity conditions, such as high packet loss rates (PLRs), may lead to aborted communications and wasted resources—decisive factors when determining whether to use SCHC in an application. However, since SCHC is a recent standard, there is a lack of knowledge about the impact of the PLR on the required number of uplink messages and the rate of successful SCHC transmissions.

This thesis proposes the definition and analysis of the following performance metrics for SCHC: the average number of uplink messages sent per SCHC transmission, and the rate of successful SCHC transmissions. Semi-empirical models are provided for both metrics. It is demonstrated the models provide a good approximation to the metrics, and that these performance metrics are useful for evaluating SCHC ACK-on-Error usage over Sigfox applications, an analysis that can also be extended to other LPWANs.

A todxs quienes han llorado en los pasillos y patios de la U,
y a todxs quienes me han visto llorar en los pasillos y patios de la U.

Agradecimientos

Quiero partir agradeciéndole a Sandra por su infinita paciencia y fe en mi trabajo, por su apoyo económico que fue crucial en los tiempos más agitados de mi carrera, y por todas las veces que me dio ese empujoncito que necesitaba para enfrentar mis inseguridades académicas. A Javier y a Niclabs por respaldar mi trabajo desde el 2019 y entregarme un puesto (y un server) en el cual pude desarrollar todo este trabajo. A les profes con vocación a quienes tuve la suerte de conocer en la facultad, y a les auxiliares motivades que tantas (pero tantas) veces se tuvieron que poner la capa para hacer la materia entendible.

A quienes colaboraron de alguna forma en esta tesis. A mis padres, Giuliana y Omar, por entenderme y siempre brindarme todo su apoyo, aun cuando tuve que pasar días sin verlos o llegando a mi casa pasadas las 00:00. Al Ian por darme los primeros *hints* de proba que pavimentarían el camino inicial de esta tesis. A Roy por matraquear conmigo. Al Coba por ayudarme con las sutilezas matemáticas de los golazos cometidos en esta tesis (es broma). A la Flo por ayudarme a usar `screen` y a optimizar algunos gráficos de esta tesis. A Cisneros y al Pipe por compartirme sus memorias para yo aprender cómo se describe un *software*. Al Blaz por prestarme su VPN. Al Dmitri por haber hecho un push en el PC que dejé prendido lejos de mi alcance. Al *Eric* por presentarme la distribución Beta. A la Ivana por el apoyo moral durante mis inseguridades tesistas. A Pablo por ayudarme a escoger funciones candidatas (y por soportar mis ranteos y desilusiones académicas).

No hubiera podido llegar hasta aquí, bajo ningún criterio, de no ser por la gente que estuvo a mi lado y con quienes pude compartir algún chiste, algún almuerzo, algún jueguito de mesa u online, algunos carretes. Agradezco a los grupos de amigues de los que fui parte, no limitándose a: Les Wituniens, las Crêmès, el Electroteam, Sumaria2, Comfeli (de quienes no soy parte), Yogis, la Ofisalita, ApendiCC/Los Cullen, Bellakeo++ y tantos otros. A aquelles con quien he tenido el agrado de tocar algún temita en la U, ya sea frente a un público o de cara a nosotres mismes. A quienes alguna vez me prestaron un sillón para yo poder dormir ahí habiéndome quedado en la U hasta horas absurdas haciendo tareas absurdas. A la gente que ya no está aquí en cuerpo, pero sí en alma. Y a las personitas que me brindaron en su momento un colchoncito emocional que atesoro todos los días.

Y ahora un poema: *He visto a las mentes más brillantes de mi generación destruidas por la locura (locura que grita el corazón cuando es víctima de la razón), histéricxs famélicxs desnudxs arrastrándonos entre las terrazas, que pobres y harapientxs y ojerosxs pasamos la noche en la oscuridad sobrenatural de los departamentos de disciplina fría. Mucho se nos habla de que estamos parados en hombros de gigantes, cuando en realidad los gigantes somos nosotres mismes.*

Table of content

List of Tables

List of Figures

Acronyms

3GPP Third Generation Partnership Project.

5G Fifth Generation.

ACK Acknowledgement.

ACK REQ ACK Request.

ALOHA Advocates of Linux Open-source Hawaii Association, or Additive Links On-line Hawaii Area.

API Application Programming Interface.

ARQ Automatic Repeat Request, or Automatic Repeat Query.

BLE Bluetooth Low Energy.

C/D Compression/Decompression.

CoAP Constrained Application Protocol.

CPU Central Processing Unit.

CRC Cycling Redundancy Check.

CSS Chirp Spread Spectrum.

DASH7 Developers' Alliance for Standards Harmonization of ISO 18000-7.

DL Downlink.

DTag Datagram Tag.

ECC Error-Correcting Code.

eNodeB Evolved Node B.

F/R Fragmentation/Reassembly.

FCN Fragment Compressed Number.

FCS Frame Check Sequence.

FLR Fragment Loss Rate.

GCP Google Cloud Platform.

HTTP Hyper-Text Transfer Protocol.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

iid Independent and Identically Distributed.

IoT Internet of Things.

IP Internet Protocol.

IPv6 Internet Protocol, version 6.

ISM Industrial, Scientific, and Medical.

ISO International Organization for Standardization.

JSON JavaScript Object Notation.

L2 Layer 2.

LoRa Long-Range.

LoRaWAN Long-Range Wide Area Network.

LOS Line of Sight.

LPWAN Low-Power Wide-Area Network.

LPWAN-AAA LPWAN Authentication, Authorization, and Accounting.

LR-WPAN Low-Rate Personal-Area Network.

LSM Least Squares Method.

LTE Long Term Evolution.

LTE-M LTE Machine Type Communication.

MAC Media Access Control.

MTU Maximum Transmission Unit.

NACK Negative Acknowledgement.

NB-IoT Narrowband IoT.

OSI Open Systems Interconnection.

PDN Packet Delivery Network.

PHY Physical Layer.

PLR Packet Loss Rate.

RAM Random Access Memory.

RC Radio Configuration.

RCS Reassembly Check Sequence.

REST Representational State Transfer.

RFC Request for Comments.

RFT Receiver-Feedback Technique.

RX Reception.

SCHC Static Context Header Compression and Fragmentation.

SNR Signal-to-Noise Ratio.

TDD Test-driven Development.

ToA Time on Air.

TX Transmission.

UDP User Datagram Protocol.

UL Uplink.

UML Unified Modeling Language.

UNB Ultra-Narrow Band.

URI Uniform Resource Identifier.

Wi-SUN Wireless Smart Ubiquitous Network.

WPAN Wireless Personal-Area Network.

Chapter 1

Introduction

The Internet of Things (IoT) is a technology term that describes network systems composed of devices—or things—embedded with sensors or actuators and able to exchange data with other devices using a network infrastructure, such as (but not limited to) the Internet. This technology paradigm enables devices to send information either to other devices or to a data processing server. This allows operators to remotely control the devices and to immediately collect and process their data. As the Internet is the most highly distributed network used by data servers, communication using the Internet protocol (IP), and other protocols of the IP protocol suite, is desired in IoT deployments.

However, the available infrastructure and resources to handle such a massive amount of data transmissions are highly constrained. Depending on the distribution of the devices, the available network technology needs to provide the adequate coverage, which is often satisfied by wireless networks. This network technology should also constrain the amount of data sent over time, since large and frequent data transmissions are prone to collisions and interference. Moreover, as the devices are meant to operate remotely and with little human intervention, the power consumption of the communication should also be minimized to extend battery life. Although Wireless Personal-Area networks (WPANs) are useful in small scale IoT systems, if the distribution of the devices expands geographically, personal-area networks are no longer suited to supply the network coverage requirements.

Low-Power Wide-Area Network (LPWAN) technologies satisfy the wide coverage requirements and are the primary type of networks used in large scale IoT systems deployment [1]. LPWANs are characterized by their small power consumption during wireless transmission and reception of data, which leads to a battery life of many years. These networks also constrain the amount of data transmitted over time —the throughput—, to reduce power consumption and interference. LPWANs are wireless, and use modulation mechanisms that maximize their coverage to many kilometers. Many LPWAN technologies exist and are deployed in urban environments, the most remarkable being LoRaWAN, Sigfox, and NB-IoT.

The remaining challenge for LPWANs to satisfy the IoT industry requirements is enabling IP communication, which is not trivial since the size of LPWAN packets is limited. IPv6 and UDP headers, which are needed to process IP packets, highly surpass the maximum packet size of most LPWANs [7], leaving little to no space for payload data. Sigfox, in particular, is the most constrained network of the above-mentioned in terms of packet length, having a maximum transmission unit (MTU) of 12 bytes, not enough to carry these headers.

Static Context Header Compression and Fragmentation (SCHC) is a framework designed to compress IPv6 or UDP headers and to fragment the result of this compression if it still exceeds the MTU of the network where SCHC is applied [5]. It is provided as an adaptation layer between the LPWAN link layer and the IP network layer, and can be extended to support other network protocols or applications with similar overhead. Its fragmentation mechanism is also applicable when no header compression is performed and when the desired payload size exceeds the network MTU. SCHC is defined as a general standard with the parameters and algorithms necessary to perform compression, fragmentation, reassembly, and decompression, but the specific values and behaviors vary among different LPWAN. The set of specific values and behaviors is called a SCHC Profile, and are left for each LPWAN to propose and configure.

As the fragmentation procedure generates many LPWAN packets to be sent over the network, reliable delivery mechanisms are defined to guarantee fragment delivery over a lossy wireless channel. To achieve this, the SCHC receiver sends acknowledgement (ACK) messages back to the sender that inform about the reception or loss of SCHC Fragments. The SCHC Profile of a network defines the header fields of SCHC Fragments and ACKs, and the timeout values to optimize this bidirectional communication. Upon reception of an error-reporting ACK, the device automatically retransmits all lost data accordingly. This behavior follows the automatic repeat request (ARQ) paradigm for reliable delivery.

1.1 Motivation and background

The SCHC framework was standardized by the Internet Engineering Task Force (IETF) LPWAN Working Group in April 2020. At the time of this writing, academic studies regarding the performance of SCHC in different networks are scarce. The implementations of this framework are also few, and have been in constant development. LoRaWAN is the only LPWAN to have its SCHC Profile standardized by the IETF [8], whilst the SCHC Profiles of Sigfox and NB-IoT are still undergoing standardization [3, 9].

The Sigfox SCHC Profile—or SCHC/Sigfox[1]—is still a work in progress, published as an Internet-Draft since April 2019. Sigfox does not use SCHC compression, focusing instead on the fragmentation and reassembly procedures. Sigfox uses the ACK-on-Error ARQ mechanism of SCHC fragmentation since it minimizes downlink channel use, which is also highly constrained in terms of maximum uplink packet size [10]. Sigfox proposes minimal changes to the ACK-on-Error algorithm to optimize network resources usage, which are addressed later in this document. A recent implementation of SCHC over Sigfox allows it to be tested from different approaches regarding the performance of the fragmentation and ARQ mechanisms [11, 12].

Performance evaluations help introduce the SCHC framework to the IoT industry, giving

[1] For brevity, in the remainder of this document, "SCHC/Sigfox" refers to the Sigfox SCHC Profile and its ACK-on-Error implementation.

quantitative analyses that make it easier to compare the behavior of SCHC ARQ algorithms used by the different LPWAN technologies. These analyses highlight the benefits and remarkable features of SCHC, facilitate the adoption of (and interest in) the framework, and make SCHC ACK-on-Error comparable to other ARQ mechanisms.

Useful performance metrics for ARQ implementations are the additional transmission delay, the queue length, and the throughput of the whole system. The channel usage and data overhead are also considered, specially in wireless networks. These metrics have been widely studied for classic ARQ techniques such as Go-Back-N or Selective Repeat. Although the main concept of using ACK messages is present in the reliable delivery mechanisms of SCHC, they have peculiarities that make them different from the mentioned protocols.

1.1.1 Definition of the problem

Given insufficient network conditions, the ARQ mechanisms of SCHC may fail. Due to excessive use of bandwidth, number of messages sent or received, energy, or other scarce resources of the network, either side of the communication can decide that the transmission should not continue. In doing so, the ARQ algorithm is finished and a SCHC Abort message is sent, thereby letting the other side of the communication know about this decision. In this way, a SCHC transmission is aborted.

When a SCHC transmission is aborted, it is most likely the case that not all SCHC Fragments have been delivered to the receiver. As aborting the transmission finalizes the ARQ mechanism, the remaining fragments are discarded, making reassembly impossible. The receiver may discard every SCHC Fragment received so far, since it is no longer possible to retrieve the information of the original SCHC Packet.

The abort mechanisms are useful in avoiding using too many resources in challenging network conditions. However, at the time of writing there has not been an analysis on how many resources are saved by aborting a SCHC transmission under different SCHC Profile configurations. Moreover, the probability of aborting a SCHC transmission has also not been studied. These metrics are inherently determined by the PLR of the network. As the theoretical and empirical analyses of the resource usages of SCHC are few, not much can be concluded regarding these performance metrics, which are important in highly constrained networks such as Sigfox, and crucial in deciding whether to incorporate SCHC in an application when challenging network conditions are present.

1.1.2 Proposed solution

This thesis defines two performance metrics under the SCHC-over-Sigfox ACK-on-Error mechanism:

- The average number of uplink messages sent in a SCHC/Sigfox transmission, N_U.
- The rate of successful SCHC/Sigfox transmission, r_S.

These metrics are non-standard for ARQ mechanisms and, in this thesis, they are proposed for the study of SCHC/Sigfox and its implementations or different configurations. These metrics can also be measured over SCHC deployments over other LPWAN technologies that use the ACK-on-Error mode.

N_U is useful to quantify the impact of aborting SCHC transmissions on the resource usage of the protocol. If measured over a scenario where SCHC transmissions are configured not to be aborted (a non-aborting scenario), it can be compared to a measurement of N_U over normal SCHC operating conditions, i.e., a scenario where transmissions can be aborted as per SCHC Profile specifications. The difference or ratio between both measurements provides a quantification of the number of uplink messages whose transmission was avoided, and it can be extended to either time or energy quantifications.

On the other hand, r_S signals the likeliness and uncertainty of a SCHC transmission to be completed under stable channel conditions, which is a performance metric useful for SCHC deployment considerations. If r_S is too low, or if its associated standard deviation is too high, it may be recommended not to perform a SCHC transmission. Depending on the network scenario, it may be better to wait for better network conditions or to consider another reliable delivery mechanism.

Both performance metrics, especially N_U over a non-aborting scenario, require many uplink transmissions to be performed and measured. This often takes too much time to make these measurements over the network. To avoid spending large amounts of time in gathering the data and calculating these metrics, a semi-empirical model for N_U over a non-aborting scenario and r_S are provided. These models fundamentally assume that the PLR of the network is fixed, and that the event of losing a Sigfox packet (and consequentially a SCHC Fragment) is modeled by an independent and identically distributed (*iid*) random variable. The parameters of the Sigfox SCHC Profile are used as variables, which helps in making these models extensible to other network configurations.

A SCHC/Sigfox implementation was developed over the course of this investigation and the Sigfox SCHC Profile standardization process, which has been useful to obtain insights and new considerations that have been incorporated into the SCHC Profile. The implementation provides a simulation of SCHC/Sigfox transmissions, as well as the software needed to enable SCHC communication over the Sigfox network architecture. This implementation was employed to measure the proposed performance metrics. The simulation environment allows data to be gathered in less time than in a real deployment. Data was also collected from a real deployment, although the sample size obtained is less than what the simulation could provide, given time and bandwidth constraints of the Sigfox technology.

1.2 Hypotheses

The following hypotheses are held for the performance metrics N_U and r_S:

- The average number of uplink messages per completed SCHC transmission is always finite, for every PLR unequal to 100 %. In other words, infinite SCHC transmissions are impossible, since the algorithm is robust enough to always end and does not contain absorbing loops.

- The success rate is independent of the total number of fragments to transmit. This is based on the fact that the only event that aborts a SCHC transmission is the loss of the last SCHC Fragment multiple times, which is thought not to depend on the total number of fragments that are generated from a SCHC Packet.

These hypotheses will be tested by measuring both performance metrics under various conditions. SCHC Packets of different sizes will be transmitted over different induced PLR values. The experiments will be repeated multiple times, gathering data of all SCHC transmissions, which allow both performance metrics and their associated standard deviations to be calculated.

1.3 Objectives

1.3.1 General objectives

The general objective is to define PLR-based performance metrics for SCHC over Sigfox: the average number of uplink messages per SCHC transmission and the success rate of SCHC transmissions, both under the ACK-on-Error configuration of the Sigfox SCHC Profile. Also, this thesis provides semi-empirical models for them. These performance metrics are measured in a simulation of SCHC over the Sigfox network using different packet sizes, and the models are validated by the same procedure. The metrics are also measured in a real deployment of the protocol.

1.3.2 Specific objectives

1. **Performance metrics:**
   (a) Define the average number of uplink transmissions in a SCHC transmission as a performance metric, which quantifies the impact of allowing SCHC transmissions under poor network conditions to be aborted.
   (b) Define the rate of successful SCHC transmissions as a performance metric, which quantifies the likeliness and uncertainty of a SCHC transmission to be complete over known channel conditions, information useful when deciding whether to use SCHC in a particular application.
2. **Model definition:**
   (a) Propose a mathematical model for the expected number of SCHC fragments transmitted, for a single fragmented packet, depending on the PLR of the network and the SCHC Profile parameters.

(b) Propose a mathematical model for the success rate of SCHC fragmented packet transmissions, depending on the PLR of the network and the SCHC Profile parameters.

3. **Validation:**

   (a) Define and execute a set of experiments using the Sigfox SCHC Profile that will provide the data to analyze.

   (b) Compare the empirical results to the performance of the mathematical models to validate their applicability.

4. **Profile definition:**

   (a) Provide a software implementation of SCHC-over-Sigfox, able to execute simulation experiments as well as transmissions over the Sigfox network using the SCHC framework.

   (b) Contribute to the definition of the SCHC-over-Sigfox Profile with insights based on the implementation process, clarifying specifications and adding more useful information needed for implementing and deploying the framework.

1.4 Thesis outline

The remainder of this document is organized as follows:

- Chapter 2 introduces technical concepts such as LPWAN, Sigfox, and SCHC; and provides an overview of related works on SCHC performance measurement and implementations.
- Chapter 3 explains how are the performance metrics defined and modeled, how was the implementation developed and how are the subsequent experiments performed.
- Chapter 4 describes the SCHC-over-Sigfox implementation developed over the course of this investigation.
- Chapter 5 define the performance metrics N_U and r_S, while also providing their respective semi-empirical models.
- Chapter 6 lays out the results obtained from the experiments carried out to calculate the performance metrics and validate their models.
- Chapter 7 analyses the results, evaluating the hypotheses presented in this section and providing guides on how to interpret the performance metrics while adopting SCHC in an application.
- Finally, Chapter 8 concludes this document by summarizing the work and highlighting the most important results and considerations, while also discussing future research on the field.

Chapter 2

Theoretical framework

This chapter addresses the definitions and properties of technical concepts regarding LPWANs, the Sigfox network, and SCHC. Additionally, a review of related articles and projects on SCHC performance evaluation is presented, which gives an overview of the context in which this work is developed.

2.1 Technical Concepts

2.1.1 Low Power Wide-Area Networks

Low-power wide-area networks (LPWAN) are network architectures characterized by their low energy consumption on the transmitter device and the wide coverage they provide, in exchange for constraints in the data rate of the transmission. These types of wireless networks, reviewed in IETF Request for Comments (RFC) 8376 [1], have achieved great interest regarding the development of IoT applications, which deploy transmitting devices in large geographic zones and thus require wide coverage and little human intervention. Low power wireless networks of reduced coverage such as Bluetooth Low Energy (BLE) or ZigBee are also useful for other IoT use cases where coverage is not an issue. These personal-area networks are labeled as wireless personal-area networks (WPAN) and are standardized by the IEEE 802.15 Working Group [13].

Many LPWANs exist and are currently operative, such as LoRaWAN, NB-IoT, Sigfox, Wi-SUN, LTE-M, DASH7, MIoTy, Weightless, and others. The IETF LPWAN Working Group categorizes and describes the first four in RFC 8376 [14] as follows:

- **LoRaWAN** is an LPWAN techology based on the Long-Range (LoRa) physical layer (PHY) modulation scheme, patented by Semtech, which applies chirp spread spectrum (CSS) modulation [15]. LoRa describes the PHY protocol, while LoRaWAN describes the medium access control (MAC) layer and the interconnection protocol to work with Internet [16]. This technology operates in industrial, scientific, and medical (ISM)

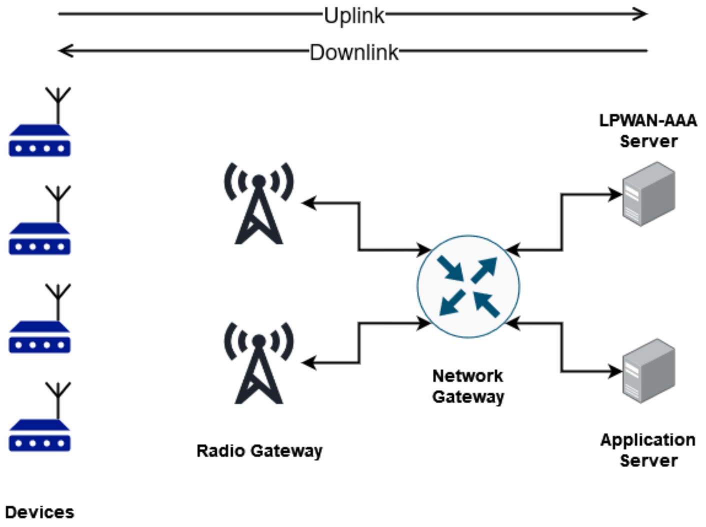


Figure 2.1: Typical LPWAN architecture. Devices communicate wirelessly with radio gateways, which are connected to the Internet via the network gateway. The arrows show the direction of the transmissions.

radio bands of 433 MHz, 868 MHz and 915 MHz, depending on the restrictions of every country.

- **Narrowband IoT (NB-IoT)**, standardized by the Third Generation Partnership Project (3GPP), is another LPWAN technology that provides less device design complexity, low cost, control over battery usage and coverage optimization, inheriting the network architecture of the Long-Term Evolution (LTE) cellular networks. It is promoted by the 3GPP as an integral component of the deployment and development of the fifth generation of mobile communication technologies (5G) [17].

- **Sigfox** is an LPWAN technology owned by the company UnaBiz, and based on ultra-narrow band (UNB) modulation systems. Its low modulation rate achieves wider coverage, and it's ideal for applications that transmit messages comprised of few bytes and with low frequency. Nevertheless, this noticeably limits the data rate of the network [18]. Additionally, this technology possesses different operating bands depending on the geographical zone: Chile is part of the radio configuration (RC) zone 4, where Sigfox transmissions are centered around 920 MHz [2].

- **Wi-SUN Alliance Field Area Networks (FANs)** are defined as IPv6 wireless mesh networks with strong security, robustness, reliability, scalability, and resilience [19]. These networks provide 2–3 km line of sight (LOS) coverage, extendable by means of multi-hop networking; bandwidth of up to 300 kb/s and latency of 0.02 s, supporting LPWAN IoT applications; low power consumption, using less than 2 uA when resting and 8 mA when listening; and scalability, deploying tens of millions of devices in urban, suburban, and rural environments.

These technologies present similar tree or "star-of-stars" topology, as shown in Figure 2.1. The transmitter devices are located at the user end of the network and generally are sensors or actuators capable of transmitting and receiving data. The messages sent by a device are called “uplink” (UL) messages, whilst messages received by devices are called “downlink” (DL) messages. UL messages are received by the radio gateways, antennas designed to capture transmissions performed in specific bandwidths. This data is forwarded via IP towards the final application, which receives and processes the information accordingly. LPWANs also have authentication, authorization, and accounting (LPWAN-AAA) servers, which manage every LPWAN transmission. A DL message is sent back to the device if the application requires it, and it follows the opposite direction. Although the functions of these components are similar, they receive different names in every technology. Table 2.1 details the specific names of LPWAN components.

Function	**IETF**	**LoRaWAN**	**NB-IoT**	**Sigfox**
Sensor or actuator	Device	End Device	User Equipment	End Point
Transceiver antenna	Radio Gateway	Gateway	Evolved Node B (eNodeB)	Base Station
Server	Network Gateway	Network Server	Packet Delivery Network (PDN) Gateway	Service Center
Authentication server	LPWAN-AAA Server	Join Server	Home Subscriber Server	Registration Authority
Application	Application	Application Server	Application Server	Network Application

Table 2.1: Terminology comparison between LPWAN technologies and the general terms used by the IETF. Adapted from [1].

The four previously defined networks are in the scope of the IETF LPWAN working group and have an interest in supporting logical communication with the Internet protocol. The study presented in this thesis focuses on the Sigfox network, which will be described below.

2.1.2 Sigfox

Sigfox is an LPWAN technology highlighted by the wide coverage it provides in exchange for strong restrictions on the data rate of every transmission. Its UNB-based modulation scheme is responsible for this, which can be configured to achieve data transmission rates of 100 bps or 600 bps, depending on the geographical zone. Its MAC mechanism is based on random selection of frequencies and time, which is inspired by ALOHA [20]. The Sigfox radio configurations (RCs) are sets of parameters that control modulation and transmission, such as data rates and central frequencies, which differ according to geographical zones. The southern zone of Latin America and oceanic regions use RC4, and the specific parameters of this zone are shown in Table 2.2.

Parameter	RC4 value
Uplink central frequency	920.8 MHz
Downlink central frequency	922.3 MHz
Uplink data rate	600 bps
Downlink data rate	600 bps

Table 2.2: Specific Sigfox parameters for RC4. Adapted from [2].

The Sigfox network is restricted to at most 140 UL and 4 DL messages per day, which makes its bandwidth a scarce resource that should be considered when deploying the network. UL messages may contain up to 12 B, and DL messages always contain 8 B. A device can request a DL message by means of an UL message with a specific flag, opening a reception window and waiting for the DL message to arrive. This is the only way a DL message is generated in the network.

UL messages are composed of a preamble, a frame synchronization field, a device identifier, the payload, an authentication code, and an error-detecting frame check sequence (FCS), the latter being verified by a cycling redundancy check (CRC) algorithm. DL messages have a similar structure, although the device identifier is replaced by an error-correcting code (ECC) field. These messages and the size of their sections are shown in Figure 2.2.

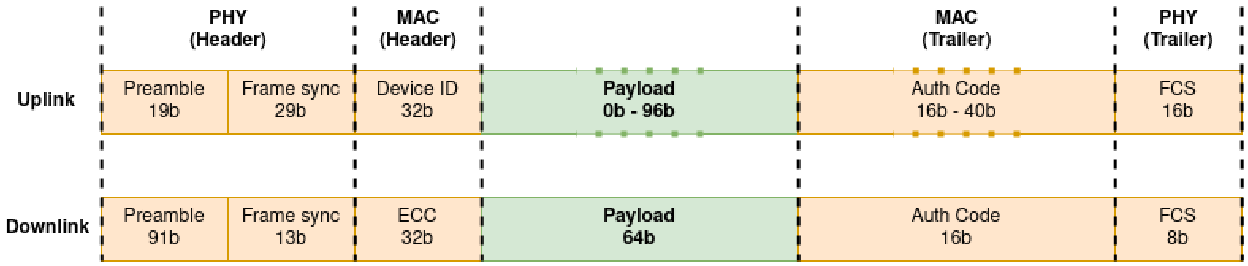


Figure 2.2: Sigfox uplink and downlink message format. Their fields and the size in bits of the fields are shown. The layers the fields belong to, according to the Open Systems Interconnection (OSI) model, are also shown. Adapted from [1].

The information received by the Sigfox network is stored in its backend, a cloud server equivalent to the network gateway of LPWAN architecture, also called service center. Every Sigfox user can configure the behavior of the backend regarding a group of devices. The messages received by the backend can be forwarded towards web applications using two application programming interfaces (APIs): the callback API, which forwards the messages to the application server using the hypertext transfer protocol (HTTP) as soon as they are received; and the representational state transfer (REST) API, which performs queries directly to the database associated with each device.

2.1.3 Static Context Header Compression and Fragmentation

SCHC (pronounced "*sheek*" [21]) is a standard presented in IETF RFC 8724 [5] which describes compression and fragmentation mechanisms for LPWAN packets larger than the maximum packet size allowed by LPWAN technologies. This protocol provides an adaptation layer between upper-layer protocols with large headers and the underlying LPWAN technology. To do so, SCHC compresses the headers of the upper-layer message and fragments it into smaller datagrams if the result of the compression still exceeds the MTU of the network.

The compression and decompression (C/D) mechanism uses a static set of parameters—a static context—in the transmitter device and in the network infrastructure. The result of the compression mechanism is called a SCHC Packet. When received, the SCHC Packet is decompressed to recover the original packet.

The fragmentation and reassembly (F/R) mechanism acts only if, after compression, the SCHC Packet still exceeds the maximum LPWAN packet size. The information is then sectioned into tiles of a specified size and sent over various messages, called SCHC Fragments. Each of these fragments contains at least one tile. When all SCHC Fragments are received, they are reassembled into the original SCHC Packet.

F/R mechanism

The fragments generated in the fragmentation process are grouped in windows and are identified by both the number of the window they belong to and their fragment compressed number (FCN), a sequence number. Both values are represented as bits and are assigned in descending order. The last message of every non-final window has an FCN comprised only of zeroes and is called an All-0 fragment. To identify the last fragment of a transmission (the last fragment of the last window), its FCN is overwritten with only ones, and is called an All-1 fragment. A "regular" SCHC Fragment is a fragment that is not an All-1.

Since the F/R mechanism requires sending many LPWAN messages, SCHC also defines operating modes to guarantee the delivery of these packets to the network, using acknowledgement (ACK) messages, like those used by automatic repeat request (ARQ) mechanisms. These modes are:

- **ACK-Always Mode**: Window numbers and FCNs are added to each fragment, which lets the receiver identify the received fragments and send a DL ACK every time the transmission reaches a fragment located at the end of a window. The ACK contains information regarding received packets. Upon reception of the ACK, the transmitting device can read its information and determine if fragments were lost, retransmitting them afterwards if needed. The All-1 is always responded with an ACK from the receiver side.
- **ACK-on-Error Mode**: Similarly to the ACK-Always mode, the receiver identifies the fragments using their headers, but it also has the ability to detect losses. After sending a certain number of fragments, the transmitter sends a message requesting a SCHC ACK (called SCHC ACK REQ messages), after which the receiver responds only if it has detected lost fragments. Otherwise, the transmission can continue normally. Again, the All-1 is always responded with an ACK from the receiver.
- **No-ACK Mode**: This mode does not guarantee the delivery of packets, but minimizes the header overhead introduced by the fragmentation sublayer, since it does not carry sequence numbers larger than one bit.

SCHC ACKs carry the information of received and lost fragments of a particular window. This information often uses little space in a DL frame, saving network resources in networks with variable DL message sizes. However, in networks where the DL message size is fixed (such as Sigfox), a single SCHC ACK can leave large amounts of unused space. To take advantage of the unused space of DL messages, the SCHC Compound ACK was defined as an extension of the regular SCHC ACK that can carry the information of received and lost fragments of many windows if needed [6].

SCHC Fragments and SCHC ACKs possess headers composed of the following fields, which are shown in Figure 2.3:

- **Rule ID**: The identifier of a Rule, a set of parameters pertaining to a SCHC transmission which specify the F/R mode to use, the reassembly mechanism algorithm, and if the fragment contains additional fields. Its size is denoted by `RULE_ID_SIZE`.
- **Datagram Tag (DTag)**: An identifier for fragments that are part of different SCHC Packets that are sent using the same Rule ID. This enables interleaving in SCHC transmissions. Its size is denoted by T.
- **Window (W)**: If windows are used, the W field carries the bit representation of the number of the window that the fragment belongs to. Its size is denoted by M. The maximum number of windows allowed is 2^M.
- **Fragment Compressed Number (FCN)** (only in fragments): The sequence number of every fragment relative to the window they belong to. The FCN comprised only by zeroes and only by ones are reserved, called All-0 and All-1 respectively. Its size is denoted by N. The maximum number of fragments in a window is $2^N - 1$, since the All-1 is only used for the very last fragment.
- **Reassembly Check Sequence (RCS)** (only in fragments): Redundancy bits added to detect errors, sent only in the All-1 fragment. Its size is denoted by U.
- **Integrity Check (C)** (only in ACKs): A one-bit field that indicates if an integrity check was performed over the reassembled SCHC Packet. It is 1 if the check was performed and successful, and it is 0 if it was failed or not performed.
- **Compressed Bitmap** (only in ACKs): A bit string used to identify received tiles. A 1 in position i of the bitmap indicates that the i-th tile was received, and a 0 indicates that it wasn't received or that it was dropped. Its size is denoted by `WINDOW_SIZE` and equals the maximum number of tiles that can be carried in a single window, which can be equal to or less than $2^N - 1$.

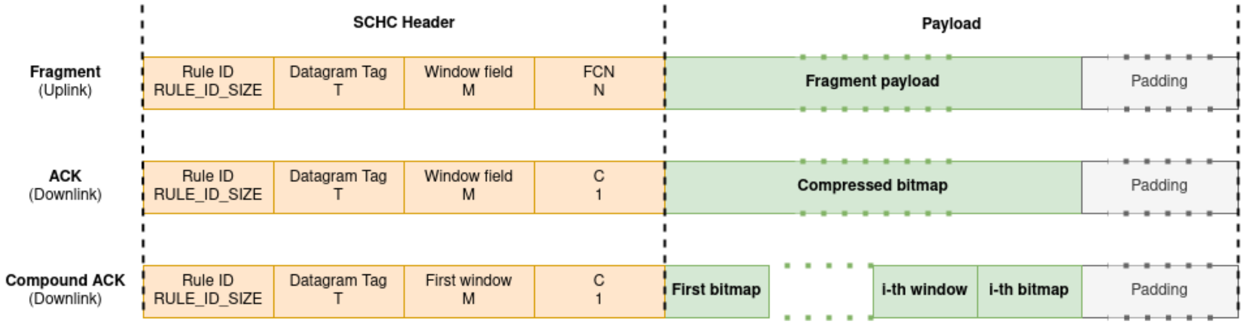


Figure 2.3: SCHC Fragment, SCHC ACK and SCHC Compound ACK message format. Their fields and the name of the variable that denotes their size in bits are shown. Adapted from [5] and [6].

Timers and timeout values are incorporated in SCHC, making it possible to decide whether a message is lost or not. The Inactivity Timer keeps track of the time that the receiver has waited for a new SCHC Fragment since the last received fragment of a particular SCHC transmission. If no SCHC Fragment is received after its timeout value, the receiver automatically aborts the transmission by means of sending a SCHC Receiver-Abort message to the device whenever possible. On the other hand, the Retransmission Timer keeps track of the time that the sender device has waited for an ACK since requesting one by sending a SCHC ACK REQ or an All-1 fragment. If no ACK is received after its timeout value, the sender requests the ACK again. This process can be repeated until reaching a maximum of attempts in a row, noted `MAX_ACK_REQUESTS` or $R_{\max}$. If this limit is reached, the sender automatically aborts the communication by means of sending a SCHC Sender-Abort message.

SCHC over Sigfox

To use SCHC, an LPWAN technology must define its own SCHC Profile, a set of configurations and parameters used by SCHC to perform the F/R procedures. Since the C/D procedure always has static context, it is independent of the SCHC Profile. At the time of writing, SCHC/Sigfox is proposed as an Internet-Draft of the IETF [3], and specifies the functioning of the F/R mechanism. Here, it is considered that one SCHC Fragment carries only one tile. Moreover, a single SCHC Fragment is sent in a single Sigfox packet. Therefore, the rate at which SCHC Fragments are lost (fragment loss rate, FLR) and the rate at which Sigfox packets are lost (packet loss rate, PLR) are equivalent terms. Since the Sigfox network has a fixed DL message size, the SCHC Compound ACKs are favored instead of regular SCHC ACKs. Furthermore, Sigfox considers no SCHC C/D procedure.

Considering DL restrictions in Sigfox, this SCHC Profile favors the ACK-on-Error F/R mode, thereby using DL resources only when necessary. It is also specified that All-0 fragments are processed similarly to SCHC ACK REQs: at the end of every window, an ACK is requested to be sent only if errors were detected; however, if this ACK is not received by the sender device, it continues sending the next window of fragments instead of retransmitting the All-0.

After receiving an All-1 fragment, the receiver must always respond with an ACK. If the All-1 or the ACK are lost or delayed, the sender device sends the All-1 again. If the ACK reports missing fragments, the sender retransmits them and sends the All-1 again, repeating the process. Finally, if the ACK does not report losses, the transmission is completed.

The sender device must maintain a counter of the number of times it has sent an ACK REQ. This is called an “attempts counter” [5], and it increases monotonically until reaching $R_{\max}$, when a SCHC Sender-Abort must be sent. The Sigfox SCHC Profile extends this part of the sending algorithm, stating that the SCHC Sender-Abort is sent if the number of repeated All-1 fragments sent in sequence reaches $R_{\max}$. This requires that the attempts counter be reset every time an ACK is received[1].

[1] At the time of writing, this is specified in Section 3.5.1.1 of the Sigfox SCHC Profile specifications [3]. The original behavior is specified in Section 8.4.3.1, paragraph 17.1, of RFC 8724 [5] and does not specify resetting the attempts counter.

Operational modes

Sigfox defines three sets of SCHC parameters for UL ACK-on-Error fragmentation, which are thought for different SCHC Packet sizes. These operational modes vary in the size of the header fields of regular SCHC Fragments and the All-1 fragment: SCHC Fragments must have a header large enough to carry the Rule ID, DTag, Window and FCN fields, whilst the All-1 must have a header large enough to carry the Rule ID, DTag, Window, FCN and RCS field. In All-1s, if the resulting header length is not a multiple of the L2 Word Size (1 B), zero-padding is added as needed. The specific parameters for each operational mode are shown in Table 2.3.

Parameter	**Meaning**	**Value (1-byte header)**	**Value (2-byte header op.1)**	**Value (2-byte header op.2)**
Maximum SCHC Packet size	Maximum size of a processable SCHC Packet	300 B	480 B	2400 B
L2 Word Size	Minimum information unit	8 b	8 b	8 b
Rule ID Size	Rule ID field size	3 b	6 b	8 b
`T`	DTag field size	0 b	0 b	0 b
`M`	Window field size	2 b	2 b	3 b
`N`	FCN field size	3 b	4 b	5 b
`U`	RCS field size	3 b	4 b	5 b
`MAX_ACK_REQUESTS`	Maximum number of times that an ACK can be requested in a row	5	5	5
`WINDOW_SIZE`	Maximum number of tiles per window	7	12	31
Regular header size	Maximum size of the header of a regular SCHC Fragment	1 B	2 B	2 B
All-1 header size	Maximum size of the header of an All-1 SCHC Fragment	2 B	2 B	3 B
Regular tile size	Maximum size of each tile	11 B	10 B	10 B
All-1 tile size	Size of the last tile	0 to 10 B	1 to 10 B	0 to 9 B
Retransmission Timeout	Maximum time the sender can wait for an ACK	12 h (recommended)	12 h (recommended)	12 h (recommended)
Inactivity Timeout	Maximum time that the receiver can wait for a new fragment	12 h (recommended)	12 h (recommended)	12 h (recommended)

Table 2.3: Sigfox SCHC Profile parameters for the ACK-on-Error F/R mode. If a field has size 0, it is not present in the message. Adapted from [3].

At the time of writing, these operational modes are presented in Section 3.6 of the Sigfox SCHC Profile specifications [3], and are summarized as follows:

- The **Single-byte SCHC Header** mode enables transmissions of SCHC Packets of up to 300 B. Regular SCHC fragments have a header size of 1 B (with no padding bits) whilst the All-1 fragment has a header size of 2 B (with 5 zero-padding bits).
- The **Two-byte SCHC Header Option 1** mode enables transmissions of SCHC Packets of up to 480 B. Regular SCHC fragments have a header size of 2 B (with 4 zero-padding bits) and the All-1 fragment also has a header size of 2 B (with no padding bits).
- The **Two-byte SCHC Header Option 2** mode enables transmissions of SCHC Packets of up to 2400 B. Regular SCHC fragments have a header size of 2 B (with no padding

bits) whilst the All-1 fragment has a header size of 3 B (with 3 zero-padding bits). This operational mode satisfies the IPv6 minimum MTU requirement.

In the Single-byte SCHC Header and Two-byte SCHC Header Option 2 modes, the size of the All-1 header is greater than the size of the header of a regular fragment, which entails a smaller payload field for the All-1 fragment. Since the All-1 cannot carry a payload of the size of a regular tile, if the SCHC Packet to be processed has a size that is multiple of the regular tile size, the All-1 produced by the fragmentation process of these modes carries no payload.

Sender and receiver algorithms

This section summarizes the algorithms followed by both the sender and receiver part of a SCHC transmission. These algorithms do not fully describe the actual SCHC algorithms described in [5], but are sufficient to understand the protocol.

The algorithm followed by the sender part of a SCHC transmission using the Sigfox SCHC Profile can be summarized as the following sequence of steps:

1. Divide a SCHC Packet into F fragments, and group them into W windows of maximum size S. Set the current window counter to $w = 1$ and the attempts counter to $R = 0$.
2. Transmit the w-th window and wait for a Compound ACK.
3. If $w < W$,
    (a) If a Compound ACK is not received after a certain amount of time, update $w = w + 1$ and continue transmitting the next window.
    (b) If a Compound ACK is received, retransmit the missing fragments reported in the Compound ACK, then update $w = w + 1$ and continue transmitting the next window.
4. If $w = W$,
    (a) Increase the attempts counter by 1, updating $R = R + 1$.
    (b) If $R = R_{\text{max}}$, send a SCHC Sender-Abort and exit the SCHC transmission with an error condition.
    (c) If a Compound ACK is not received after a certain amount of time, send the last fragment again to wait for a Compound ACK and repeat step 4.
    (d) If a Compound ACK is received,
        i. Reset the attempts counter, setting $R = 0$.
        ii. If losses are reported, retransmit the missing fragments and send the last fragment again to wait for a Compound ACK and repeat step 4.
        iii. If no losses are reported, exit the SCHC transmission.

On the other hand, the algorithm of the receiver part of the algorithm can be summarized in the following steps:

1. Upon reception of a new fragment, check if the inactivity timer has expired. If it has, respond with a SCHC Receiver-Abort whenever possible, exiting the SCHC transmission.
2. Store the fragment in memory.
3. If the fragment was previously requested for retransmission, wait for a new fragment.
4. If it was not,
   (a) If the fragment is not located at the end of a window, wait for a new fragment.
   (b) If the fragment is located at the end of any window, check for lost fragments.
      i. If losses are detected, respond with a Compound ACK reporting all lost fragments and then wait for a new fragment.
      ii. If no losses are detected,
         A. If the fragment is located at the end of the last window, respond with a Compound ACK that reports no losses, and then reassemble the SCHC Packet using the fragments in memory, exiting the SCHC transmission.
         B. If the fragment is not at the end of the last window, continue waiting for a new fragment.

The Sigfox ACK-on-Error sending algorithm is modeled as a system that consist of transmission cycles, intermediate retransmission cycles, and final retransmission cycles [22]. The sender starts in a transmission cycle, sending every fragment of the first window. If an ACK is received at the end of the window, the lost fragments reported in the ACK are sent in an intermediate retransmission cycle. This process is repeated for every window until reaching the last one. After sending the last fragment—the All-1 fragment—, if an error-reporting ACK is received then the sender retransmits lost fragments in a final retransmission cycle, which ends by sending the All-1 fragment once more. The final retransmission cycle is repeated each time an error-reporting ACK is received, and finalizes when an ACK that doesn't report losses is received.

2.2 Related Work

2.2.1 Network performance studies

The intellectual property restrictions of technologies such as Sigfox or LoRa make it difficult to directly model network performance metrics. These studies often gather performance data experimentally. An experimental study performed in Germany is presented in [23], which compares the LoRaWAN, MIoTy, Sigfox and NB-IoT networks in terms of average signal-to-noise ratio (SNR), received signal strength, packet loss rate (PLR), packet size flexibility and delay in the communication. The PLR is measured only in the uplink direction with a fixed packet size, where Sigfox shows the most resistance to signal deterioration due to distance. However, the packet size flexibility of Sigfox is limited and widely surpassed by that of LoRaWAN.

Although the low power consumption requirement demands low data rates in LPWANs, the use of ISM bands is subject to legal regulations that may vary between geographical zones. The impact of such limitations is reviewed in [24], addressing technologies such as LoRaWAN, Sigfox, IEEE 802.15.4 and DASH7. It is noted that these regulations vary on various levels, since multiple institutions are responsible for defining them. This study is focused on European Union regulations, which rely on duty cycle and maximum transmission power.

2.2.2 Fragmentation in LPWAN

Fragmentation mechanisms have been of interest in LPWANs since the packet size restrictions of most LPWANs impedes large amounts of data from being sent, which is of interest in IoT applications. Packet fragmentation, however, comes at the expense of increased energy consumption, communication overhead, and access attempts in the network. These drawbacks are studied in [25], where it is found that packet fragmentation increases communication reliability, an important network parameter for dense networks.

An aggressive fragmentation scheme using group NACKs for constrained LPWANs is proposed in [26], which provides increased goodput and energy efficiency. In this scheme, a packet is fragmented even if its length is less than the network MTU. Different strategies according to the network density are provided, showing that no fragmentation is preferable in small networks and IoT applications that require sending packets smaller than the proposed fragment sizes. This fragmentation scheme was developed using the NS3 simulator and is based on the buffered ALOHA framework.

2.2.3 SCHC definition

The SCHC standard has been drafted since September 2016, and was standardized by the IETF in April 2020 as RFC 8724 [5]. The definitions of the SCHC Profiles for LPWAN technologies start being developed as Internet-Drafts, preliminary documents of standards in process, which after being formalized and reviewed are published as RFCs. This is the case of the SCHC Profile of LoRaWAN, published as the RFC 9011 [8]. CoAP is an application protocol oriented to microcontrollers of constrained memory which uses the REST software architecture [27], and its specifications to make SCHC applicable over it have also been published [28]. At the time of writing, the SCHC Profile of Sigfox is in its final stages to become a Proposed Standard and is available as an Internet-Draft in [3].

Acklio is a company dedicated to the development of IoT applications, has been the main driving force behind the development of SCHC as a standard [21]. It its web page dedicated to SCHC, they mention the challenges of using the Internet protocol over LPWANs, which arise from the fact that the minimum amount of information needed to enable IPv6 communication—the IPv6 minimum MTU requirement—is larger than the amount of information that most LPWANs can carry in a single packet. They emphasize that the principle of SCHC is to allow full compatibility with IP for LPWAN technologies, making these networks interoperable.

SCHC is presented as a protocol with potential in the development of IoT thanks to the C/D and F/R mechanisms, and several challenges in the adoption of this standard have been discussed. The difficulty of optimizing the parameters of SCHC Profiles is addressed in [29], which arises since every network has its own configurations and limitations regarding data rates and energy consumption. Therefore, SCHC was defined as a generic framework, able to be implemented in many technologies such as LoRaWAN, Sigfox, NB-IoT and low-rate wireless personal-area networks (LR-WPAN), the latter being defined in the IEEE 802.15.4 standard. Furthermore, it is mentioned that the compression mechanism was designed with IPv6, Used Datagram Protocol (UDP) and CoAP in mind, but is extendable to other protocols, including those employed in non-IP-based traffic [12, 9].

Regarding the fragmentation and reliable delivery of fragments, in [29] it is mentioned that, in an ideal case, the reception of many fragments could be confirmed in only one downlink message, but the restrictions in packet size of LPWANs make this task difficult. Therefore, F/R modes are defined to detect fragment losses, and it is also addressed that every SCHC Profile must specify its preferred F/R mode. Finally, security aspects and vulnerabilities of the protocol are addressed and detailed in the RFC.

2.2.4 SCHC performance studies

The definition of SCHC has been studied in few articles from a theoretical point of view with mathematical models that involve the parameters of the protocol, which quantify its performance. In [30], a study regarding the optimization of the ACK-on-Error F/R mode parameters in LoRaWAN and Sigfox is presented. The optimal maximum sizes of fragments, window number fields and sequence number fields are also discussed.

Performance metrics such as the channel occupancy, goodput and time overhead for different F/R modes of SCHC over LoRaWAN are studied in [31]. It is found that No-ACK is the F/R mode with the lowest channel occupancy, highest goodput and lowest time overhead, although it does not provide reliable delivery of SCHC Fragments. It is also found that ACK-on-Error provides the most goodput, whilst ACK-Always has the same total delay and similar channel occupancy as ACK-on-Error.

The study in [32] evaluates an implementation of the C/D mechanism over LoRaWAN. It addresses the compression rate for different parameters of the SCHC Profile of said technology. A similar analysis is presented in [33], which addresses the F/R mechanism transmission times and proposes modifications in the receiver behavior.

The impact of header compression, data rate configuration and fragment size of SCHC over LoRaWAN is addressed in [34]. An improvement in LPWAN reliability for low data rates is highlighted. However, it is noted that fragmentation imposes an additional overhead that limits the overall performance gain. Therefore, it is necessary to consider a trade-off between energy consumption, channel reliability, bandwidth usage and fragment sizes when using SCHC Fragmentation in IoT applications.

A model for channel occupation efficiency for SCHC over LoRaWAN is presented in [35]. The model proves that, for all LoRaWAN spreading factors, the channel efficiency decreases as the probability of losing a SCHC Fragment increases. It is also shown that the efficiency is directly linked to the transmission time used for each SCHC Fragment. Defining the efficiency of the transmission as the ratio between the effective transmission rate and the channel transmission rate, this study finds that the efficiency of SCHC is not linearly affected by the LoRaWAN spreading factor.

The performance of the SCHC ACK-on-Error mode over the Sigfox network is evaluated in [12] in terms of transfer time and messages exchanged. This article presents an analytical approach to the transfer time of a SCHC Packet and provides empirical evaluations of the time and number of messages exchanged for SCHC Packets of different lengths. This study shows that small changes in packet sizes may have a significant impact in the transmission time of a whole SCHC Packet. It is also shown that an increase in the fragment loss rate may reduce the transmission time, since retransmissions can be performed in less time than the time used to wait for an ACK when there are no errors.

A model for the energy consumption of SCHC transmissions over the Sigfox network is presented in [36]. The model is based on hardware measurements and determines the impact of different parameter and algorithm configurations on the energy consumption of SCHC over Sigfox, providing a relation between the different parameters of SCHC/Sigfox and the device lifetime. This study finds that average current consumption of transferring a SCHC Packet decreases as the size of the SCHC Packet increases.

SCHC uses a compressed bitmap that contains the information of lost fragments in its ACKs, which is an example of a receiver-feedback technique (RFT). The study in [37] evaluates different RFTs for SCHC, such as a list of lost fragments, a list of deltas and an uncompressed bitmap. These RFTs are tested over LoRaWAN in terms of ACK payload size, number of L2 frames per ACK, and the time on air (ToA) of each ACK. It is concluded that RFTs that are different from a compressed bitmap offer performance improvement in many scenarios.

The studies presented in this section help in understanding the different configurations of SCHC and provide useful insights on the efficiency of the compression algorithms and the time spent in SCHC transmissions. However, at the time of writing there is no study regarding the rate of completed SCHC transmissions nor the total number of fragments sent per SCHC transmission for a certain PLR, metrics that would be directly related to the applicability of SCHC in constrained networks. A state machine for the Sigfox implementation of ACK-on-Error is presented in [22] and is proposed to carry a probabilistic analysis over these performance metrics.

The studies presented in this section are categorized in Table 2.4, indicating the LPWAN technology that they employed, whether the study is theoretical or empirical, whether packet losses are considered in the model definitions, and which performance metrics are addressed.

Reference	Year	LPWAN Technology	Type of study	Packet losses?	Performance metric
[31]	2019	LoRaWAN	Empirical (Simulated)	No	Channel occupancy Goodput Total delay
[32]	2019	LoRaWAN	Empirical	No	Compression rate
[33]	2019	LoRaWAN	Empirical	No	SCHC Fragment transmission times
[30]	2020	Sigfox, LoRaWAN	Theoretical, Empirical	Yes	Number of ACKs required
[34]	2020	LoRaWAN	Empirical	Yes	Delivery ratio Time overhead Payload overhead
[12]	2021	Sigfox	Theoretical, Empirical	No	SCHC Packet transfer time Number of total uplink messages Number of total downlink messages
[35]	2022	LoRaWAN	Theoretical	Yes	Channel efficiency
[36]	2022	Sigfox	Theoretical	No	Energy consumption
[37]	2022	LoRaWAN	Empirical (Simulated)	No	ACK payload size L2 frames per ACK ACK ToA
This study	2023	Sigfox	Theoretical, Empirical	Yes	Uplink messages per completed SCHC transmission Rate of completed SCHC transmissions

Table 2.4: Notable SCHC performance studies to the date of writing.

2.2.5 SCHC implementations

OpenSCHC is an open-source project written in the Python programming language, which aims to implement SCHC in a stable way. It is developed by contributions in the GitHub platform [38]. This code has not been fully adapted to other LPWAN technologies aside from LoRaWAN. There also exists a complete implementation for LoRaWAN [39, 35], PySCHC, which was used to evaluate the performance of SCHC over LoRaWAN theoretically and experimentally.

Inspired by the implementation presented in [39, 35], an implementation of the F/R mechanism for Sigfox was devised [11, 40]. This Python implementation aims to provide an interface between SCHC and Sigfox [40]. It encapsulates every component of a SCHC transmission as a set of parameters and methods. Furthermore, it provides scripts that use the Flask framework to simulate SCHC/Sigfox transmissions. The sender side of this implementation is deployable over Pycom LoPy4 devices, whilst the receiver side is deployable over Google Cloud Platform (GCP). The analysis of SCHC presented in this thesis uses this code for carrying experiments out. This implementation was developed over the course of this study and the definition of SCHC/Sigfox, and is described in detail in Chapter 4.

The implementation and simulation of SCHC/Sigfox developed over the course of this investigation has been used to contribute in the Sigfox SCHC Profile and the SCHC Compound ACK standards. It has also been employed to perform performance evaluation studies and theoretical modeling. Publications derived from this work are laid out in Appendix A.

Chapter 3

Methodology

This section provides an overview on the methods used in developing the SCHC-over-Sigfox implementation, defining the performance metrics, and validating the semiempirical models.

3.1 Software development process

Over the course of both the standardization of SCHC/Sigfox and this investigation, an implementation was devised. This implementation has been developed with close guidance from SCHC/Sigfox coauthors. It was first presented in [11], was employed in [12], and is publicly available at [40]. This implementation provides code for the sender side of the communication, the receiver side of the communication, and a local simulation of the whole system. The sender code is executable in Pycom LoPy4 devices, the receiver code is deployable over GCP Cloud Functions and Firebase Realtime Database, and the simulation is executable in a Python 3.9 environment. The Python language was chosen to develop the implementation since the LoPy4 is programmable in a Python implementation, Micropython, and GCP allows Python code to be executed in Cloud Functions.

The code was developed by following a class-based approach, where most of the sending and receiving logic is encapsulated into objects consisting of parameters and methods. The constant modifications of the SCHC/Sigfox standardization process required stability and resilience throughout the code development, and constant validation of all changes incorporated in each modification. Therefore, a test-driven development (TDD) scheme was carried out, in which methods and algorithms are formulated as test cases before being implemented and deployed.

Since the receiver side of the project is meant to be run over Cloud Functions and Firebase Realtime Database, the receiver side of the simulation was developed in a way that emulates how both platforms work: A local Flask server is run, which executes memory-less HTTP functions every time a message is received. A nested JSON object is used as storage, which emulates how Firebase Realtime Database works. On the other hand, the sender side of the project uses Pycom libraries for Sigfox communication. Its functioning is emulated in the simulation environment by objects that encapsulate the functions of a Python library for HTTP communication, `requests`, which directly communicates with the HTTP endpoint of the receiver side.

The project is composed of four different environments: the simulation environment, a partial Realtime Database integration environment, the GCP (receiver) environment, and the LoPy4 (sender) environment. When the project requires modifications, these environments are developed in sequence, as shown in Figure 3.1. The simulation was the first environment to be developed, and all changes are first implemented and tested there. After the implementation of new features or changes, these are validated by running unit tests: individual tests of all methods and modules present in the code. After unit test validation, the simulation can be run as a local client-server application. Special tests for known corner cases are run to verify the functioning of the sender and receiver algorithms.

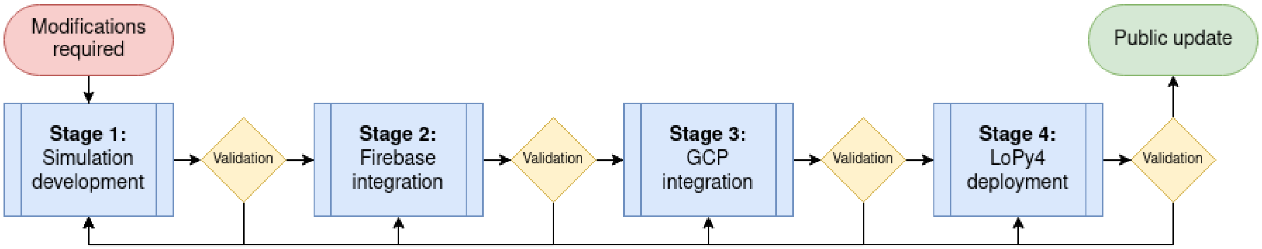


Figure 3.1: Development procedure of the SCHC/Sigfox implementation. When validations are successful, the changes are adapted into the next environment. If validation fails, development falls back to where the failure occurred and the process is carried out again.

If the simulation validation is satisfactory, its changes are then applied to a partial integration with Firebase Realtime Database, which is a Google platform used by the receiver side as storage. The code is validated again by means of unit testing and corner tests cases verification, to correct potential errors that were introduced in the integration process. If this validation is successful, the changes of the receiver side of the simulation are applied to the receiver environment.

The receiver environment integrates Cloud Functions with Firebase Realtime Database. After applying the necessary changes, the code is deployed into Cloud Functions and a uniform resource identifier (URI) of its HTTP endpoint is obtained. To communicate with the receiver, an HTTP POST request must be sent towards that URI. The sender side of the simulation is modified to send an HTTP POST request to the receiver endpoint. Again, tests are carried out to verify the functioning of the receiver side deployment.

Finally, the sender side of the simulation is adapted into Micropython. Micropython is a Python-based programming language for microcontrollers, used by the Pycom LoPy4 device as a programming interface [41, 42]. The LoPy4 board is enabled to send messages over the Sigfox network using the Sigfox wireless modulation. A Sigfox HTTP callback is configured to forward the messages received by the Sigfox base stations to the receiver endpoint as an HTTP request. A final verification of the code is done by performing SCHC transmissions over the Sigfox network and observing the behavior of all algorithms. If this validation is satisfactory, the code is publicly updated in the GitHub repository.

3.2 Performance metrics

Two performance metrics are proposed in this work: the average number of uplink messages per SCHC transmission, N_U, and the rate of successfully finished SCHC/Sigfox transmissions, r_S. Both performance metrics are to be measured empirically, and a semi-empirical approach is performed to provide a mathematical approximation for their values. N_U is calculated performing n SCHC transmissions, counting the number of uplink messages sent until the transmission is either completed or aborted, and then calculating the average over the n samples. The success rate r_S is calculated by performing n SCHC transmissions, counting the number of completed (not aborted) transmissions and dividing it by n. Both metrics depend on the rate at which SCHC Fragments are lost—the FLR—, the size L of the SCHC Packet, and other network parameters. Since SCHC/Sigfox specifies that a single SCHC Fragment is always carried in a single Sigfox packet, the FLR is equivalent to the uplink PLR of the Sigfox network.

3.2.1 Justification

N_U is proposed as a performance metric that quantifies the impact of the maximum number of ACK requests sent in a row, `MAX_ACK_REQUESTS` or $R_{\max}$, over network resources usage. This requires N_U to be calculated in two scenarios: a scenario where aborting a transmission is not possible (a non-aborting scenario), and the scenario where transmissions are aborted if $R_{\max}$ is reached (the base scenario).

The calculation of N_U over a non-aborting scenario requires that the $R_{\max}$ limit be bypassed: Instead of sending a SCHC Sender-Abort message after the attempts counter reaches $R_{\max}$, the SCHC transmission is continued normally until SCHC Packet reassembly is possible at the receiver end. This requires excessive use of network resources for high FLR values, which is why, in this study, it is only calculated over a simulation environment and not the actual network. Its comparison respect to N_U when calculated over the base scenario indicates the amount of uplink messages whose transmission was avoided.

The calculation of N_U over the base scenario, aside from being used for comparison with the non-aborting scenario, is a metric of interest in comparing the ACK-on-Error F/R mode to other reliable delivery methods. Reliable delivery methods use control messages and retransmissions that inevitably add additional message and time overhead. Particularly, the ACK-on-Error F/R mode retransmits every lost fragment until it is received, which requires sending an additional uplink message each time. Furthermore, the last SCHC Fragment generated from a SCHC Packet requests an ACK after every final retransmission cycle, which adds additional overhead.

On the other hand, the success rate r_S is a metric of interest since, when calculated with a large enough sample size, it signals the likeliness of a SCHC transmission to be completed in stable channel conditions. Its calculation is only performed over the base scenario. When network conditions are insufficient for SCHC transmission completion, the transmission is aborted by means of the SCHC Abort messages, which make the transmissions end prematurely and trigger the deletion of all SCHC Fragments processed up to that point.

This potentially rules out the possibility of recovering the information received in incomplete SCHC transmissions. Given the constraints of the Sigfox network in terms of bandwidth usage, knowing the success rate of SCHC transmissions given certain channel conditions is a crucial metric for determining whether incorporating SCHC in a Sigfox application is worth its bandwidth usage.

3.2.2 Modeling process

Both metrics are modeled in Chapter 5 in a similar way. The model for N_U is defined for the non-aborting scenario, whilst the model for r_S is defined for the base scenario. First, a mathematical justification for these metrics to be used as predictions of their theoretical counterparts is provided. Random variables that correspond to a single sample of these metrics are defined. Then, a sequence of n realizations of these variables is defined, and by applying the strong law of large numbers it is concluded that as n increases, these performance metrics converge almost surely to their theoretical values: the expected number of uplink messages in a SCHC transmission and the success probability, respectively.

Second, a probabilistic analysis is carried out. N_U over a non-aborting scenario requires that all fragments be correctly delivered to the receiver side, and r_S requires that the All-1 fragment attempts counter does not exceed $R_{\max}$. This analysis provides the theoretical component of both metrics, $N_{U_T}(p)$ and $r_{S_T}(p)$ respectively.

Third, the theoretical components are compared with preliminary results obtained in the simulation of SCHC/Sigfox, where a fixed FLR is induced. The shape of the absolute difference curves is analyzed, and candidate functions are proposed as empirical adjustments. These functions are shaped by various parameters, which are optimized using the least squares method (LSM) and then adjusted by inspection not to exceed theoretical limits. This analysis provides the empirical component of both metrics, $N_{U_E}(p)$ and $r_{S_{E(p)}}$ respectively.

Finally, the semi-empirical models for both performance metrics are defined as the sum of their theoretical parts and their empirical parts. These models are later validated by comparing them to experimental results in terms of the absolute difference between the models and the empirical calculations of the metrics.

3.3 Data collection and analysis

3.3.1 Experimental design

A SCHC transmission consists of sending and retransmitting SCHC Fragments pertaining to the same SCHC Packet until all fragments are correctly delivered to the receiver or until the transmission is aborted by the algorithm. To obtain data for the proposed performance metrics, several repetitions of SCHC transmissions for different values of SCHC Packet size and FLR are performed.

Experiments were performed in the simulation environment and over the Sigfox network. Due to the simulation environment being a local client-server application, the transmission times are much faster than in the real deployment. This makes it possible to configure ACK timeout values to minimal values, which provides faster results. On the other hand, the real deployment has longer transmission times and timeout values, which makes it difficult to perform the same amount of SCHC transmissions in both environments within the same time. In the simulation, both `SIGFOX_DL_TIMEOUT` and `RETRANSMISSION_TIMEOUT` were configured to 0.1 s, whilst in the real deployment both timeout values are set to 60 s. The change in the timeout value over the simulation environment is not thought to impact these performance metrics. An additional set of experiments with a timeout of 1 s was performed to calculate r_S to support this claim.

This work employs the 1-byte header operational mode of SCHC/Sigfox. This operational mode is able to perform SCHC transmissions from 1 fragment up to 28 fragments. Eight evenly-spaced SCHC Packet sizes are selected in the range of those admitted by this operational mode. Other operational modes of SCHC/Sigfox were not tested since a larger number of SCHC Fragments implies larger transmission time of SCHC Packets, both in the simulation environment and in the real deployment.

In the simulation environment, SCHC transmissions are performed for ten evenly-spaced values of the FLR, ranging from 0% to 90%. An FLR value of 100% was not tested since it trivially produces infinite SCHC transmissions while measuring N_U and also produces a null r_S. A number of 10,000 SCHC transmissions were performed for each combination of SCHC Packet size and FLR value. Such a large number of experiments was chosen to obtain stability of average values and standard deviations, achieving statistical significance and validating the use of the law of large numbers in the reasoning behind the semi-empirical models. These benefits are obtained by taking advantage of the reduced amount of time needed for each experiment in the simulation.

On the other hand, as the FLR is not directly controllable in the real deployment, SCHC transmissions were performed with the device located in three different positions inside the laboratory the experiments were carried out in. Spatial distribution of the transmitter device and the surrounding environment impacts channel conditions and the FLR, which is calculated after obtaining the results. In this configuration, 100 SCHC transmissions were performed for each combination of SCHC Packet size and device location. Although this number of experiments is far smaller than the number of experiments carried out in the simulation (and may not achieve statistical significance), it was chosen since SCHC transmissions in the real deployment are restricted in bandwidth usage and take significantly larger amounts of time. Moreover, results from experiments over a real deployment of SCHC/Sigfox are needed to make a comparison with the results of the simulation.

The selected values of the SCHC Packet sizes and number of repetitions performed to calculate N_U and r_S are reported in Table 3.1 for the simulation environment, and in Table 3.2 for the deployment over the Sigfox network. Experiments add up to 800,000 SCHC transmissions in the simulation environment, and up to 2,400 SCHC transmissions in the real deployment.

Although the input of the SCHC implementation is a SCHC Packet, the results are

SCHC Packet size [B]	Fragments	Windows	Repetitions	Total SCHC transmissions
1	1	1	10,000 per FLR value in $[0\%, 10\%, \cdots, 90\%]$	800,000
45	5	1		
88	9	2		
132	13	2		
176	17	3		
220	21	3		
263	24	4		
307	28	4		

Table 3.1: Configuration of the SCHC Packet size and FLR values for each experiment in the simulation environment.

SCHC Packet size [B]	Fragments	Windows	Repetitions	Total SCHC transmissions
1	1	1	100 per run	2400
45	5	1		
88	9	2		
132	13	2		
176	17	3		
220	21	3		
263	24	4		
307	28	4		

Table 3.2: Configuration of the SCHC Packet size for each experiment in the real deployment environment.

shown in relation to the number of fragments. This is because each number of fragments comprehends a range of SCHC Packet sizes. Moreover, the SCHC sending and receiving algorithms only interact with fragments. Additionally, although a SCHC Packet size of 307 B was tested (which is greater than the maximum packet size recommended in this configuration of SCHC/Sigfox), it generates the same number of fragments as a SCHC Packet of size 300 B. This change in SCHC Packet size is not thought to generate any implications on the results.

N_U was measured over a non-aborting scenario and over the base scenario. By using the same results of the base scenario, r_S is measured. Both performance metrics are calculated over the real deployment, although experiments were not performed in the non-aborting scenario. This is due to the longer transmission times and the excessive use of network resources required in this configuration.

3.3.2 Experimental setup

As previously stated, the experiments of this work are carried out both in a simulation setup and in a real deployment setup. All experiments of the simulation environment were executed in a remote server. The specifications of the operating system, architecture, CPU and others were obtained with the `lscpu` Linux command and are reported in Table 3.3.

Experiments carried out over the real deployment were run at the Faculty of Physical

Parameter	Value
Operating system	Debian GNU/Linux 10 (buster)
Architecture	x86_64
Number of CPU cores	36
Threads per core	2
Model name	Intel(R) Xeon(R) Gold 5220 @ 2.20 GHz
Maximum Hz	3900 Hz
Memory (RAM + swap)	213 GiB
Disk space	1.5 TiB

Table 3.3: Specifications of the server in which the simulation experiments were carried out.

and Mathematical Sciences, Universidad de Chile. The LoPy4 device was indoors, near a window facing west, approximately 250 m to the west of the nearest Sigfox base station. The distance between the device and the nearest window impacts channel conditions and FLR; therefore, the device was located in three different positions inside the laboratory, according to spatial availability of the room. Experiments were performed at roughly 20 cm, 230 cm and 470 cm from the nearest window. The specifications of the LoPy4 device are shown in Table 3.4.

Parameter	Value
CPU	Xtensa(R) LX6 microprocessor(s)
Architecture	32-bit
Number of CPU cores	2
Memory (RAM)	4 MB
Flash memory	8 MB
Sigfox node range	Up to 50 km
Sigfox maximum TX power (RC4)	+20 dBm
Sigfox data rate (RC4)	600 bps
Sigfox RX sensitivity	−126 dBm
Sigfox current draw (RC4)	125 mA (TX), 11.2 mA (RX)

Table 3.4: Specifications of the LoPy4 device and its Sigfox modem performance. Adapted from [4].

3.3.3 Data analysis

During each SCHC transmission, the sender program records data in an internal dictionary object, which is updated every time a SCHC Fragment is sent and every time a SCHC ACK is received. After finishing each transmission, be it successful or not, the sender exports the dictionary object into the file storage of the platform in which the code is being run. The stored files are JSON-formatted and contain information regarding the SCHC Packet size, the number of SCHC Fragments generated, the induced FLR, the total number of uplink transmissions performed, whether the SCHC transmission was ended successfully, and other data.

The data was processed with NumPy and Matplotlib, Python libraries for numerical data analysis. Empirical calculations of N_U and r_S were required aggregating data for the same F and FLR value pair, calculating the mean and the associated standard deviation. For each value of F, data for N_U and r_S respect to the FLR value was plotted. The empirical data was plotted along the models proposed in Chapter 5, as well as the absolute difference between both curves. The evolution of the standard deviation of the calculations of N_U and r_S for multiple FLR values was also plotted.

Chapter 4

SCHC-over-Sigfox implementation

This chapter describes the implementation of SCHC/Sigfox, detailing both the simulation environment and the real deployment environment. The simulation environment is the first system to be modified when changes are needed and is developed in a way that emulates the functionality of the real deployment, which makes it easier to integrate the modifications into the deployment. Both architectures are developed primarily in the Python programming language.

4.1 Module description

The simulation and the real deployment environments share the same model and class structure, although the implementation of some classes and methods differ. This is because the simulation is developed to emulate how LoPy4, Cloud Functions, and Firebase Realtime Database work. The real deployment uses the actual libraries and frameworks implemented in those technologies. The modules present in the SCHC/Sigfox implementation are shown in Figure 4.1 and are described in the following paragraphs.

The `config` module contains configuration modules, `schc.py` and `gcp.py`, which control the functioning of SCHC and GCP respectively. `schc.py` contains application-specific configurations such as the timeout values, the time to wait between two consecutive fragment transmissions, and global constants. On the other hand, `gcp.py` contains information used by GCP to perform the deployment and operate the Cloud Functions, such as the endpoints and the authentication credentials.

The `db` module contains classes that manage data storage in different environments. It contains the classes `FileStorage` and `CommonFileStorage`, which are used by the sender and manage local file systems; and the classes `JSONStorage`, `LocalStorage`, and `FirebaseRTDB`, which are used by the receiver and are programmed to read and save from a JSON-formatted database. Particularly, `LocalStorage` emulates the functioning of `FirebaseRTDB` by creating a locally stored JSON object that is updated when the receiver otains a new SCHC Fragment or when it replies with a Compound ACK. `FirebaseRTDB` uses the `firebase_admin` Python

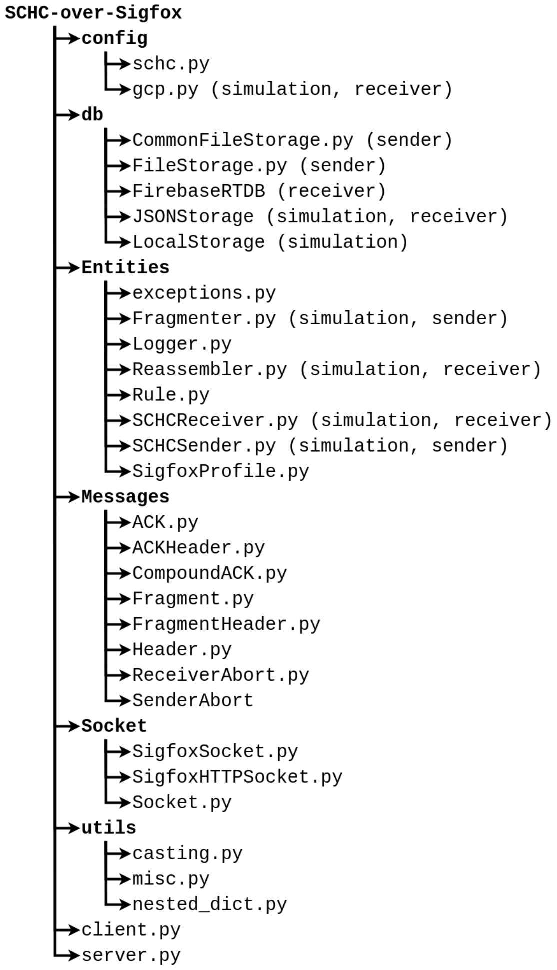


Figure 4.1: Module diagram of the SCHC-over-Sigfox implementation. Python scripts are listed along with the environment they employed in. If not present, they are employed in all environments.

library, which allows the code to modify the online storage object.

The `Entities` module contains classes that contain data pertaining to a particular SCHC transmission and perform the F/R procedure. `exceptions.py` contains custom exceptions of the project, raised under certain circumstances to control the flow of the F/R algorithms. `Rule` and `SigfoxProfile` are data classes that are instantiated at the start of every SCHC transmission and configure the values and parameters of the SCHC Profile. `Logger` is an object that displays messages into the console and stores statistics of SCHC transmissions, which can be exported for data analysis. The `Fragmenter` class performs the fragmentation procedure, with a SCHC Packet as input and a list of SCHC Fragments as outputs; the `Reassembler` class performs the opposite operation. `SCHCSender` is a class that executes the SCHC sender algorithm by means of its `start_session()` method, which iterates over the list of generated fragments. Finally, `SCHCReceiver` is responsible for executing the SCHC receiver algorithm. The `schc_recv()` method is to be called every time the receiver side of the implementation obtains a new SCHC Fragment.

The `Messages` module is composed of classes that encapsulate the information and functions of all types of messages of SCHC/Sigfox. The `Header` class contains common information for SCHC Headers, and is extended into the `FragmentHeader` and `ACKHeader` classes, which contain specific information of SCHC Fragments and SCHC ACKs. These classes are used in the `Fragment` and `ACK` classes, respectively. The `Fragment` class represents a SCHC Fragment and is extended into the `SenderAbort` class, which has the same field structure. On the other hand, the `ACK` class represents a SCHC ACK and is extended into the `CompoundACK` class and the `ReceiverAbort` class. The `CompoundACK` class is the one instantiated by the `SCHCReceiver` when a Compound ACK is generated, and contains more information than the `ACK` class.

The `Sockets` module contains different implementations of the communication procedure between each component of the system. `Socket` is an abstract class that declares the `send()`, `recv()`, `set_reception()` and `set_timeout()` methods, but does not implement them. Instead, they are implemented in the `SigfoxHTTPSocket` and the `SigfoxSocket` classes. The `SigfoxHTTPSocket` class emulates the functioning of the LoPy4 Sigfox sockets and the message forwarding of the Sigfox HTTP Callbacks by creating an HTTP request that is directly sent to the receiver endpoint. On the other hand, `SigfoxSocket` encapsulates the actual LoPy4 Sigfox sockets, and is only responsible for sending and receiving messages over the network. The HTTP callbacks are performed by the Sigfox backend itself.

Finally, the `utils` module contains functions used throughout the project. The `casting` module contains functions used to transform information between different data types used in the project, such as binary, byte, hexadecimal, integer, and string representations. The `misc` module contains miscellaneous functions used in the project. The `nested_dict` module contains functions that perform read and write operations in an arbitrarily nested dictionary, used by the `JSONStorage` class and its subclasses.

Unified Modeling Language (UML) diagrams of the `db`, `Entities`, `Messages` and `Sockets` modules are provided in Appendix B, Figures B.1, B.2, B.4 and B.3, respectively.

4.2 Simulation

The simulation environment can be run entirely within a single computer. It is an HTTP client-server application that uses Flask, a minimal Python framework for web services, also used by GCP to instantiate Cloud Functions [43]. Since the system is self-hosted, transmission times are multiple times faster than in the real deployment, which allows timers to be configured with minimal values, performing SCHC transmissions faster. This advantage is particularly useful when developing new changes or refactoring the code, since the changes can be tested quickly.

An overview of the local simulation system is provided in Figure 4.2. Within the same computer, a SCHC Packet is generated and processed by the `Fragmenter`, which generates a list of SCHC Fragments. These fragments are processed by the `SCHCSender` and sent using a `SigfoxHTTPSocket` object, which communicates with the receiver using HTTP. The HTTP message is received at the HTTP receiver endpoint, and is passed to the `SCHCReceiver`. When all fragments are received, the `Reassembler` object produces the original SCHC Packet. If

Compound ACKs are to be sent back to the sender side, the communication is done in the opposite direction. Both the sender and the receiver side share the same storage but use different interfaces to read from and write into it.

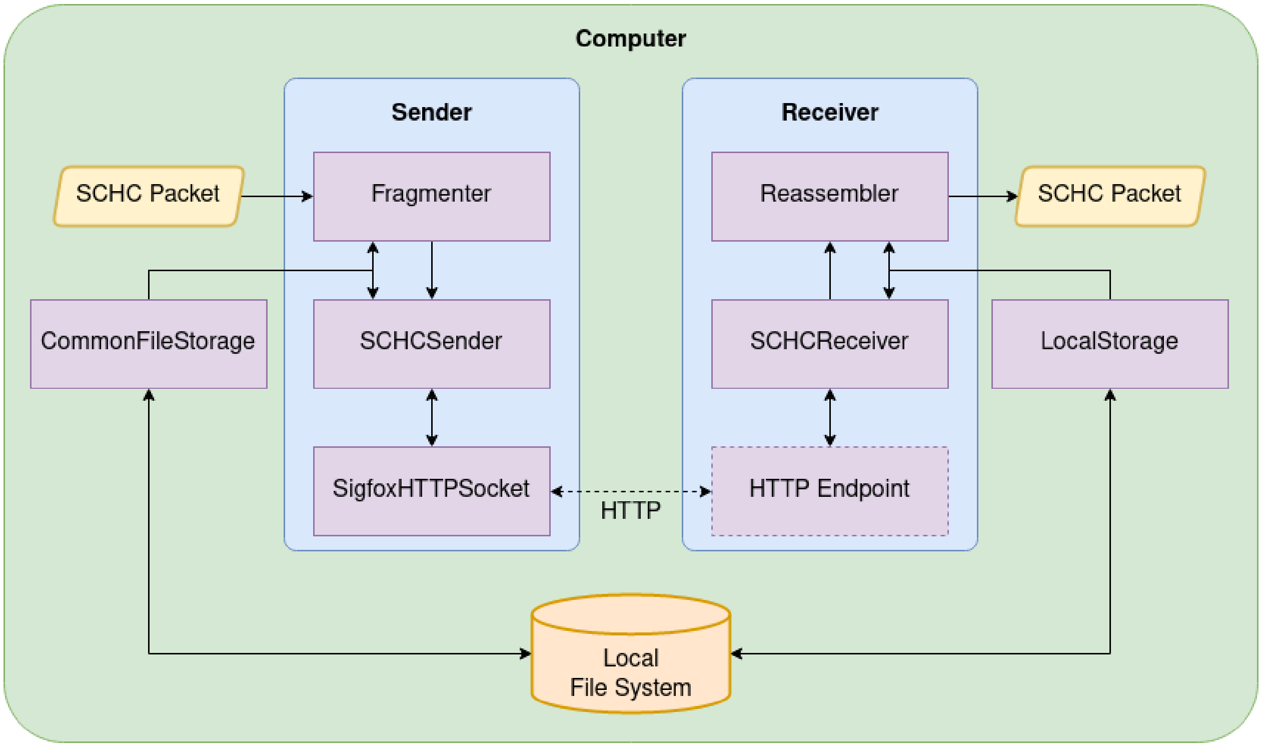


Figure 4.2: Overview of the simulation system.

4.3 Real deployment

The real deployment environment uses a LoPy4 as the sender side of the system, and an integration with Cloud Functions and Firebase Realtime Database as the receiver side of the system. The LoPy4 is programmed in Micropython, a Python implementation for microcontrollers [41]. An HTTP Callback must be configured in the Sigfox backend to forward the messages received by the network into the GCP environment, which communicates using HTTP.

An overview of the local simulation system is provided in Figure 4.3. The sender side of the code is deployed into a LoPy4 device, which has its own local file system and uses the `CommonFileStorage` class to interact with it. A SCHC Packet is generated and fragmented, and the resulting SCHC Fragments are sent using the `SigfoxSocket` object, which uses Pycom-specific methods to send messages using the Sigfox wireless modulation. These messages are received by a Sigfox base station, which forwards them using IP into the Sigfox backend. Here, the messages are formatted as a JSON object and forwarded into the endpoint of the receiver Cloud Function using HTTP. When Compound ACKs are to be sent back to the receiver, the communication is done in the opposite direction. The receiver uses the `FirebaseRTDB` class to communicate with the storage at Firebase, and produces a SCHC

Packet when all fragments are received.

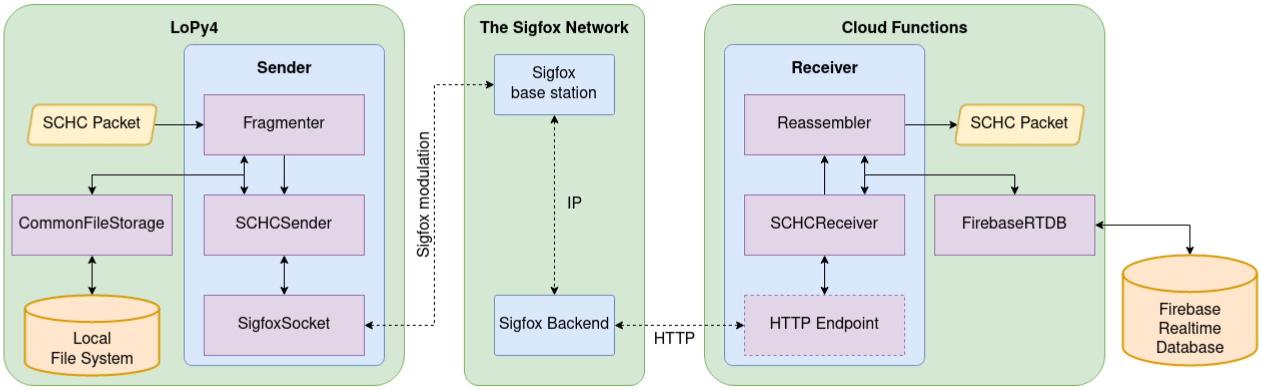


Figure 4.3: Overview of the real deployment system.

Chapter 5

Performance metrics

A SCHC transmission is defined as the transmission of a single SCHC Packet, which may be fragmented, and is finished when the packet is received in its entirety at the receiver end or when the transmission is aborted. The SCHC transmission of a fragmented SCHC Packet considers the transmissions of all the SCHC Fragments generated and all the SCHC ACKs sent by the receiver, either until completion or failure to deliver the entire packet.

This chapter defines the two performance metrics proposed in this work: the average number of uplink messages per SCHC transmission and the rate of completed SCHC transmissions. Both models consist of a theoretical part and an empirical part, the latter being approximated by comparing partial results with the theoretical part, defining a candidate function, and adjusting its parameters using LSM and manual inspection.

5.1 SCHC F/R process overview

This section introduces the variables that will be used in the models by describing the fragmentation process. The following constants are defined:

- Let L_{MTU} be the MTU of the network in bytes.
- Let H_R be the length in bytes of the header of a regular SCHC Fragment
- Let H_A be the length in bits of the header of an All-1 fragment.
- Let N be the length in bits of the FCN header field.
- Let M be the length in bits of the window header field.
- Let $R_{\max}$ be equal to `MAX_ACK_REQUESTS`.

With the previous definitions, the following variables are calculated:

- The maximum number of fragments that can be grouped in a single window is calculated as $S = 2^N - 1$.
- The maximum window number is $W_{\max} = 2^M$.
- The maximum number of fragments that can be generated with this configuration is $F_{\max} = W_{\max} S$.

The different operational modes of SCHC/Sigfox, presented in Section 2.1.3, may specify different header and payload sizes for regular SCHC Fragments and the All-1 fragment. The length in bytes of the payload of regular fragments is $P_R = L_{\text{MTU}} - H_R$, whilst the maximum length in bytes of the payload of the All-1 fragment is $P_A = L_{\text{MTU}} - H_A$. Note that if $H_R = H_A$, $P_R = P_A$. Since a SCHC fragmentation process always generates an All-1 fragment, the maximum SCHC Packet size in bytes can be calculated as

$$L_{\max} = P_0(F_{\max} - 1) + P_A. \tag{5.1}$$

Note that $L_{\max}$ may not be equal to the maximum SCHC Packet size defined in [3] and reported in Table 2.3.

In header configurations where $H_R < H_A$, such as in the Single-byte SCHC Header and the Two-byte SCHC Header Option 2 modes, the All-1 fragment carries no payload if the size of the SCHC Packet is a multiple of P_0. Therefore, the number of fragments generated from a SCHC Packet of size L is calculated as

$$F = \begin{cases} \left\lceil \frac{L}{P_0} \right\rceil + 1 & \text{if } H_R = H_A \wedge L \mod P_0 = 0 \\ \\ \left\lceil \frac{L}{P_0} \right\rceil & \text{otherwise} \end{cases}. \tag{5.2}$$

Subsequently, the number of windows obtained by grouping the F fragments into windows of size S is calculated as

$$W = \left\lceil \frac{F}{S} \right\rceil. \tag{5.3}$$

Every window but the last one contains exactly S fragments. The number of fragments in the last window is not directly calculated as the remainder of dividing F by S, since all windows contain at least one fragment and in cases where F is a multiple of S, the remainder is 0. Instead, one fragment is taken from F and is added back after calculating the reminder. Therefore, the number of fragments in the last window is calculated as

$$U = [(F - 1) \mod S] + 1. \tag{5.4}$$

Finally, let X denote a random variable that represents whether a single fragment is lost, with a discrete space $\Omega_X = \{0, 1\}$. The decimal representation of the FLR is assumed as the probability of this happening and is labeled p. More specifically,

$$\mathbb{P}(X = 1) = p \qquad \text{(The fragment is lost)} \tag{5.5}$$

$$\mathbb{P}(X = 0) = 1 - p \qquad \text{(The fragment is delivered).} \tag{5.6}$$

Note that $X \sim \mathrm{B}(1, p)$, because each attempt of sending a particular fragment is a binomial trial. Also note that p quantifies losses of Sigfox messages in the uplink stream; losses in the downlink stream do not necessarily follow the same rate and are not considered in this study.

The variable names and meanings are summarized in Table 5.1. A set of examples for L, F, W and U are shown in Table 5.2, considering the values of the 1-byte header configuration of SCHC/Sigfox. These examples were selected by taking eight evenly-spaced integer values for L ranging from the smallest SCHC Packet size, 1 B, to the largest practical SCHC Packet size, 307 B.

Variable name	**Type**	**Unit**	**Meaning**
F	Function	1	Number of fragments generated
$F_{\max}$	Function	1	Maximum number of fragments
H_A	Constant	Byte	All-1 fragment header size
H_R	Constant	Byte	Regular fragment header size
L	Constant	Byte	Length of a particular SCHC Packet
$L_{\max}$	Function	Byte	Practical maximum SCHC Packet size
L_{MTU}	Constant	Byte	MTU of the network
M	Constant	Bit	Length of the window header field
N	Constant	Bit	Length of the FCN header field
P_A	Function	Byte	All-1 fragment payload size
P_R	Function	Byte	Regular fragment payload size
$R_{\max}$	Constant	1	Maximum number of ACK requests in a row (`MAX_ACK_REQUESTS`)
S	Function	1	Maximum number of fragments per window
U	Function	1	Number of fragments in the last window
W	Function	1	Number of windows obtained
$W_{\max}$	Function	1	Maximum number of windows

Table 5.1: Variable names in alphabetical order.

L [B]	**F**	**W**	**U**
1	1	1	1
45	5	1	5
88	9	2	2
132	13	2	6
176	17	3	3
220	21	3	7
263	24	4	3
307	28	4	7

Table 5.2: Values of F, W and U for different values for L, calculated for the 1-byte header configuration of SCHC/Sigfox.

The performance metrics are defined under the following assumptions:

- X is independent and identically distributed
- All Compound ACKs are correctly delivered to the sender

- The retransmission timer does not expire.

Although these assumptions are not necessarily true, they make the models mathematically tractable and provide room for improvement. Further implications are discussed later in the document.

5.2 Average number of uplink messages per SCHC transmission

The average number N_U of uplink messages per completed SCHC transmission is proposed as a performance metric that quantifies the uplink message overhead of SCHC fragmentation under lossy channels. It is defined as the average number of messages sent over many SCHC transmissions, either until they finish successfully or until they are aborted. It can also be measured in a scenario where the $R_{\max}$ limit is bypassed, which avoids aborting the transmissions (a non-aborting scenario).

The measurement over the non-aborting scenario, if compared to a measurement over the base scenario, quantifies the impact of aborting SCHC transmissions in terms of uplink messages whose transmission was avoided, since network conditions were too harsh. With this modification, it is expected, theoretically and for $p < 1$, that every SCHC transmission would eventually be successful. This modification prevents the SCHC algorithm from reaching step 4b of the sender algorithm shown in Section 2.1.3.

This model is composed of a theoretical part $N_{U_T}(p)$ and an empirical part $N_{U_E}(p)$. $N_{U_T}(p)$ is obtained by performing a probabilistic analysis over the expected number of times a particular SCHC Fragment needs to be sent until it is received. On the other hand, $N_{U_E}(p)$ is estimated by comparing $N_{U_T}(p)$ to preliminary empirical results. The semiempirical model for N_U is then defined as

$$N_U(p) = N_{U_T}(p) + N_{U_E}(p). \tag{5.7}$$

5.2.1 Justification

This section lays out the importance of this metric to quantify SCHC Fragmentation performance. The sender algorithm of SCHC/Sigfox, after transmitting the last fragment—the All-1 fragment—, starts a retransmission phase for every lost fragment in the corresponding Compound ACK, repeating this process until all fragments are delivered correctly to the receiver or until the All-1 is sent $R_{\max}$ times in a row. On its own, this performance metric quantifies the average message overhead in a SCHC transmission per FLR value. By comparing the two measurements of N_U (over a non-aborting scenario and over the base scenario), this metric is a way to quantify the number of UL messages that are saved, i.e., not sent over harsh network conditions, avoiding excessive use of network resources and aborting the transmission instead.

Depending on the FLR, the number of messages needed to do so is expected to increase, but for $p < 1$ this number should not be infinite unless the SCHC algorithm contains loops. Its calculation is also useful to find absorbing loops or errors in the SCHC algorithm.

A measurement of N_U over a non-aborting scenario cannot be used to predict the actual number of fragments that would be sent in a real deployment of SCHC/Sigfox, since ignoring the $R_{\max}$ limit drastically changes the functioning of the algorithm. Therefore, this scenario provides an upper bound to the actual average number of messages sent per SCHC transmission.

If calculated over the base scenario with a sample size large enough, N_U can be used as a prediction of its theoretical counterpart, the expected number of uplink messages per SCHC transmission. Let N_{U_n} denote the average number of uplink messages per SCHC transmission measured by performing n transmissions. Let Y be a random variable that quantifies the number of uplink messages sent in a single SCHC transmission, with a discrete sample space $\Omega_Y \subseteq \mathbb{N}\backslash\{0\}$ and with expected value $\mathbb{E}(Y) = \mu \in \mathbb{R}$. Let $(Y_k)_{k=1}^n = (Y_1, Y_2, \cdots, Y_n)$ be a random process consisting of a sequence of *iid* random variables with the same expected value $\mathbb{E}(N_1) = \mathbb{E}(N_2) = \cdots = \mu$. According to the strong law of large numbers, the average of $(Y_k)_{k=1}^n$, $\overline{Y_n}$, converges almost surely to the expected value, μ, as n increases towards infinity. Note that $\overline{Y_n}$ is a calculation of the average uplink message count for n SCHC transmissions, this is, $\overline{Y_n} = N_{U_n}$. With this,

$$\mathbb{P}\left(\lim_{n\longrightarrow\infty} N_{U_n} = \mu\right) = 1. \tag{5.8}$$

Equation 5.8 implies that, as the sample size used to calculate N_{U_n} increases, its value approaches the expected value for the distribution of Y. This makes it possible to compare probabilistic calculations with empirical results for a large sample size.

5.2.2 Theoretical model

A model for N_U when calculated over a non-aborting scenario is provided in the following paragraphs. Let Z be a random variable that quantifies the number of times a particular SCHC Fragment needs to be sent until it is successfully delivered to the receiver, with a discrete space $\Omega_Z \subseteq \mathbb{N}\backslash\{0\}$. Since Z includes the successful attempt, $Z \geq 1$. Note that each attempt to send a fragment is a realization of X, a binomial trial of success probability $1-p$, and that the count stops as soon as $X = 0$ according to Equation 5.6. Therefore, Z follows a geometric probability distribution of associated probability $1 - p$, i.e., $Z \sim \text{Geo}(1 - p)$. It follows that

$$\mathbb{E}(Z) = \frac{1}{1-p}. \tag{5.9}$$

Equation 5.9 implies that for a single SCHC Fragment and for a value of p, the expected number of times that the SCHC Fragment is transmitted until reception is $\frac{1}{1-p}$. As per assumption 5.1, the retransmission attempts could be performed at every Compound ACK reception, but the moment in which they are performed does not affect the values that Z can take. Therefore, Z is independent of the number of received Compound ACKs and, therefore, of the number of windows.

In a SCHC transmission of F fragments, the random variable Z is realized for each separate fragment. Let $(Z_k)_{k=1}^F = (Z_1, Z_2, \cdots, Z_F)$ be a sequence of realizations of Z for every fragment of a SCHC Transmission, *iid* random geometric variables of associated probability $1-p$. Each fragment is sent independently of other fragments, thus, these realizations of Z are independent. Note that the sum of these realizations, $\sum_{k=1}^F Z_k$, is random variable that quantifies the total number of uplink transmissions until reception of the F fragments By the linearity of expectation, it follows that

$$\mathbb{E}\left(\sum_{k=1}^{F} Z_k\right) = \sum_{k=1}^{F} \mathbb{E}\left(Z_k\right) = \sum_{k=1}^{F} \frac{1}{1-p} = F\left(\frac{1}{1-p}\right). \tag{5.10}$$

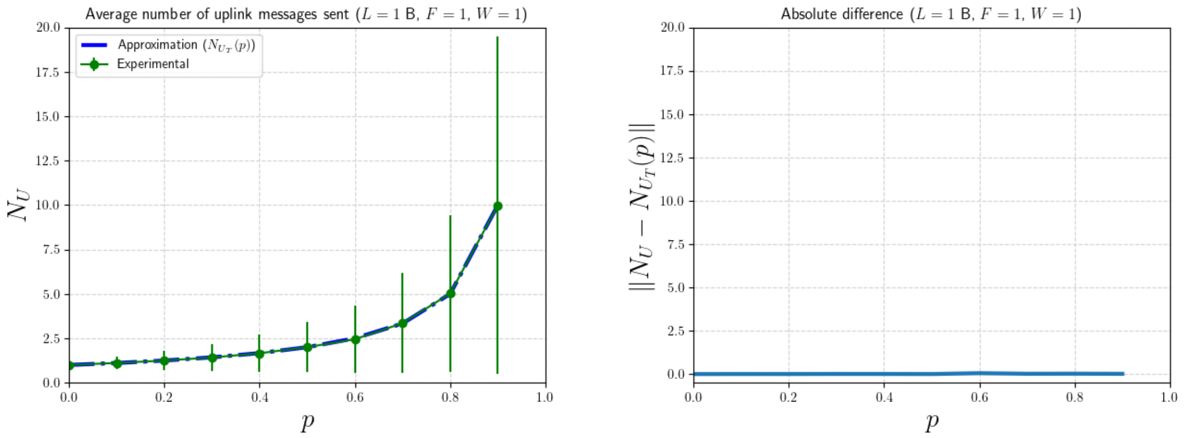
With this, it is proposed that

$$N_{U_T}(p) = F\left(\frac{1}{1-p}\right). \tag{5.11}$$

5.2.3 Validation

The theoretical component of $N_U(p)$, $N_{U_T}(p)$, was compared with measurements of N_U performed in the SCHC/Sigfox simulation. The experiments consisted of repeating 10,000 SCHC transmissions for every combination of number of SCHC Fragments (F) and induced FLR value (p), as stated in Table 3.1. For each combination of F and p, the mean and the standard deviation of the data is calculated and displayed. This section displays the results for $F = 1$ and $F = 17$ B in Figure 5.1 and Figure 5.2, respectively. Other packet sizes behave similarly as the case for $F = 17$ B but differ from the case for $F = 1$, and are reported in Appendix C.

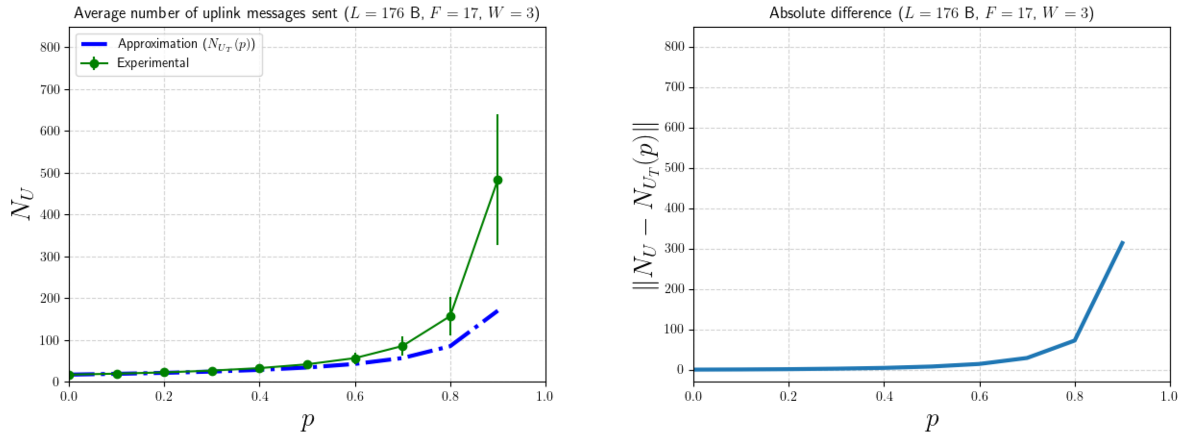
As shown in Figure 5.1a, $N_{U_T}(p)$ closely follows empirical results for $F = 1$, with a peak difference of 0.051 found at $p = 0.9$, as shown in Table 5.3. However, a clear difference appears in the cases where $F > 1$, as seen in Figure 5.2a. Experimental curves show a rapid increase and separate from the theoretical curve, the difference being higher as F increases. The shape of the difference curves is analyzed to propose a candidate function that compensates the differences.



(a) Comparison between $N_{U_T}(p)$ and N_U.

(b) Difference between simulations and the theoretical model.

Figure 5.1: Simulation environment results compared to the theoretical component N_{U_T}(p). $L = 1$ B, $F = 1$, $W = 1$



(a) Comparison between $N_{U_T}(p)$ and N_U.

(b) Difference between simulations and the theoretical model.

Figure 5.2: Simulation environment results compared to the theoretical component N_{U_T}(p). $L = 176$ B, $F = 17$, $W = 3$

5.2.4 Empirical adjustment

For $F = 1$, the difference is negligible; however, for $F > 1$, the difference curves reach up to 364.445. As shown in and Table 5.3, the maximum difference increases noticeably as both L and consequently F increase. A candidate function should follow this behavior: negligible or null for $F = 1$ and increasing for $F > 1$.

All difference curves for $F > 1$ show a similar shape, which only differs slightly between the curves as F increases. The shape of the curves is found to be similar to that of curves of

F	Max. difference	p
1	0.051	0.6
5	194.024	0.9
9	251.376	0.9
13	292.191	0.9
17	313.431	0.9
21	334.390	0.9
24	350.586	0.9
28	364.445	0.9

Table 5.3: Maximum difference between $N_{U_T}(p)$ and N_U for each value of F, along with the value of p where the maximum was found.

the form $y(x) = \frac{x}{1-x}$. More specifically, the following function is proposed:

$$N_{U_E}(p) = f_N(F)\left(\frac{p}{(1-p)^a}\right), \tag{5.12}$$

where a is constant and $f_N(F)$ is a function that is non-zero for $F > 1$ and that increases as F increases. The parameter a controls the exponential increment of the curve.

It is proposed that

$$f_N(F) = F - 1. \tag{5.13}$$

Note that if $F = 0$, $f_r(F) = 0$. The optimal value of a is obtained by means of the LSM for every value of F. A single value of a was selected from this set of optimum a, which makes $N_U(p)$ not exceed the values of N_U, was selected. $a = 1.14$ is chosen. The values of the optimal a for every F are reported in Table 5.4.

Criteria	a
F = 5	1.732
F = 9	1.545
F = 13	1.428
F = 17	1.330
F = 21	1.246
F = 24	1.200
F = 28	1.145
Mean	1.375
Std. dev.	0.193
Adjusted value	1.140

Table 5.4: Values for a obtained by using LSM. The final adjustment by inspection is shown in the last row.

With the previous decisions, $N_{U_E}(p)$ becomes

$$N_{U_E}(p) = (F-1)\left(\frac{p}{(1-p)^{1.14}}\right). \tag{5.14}$$

Combining Equation 5.11 and Equation 5.14 into Equation 5.7 yields the proposed semiempirical model for N_U when calculated over a non-aborting scenario:

$$N_U(p) = F\left(\frac{1}{1-p}\right) + (F-1)\left(\frac{p}{(1-p)^{1.14}}\right) \tag{5.15}$$

5.3 Rate of successful SCHC transmissions

The rate of completed SCHC transmissions, or success rate r_S, is defined as the ratio between the amount of successful SCHC transmissions and the total amount of SCHC transmissions performed for a fixed value of L and p. A successful SCHC transmission is defined as a SCHC transmission where the sender has received the finalizing ACK from the receiver. This requires that all fragments get from the sender end to the receiver end without errors, and that the transmission is not aborted, i.e., the sender and the receiver should reach steps 4(d)iii and 4(b)iiA of the algorithms shown in Section 2.1.3, respectively.

Similarly to the model for N_U, this model is composed of a theoretical part and an empirical part, $r_{S_T}(p)$ and $r_{S_E}(p)$ respectively. $r_{S_T}(p)$ is proposed in the following definition section by means of a probabilistic analysis and $r_{S_E}(p)$ is estimated by comparing $r_{S_T}(p)$ to preliminary empirical results. The semiempirical model for the success rate is then composed as a function of p,

$$r_S(p) = r_{S_T}(p) + r_{S_E}(p). \tag{5.16}$$

5.3.1 Justification

The importance of the success rate as a performance metric is laid out in the following paragraphs. Let r_{S_n} denote the success rate measured by performing n SCHC transmissions. A SCHC transmission can be either successful or aborted. Therefore, the event of a SCHC transmission being successful is a binomial trial with a certain associated success probability, p_S. Let V be a random binomial variable that quantifies whether a single SCHC transmission is successful or not, with a discrete space $\Omega_V = \{0, 1\}$. It follows that $V \sim \mathrm{B}(1, p_S)$.

Let $(V_k)_{k=1}^n = (V_1, V_2, \cdots, V_n)$ be a sequence of *iid* random variables, where $V_k \sim \mathrm{B}(1, p_S)$ for $k \in [1, n]$. Each V_k corresponds to a particular realization of V. According to the strong law of large numbers, the average of $(V_k)_{k=1}^n$, $\overline{V_n}$, converges almost surely to the expected value, p_S, as n tends to infinity. Note that $\overline{V_n}$ is a calculation of the success rate with n SCHC transmissions, this is, $\overline{V_n} = r_{S_n}$. With this,

$$\mathbb{P}\left(\lim_{n\longrightarrow\infty} r_{S_n} = p_S\right) = 1. \tag{5.17}$$

Equation 5.17 implies that, as the sample size used to calculate the r_S increases, r_S approximates the success probability p_S. This makes it possible to use an empirically calculated success rate as a prediction of the probability of a SCHC transmission to be complete, under the same conditions in which r_S was measured. The success rate of certain SCHC configurations can then be taken into consideration before deploying a SCHC application if the channel conditions are known and stable, providing information that can potentially save network resources in scenarios where the success rate is low or when its variance is high.

5.3.2 Theoretical model

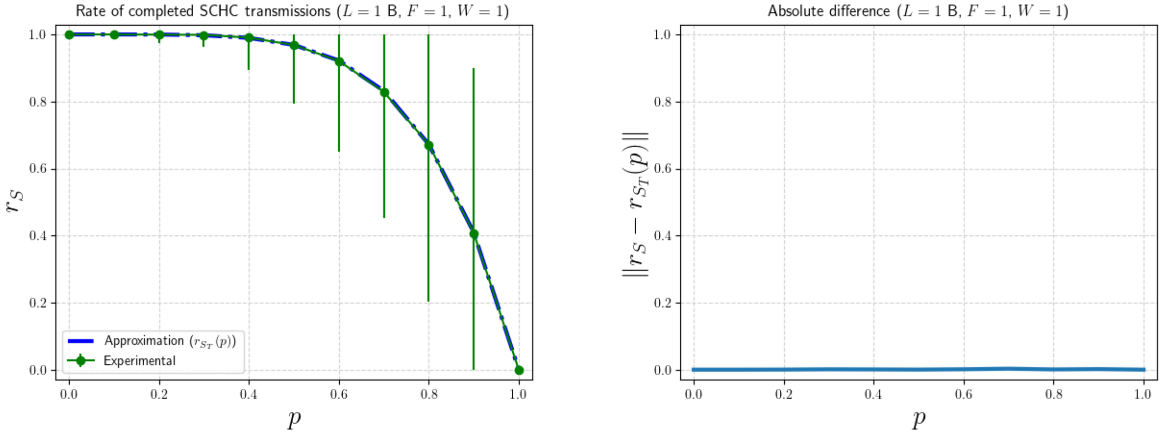
Since retransmission timer expiration is not considered in this analysis, the only way for a SCHC transmission to be aborted is by means of a SCHC Sender-Abort message, which is sent after reaching the maximum number of times that an ACK can be requested in a row, $R_{\max}$. The Sigfox SCHC Profile states that the attempts counter is reset every time an ACK is received by the device, and is only updated after sending an All-1. The limit for the attempts counter can be exceeded by any combination of losing the All-1 or the ACKs that make the attempts counter reach $R_{\max}$. As per assumption 5.1, this analysis is centered in the event of losing the All-1 $R_{\max}$ times in a row.

The Sigfox SCHC Profile states that the SCHC Sender-Abort is sent only when the number of repeated All-1 transmissions in sequence reaches $R_{\max}$ without receiving a Compound ACK. This requires one All-1 fragment to be lost $R_{\max}$ in a row. The probability of any fragment to be lost r times in a row is simply p^r, and the probability of the opposite event—not losing the fragment r times in a row—is $1-p^r$. This is true for any fragment, particularly the All-1 fragment. The event of losing an All-1 fragment $R_{\max}$ times in sequence is necessary for a SCHC transmission to be aborted, thus, this event should not happen for a SCHC transmission to be successful. It is proposed that

$$r_{S_T} = 1 - p^{R_{\max}}. \tag{5.18}$$

5.3.3 Validation

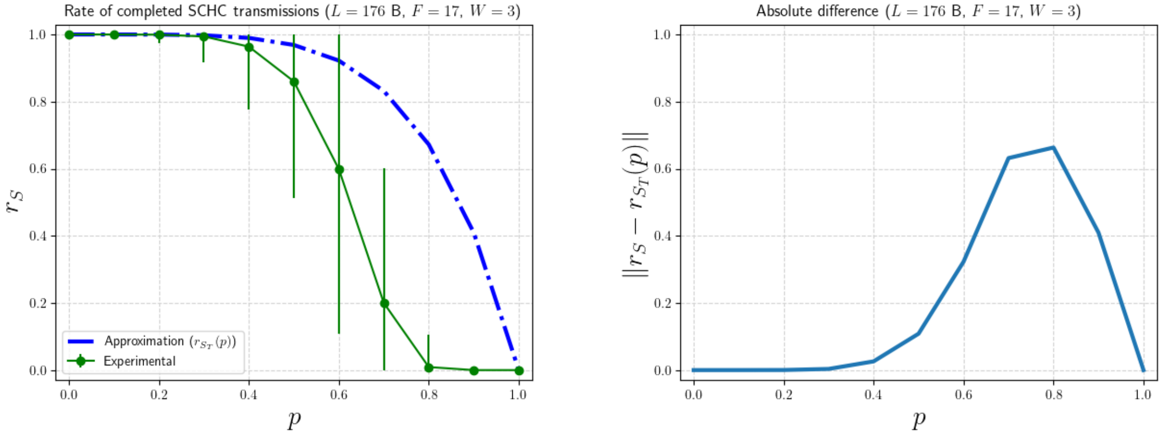
The theoretical component of the success rate was compared with several measurements of r_S in the SCHC/Sigfox simulation. The experiments performed in the simulation environment consisted of repeating 10,000 SCHC transmissions for every combination of number of SCHC Fragments (F) and induced FLR value (p), as stated in Table 3.2. The mean and the standard deviation of the data obtained from the experiments is calculated and displayed. This section displays the results for SCHC Packet sizes of $F = 1$ and $F = 17$ B in Figures 5.3 and 5.4. Other packet sizes behave similarly as the $F = 17$ case but differ from the $F = 1$ case, and are reported in Appendix G.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

Figure 5.3: Simulation environment results compared to the theoretical component r_{S_T}(p). $L = 1$ B, $F = 1$, $W = 1$



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

Figure 5.4: Simulation environment results compared to the theoretical component r_{S_T}(p). $L = 176$ B, $F = 17$, $W = 3$

As shown in Figure 5.3a, $r_{S_T}(p)$ closely follows empirical results for $F = 1$, with a peak difference of 0.003 found at $p = 0.7$, as shown in Table 5.5. However, a clear difference appears in the cases where $F > 1$, as seen in Figure 5.4a. The curves show a steep decrease starting around $p = 0.4$ and present an inflection point. The shape of the difference curves is analyzed to propose a candidate function that compensates the differences.

F	Max. difference	p
1	0.003	0.7
5	0.602	0.8
9	0.646	0.8
13	0.658	0.8
17	0.663	0.8
21	0.665	0.8
24	0.667	0.8
28	0.676	0.7

Table 5.5: Maximum difference between $r_{S_T}(p)$ and r_S for each value of F, along with the value of p where the maximum was found.

5.3.4 Empirical adjustment

For $F = 1$, the difference is negligible; however, for every $F > 1$, the difference curves reach up to 0.676. As shown in Table 5.5, the maximum difference increases slightly as both L and consequently F increase. A candidate function should follow this behavior: negligible or null for $F = 1$ and slightly increasing for $F > 1$.

All difference curves for $F > 1$ show a similar shape, which only differs slightly between the curves as F increases. The shape of the curves is found to be similar to that of curves such as $y(x) = x(1 - x)$. More specifically, the following function is proposed:

$$r_{S_E}(p) = -f_r(F)\left[\gamma p^{\alpha}(1-p)^{\beta}\right]^{\delta}, \tag{5.19}$$

where α, β, γ and δ are constants, and $f_r(F)$ is a function that is non-zero for $F > 1$ and that increases slowly as F increases. The negative sign is added since $r_{S_T}(p)$ was found to be always greater than the empirical results. α controls the aperture to the left of the curve, β controls the aperture to the right of the curve, γ controls the rate at which the function increases and decreases and δ is a scaling factor.

It is proposed that

$$f_r(F) = \frac{F-1}{5F}. \tag{5.20}$$

Note that if $F = 0$, $f(F) = 0$. The optimal values of α, β, γ, δ and ε were obtained by means of LSM for every value of F, and the mean for each constant was selected. Then, they were adjusted by inspection to avoid obtaining negative results for $r_S(p)$. The adjusted values differ from the ones found by LSM but provide a more realistic fit for the curve. The values of these constants are reported in Table 5.6, whilst the results of LSM are reported in Table G.1.

With the previous decisions, $r_{S_E}(p)$ becomes

$$r_{S_E}(p) = -\frac{F-1}{5F}\left[4.8p^{2.00}(1-p)^{0.49}\right]^{3.20}. \tag{5.21}$$

Combining Equation 5.18 and Equation 5.21 into Equation 5.16 yields the proposed

Constant	**Value**
α	2.00
β	0.49
γ	4.80
δ	3.20

Table 5.6: Adjusted values for α, β, γ and δ.

semiempirical model for the success probability of a SCHC transmission:

$$r_S(p) = 1 - p^{R_{\max}} - \frac{F-1}{5F}\left[4.80p^{2.00}(1-p)^{0.49}\right]^{3.20}. \tag{5.22}$$

Chapter 6

Results

This chapter provides the data obtained from the experiments, which show a comparison of the measured values of performance metrics N_U and r_S respect to their semi-empirical models, $N_U(p)$ and $r_S(p)$. Useful performance data of SCHC/Sigfox is also obtained.

6.1 Average number of uplink messages sent per SCHC transmission

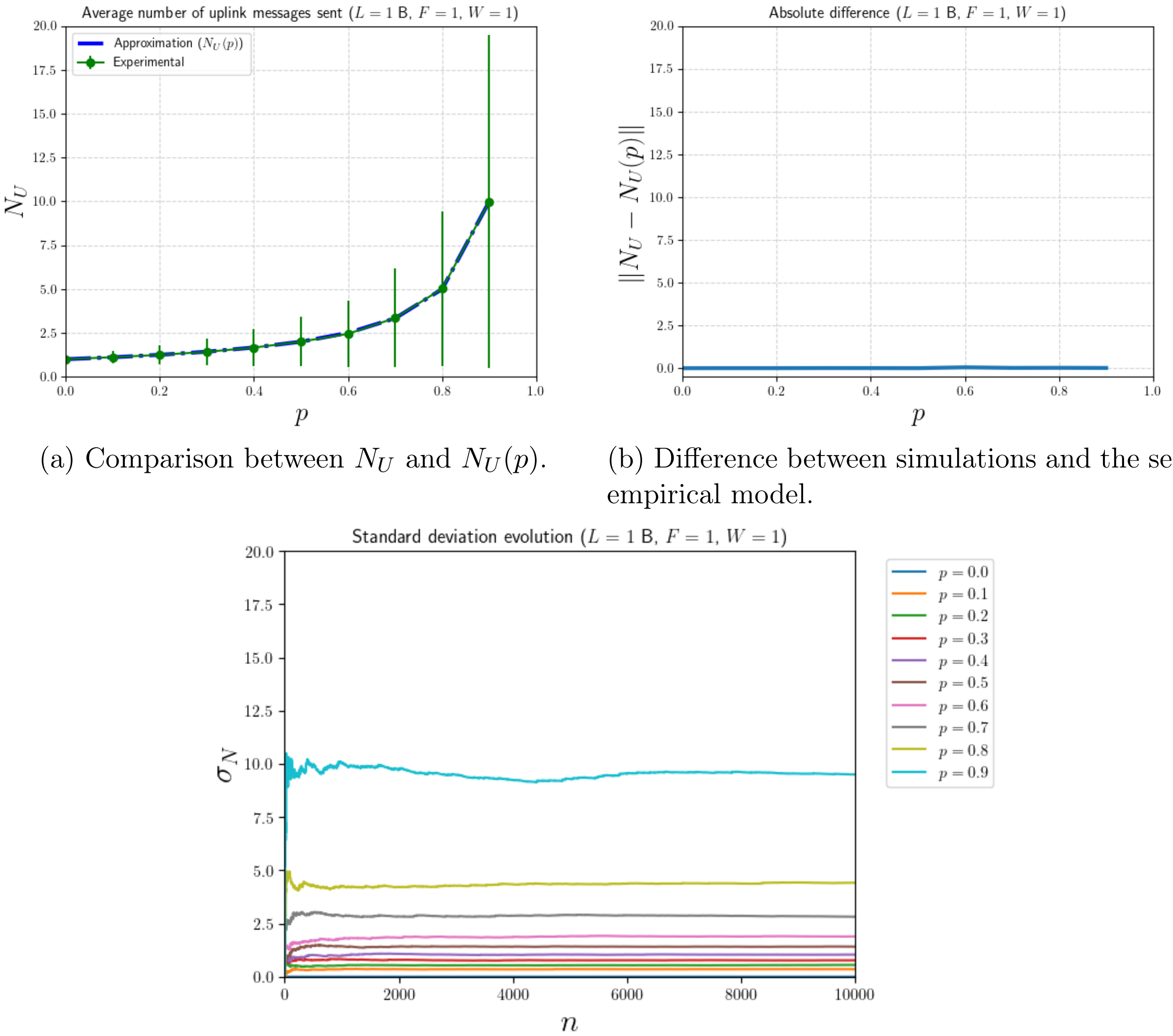
This section provides the results of the experiments performed to calculate the average number of uplink messages sent per SCHC transmissions, N_U, as described in Section 3.3.1. The results of the semi-empirical model $N_U(p)$, described in Section 5.2, are also provided.

6.1.1 Simulation results (non-aborting scenario)

These results were obtained by disabling the $R_{\max}$ limit check when the attempts counter increases. Representative results are shown in Figures 6.1 and 6.2 for $F = 1$ and $F = 17$. Other numbers of SCHC Fragments show similar behavior to that of $F = 17$, and are shown in Appendix D.

Figure 6.1a shows both curves for $F = 1$. The curve of N_U has error bars of $\pm\sigma_N$, where σ_N is the standard deviation obtained for this calculation. Figure 6.1b shows the difference between the two curves, which peaks at 0.051 for $p = 0.6$. On the other hand, Figure 6.2a shows both curves for $F = 17$, and Figure 6.2b presents the difference, which peaks at 114.656 for $p = 0.9$. The evolution of σ_N as the sample size n increases is shown in Figures 6.1c and 6.2c.

The values of N_U obtained in the simulation environment for different number of fragments are shown in Table 6.1. The values for the standard deviation obtained for each combination of L and p are shown in Table 6.2. The maximum differences between the curves for each value of L are shown in Table 6.3.

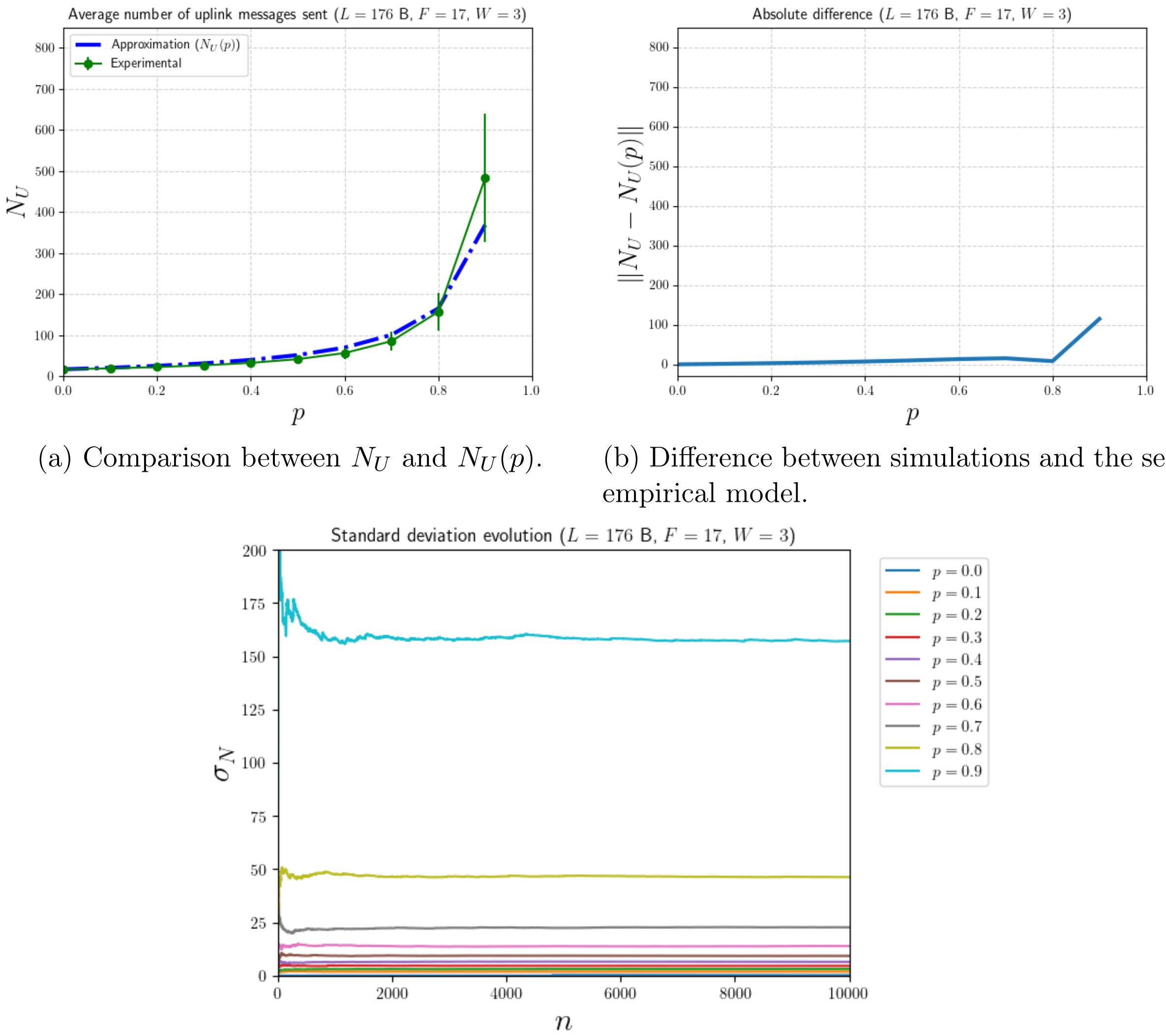


(a) Comparison between N_U and $N_U(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure 6.1: Simulation environment results for N_U and $N_U(p)$, $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between N_U and $N_U(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure 6.2: Simulation environment results for N_U and $N_U(p)$, $L = 176$ B, $F = 17$, $W = 3$.

		p									
	N_U	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
F	**1**	1.000	1.114	1.248	1.422	1.661	2.002	2.449	3.350	5.022	9.986
	5	5.000	5.973	7.217	8.836	11.235	15.051	21.435	34.543	69.304	244.024
	9	9.000	10.280	12.086	14.681	18.420	23.978	33.702	53.001	102.515	341.376
	13	13.000	15.070	17.690	21.148	26.057	33.519	46.049	70.111	131.625	422.191
	17	17.000	19.307	22.353	26.632	32.612	41.660	56.516	85.507	156.990	483.431
	21	21.000	24.086	27.887	32.823	40.060	50.583	67.988	101.127	182.028	544.390
	24	24.000	27.119	31.251	36.747	44.612	56.201	75.443	111.808	198.620	590.586
	28	28.000	31.861	36.574	43.009	51.921	65.007	86.473	126.850	223.748	644.445

Table 6.1: Values of N_U calculated by performing $n = 10,000$ transmissions per combination of F and p in the simulation scenario. Each row is coded with a green–red color gradient ranging from its minimum value to its maximum value.

		p									
	σ_N	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
L	**1**	0.000	0.353	0.556	0.773	1.040	1.414	1.894	2.819	4.414	9.512
	5	0.000	1.373	2.170	3.184	4.497	6.711	10.344	17.686	37.590	137.585
	9	0.000	1.485	2.516	3.803	5.465	7.828	12.041	20.122	41.618	146.696
	13	0.000	1.860	2.921	4.209	5.958	8.620	13.031	21.443	43.748	154.848
	17	0.000	1.903	3.143	4.595	6.491	9.276	13.902	22.713	46.299	157.176
	21	0.000	2.146	3.413	4.903	6.948	9.804	14.575	24.126	47.954	160.739
	24	0.000	2.161	3.546	5.180	7.213	10.201	15.144	24.768	48.662	161.792
	28	0.000	2.468	3.690	5.386	7.592	10.743	15.579	25.584	49.748	167.911

Table 6.2: Standard deviation values of N_U calculated by performing $n = 10,000$ transmissions per combination of F and p in the simulation scenario. Each row is coded with a green–red color gradient ranging from its minimum value to its maximum value.

F	**Max. difference**	**p**
1	0.051	0.6
5	144.330	0.9
9	151.988	0.9
13	143.110	0.9
17	114.656	0.9
21	85.921	0.9
24	64.847	0.9
28	51.546	0.8

Table 6.3: Maximum difference between $N_U(p)$ and N_U for each value of F, along with the value of p where the maximum was found.

6.1.2 Simulation results (base scenario)

In this section, results were obtained in the base scenario, where the $R_{\max}$ limit is not bypassed. Completed SCHC transmissions are reported as well as the total number of transmissions, the latter including completed and aborted transmissions. Representative results are shown in Figures 6.3 and 6.4 for $F = 1$ and $F = 17$. Other SCHC Packet sizes show similar behavior to that of $F = 17$, and are shown in Appendix E.

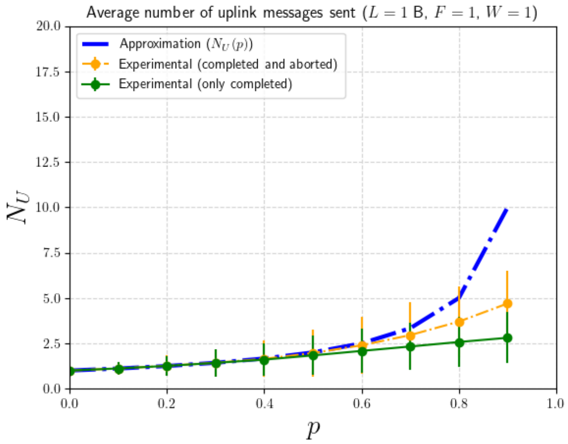


Figure 6.3: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 1$ B, $F = 1$, $W = 1$.

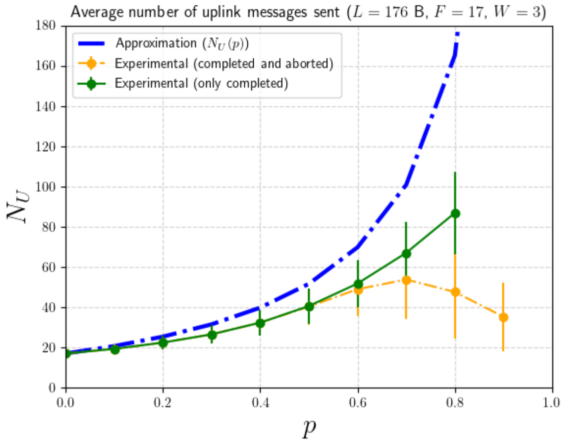


Figure 6.4: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 176$ B, $F = 17$, $W = 3$.

Figures 6.3 and 6.4 show N_U calculated both considering all SCHC transmissions and considering only completed SCHC transmissions. Note that for high values of p and for $F = 1$, the total average surpasses the average calculated in successful SCHC transmissions, whilst for $F = 17$ and other values, the average calculated in successful transmission surpasses the total average of uplink transmissions. The ratios between the calculations of N_U over the base scenario and over the non-aborting scenario are shown in Figure 6.5. This ratio considers the average number of uplink transmissions performed in all experiments, considering completed and aborted SCHC transmissions.

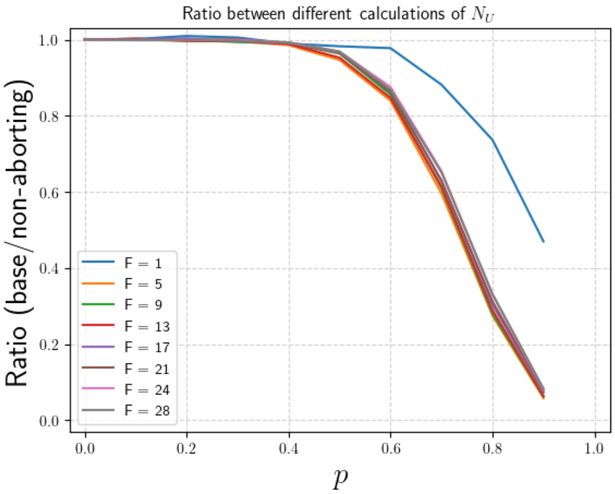
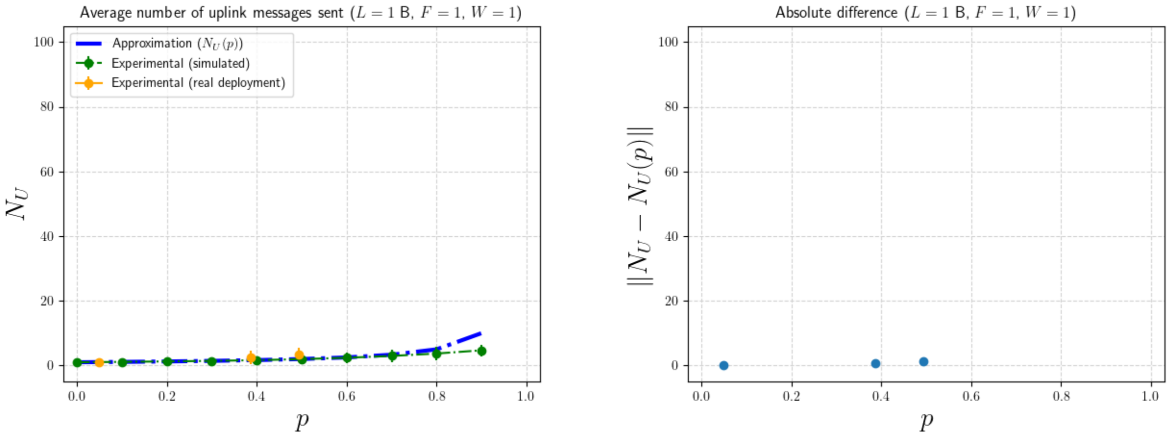


Figure 6.5: Ratio between the calculations of N_U over the base scenario and over the non-aborting scenario

6.1.3 Real deployment results

Representative results are shown in Figures 6.6 and 6.7, which display a comparison between N_U and $N_U(p)$ for $F = 1$ and $F = 17$. Results for other SCHC Packet sizes are shown in Appendix F.

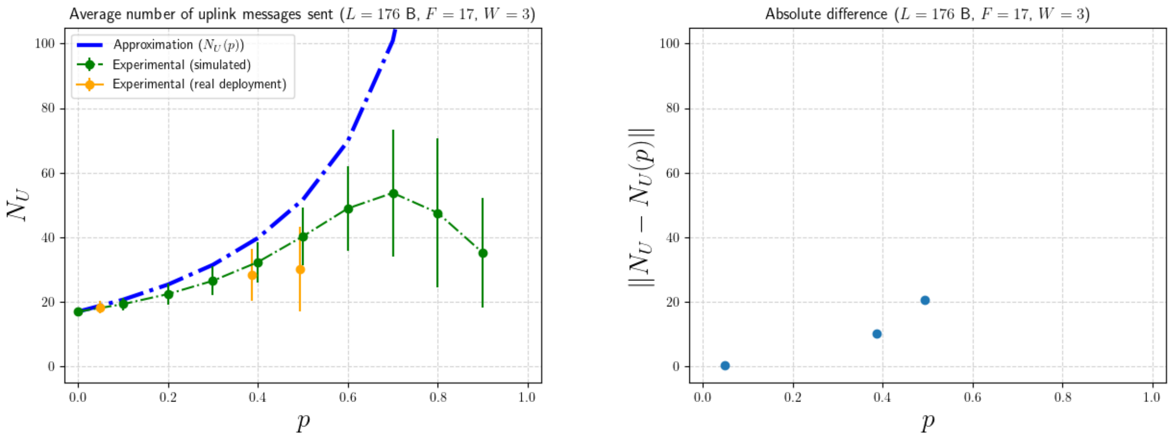
The uplink and downlink PLR were calculated by counting the number of sent and received uplink and downlink messages at both ends of the communication, respectively. The uplink PLR was calculated as the ratio between the UL count at the receiver and the UL count at the sender, whilst the downlink PLR was calculated as the ratio between the DL count at the sender and the DL count at the receiver. These calculations are shown in Table 6.4, and the value of p shown in the subsequent results is chosen as the decimal representation of these percentages.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

Figure 6.6: Real deployment results for N_U and $N_U(p)$, $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

Figure 6.7: Real deployment results for N_U and $N_U(p)$, $L = 176$ B, $F = 17$, $W = 3$.

Figures 6.6 and 6.7 show the curve of $r_S(t)$, simulation results as a reference, and the data obtained from the experiments as singular dots for $F = 1$ and $F = 17$, respectively. Both the simulation results and the real deployment results count data of all (completed and aborted) SCHC transmissions. Figures 6.6b and 6.7b show the respective differences between N_U and $N_U(p)$. The precise values for N_U and its associated standard deviation σ_N calculated in this scenario are shown in Table 6.5. As shown in Table 6.6, the difference for $F = 1$ peaks at 1.366, found at $p = 0.493$, whilst the difference for $F = 17$ B peaks at 20.586, found at $p = 0.493$.

Run	**First**	**Second**	**Third**
UL count (receiver)	12,151	10,464	12,287
UL count (sender)	12,790	20,658	20,042
UL PLR (%)	**4.996**	**49.347**	**38.695**
DL count (sender)	1,278	1,898	2,359
DL count (receiver)	1,281	1,900	2,369
DL PLR (%)	**0.234**	**0.105**	**0.422**

Table 6.4: Calculated PLR values in the uplink and the downlink direction for each run of experiments in the real deployment scenario.

		p		
	$N_U; \sigma_N$	**0.050**	**0.387**	**0.493**
F	**1**	1.020; 0.140	2.470; 2.012	3.340; 2.273
	5	5.530; 0.943	10.040; 3.197	11.940; 5.455
	9	9.550; 0.942	16.390; 5.234	17.300; 6.409
	13	13.870; 1.815	23.380; 7.731	29.970; 10.446
	17	18.370; 1.869	28.390; 8.103	30.120; 13.059
	21	22.870; 4.095	35.360; 9.101	35.490; 13.687
	24	26.320; 6.854	39.020; 10.842	39.420; 14.008
	28	30.370; 3.466	45.370; 12.194	45.000; 16.749

Table 6.5: Values of N_U and σ_N obtained in the real deployment scenario by performing $n = 100$ transmissions per number of SCHC Fragments.

F	**Max. difference**	**p**
1	1.366	0.493
5	2.217	0.493
9	9.040	0.493
13	8.553	0.493
17	20.586	0.493
21	27.399	0.493
24	32.607	0.493
28	39.210	0.493

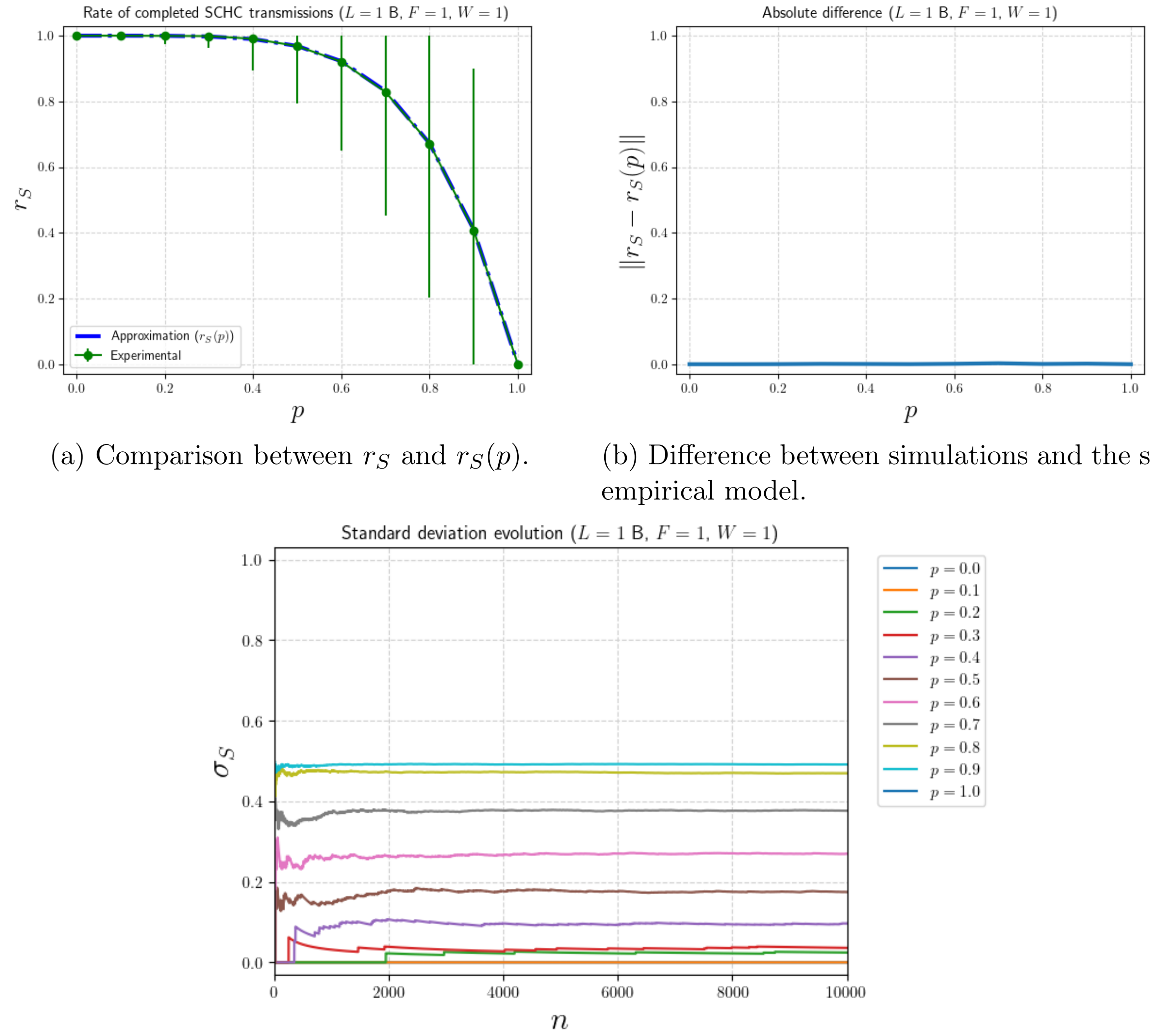
Table 6.6: Maximum difference between $N_U(p)$ and N_U found in the real deployment scenario for each value of F, along with the value of p where the maximum was found.

6.2 Rate of successful SCHC transmissions

This section provides the results of the experiments performed to calculate the rate of successful SCHC transmissions, r_S, as described in Section 3.3.1. The results of the semi-empirical model $r_S(t)$, described in Section 5.3, are also provided.

6.2.1 Simulation results

Representative results are shown in Figures 6.8 and 6.9, which display a comparison between r_S and $r_S(t)$ for $F = 1$ and $F = 17$. Other SCHC Packet sizes show similar behavior to that of $F = 17$, and are shown in Appendix H.

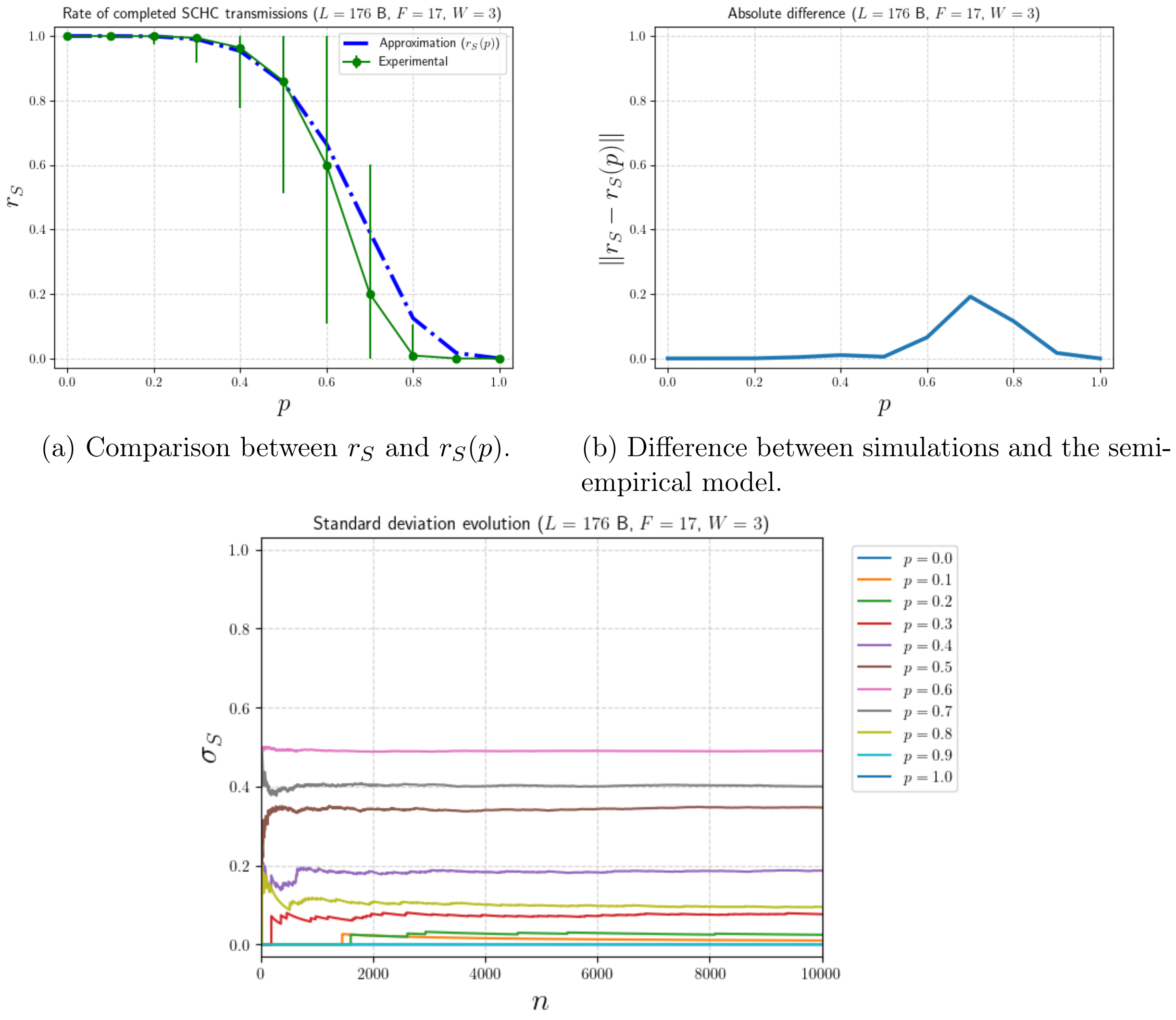


(a) Comparison between r_S and $r_S(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure 6.8: Simulation environment results for r_S and $r_S(p)$, $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between r_S and $r_S(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure 6.9: Simulation environment results for r_S and $r_S(p)$, $L = 176$ B, $F = 17$, $W = 3$.

Figure 6.8a shows both curves for $F = 1$. The curve of r_S has error bars that correspond to a single standard deviation, $\pm\sigma_S$, which are capped not to exceed the $[0, 1]$ interval. Figure 6.8b shows the difference between the two curves, which peaks at 0.003 for $p = 0.7$. On the other hand, Figure 6.9a shows both curves for $F = 17$ B, and Figure 6.9b presents the difference, which peaks at 0.192 for $p = 0.7$. The evolution of the standard deviation of r_S, σ_S, as the sample size n increases is shown in Figures 6.8c and 6.9c.

The values of r_S obtained in these experiments are shown in Table 6.7. The values for the standard deviation obtained for each combination of F and p are shown in Table 6.8. The maximum differences between the curves for each value of F are shown in Table 6.9.

		p									
	r_S	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
F	**1**	1.000	1.000	0.999	0.999	0.990	0.968	0.921	0.829	0.672	0.408
	5	1.000	1.000	1.000	0.993	0.973	0.898	0.698	0.355	0.070	0.001
	9	1.000	1.000	1.000	0.993	0.972	0.887	0.650	0.274	0.027	0.000
	13	1.000	1.000	0.999	0.994	0.965	0.864	0.607	0.228	0.015	0.000
	17	1.000	1.000	0.999	0.994	0.964	0.860	0.599	0.200	0.009	0.000
	21	1.000	1.000	0.999	0.994	0.962	0.844	0.574	0.177	0.008	0.000
	24	1.000	1.000	1.000	0.992	0.964	0.849	0.570	0.169	0.006	0.000
	38	1.000	1.000	0.999	0.992	0.956	0.841	0.547	0.156	0.003	0.000

Table 6.7: Values of r_S calculated by performing $n = 10,000$ transmissions per combination of F and p in the simulation scenario. Values close to 1, 0.5 and 0 are coded with a green–white–red color gradient.

		p									
	σ_S	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
F	**1**	0.000	0.000	0.024	0.036	0.097	0.175	0.270	0.377	0.470	0.491
	5	0.000	0.000	0.022	0.081	0.163	0.302	0.459	0.478	0.255	0.033
	9	0.000	0.000	0.022	0.081	0.165	0.316	0.477	0.446	0.161	0.000
	13	0.000	0.000	0.035	0.076	0.183	0.343	0.488	0.420	0.120	0.000
	17	0.000	0.010	0.024	0.077	0.187	0.347	0.490	0.400	0.095	0.000
	21	0.000	0.000	0.024	0.079	0.192	0.363	0.494	0.382	0.088	0.000
	24	0.000	0.000	0.017	0.087	0.186	0.358	0.495	0.375	0.075	0.000
	28	0.000	0.000	0.030	0.091	0.204	0.365	0.498	0.363	0.056	0.000

Table 6.8: Standard deviation values of r_S calculated by performing $n = 10.000$ transmissions per combination of F and p in the simulation scenario. Values close to 0 and 0.5 are coded with a green–red color gradient.

F	**Max. difference**	**p**
1	0.003	0.7
5	0.137	0.8
9	0.142	0.7
13	0.172	0.7
17	0.192	0.7
21	0.210	0.7
24	0.215	0.7
28	0.225	0.7

Table 6.9: Maximum difference between $r_S(p)$ and r_S for each value of F, along with the value of p where the maximum was found.

6.2.2 Simulation results with different timeouts

Additional experiments were performed to calculate r_S with timer values set to 1 s instead of 0.1 s. The maximum differences between both calculations found for different configurations of F and p are shown in Table 6.10.

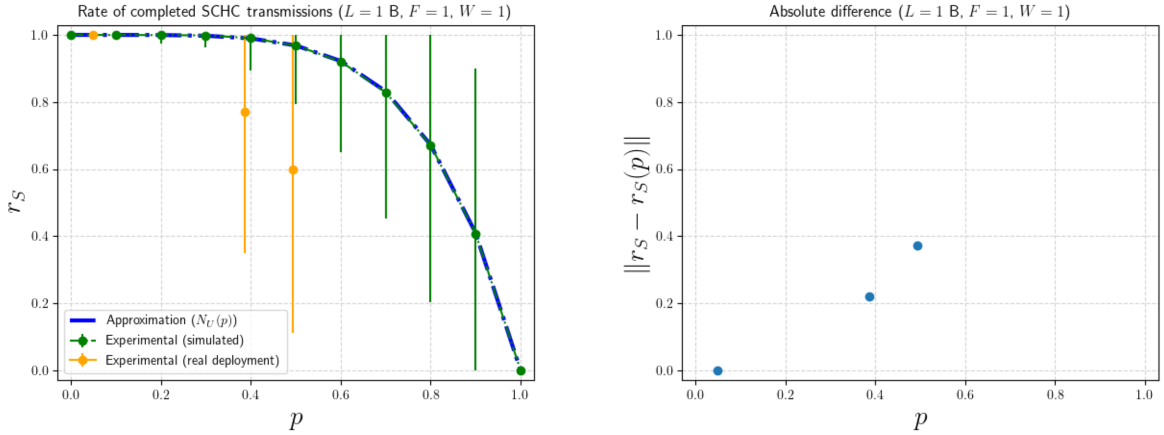
F	**Max. difference**	**p**
1	0.014	0.7
5	0.015	0.5
9	0.025	0.5
13	0.022	0.6
17	0.010	0.6
21	0.007	0.4
24	0.023	0.5
28	0.015	0.5

Table 6.10: Maximum difference between N_U calculated for timers of 0.1 s and of 1 s for each value of F, along with the value of p where the maximum was found.

6.2.3 Real deployment results

Representative results are shown in Figures 6.10 and 6.11, which display a comparison between r_S and $r_S(p)$ for $F = 1$ and $F = 17$, as well as displaying r_S calculated over the simulation environment as a reference. Results for other SCHC Packet sizes are shown in Appendix I.

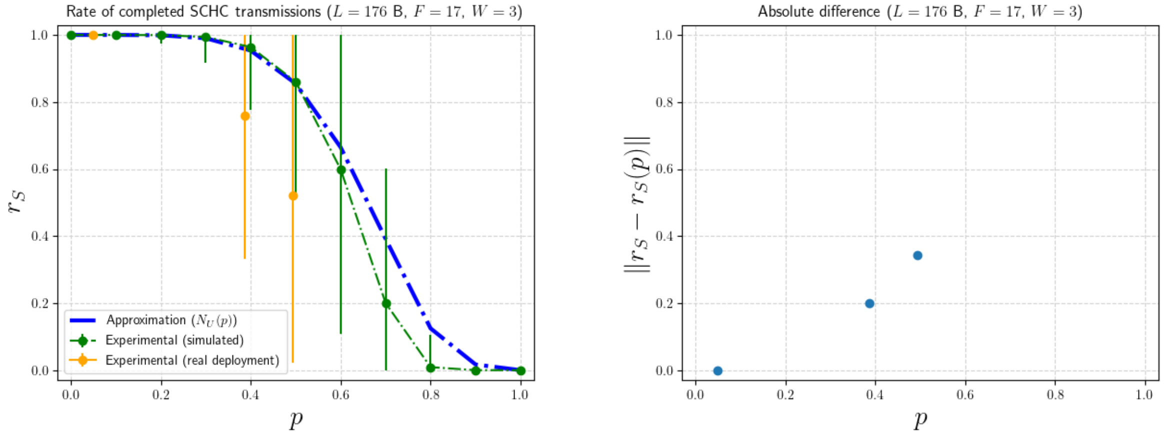
The uplink and downlink PLR are the same as those reported in Table 6.4. Figures 6.10 and 6.11 show the curve of $r_S(t)$ and the data obtained from the experiments as singular dots for $F = 1$ and $F = 17$, respectively. Figures 6.10b and 6.11b show the respective differences between r_S and $r_S(p)$. The precise values for r_S and its associated standard deviation σ_S calculated in this scenario are shown in Table 6.11. As shown in Table 6.12, the difference for $F = 1$ peaks at 0.371, found at $p = 0.493$, whilst the difference for $F = 17$ peaks at 0.344, found at 0.493.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

Figure 6.10: Real deployment results, $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

Figure 6.11: Real deployment results, $L = 176$ B, $F = 17$, $W = 3$.

			p	
	r_S; σ_S	**0.050**	**0.387**	**0.493**
F	**1**	1.000; 0.000	0.600; 0.421	0.770; 0.490
	5	1.000; 0.000	0.560; 0.427	0.760; 0.496
	9	1.000; 0.000	0.540; 0.421	0.770; 0.498
	13	1.000; 0.000	0.500; 0.421	0.770; 0.500
	17	1.000; 0.000	0.520; 0.427	0.760; 0.499
	21	1.000; 0.000	0.500; 0.444	0.730; 0.500
	24	1.000; 0.000	0.470; 0.454	0.710; 0.499
	28	1.000; 0.000	0.440; 0.466	0.680; 0.496

Table 6.11: Values of r_S and σ_S obtained in the real deployment scenario by performing $n = 100$ transmissions per number of SCHC Fragments.

F	**Max. difference**	**p**
1	0.371	0.493
5	0.320	0.493
9	0.330	0.493
13	0.366	0.493
17	0.344	0.493
21	0.363	0.493
24	0.392	0.493
28	0.421	0.493

Table 6.12: Maximum difference between $r_S(p)$ and r_S found in the real deployment scenario for each value of F, along with the value of p where the maximum was found.

Chapter 7

Analysis

This section provides an analysis of the results obtained in Chapter 6, validating the semi-empirical models defined in Chapter 5 and the hypotheses proposed in Chapter 1.

7.1 Average number of uplink messages per SCHC transmission

7.1.1 Semi-empirical model and non-aborting scenario

In Section 1.2, it was hypothesized that the number of messages needed to complete a SCHC transmission was always finite, for every PLR unequal to 100 %. A theoretical proof for this hypothesis could not be provided, since it requires defining a purely theoretical model for N_U. However, the empirical evidence obtained in the results laid out in Section 6.1 supports this claim, as no infinite SCHC transmissions were detected for $p \neq 1$ during the experiments. Moreover, the evolution of the standard deviation of N_U regarding sample size reached stabilization in all experiments. As will be discussed in the following paragraphs, this implies that the measurements of N_U do not signal irregular loops neither in the SCHC/Sigfox algorithm nor its implementation.

Every experiment performed for calculating N_U over the non-aborting scenario (bypassing the $R_{\max}$ limit), for every combination of p and F, finished successfully. This shows that the SCHC/Sigfox algorithm does not contain conditions that would require sending an infinite number of SCHC Fragments, as long as $p \neq 1$. The case for $p = 1$ was not executed, since it would trivially lead to infinite SCHC transmissions; however, infinite SCHC transmissions were not detected for other values of p.

The difficulty in providing a theoretical proof resides in determining the difference between $N_{U_T}(p)$ and N_U. As explained in Section 5.2, $N_{U_T}(p)$ considers all the transmissions of SCHC Fragments until they are correctly delivered. However, the All-1 fragment needs to be sent every time a final retransmission cycle is performed, even if it was correctly received previously. The event of sending the All-1 until it is received is repeated each time a final retransmission cycle is performed and adds additional uplink message overhead. This study did not address the number of final retransmission cycles for a value of p, which is directly related to this event.

$N_{U_T}(p)$ closely resembles the calculated N_U over the non-aborting scenario for $F = 1$ with a minimal difference, as shown in Figure 6.1. However, other values of F show a significant difference. This difference is thought to arise from the final retransmission cycles, particularly from the additional retransmissions of the All-1 fragment. Note that the difference is negligible for $F = 1$ since there are no retransmission cycles in a single-fragment SCHC transmission.

The difference was compensated by proposing $N_{U_E}(p)$ and defining the semi-empirical model as $N_U(p) = N_{U_T}(p) + N_{U_E}(p)$. Although there is still a difference between N_U and $N_U(p)$, the empirical adjustment reduces it from a global maximum of 364.445 ($L = 307$ B, $p = 0.9$) to a global maximum of 151.988 ($L = 88$ B, $p = 0.9$), as shown in Tables 5.3 and 6.3. As the variables of N_{U_E} were chosen not to exceed the results, the curves for $L = 307$ B show the most similarity, as shown in Figure D.8. Globally, the model $N_U(p)$ shows the most similarity to N_U for $p \in [0.0, 0.8]$. $N_U(p)$ is therefore concluded to provide a good approximation for N_U, especially for $p \in [0.0, 0.8]$.

As the FLR increases, the associated standard deviation also increases. Such high standard deviation values for FLR greater than 80% impede using N_U as an empirical prediction for high values of FLR. The standard deviation plots of Figures 6.1c and 6.2c, and those found in Appendix D show a stabilization roughly around $n = 2,000$ samples, with minimal variations after that threshold. The final values of σ_N were reported in Table 6.2. High values of p show the largest standard deviations for every L. Although the standard deviation can be large for high values of p and L, the stabilization of these curves signals that the results always appear within the same range, and do not show trends that could signal unwanted absorbing loops in the algorithm.

7.1.2 Base scenario

The values of N_U are lower when calculated over the base scenario (where the $R_{\max}$ limit is considered), as shown in Figures 6.3, 6.4 and Appendix E. This is a direct consequence of SCHC/Sigfox aborting the transmissions when network conditions are insufficient: Transmissions that would have taken a large number of uplink messages to be completed—consuming large amounts of time and energy—are not executed. Aborting SCHC transmissions after exceeding the limit consumes, on average, less resources.

In the base scenario, N_U is calculated in two ways: considering all SCHC transmissions and considering only completed SCHC transmissions. As will be discussed in the analysis of r_S, as both p and F increase, the number of completed SCHC transmissions decreases. This leads to calculations of N_U that are significantly lower than the semi-empirical model $N_U(p)$, a consequence of the $R_{\max}$ limit impeding the delivery of large numbers of uplink messages. Moreover, since the channel is closed when an excessive number of uplink transmissions is to be avoided, successful SCHC transmissions are fewer. Note that for $F \geq 13$ and $p = 0.9$ no completed SCHC transmission was recorded, which is reinforced by the r_S calculations being close to 0.

In Figures 6.3, 6.4 and Appendix E, a particular behavior of the calculations of N_U is noted: For $F = 1$, the calculation of N_U considering completed and aborted transmissions is greater than the calculation considering only completed transmissions; on the other hand, for $F > 1$, both calculations show the opposite relation.

The behavior of N_U for $F = 1$ arises from successful transmissions consisting only of transmissions and retransmissions of the first and only fragment, while aborted transmissions spend more messages before finishing since the fragment needs to be retransmitted (and lost) multiple times. Calculating N_U while considering the larger number of uplink messages sent in aborted transmissions increases its average value.

On the other hand, for $F > 1$, N_U for completed transmissions is greater than for all transmissions. This is attributed to final retransmission cycles. These cycles imply a considerably larger number of uplink retransmissions in completed SCHC transmissions. However, aborted SCHC transmissions are more likely to happen at high values of p, and often take less messages to close the channel. Calculating N_U while considering this low number of uplink messages in aborted SCHC transmissions increases its average value.

The ratio between the calculations of N_U over the non-aborting scenario and the base scenario is often 1 or less, as shown in Figure 6.5. For $F = 1$, there were cases where the ratio was greater than 1 ($p = 0.2$ and $p = 0.3$), meaning that the messages sent over the base scenario were slightly more than the messages sent over the non-aborting scenario. This difference, however, is not thought to have an important significance. In all other cases, the ratio signals that N_U, when calculated over the non-aborting scenario, acts as an upper bound of N_U over the base scenario.

7.1.3 Real deployment

The results of the real deployment scenario, shown in Appendix F, fail to resemble the results of the model. Although the results of the simulation scenario have a smaller difference with the results of the real deployment, it is still considerable for large SCHC Packet sizes. This is attributed to the sample size, $n = 100$, being too small to provide reliable results, since it is significantly lower than the sample size of the simulation results, $n = 10,000$. Moreover, this challenges the assumption that the loss of a SCHC Fragment is modeled by an *iid* random variable. Downlink losses were found, as shown in Table 6.4, which also challenges the assumption that all Compound ACKs are correctly delivered to the sender; however, the low PLR calculated on the downlink is not considered to have a significant influence in these results.

7.1.4 Interpretation

The behavior of the ratio between the calculations of N_U over the non-aborting scenario and over the base scenario signals that aborting SCHC transmissions under harsh network conditions can reduce the average number of uplink messages sent down to 50% for $F = 1$ and down to around 10% for $F > 1$. Furthermore, since the transmission of uplink messages entails usage of network resources such as device battery, channel occupancy, transmission delay, processing power, and others, it is concluded that aborting SCHC transmissions consumes a significantly lower amount of network resources than waiting for SCHC transmissions to finish successfully.

7.2 Rate of successful SCHC transmissions

7.2.1 Semi-empirical model and simulation results

As previously stated, the shape of the curves changes with F. The curve for $F = 1$ shows a significantly different behavior to that of other values of F. Moreover, as F increases, the curves slightly change their values. This led to including F in the proposed model $r_S(p)$, and refutes the hypothesis proposed for this metric, that is, that the rate of successful SCHC/Sigfox transmissions is independent of the number of fragments generated.

Figures 5.3a and 6.8a show that the theoretical part of $r_S(p)$ closely resembles the calculated r_S when $F = 1$, showing a minimal difference. The theoretical part $r_{S_T}(p)$ accurately models the behavior of a SCHC/Sigfox transmission of a single SCHC Fragment, the All-1, whose only success condition is not being lost $R_{\max}$ times in a row. A difference still appears, however, as more fragments are considered in the model.

Although the results for other fragment sizes still show a difference between r_S and the model, the empirical adjustment reduces it from a global maximum of 0.676 ($F = 28$, $p = 0.7$) to 0.225 ($F = 28$ B, $p = 0.7$), shown in Tables 5.5 and 6.9. The model $r_S(p)$ shows the most similarity to the results for $p \in [0.0, 0.5] \cup [0.9, 0.0]$. The range $p \in [0.6, 0.8]$ shows the most difference, which could not be satisfied without further complicating the semi-empirical model. $r_S(p)$ is concluded to be a good model for r_S, especially in the ranges $p \in [0.0, 0.5] \cup [0.9, 0.0]$ but still valid for $p \in [0.6, 0.8]$.

The results show different curve shapes for $F = 1$ and for $F > 1$. This abrupt change in the curves is thought to stem from final retransmission cycles, similarly to the analysis of $N_U(p)$. Although it is true that a SCHC/Sigfox transmission is aborted when the All-1 is lost $R_{\max}$ times, every final retransmission cycle presents an additional opportunity for this event to happen. This is only possible if $F > 1$, since by definition there are no possible retransmission cycles for $F = 1$. Therefore, the higher the number of retransmission cycles, the more trials of sending the All-1 are executed, and the more likely it is for a SCHC transmission to be aborted. This study did not address the number of final retransmission cycles for a value of p. Further theoretical studies require to address the number of final retransmission cycles in a SCHC/Sigfox transmission as a function of p, which is thought to lead the way to a purely theoretical model for r_S.

The standard deviation associated to r_S, σ_S, peaks when $r_S = 0.5$ and signals complete uncertainty over SCHC transmission completion, therefore impeding using r_S as a prediction when the FLR is associated with a high σ_S. The standard deviation plots of Figures 6.8c, 6.9c, and those found in Appendix H show a stabilization roughly around $n = 2,000$ samples, with minimal variations after that threshold. The final values of σ_S were reported in Table 6.8, and closely approximate the values of the standard deviation of a binomial distribution of associated probability r_S. Note that for values of r_S close to 0 or to 1, σ_S approaches 0, which further implies that the vast majority (if not all) of the experiments performed for that combination of p and F resulted either in success or failure. This relation between r_S and σ_S arises from the success (or failure) of a single SCHC/Sigfox transmission being modeled by a binomial random variable. This reinforces the statement of Equation 5.17, that is, r_S approximates the success probability p_S as the sample size increases.

7.2.2 Results with different timeouts

The maximum differences between r_S calculated for timeout values of 0.1 s and of 1 s peak at 0.025 ($L = 88$ B, $p = 0.5$), as reported in Table 6.10. These differences are not significant, therefore, this performance metric is concluded not to depend on the timeout values. No clear dependency of r_S neither on the number of windows W nor on the number of fragments of the last window U was found.

7.2.3 Real deployment

Similarly to the results of the real deployment scenario for N_U, the results for r_S, shown in Appendix I, fail to resemble the results of the simulation scenario. This is again attributed to the sample size being far smaller than the sample size of the simulation results.

Since the success of a single SCHC transmission is modeled by a binomial distribution, as r_S reaches 0.5, its variability increases up to 0.5. If network conditions are estimated to provide a low r_S or a high σ_S, it is recommended either to use another reliable delivery mechanism or to wait for better channel conditions to perform the transmission. For example, if $0.4 \leq r_S \leq 0.6$, $\sigma_S \geq \sqrt{0.4 \times 0.6} = 0.489$, which is a standard deviation close enough to 0.5 to consider the success or failure of a SCHC transmission highly uncertain.

7.2.4 Interpretation

The following guides are proposed to interpret this performance metric:

- If $1 \geq r_S \geq 0.9$, then $\sigma_S \leq 0.3$. SCHC transmissions are likely to be completed with low uncertainty.

- If $0.9 > r_S \geq 0.7$, then $0.3 > \sigma_S \leq 0.458$. SCHC transmissions are likely to be completed; however, care must be taken not to overly use network resources.
- If $0.7 > r_S \geq 0.5$, then $0.458 > \sigma_S \leq 0.5$. Although a SCHC transmission can be completed, it is recommended to wait for better channel conditions.
- If $0.5 > r_S$, SCHC transmissions are unlikely to be completed successfully. It is recommended to use another reliable delivery method or to wait for better conditions.

7.3 General observations

The models for both performance metrics consist of a theoretical component and an empirical adjustment. The empirical adjustment was needed because the theoretical approaches do not represent the behavior of the empirical results. This difference is thought to stem from the final retransmission cycles of a SCHC transmission, which add additional message overhead that was not considered in the formulation of the theoretical approach. A purely theoretical model for N_U should consider the additional retransmissions of the All-1 fragment, whilst a purely theoretical model for r_S should consider the expected number of final retransmission cycles.

The variables and models presented in Chapter 5 consider that a single tile is carried in a single SCHC Fragment, as per SCHC/Sigfox specifications. This is not always the case for other LPWANs, since the generic definition of SCHC states that a SCHC Fragment carries at least one tile. When employing a similar analysis in other networks such as LoRaWAN or NB-IoT, care must be taken with the definition of variables such as the number of fragments and windows generated.

The TDD approach used in the development of the SCHC/Sigfox implementation provided stability and robustness for the software. Constant testing before and during the developing process allowed most errors to be found before migrating development to the next phases. Not doing so would often imply difficult debugging during long periods of time. The separation of the development process into different phases have also contributed to this.

The simulation provides faster results than the real implementation. It is also independent of the real network channel conditions, allowing SCHC transmissions to be tested in a controlled environment. It has proven to be a useful tool which facilitates developing and testing, specially in zones where Sigfox coverage is unstable or insufficient.

Chapter 8

Conclusions and Future Work

This chapter provides conclusions on the work presented in this thesis by summarizing the analyses of the results obtained from the SCHC/Sigfox implementation developed in the course of this investigation. The limitations of the experiments and the software are addressed, and future work is proposed for following studies on SCHC/Sigfox.

8.1 Conclusions

This thesis proposed performance metrics for SCHC/Sigfox: the average number of uplink messages sent per SCHC transmission, N_U, and the rate at which SCHC/Sigfox transmissions are completed successfully, r_S. When calculated with a sample size large enough, estimated around $n = 2,000$, these performance metrics converge to their theoretical counterparts: the expected number of uplink messages per SCHC/Sigfox transmission and the success probability of a SCHC/Sigfox transmission.

The average number of uplink messages per SCHC transmission, N_U, is useful in quantifying the impact of placing a limit in ARQ retransmissions that aborts the communication instead of spending large amounts of network resources trying to deliver a packet over a channel with insufficient network conditions. It also provides a way to calculate the efficiency of an ARQ protocol in terms of uplink message overhead, and can be used to compare SCHC/Sigfox ACK-on-Error with other ARQ protocols.

The rate of successful SCHC transmissions, r_S, is a useful performance metric for SCHC deployments. It signals the probability of a SCHC transmission to be complete. If r_S is estimated to be under 0.7, it is recommended not to perform a SCHC transmission or to wait for better channel conditions. If a reliable transmission is desired under those channel conditions, other ARQ mechanisms with better r_S could be chosen.

Both performance metrics are justified as predictions of their theoretical counterparts, the expected number of uplink messages per SCHC transmission and the probability of a SCHC transmission to be completed successfully. The values obtained for these metrics can be used to predict the outcome of a SCHC transmission only if the associated standard deviation is not large enough to signal high uncertainty, which is the case for N_U when FLR increases over 80% and for r_S when r_S is near 0.5.

Semi-empirical models are provided for both performance metrics. The theoretical components of both models are calculated analytically by means of a probabilistic analysis. Empirical adjustments are performed by proposing a candidate function that compensates the difference between the theoretical model and the measured values.

The model $N_U(p)$ is concluded to provide a good approximation for N_U when calculated in a scenario where SCHC transmissions are configured not to be aborted, especially for $p \in [0.0, 0.8]$. As the measurement of N_U in this scenario takes a large number of SCHC transmissions, the model $N_U(p)$ is provided to save network resources instead. Aborting SCHC transmissions in harsh network conditions was found to reduce the average number of uplink messages sent down to 50% for single-fragment SCHC transmissions and down to around 10% for other numbers of SCHC Fragments. Since transmitting uplink messages consumes network resources such as device battery, channel occupancy, transmission delay, processing power, and others, it is concluded that network resources usage is significantly lower when aborting difficult SCHC transmissions.

The model $r_S(p)$ is concluded to provide a good approximation for r_S, especially in the ranges $p \in [0.0, 0.5] \cup [0.9, 0.0]$ but still valid for $p \in [0.6, 0.8]$. It is proposed that the results obtained for r_S and $r_S(t)$ be taken into consideration when deploying SCHC applications if channel conditions are known. The definition of this model to calculate this performance metric also save network resource usages instead of performing a large number of SCHC transmissions.

The implementation developed over the course of SCHC/Sigfox standardization and this study has proven to be a useful tool which facilitates testing new changes in the SCHC Profile definition and their impact on the whole protocol, as well as studying the performance of the SCHC fragmentation mechanism. Thanks to this implementation, a number of contributions on SCHC/Sigfox were made in collaboration with the co-authors of the Profile, which led the way into its current standardization process. This implementation also made possible executing other empirical studies on SCHC/Sigfox [12], and has recently incorporated the architecture proposed in [22]. The new additions to SCHC/Sigfox were discussed with other members of the IETF LPWAN Working Group to obtain feedback, which has been quickly integrated into SCHC/Sigfox thanks to the simulation environment, a key part of the implementation.

The software implementation led to contribute to the Sigfox SCHC Profile definition. The performance metrics N_U and r_S were defined, justified and measured. Moreover, the impact of aborting SCHC transmissions over N_U was addressed. Semi-empirical models were proposed for both performance metrics, which provide a good approximation to their measured values. In summary, the objectives proposed in Section 1.3 were fulfilled.

The following list summarizes the contributions of this thesis in the fields of novel ARQ protocols over constrained networks, ARQ performance modeling and evaluation, IoT deployments over LPWANs, and SCHC standardization:

- Proposed two performance metrics for the ACK-on-Error mode of SCHC over the Sigfox network: The average number of uplink messages sent per SCHC transmission, N_U, and the rate at which transmissions of fragmented SCHC Packets are performed successfully over the Sigfox network, r_S.
- Provided N_U as a means to calculate the efficiency of the ACK-on-Error ARQ protocol regarding channel usage overhead.
- Provided a guide to interpret r_S, which can be used by network operators with access to r_S measurements to decide whether to incorporate SCHC ACK-on-Error into a particular Sigfox application.
- Defined semi-empirical models for both performance metrics, which can be used as approximations over stable channel conditions.
- Developed a publicly available implementation of SCHC/Sigfox, which is deployable over Sigfox-enabled transmitter devices and over the Sigfox backend.
- Developed a simulator of SCHC/Sigfox, which can be used to perform experiments over a controlled local network link in a significantly lower amount of time than over the actual Sigfox network.
- Co-authored two specifications for the Sigfox SCHC Profile, both of them on the track to become a Proposed Standard at the IETF.

8.2 Limitations and future work

This study did not address the number of final retransmission cycles for different SCHC Packet sizes and FLR values. This is thought to have a fundamental impact on the difference between the measured performance metrics and their theoretical models. This led to the definition of empirical adjustments by proposing a candidate function and adjusting its parameters using LSM and inspection. A further analysis on the number of retransmission cycles as a function of the SCHC Packet size and the FLR is proposed for further studies, which is thought to lead to a purely theoretical model for N_U and r_S. Moreover, a purely theoretical model for the performance metrics proposed in this work would also be applicable to other operational modes of SCHC/Sigfox.

The `MAX_ACK_REQUESTS` limit, $R_{\max}$, is set to 5 as per SCHC/Sigfox specifications [3]. The impact of different values of $R_{\max}$ on r_S is out of the scope of this thesis and was not addressed. However, it is thought that if $R_{\max}$ is raised, r_S would be higher for high values of FLR, although it would use more network resources. It is also thought that if a purely theoretical model for r_S is developed, that relation will be clear.

The sample size of the experiments over the real deployment scenario was too small to resemble the results of the simulation scenario, which is attributed to the sample size, $n = 100$, being far lower than the proposed stability threshold, $n = 2000$. It is thought that under stable channel conditions, performing more experiments in the real deployment would make the results resemble the ones obtained for the simulation scenario. This also highlights the importance of the simulation in testing the protocol, since more experiments can be performed in less time.

Further development is required for the SCHC/Sigfox implementation to be usable as a library. Although it can be used for real SCHC/Sigfox applications, it requires manual configuration of certain variables. This is undesirable in a public library, where all the internal variables are configured not by the user but by the software itself.

Further works on Sigfox quality of service should challenge the assumption that the FLR can be modeled as an independent and identically distributed random variable. Given the intellectual property rights of the Sigfox modulation scheme, the impact of the error-correcting mechanisms of Sigfox over the PLR is not known and has not been studied. In this work, it is assumed that the PLR is independent of the packet size; therefore, the results are provided in relation to the number of SCHC Fragments instead of the size of the SCHC Packet. However, the relation between the PLR and the Sigfox packet size has also not been studied. If a model is proposed for the Sigfox PLR, an extension of this study that supports a variable PLR can be performed.

Bibliography

[1] S. Farrell, "IETF RFC8376: Low-Power Wide Area Network (LPWAN) Overview," 2018. [Online]. Available: https://datatracker.ietf.org/doc/rfc8376/

[2] Sigfox, "Radio Configurations | Sigfox build." [Online]. Available: https://build.sigfox.com/sigfox-radio-configurations-rc

[3] J. C. Zúñiga, C. Gomez, S. Aguilar, L. Toutain, S. Céspedes, D. Wistuba, and J. Boite, "SCHC over Sigfox LPWAN (IETF Internet-Draft)," 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lpwan-schc-over-sigfox/

[4] Pycom, "Lopy4 datasheet version 1.1," 2020. [Online]. Available: https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf

[5] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J. C. Zúñiga, "IETF RFC8724: SCHC: Generic Framework for Static Context Header Compression and Fragmentation," 2020. [Online]. Available: https://datatracker.ietf.org/doc/rfc8724/

[6] J. C. Zúñiga, C. Gomez, S. Aguilar, L. Toutain, S. Céspedes, and D. Wistuba, "SCHC Compound ACK (IETF Internet-Draft)," 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lpwan-schc-compound-ack/

[7] S. Deering and R. Hinden, "Ietf rfc8200: Internet protocol, version 6 (ipv6) specification," 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8200.html

[8] O. Gimenez and I. Petrov, "IETF RFC9011: Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN." [Online]. Available: https://datatracker.ietf.org/doc/rfc9011/

[9] E. Ramos and A. Minaburo, "SCHC over NB-IoT (IETF Internet-Draft)," 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lpwan-schc-over-nbiot/

[10] Sigfox, "Qualification | Sigfox build." [Online]. Available: https://build.sigfox.com/study

[11] D. Wistuba, S. Céspedes, J. C. Zúñiga, R. Muñoz, S. Aguilar, C. Gomez, and R. Vidal, "An Implementation of IoT LPWAN SCHC Message Fragmentation and Reassembly," *CEUR Workshop Proceedings*, vol. 2988, 2021.

[12] S. Aguilar, D. Wistuba, A. Platis, R. Vidal, C. Gomez, S. Céspedes, and J. C. Zúñiga, "Packet Fragmentation Over Sigfox: Implementation and Performance Evaluation of SCHC ACK-on-Error," *IEEE Internet of Things Journal*, vol. 9, 2021.

[13] Institute of Electrical and Electronics Engineers, "IEEE 802.15 Working Group for Wireless Personal Area Networks (WPANs)." [Online]. Available: https://ieee802.org/15/index.html

[14] IETF LPWAN Working Group, "IPv6 over Low Power Wide-Area Networks (LPWAN)," 2016. [Online]. Available: https://datatracker.ietf.org/wg/lpwan/about/

[15] Semtech, "LoRa® and LoRaWAN®: A Technical Overview," 2019. [Online]. Available: https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf

[16] L. Alliance®, "What is LoRaWAN®," 2015. [Online]. Available: https://lora-alliance.org/resource_hub/what-is-lorawan/

[17] S. A. Gbadamosi, G. P. Hancke, and A. M. Abu-Mahfouz, "Building upon NB-IoT Networks: A Roadmap towards 5G New Radio Networks," *IEEE Access*, vol. 8, 2020.

[18] Sigfox, "Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)." [Online]. Available: https://www.sigfox.com/

[19] Wi-SUN Alliance, "Wi-SUN Alliance." [Online]. Available: https://wi-sun.org/

[20] N. Ahmed, D. De, F. A. Barbhuiya, and M. I. Hussain, "MAC Protocols for IEEE 802.11ah-Based Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 9, 2022.

[21] Acklio, "Acklio - SCHC IETF 8724." [Online]. Available: https://www.ackl.io/technology/schc

[22] D. Wistuba, S. Céspedes, and J. Bustos-Jiménez, "Modeling SCHC ACK-on-Error Fragment Delivery over Sigfox," *Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 115–119, 10 2022.

[23] E. J. Sebastian, A. Sikora, M. Schappacher, and Z. Amjad, "Test and Measurement of LPWAN and Cellular IoT Networks in a Unified Testbed," *IEEE International Conference on Industrial Informatics (INDIN)*, vol. 2019-July, 2019.

[24] M. Saelens, J. Hoebeke, A. Shahid, and E. D. Poorter, "Impact of EU Duty Cycle and Transmission Power Limitations for sub-GHz LPWAN SRDs: An Overview and Future Challenges," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, p. 219, 12 2019.

[25] I. Suciu, X. Vilajosana, and F. Adelantado, "An Analysis of Packet Fragmentation Impact in LPWAN," *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 4 2018.

[26] ——, "Aggressive Fragmentation Strategy for Enhanced Network Performance in Dense LPWANs," *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1833–1838, 9 2018.

[27] Z. Shelby, K. Hartke, and C. Bormann, “IETF RFC7252: The Constrained Application Protocol (CoAP),” 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7252.html

[28] A. Minaburo, L. Toutain, and R. Andreasen, “IETF RFC8824: Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP),” 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8824.html

[29] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zúñiga, “IPv6 over LPWANs: Connecting Low Power Wide Area Networks to the Internet (of Things),” *IEEE Wireless Communications*, vol. 27, 2020.

[30] S. Aguilar, P. Maille, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, “Performance Analysis and Optimal Tuning of IETF LPWAN SCHC ACK-on-Error Mode,” *IEEE Sensors Journal*, vol. 20, 2020.

[31] S. Aguilar, A. Marquet, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, “LoRaWAN SCHC Fragmentation Demystified,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11803 LNCS, 2019.

[32] N. Maturana, “Implementation and Evaluation of Static Context Header Compression for IPv6 Packets within a LoRaWAN Network,” 2019. [Online]. Available: https://repositorio.uchile.cl/handle/2250/170134

[33] J. J. Millán, “Implementación y Evaluación de Fragmentación de Paquetes IPv6 para un Enlace LORAWAN,” 2019. [Online]. Available: https://repositorio.uchile.cl/handle/2250/173779

[34] J. Sanchez-Gomez, J. Gallego-Madrid, R. Sanchez-Iborra, J. Santa, and A. Skarmeta, “Impact of SCHC Compression and Fragmentation in LPWAN: A Case Study with LoRaWAN,” *Sensors*, vol. 20, p. 280, 1 2020.

[35] R. Muñoz, J. S. Hidalgo, F. Canales, D. Dujovne, and S. Céspedes, “SCHC over LoRaWAN Efficiency: Evaluation and Experimental Performance of Packet Fragmentation,” *Sensors*, vol. 22, 2022.

[36] S. Aguilar, A. Platis, R. Vidal, and C. Gomez, “Energy Consumption Model of SCHC Packet Fragmentation over Sigfox LPWAN,” *Sensors*, vol. 22, p. 2120, 3 2022.

[37] S. Aguilar, R. Vidal, and C. Gomez, “Evaluation of Receiver-Feedback Techniques for Fragmentation Over LPWANs,” *IEEE Internet of Things Journal*, vol. 9, pp. 6866–6878, 5 2022.

[38] OpenSCHC, “OpenSCHC.” [Online]. Available: https://github.com/openschc/openschc

[39] R. Muñoz, “Modelado y Evaluación de la Eficiencia del Estándar SCHC para el Transporte de Paquetes IP sobre LoRaWAN,” 2020. [Online]. Available: https://repositorio.uchile.cl/handle/2250/177977

[40] SCHC-over-Sigfox, “SCHC-over-Sigfox.” [Online]. Available: https://github.com/schc-over-sigfox/schc-over-sigfox

[41] MicroPython, “MicroPython - Python for microcontrollers.” [Online]. Available: https://micropython.org/

[42] Pycom, “LoPy4 - Pycom - Quadruple Bearer MicroPython enabled Dev Board.” [Online]. Available: https://pycom.io/product/lopy4/

[43] Google Cloud Platform, “Quickstart: Create and deploy an HTTP function by using Python | Cloud Functions Documentation | Google Cloud,” 12 2022. [Online]. Available: https://cloud.google.com/functions/docs/create-deploy-http-python

Annex A

Publications derived from this thesis

- **Software:**
  - [40] "SCHC-over-Sigfox": Implementation and simulation of SCHC-over-Sigfox, deployable over Pycom LoPy4 devices (sender) and Google Cloud Platform (receiver).
- **Conference papers:**
  - [11] D. Wistuba, S. Céspedes, J. C. Zúñiga, R. Muñoz, S. Aguilar, C. Gomez, and R. Vidal, "An Implementation of IoT LPWAN SCHC Message Fragmentation and Reassembly," *CEUR Workshop Proceedings*, vol. 2988, 2021.
  - [22] D. Wistuba, S. Céspedes, and J. Bustos-Jiménez, "Modeling SCHC ACK-on-Error Fragment Delivery over Sigfox," *Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 115–119, 10–2022.
- **Journal papers:**
  - [12] S. Aguilar, D. Wistuba, A. Platis, R. Vidal, C. Gomez, S. Cespedes, and J. C. Zúñiga, "Packet Fragmentation Over Sigfox: Implementation and Performance Evaluation of SCHC ACK-on-Error," *IEEE Internet of Things Journal*, vol. 9, 2021.
- **Standards:**
  - [3] J. C. Zúñiga, C. Gomez, S. Aguilar, L. Toutain, S. Céspedes, D. Wistuba, and J. Boite, "SCHC over Sigfox LPWAN (IETF Internet-Draft)," 2022.
  - [6] J. C. Zúñiga, C. Gomez, S. Aguilar, L. Toutain, S. Céspedes, and D. Wistuba, "SCHC Compound ACK (IETF Internet-Draft)," 2022.

Annex B

Implementation UML diagrams

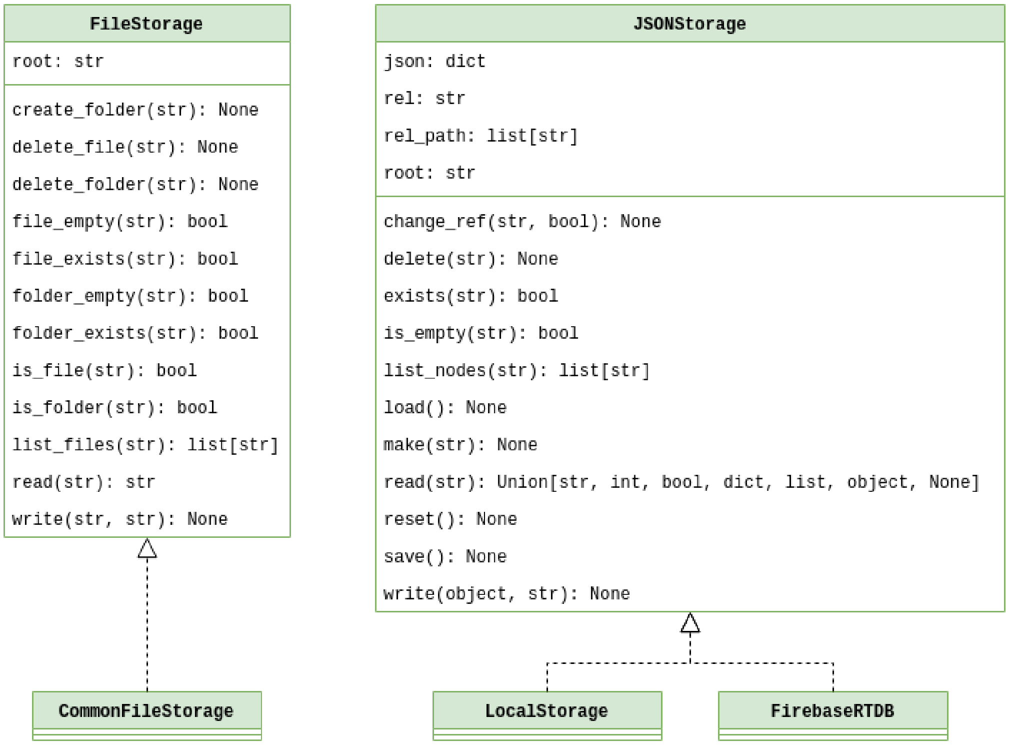


Figure B.1: UML class diagram for the DB module.

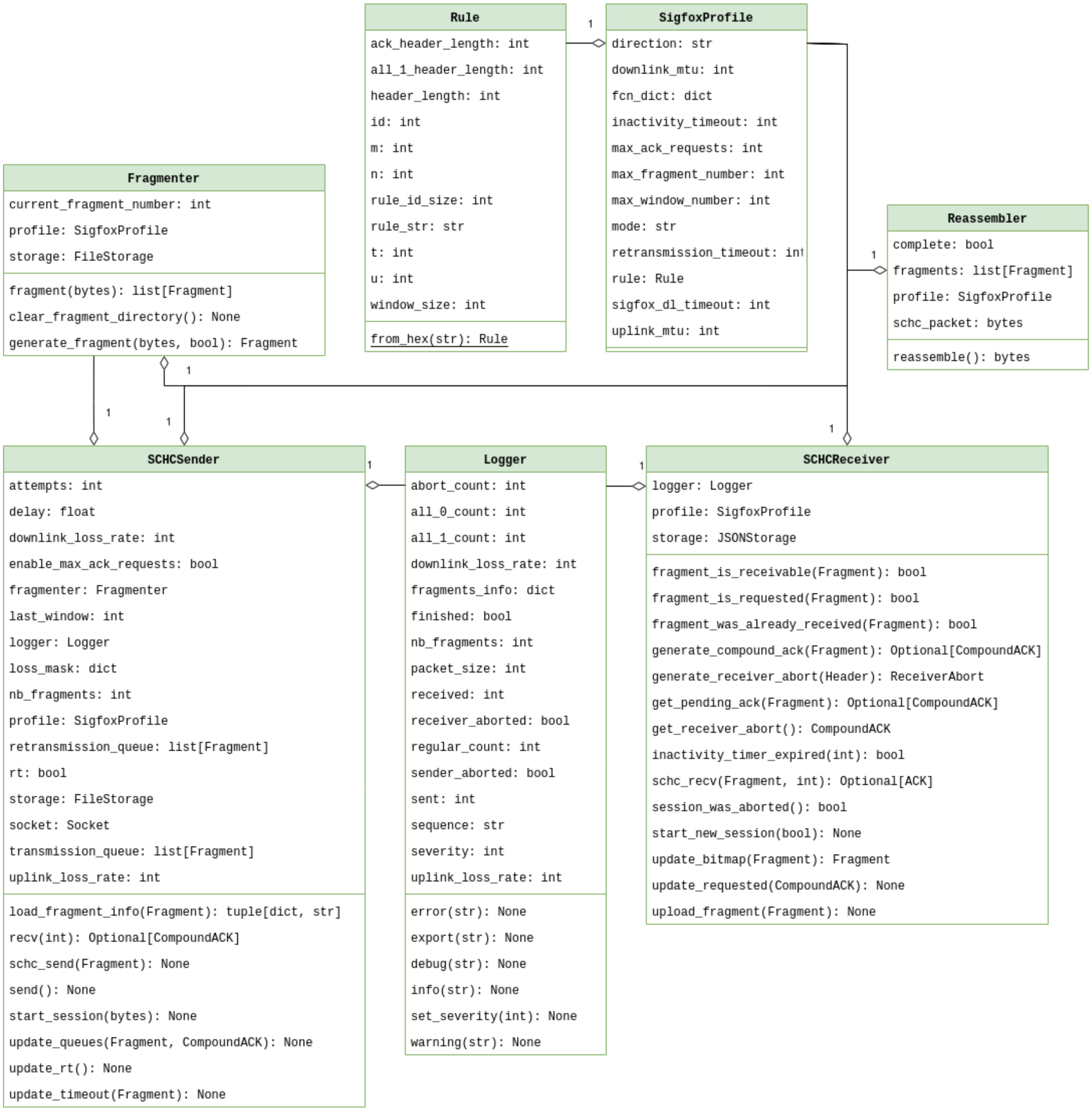


Figure B.2: UML class diagram for the Entities module.

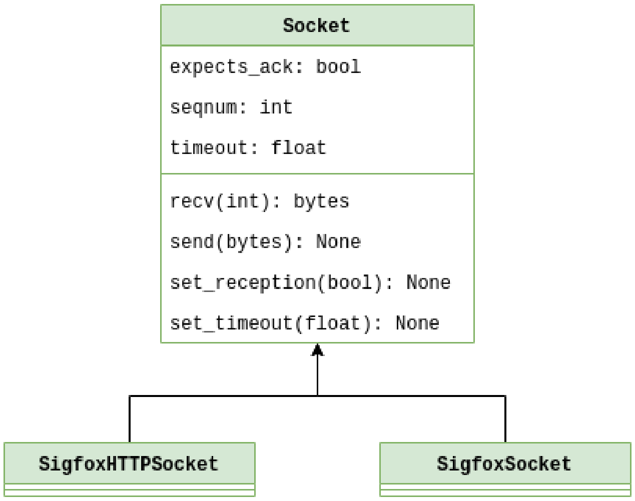


Figure B.3: UML class diagram for the Sockets module.

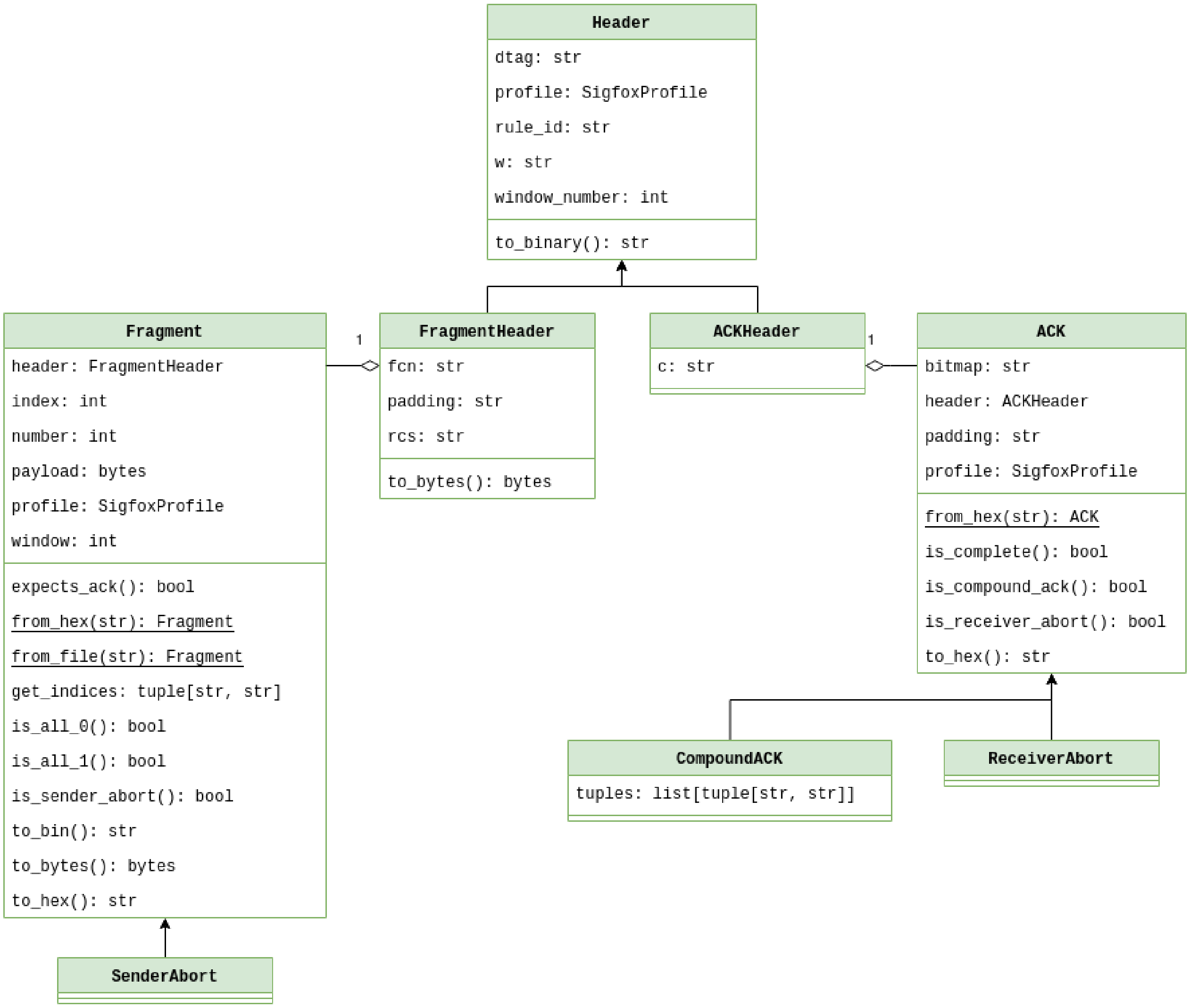
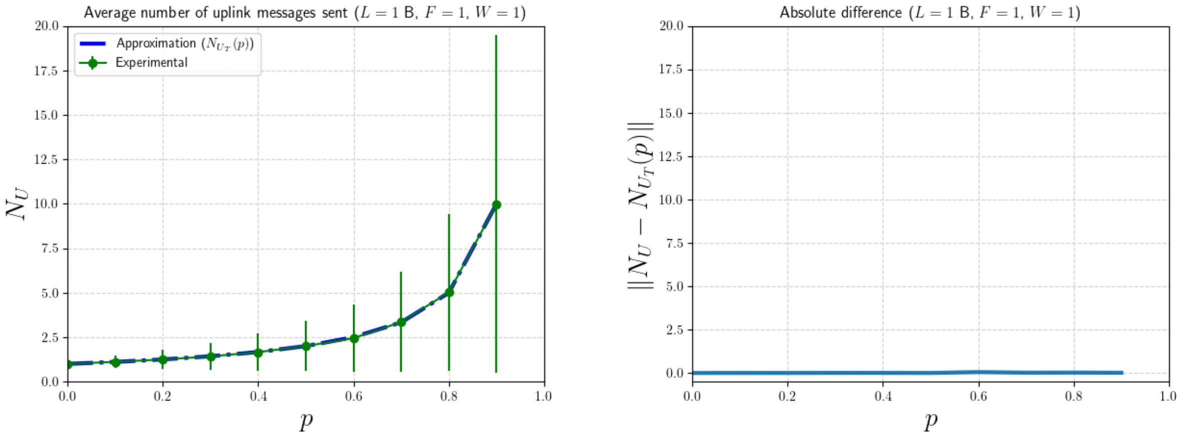


Figure B.4: UML class diagram for the Messages module.

Annex C

Validation of $N_{U_T}(p)$

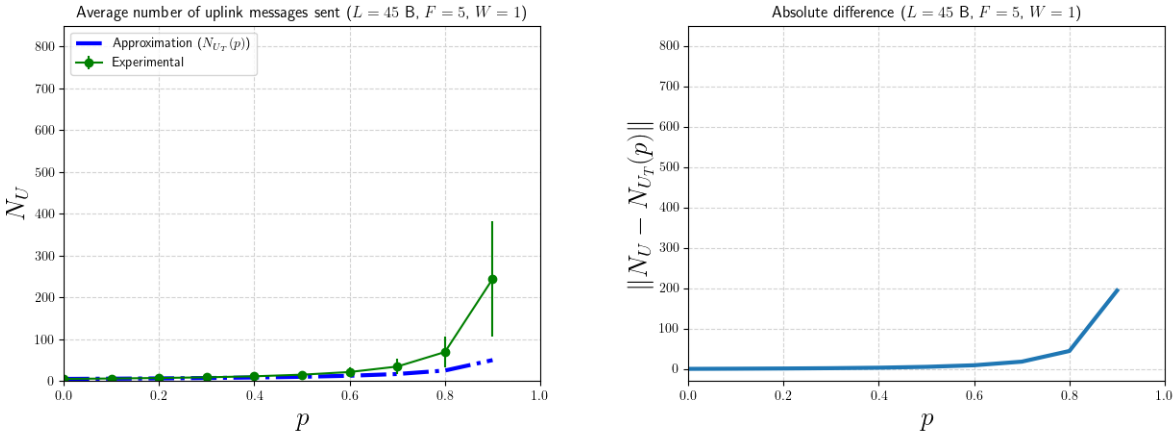
This appendix provides a comparison between $N_{U_T}(p)$ and empirical results for N_U with 10,000 samples per combination of F and p. Vertical error bars of Figures C.1, C.2, C.3, C.4, C.5, C.6, C.7, and C.8 correspond to a single standard deviation $\pm\sigma_N$.



(a) $L = 1$ B

(b) Difference between simulations and the theoretical model.

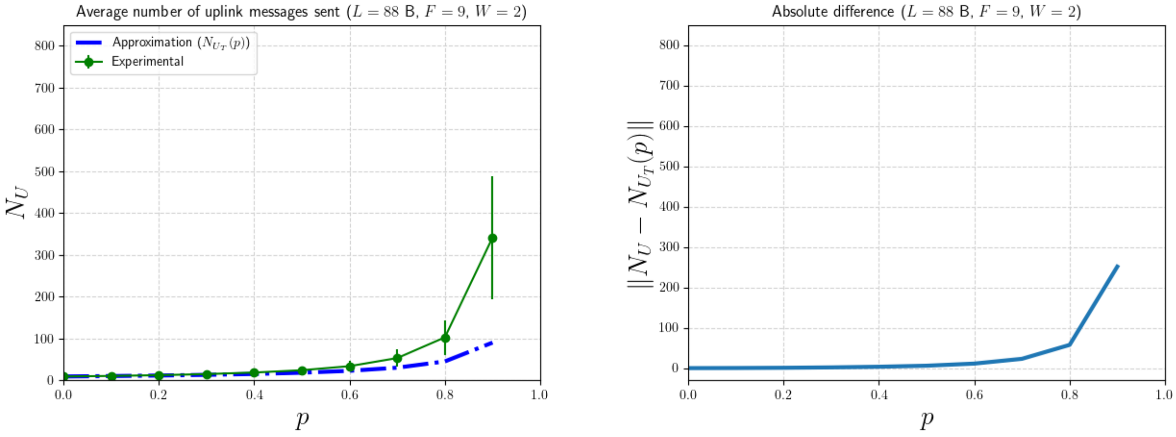
Figure C.1: Simulation results for N_U and $N_{U_T}(p)$. $L = 1$ B, $F = 1$, $W = 1$.



(a) $L = 45$ B

(b) Difference between simulations and the theoretical model.

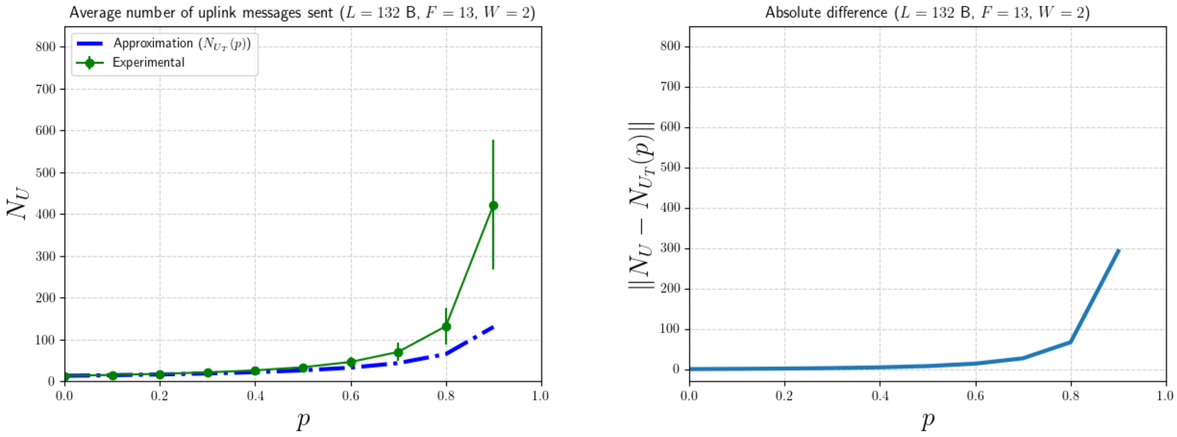
Figure C.2: Simulation results for N_U and $N_{U_T}(p)$. $L = 45$ B, $F = 5$, $W = 1$.



(a) $L = 88$ B

(b) Difference between simulations and the theoretical model.

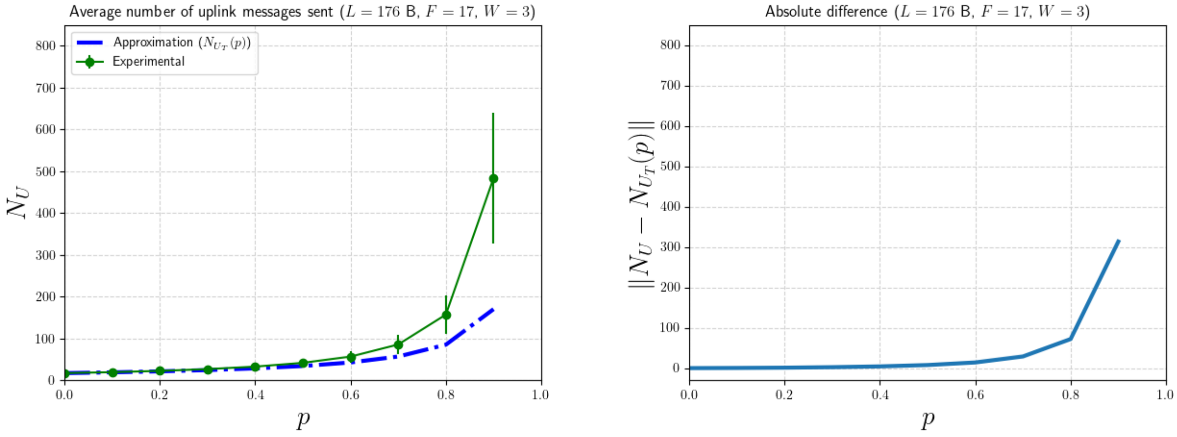
Figure C.3: Simulation results for N_U and $N_{U_T}(p)$. $L = 88$ B, $F = 9$, $W = 2$.



(a) $L = 132$ B

(b) Difference between simulations and the theoretical model.

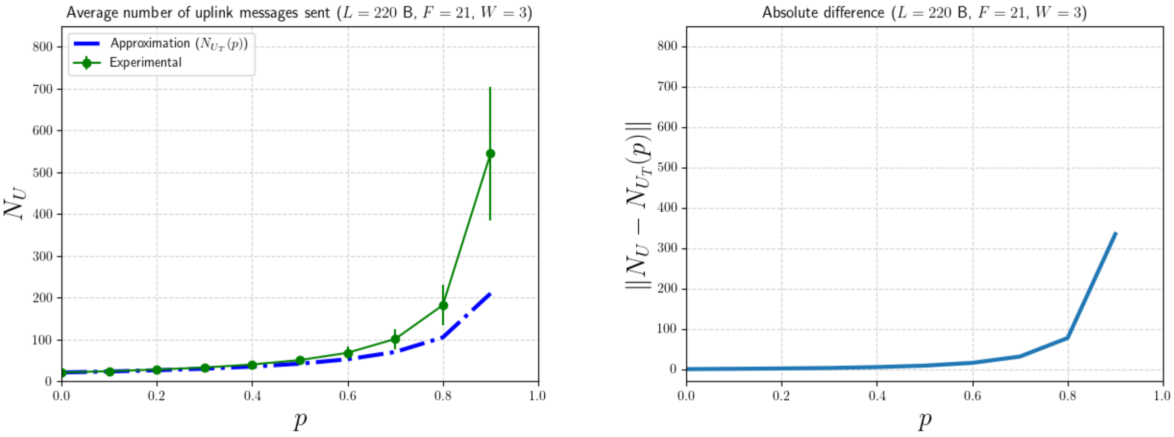
Figure C.4: Simulation results for N_U and $N_{U_T}(p)$. $L = 132$ B, $F = 13$, $W = 2$.



(a) $L = 176$ B

(b) Difference between simulations and the theoretical model.

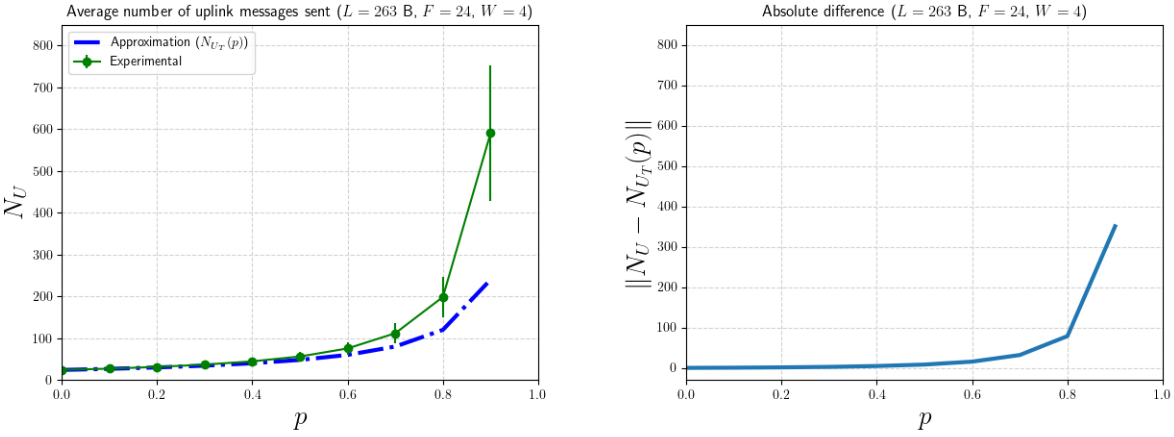
Figure C.5: Simulation results for N_U and $N_{U_T}(p)$. $L = 176$ B, $F = 17$, $W = 3$.



(a) $L = 220$ B

(b) Difference between simulations and the theoretical model.

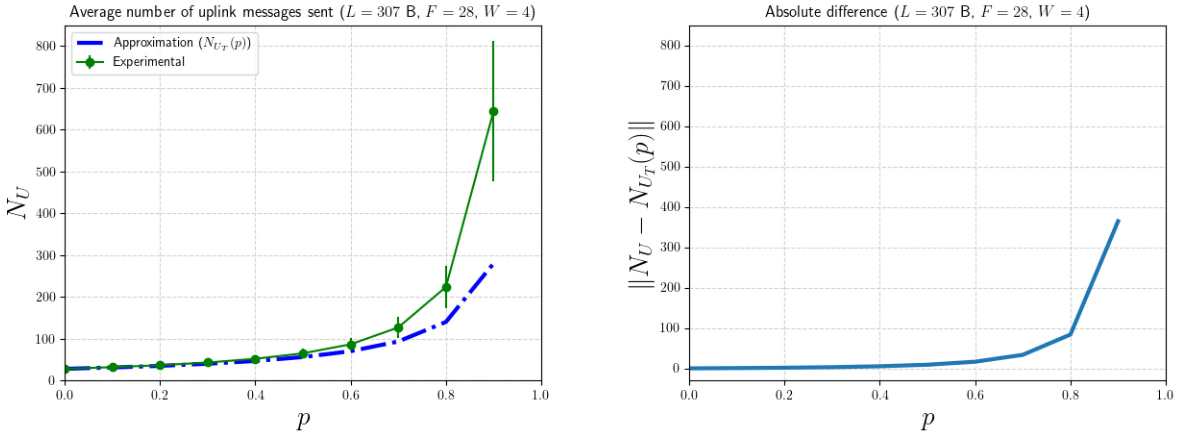
Figure C.6: Simulation results for N_U and $N_{U_T}(p)$. $L = 220$ B, $F = 21$, $W = 3$.



(a) $L = 263$ B

(b) Difference between simulations and the theoretical model.

Figure C.7: Simulation results for N_U and $N_{U_T}(p)$. $L = 263$ B, $F = 24$, $W = 4$.



(a) $L = 307$ B

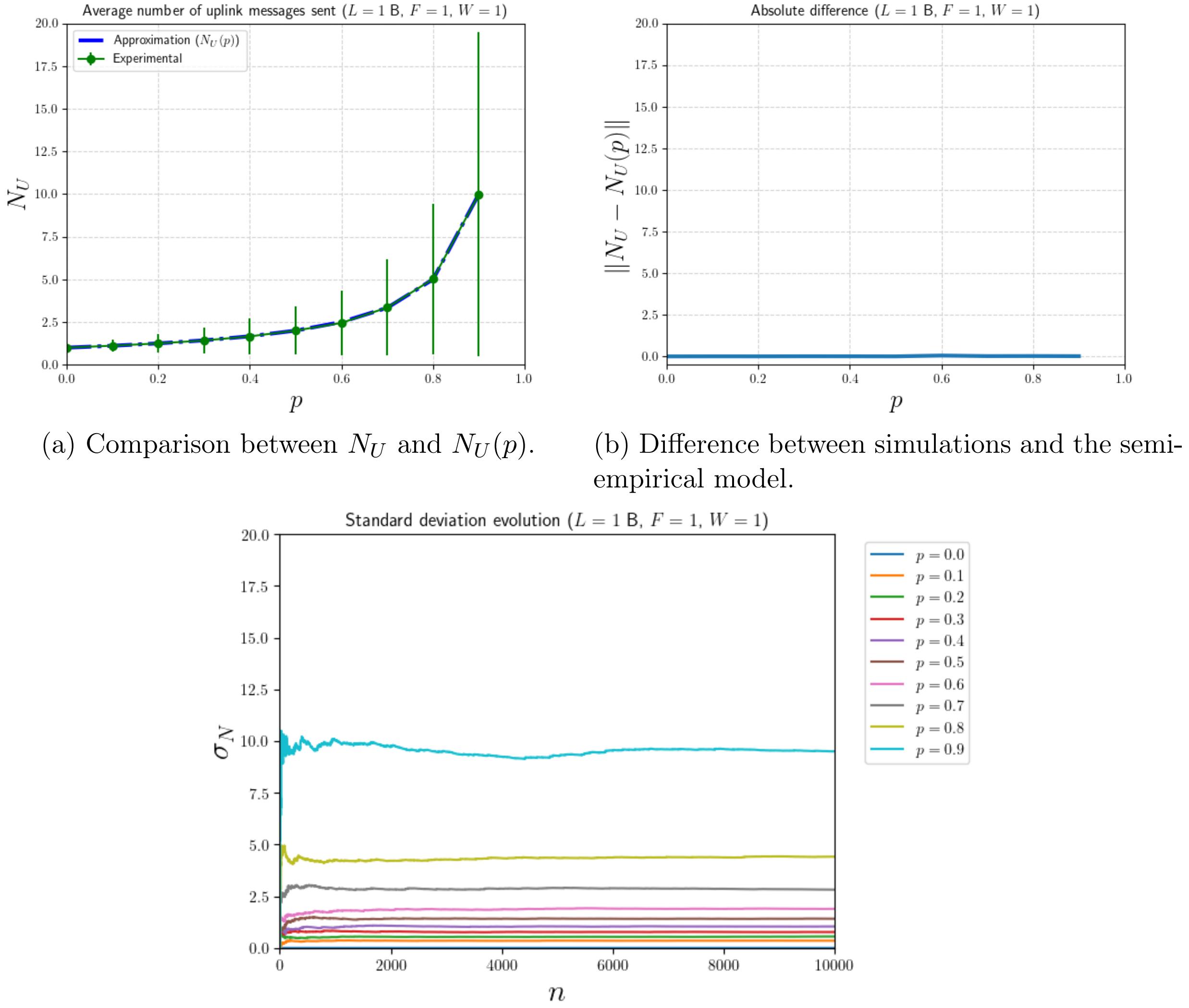
(b) Difference between simulations and the theoretical model.

Figure C.8: Simulation results for N_U and $N_{U_T}(p)$. $L = 307$ B, $F = 28$, $W = 4$.

Annex D

Validation of $N_U(p)$ over the non-aborting scenario

This appendix provides a comparison between $N_U(p)$ and empirical results for N_U by bypassing the `MAX_ACK_REQUESTS` limit (the non-aborting scenario) with 10,000 samples per combination of F and p. Vertical error bars of Figures D.1, D.2, D.3, D.4, D.5, D.6, D.7, and D.8 correspond to a single standard deviation $\pm\sigma_N$.

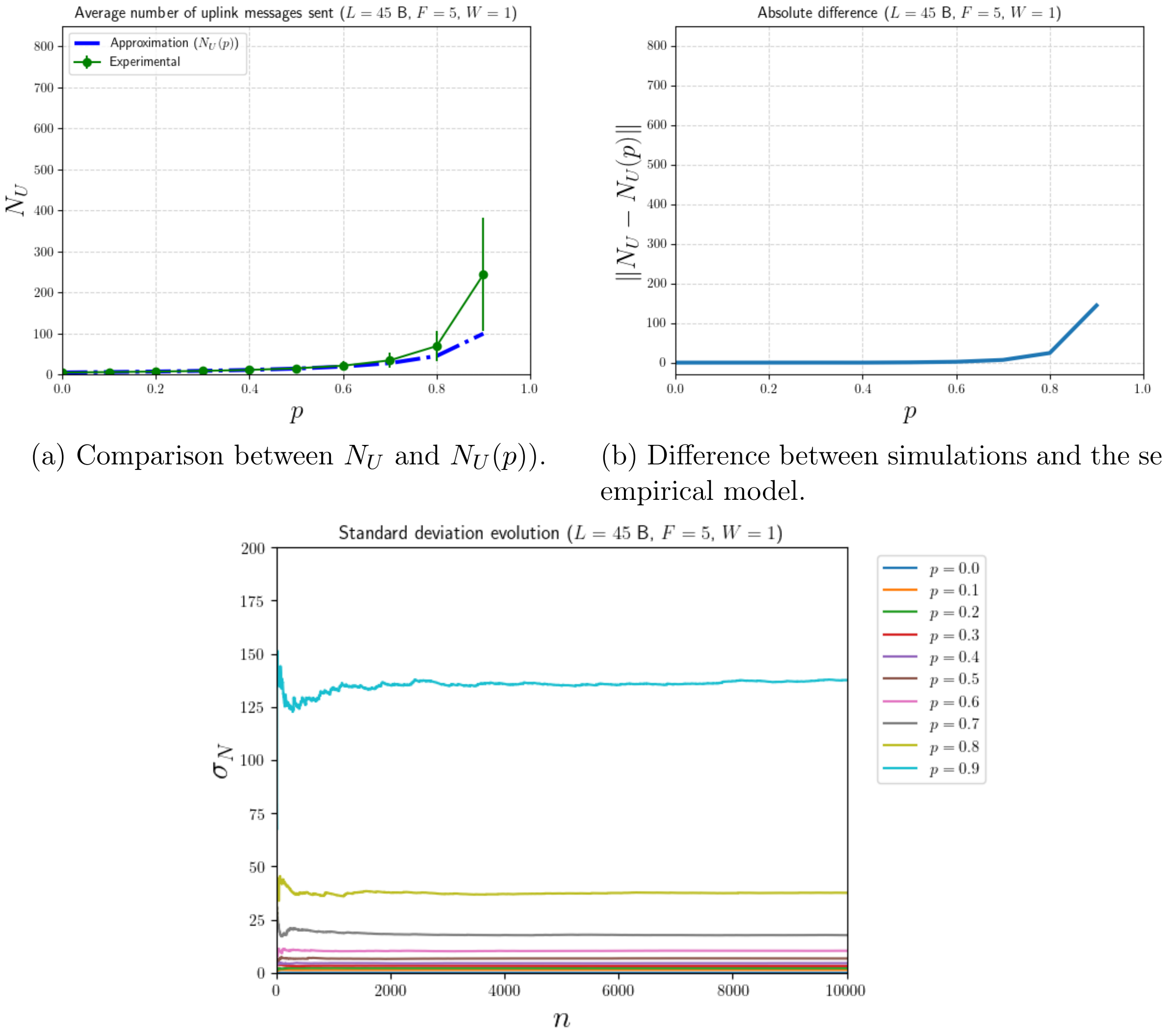


(a) Comparison between N_U and $N_U(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.1: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 1$ B, $F = 1$, $W = 1$.

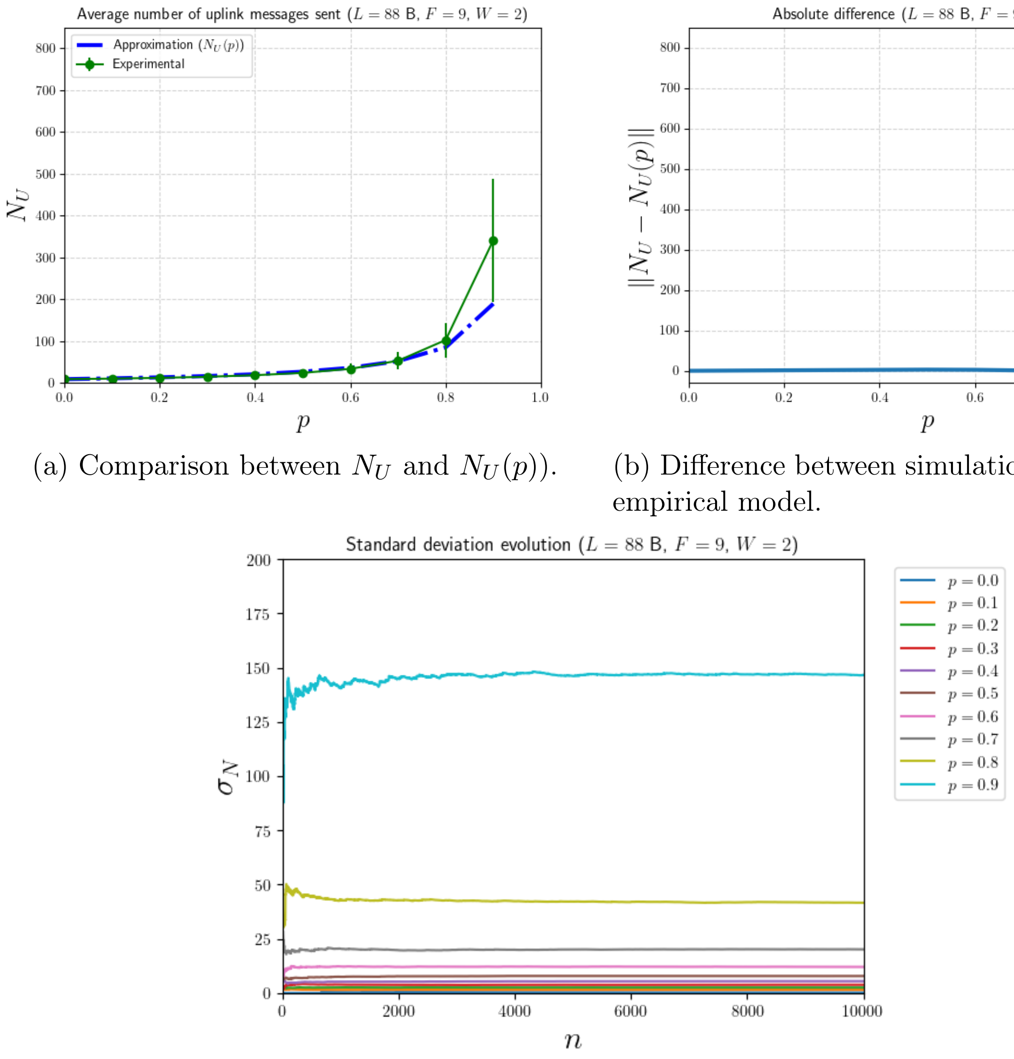


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.2: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 45$ B, $F = 5$, $W = 1$.

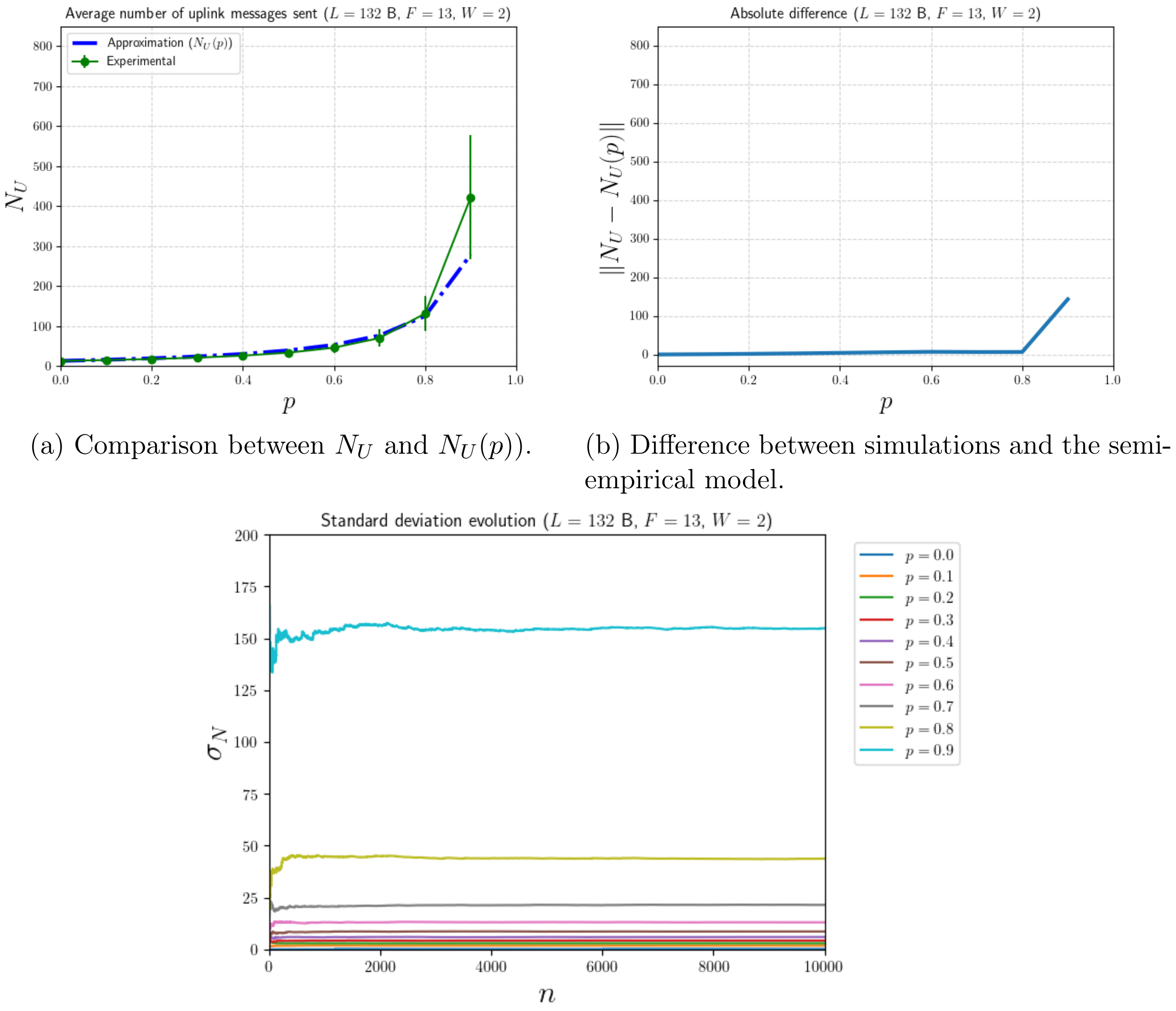


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.3: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 88$ B, $F = 9$, $W = 2$.

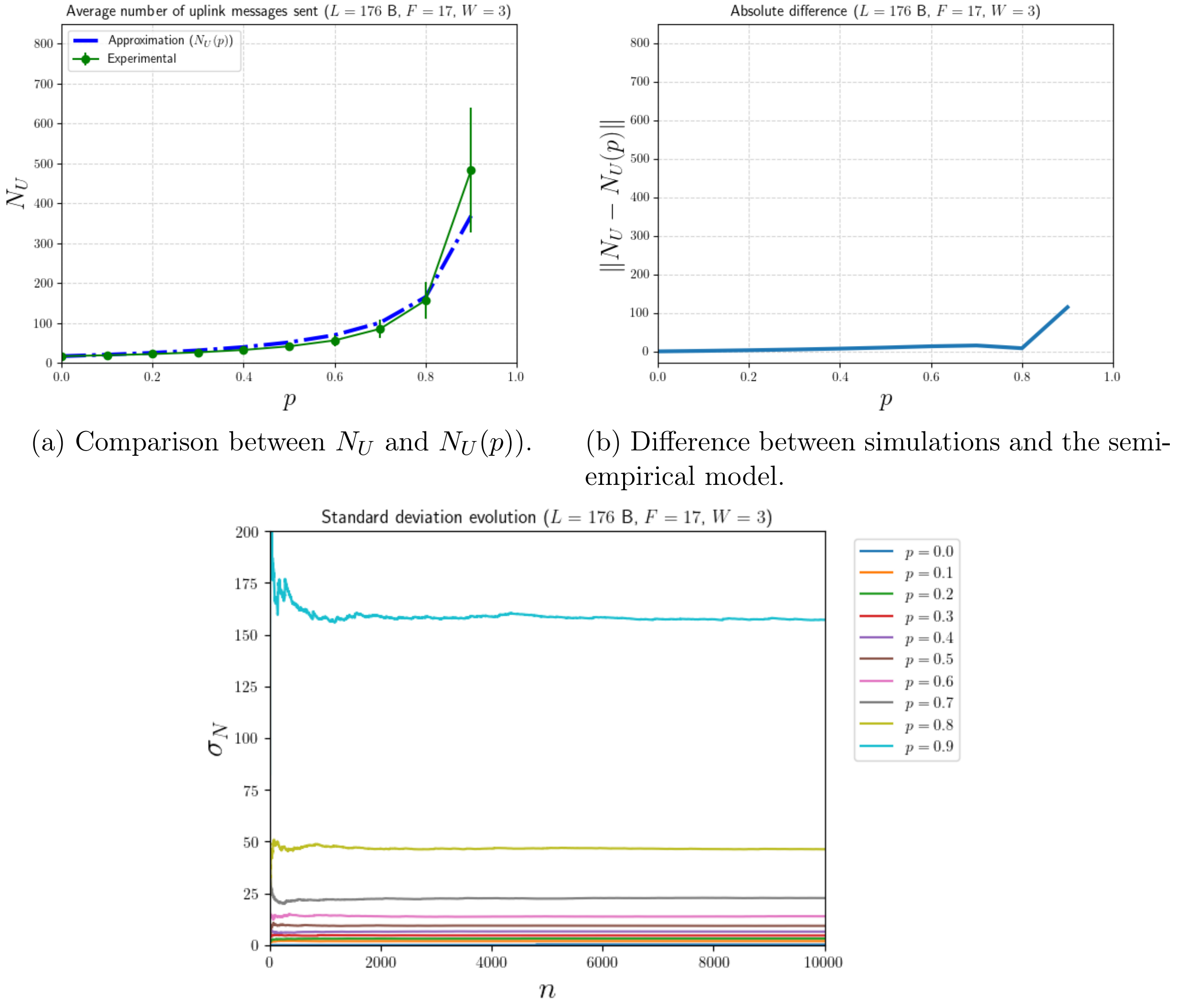


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.4: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 132$ B, $F = 13$, $W = 2$.

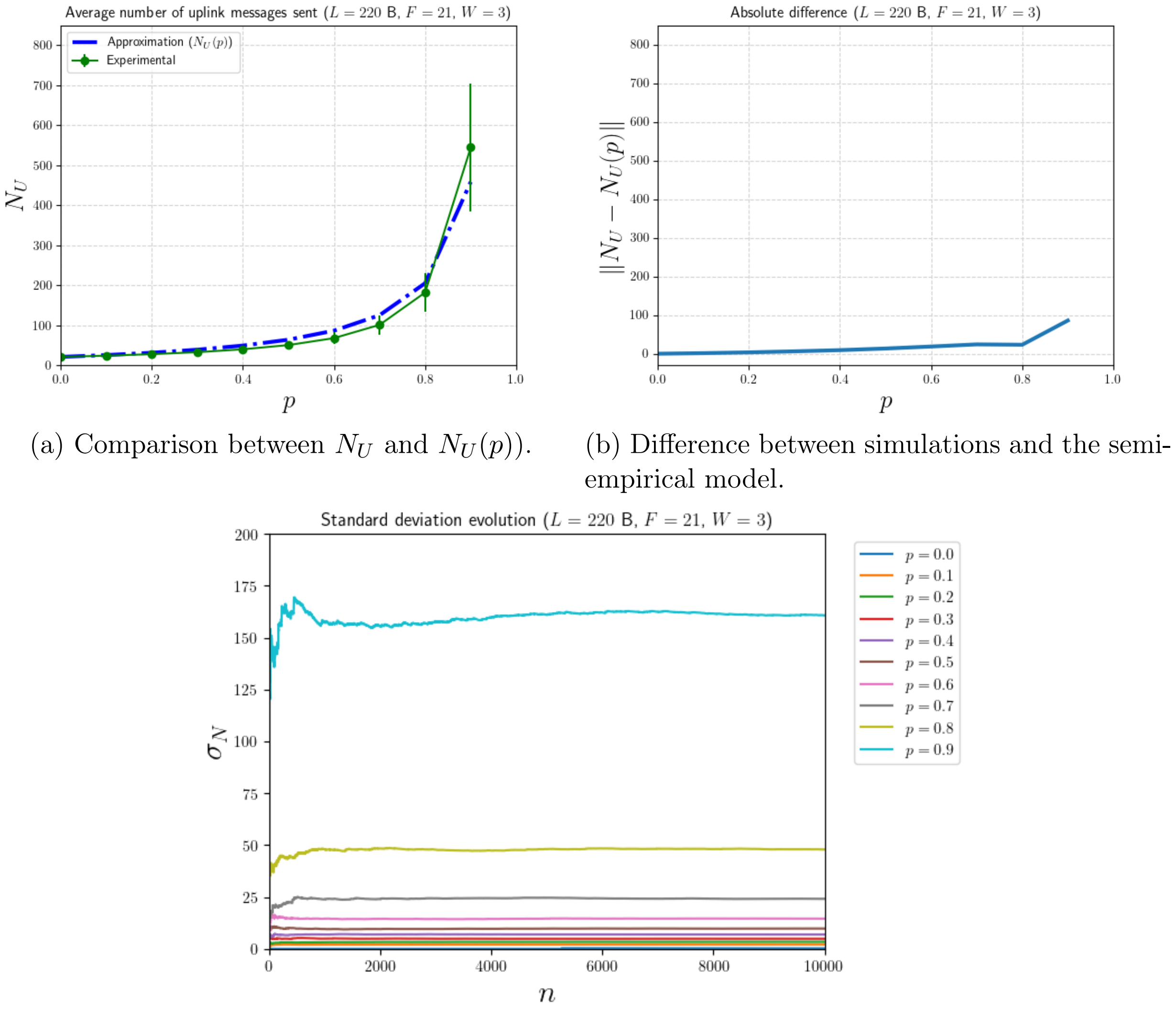


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.5: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 176$ B, $F = 17$, $W = 3$.

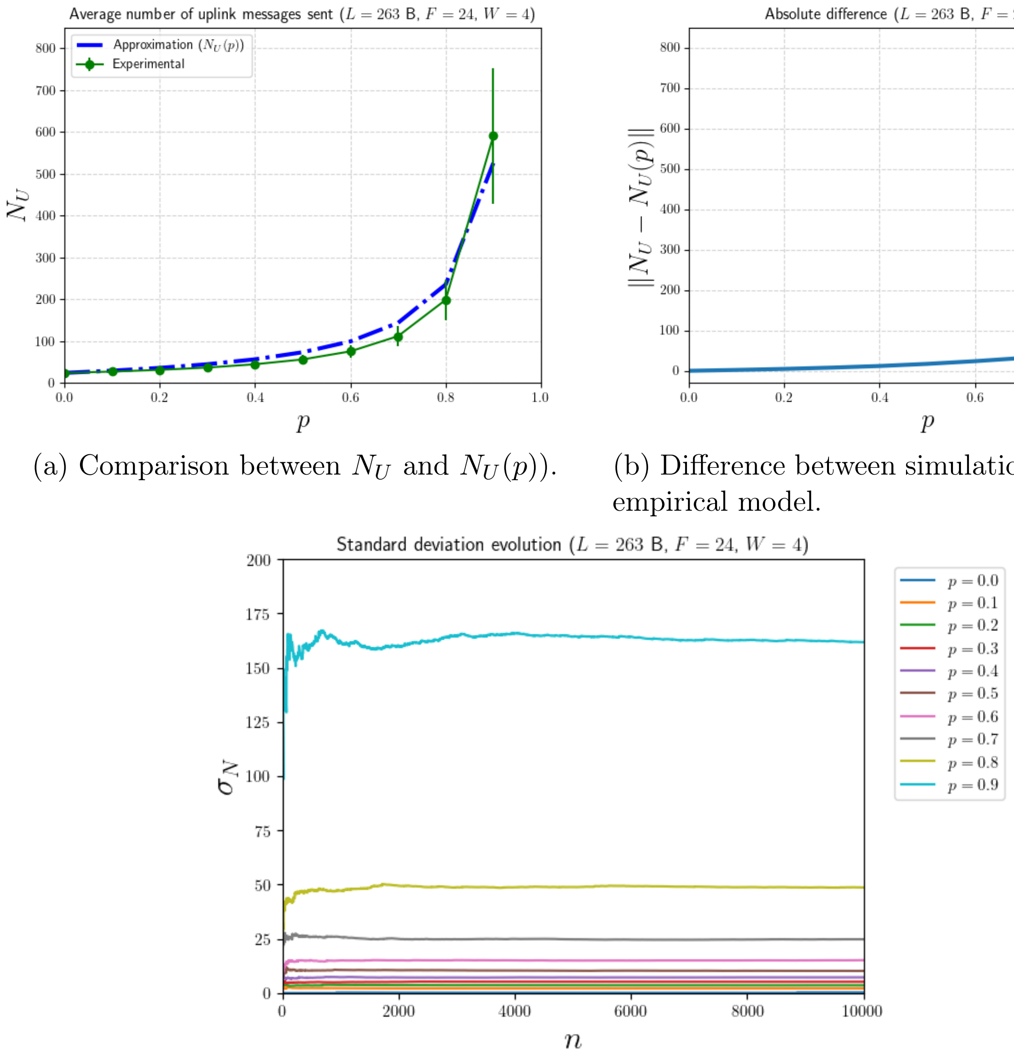


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.6: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 220$ B, $F = 21$, $W = 3$.

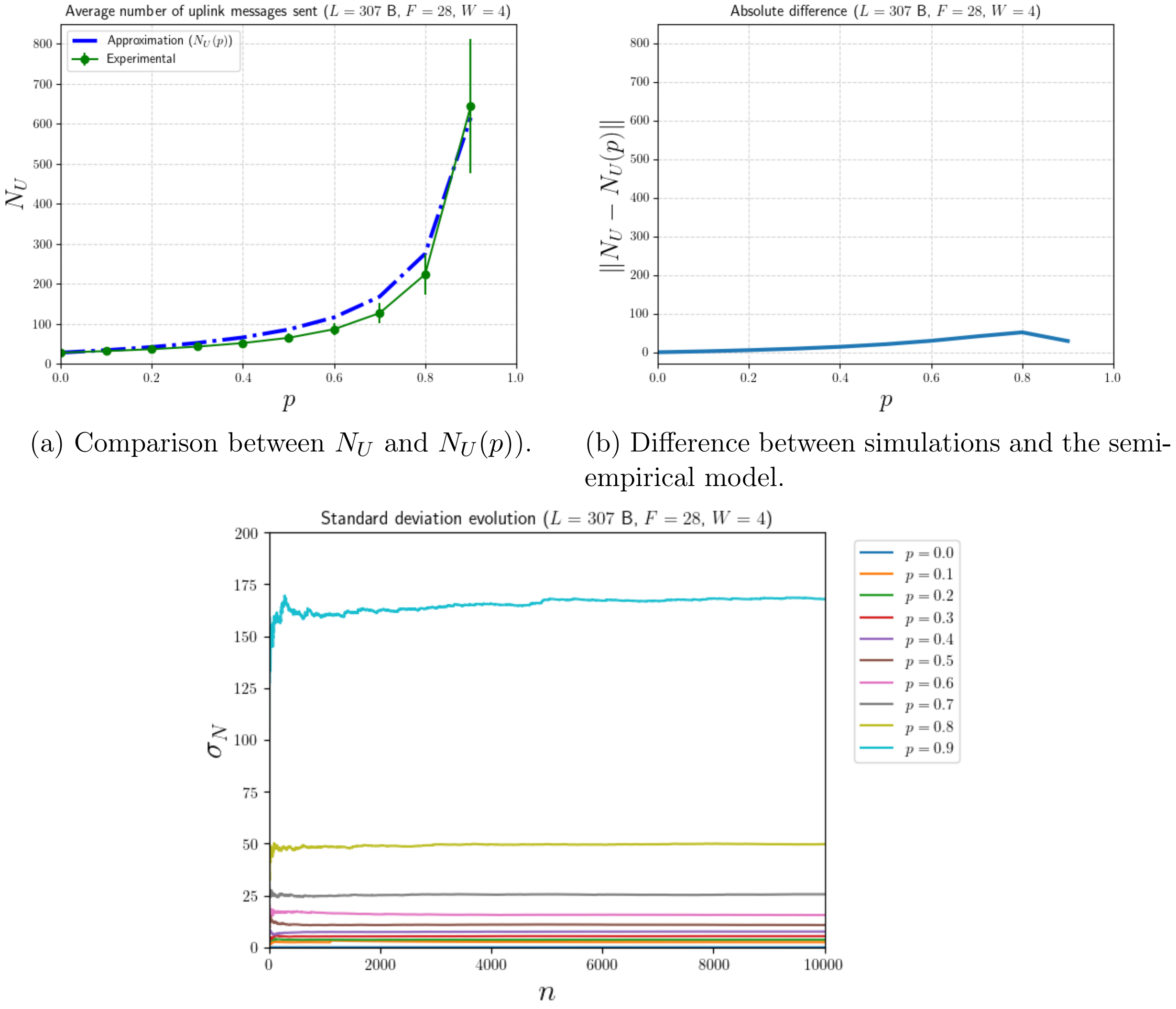


(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.7: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 263$ B, $F = 24$, $W = 4$.



(a) Comparison between N_U and $N_U(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure D.8: Simulation results for N_U and $N_U(p)$ in a non-aborting scenario. $L = 307$ B, $F = 28$, $W = 4$.

Annex E

Validation of $N_U(p)$ over the base scenario

This appendix provides a comparison between $N_U(p)$ and empirical results for N_U by considering the `MAX_ACK_REQUESTS` limit (the base scenario), with 10,000 samples per combination of F and p. Vertical error bars of Figures E.1, E.2, E.3, E.4, E.5, E.6, E.7, and E.8 correspond to a single standard deviation $\pm\sigma_N$.

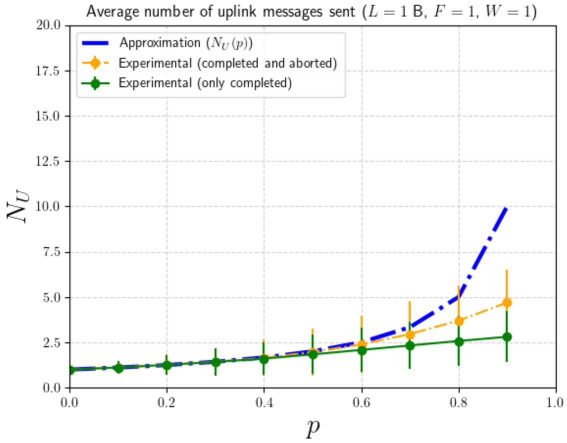


Figure E.1: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 1$ B, $F = 1$, $W = 1$.

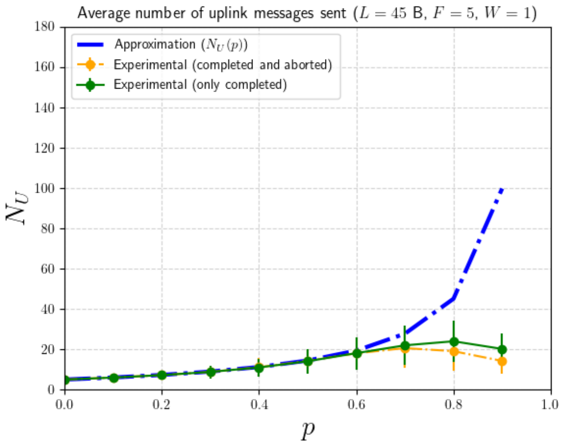


Figure E.2: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 45$ B, $F = 5$, $W = 1$.

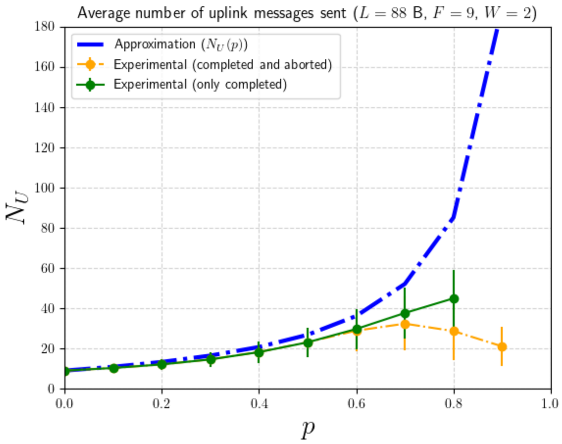


Figure E.3: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 88$ B, $F = 9$, $W = 2$.

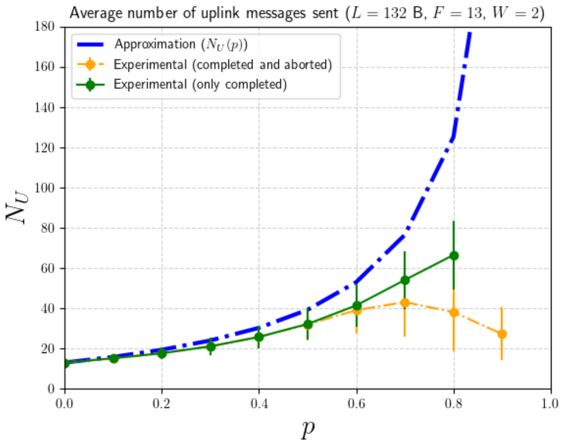


Figure E.4: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 132$ B, $F = 13$, $W = 2$.

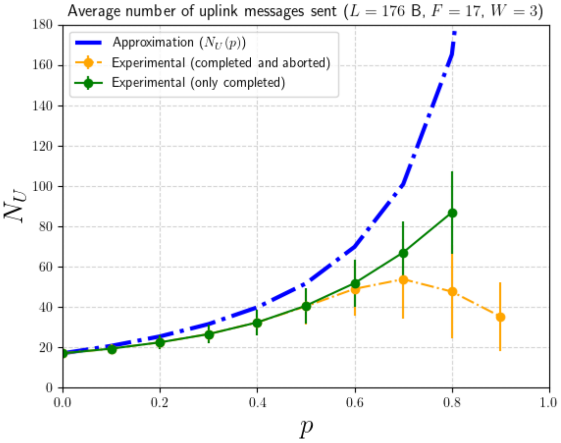


Figure E.5: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 176$ B, $F = 17$, $W = 3$.

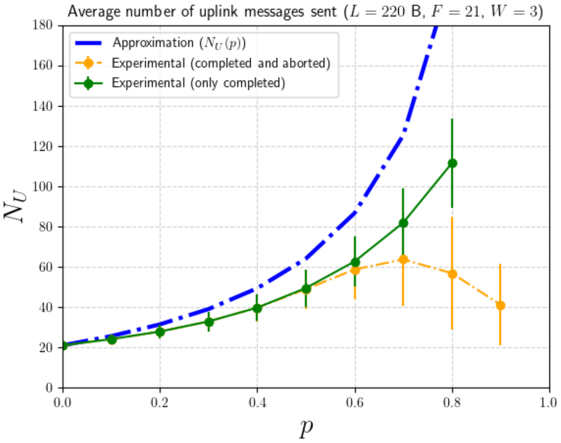


Figure E.6: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 220$ B, $F = 21$, $W = 3$.

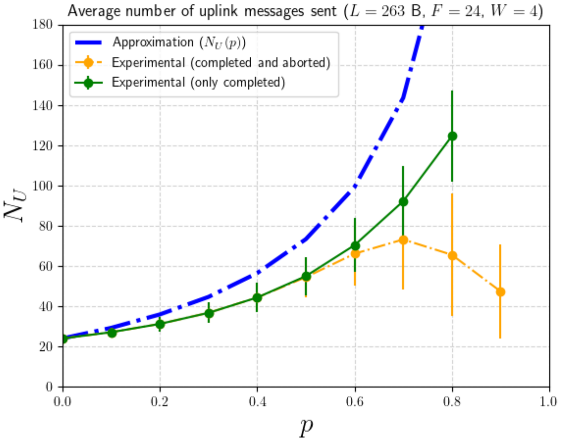


Figure E.7: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 263$ B, $F = 24$, $W = 4$.

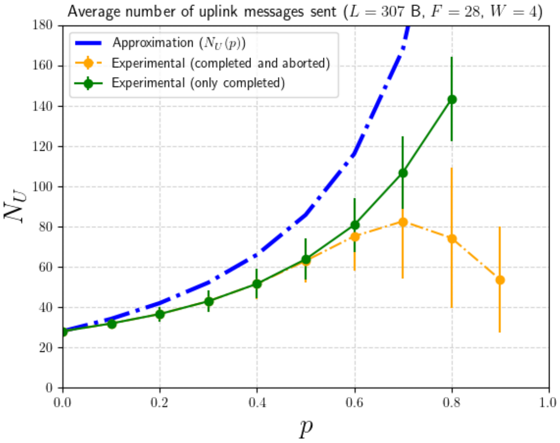
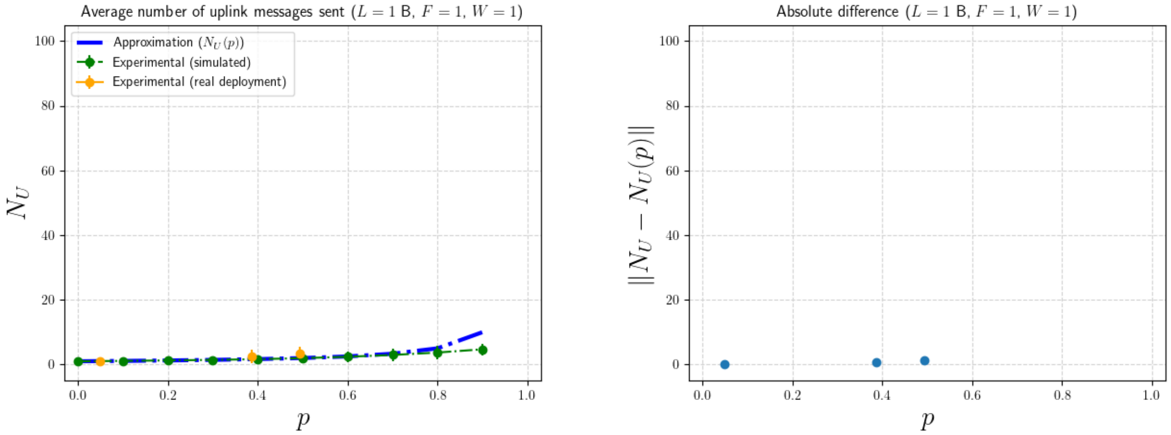


Figure E.8: Simulation environment results over the base scenario for N_U and $N_U(p)$. $L = 307$ B, $F = 28$, $W = 4$.

Annex F

Real deployment results of N_U

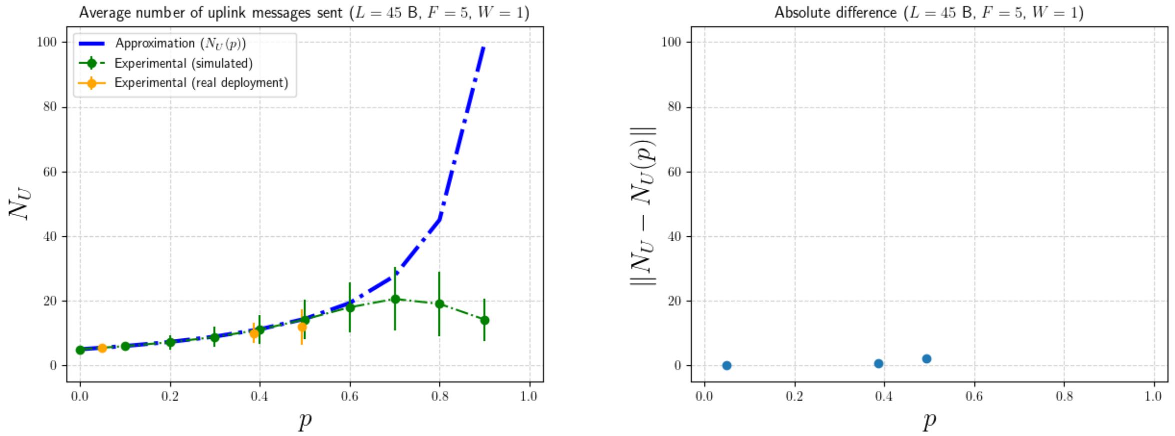
This appendix provides a comparison between $N_U(p)$ and the results for N_U obtained in the real deployment scenario by performing 100 SCHC/Sigfox transmissions per value of F in three separate runs. Vertical error bars of Figures F.1, F.2, F.3, F.4, F.5, F.6, F.7, and F.8 correspond to $\pm\sigma_N$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

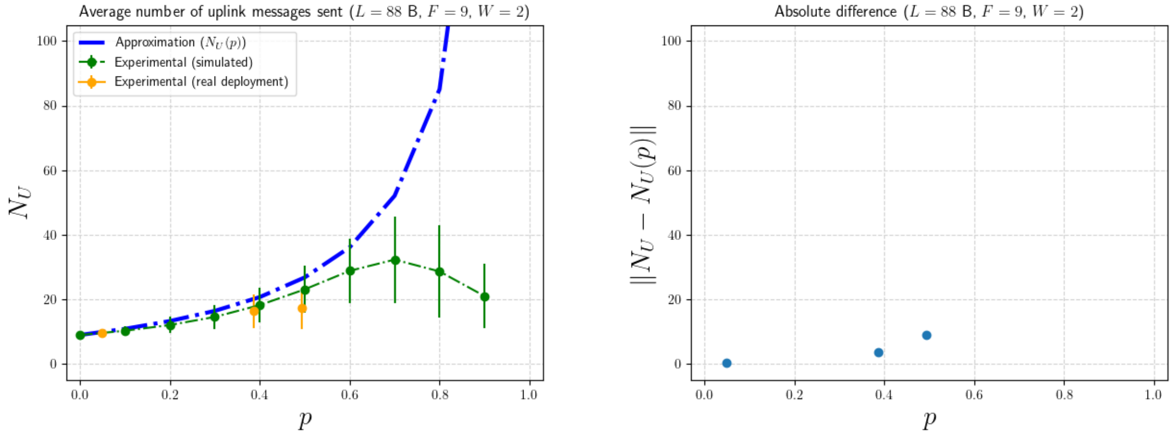
Figure F.1: Real deployment results for N_U and $N_U(p)$. $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

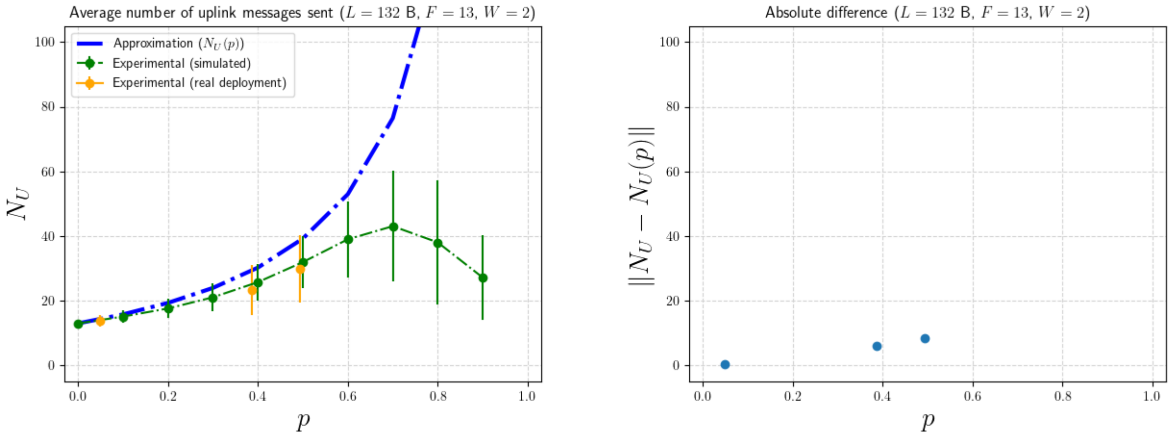
Figure F.2: Real deployment results for N_U and $N_U(p)$. $L = 45$ B, $F = 5$, $W = 1$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

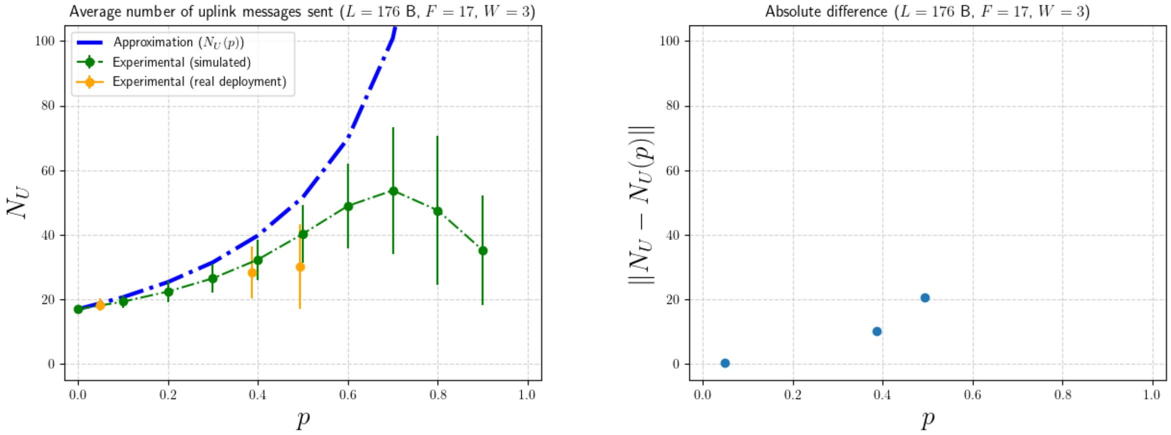
Figure F.3: Real deployment results for N_U and $N_U(p)$. $L = 88$ B, $F = 9$, $W = 2$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

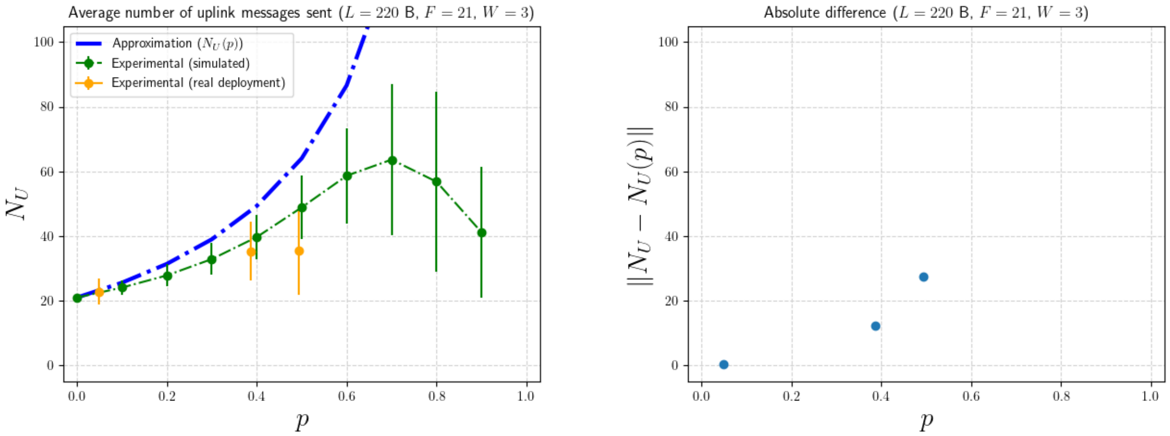
Figure F.4: Real deployment results for N_U and $N_U(p)$. $L = 132$ B, $F = 13$, $W = 2$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

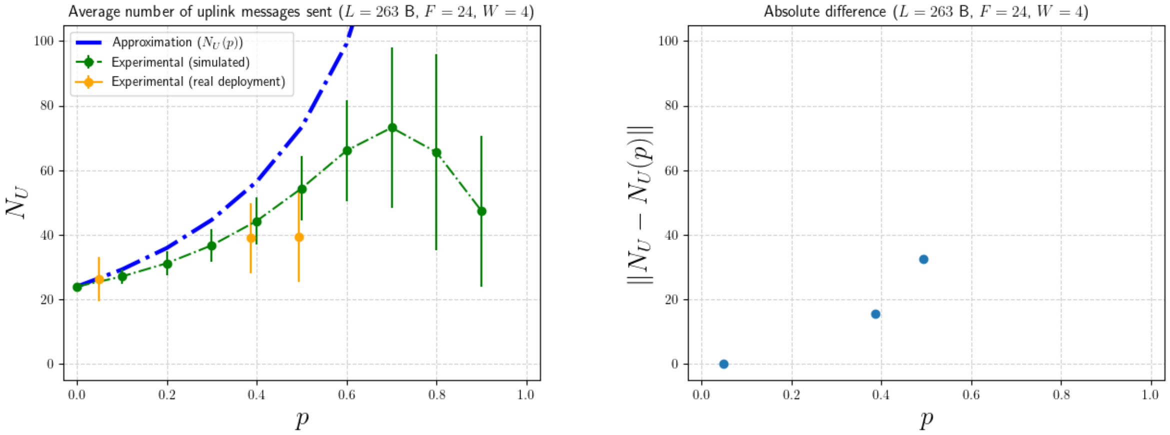
Figure F.5: Real deployment results for N_U and $N_U(p)$. $L = 176$ B, $F = 17$, $W = 3$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

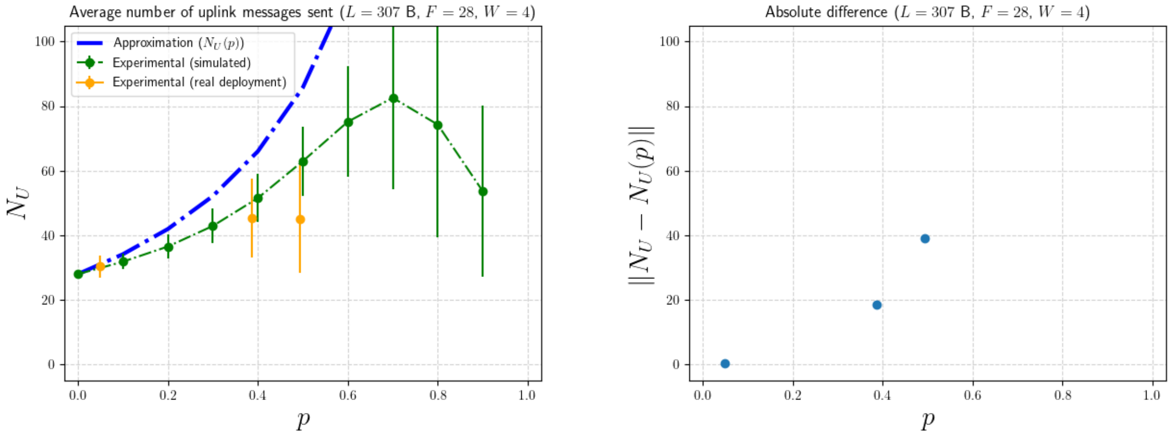
Figure F.6: Real deployment results for N_U and $N_U(p)$. $L = 220$ B, $F = 21$, $W = 3$.



(a) Comparison between $N_U(p)$ and N_U.

(b) Difference between real deployment and semi-empirical model.

Figure F.7: Real deployment results for N_U and $N_U(p)$. $L = 263$ B, $F = 24$, $W = 4$.



(a) Comparison between $N_U(p)$ and N_U.

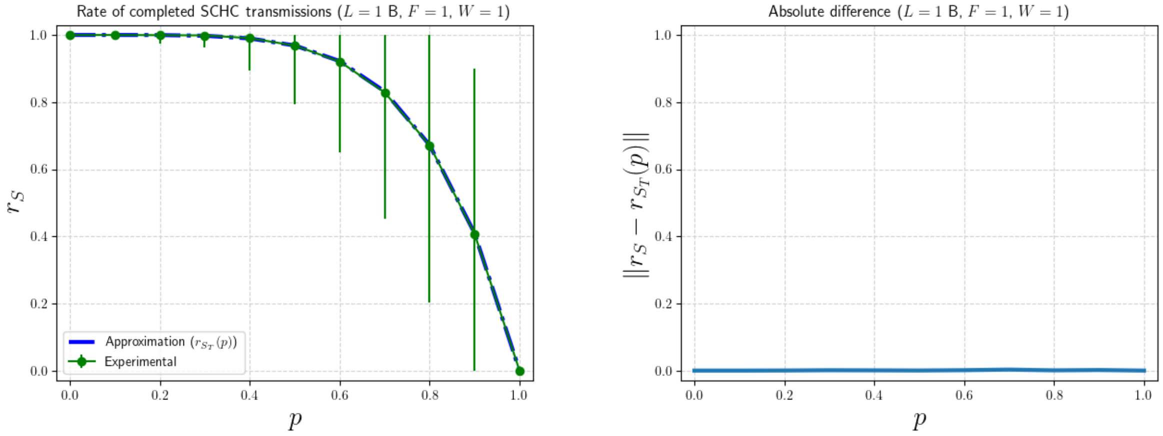
(b) Difference between real deployment and semi-empirical model.

Figure F.8: Real deployment results for N_U and $N_U(p)$. $L = 307$ B, $F = 28$, $W = 4$.

Annex G

Validation of $r_{S_T}(p)$

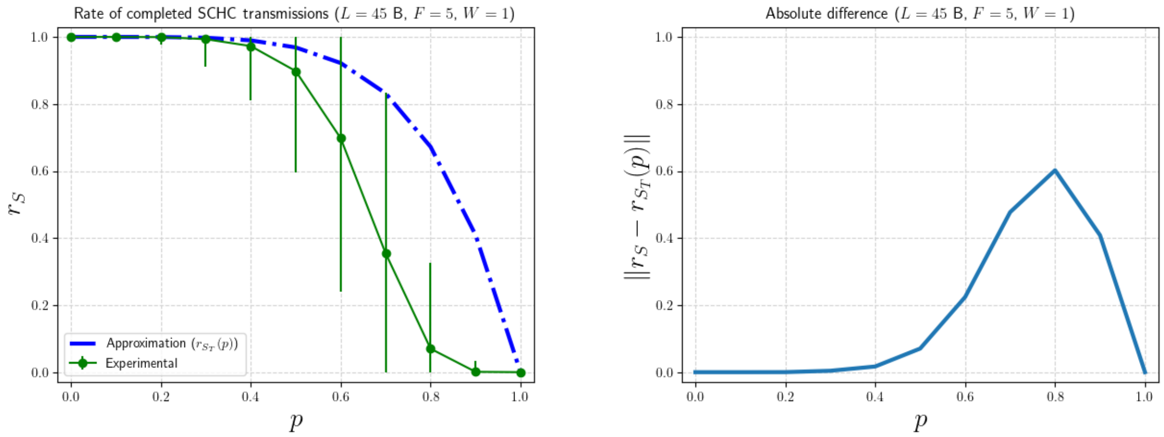
This appendix provides a comparison between $r_{S_T}(p)$ and empirical results for r_S with 10,000 samples per combination of F and p. Vertical error bars of Figures G.1, G.2, G.3, G.4, G.5, G.6, G.7, and G.8 correspond to a single standard deviation $\pm\sigma_S$, capped not to exceed the $[0, 1]$ interval.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

Figure G.1: Simulation results for r_S and $r_{S_T(p)}$. $L = 1$ B, $F = 1$, $W = 1$.



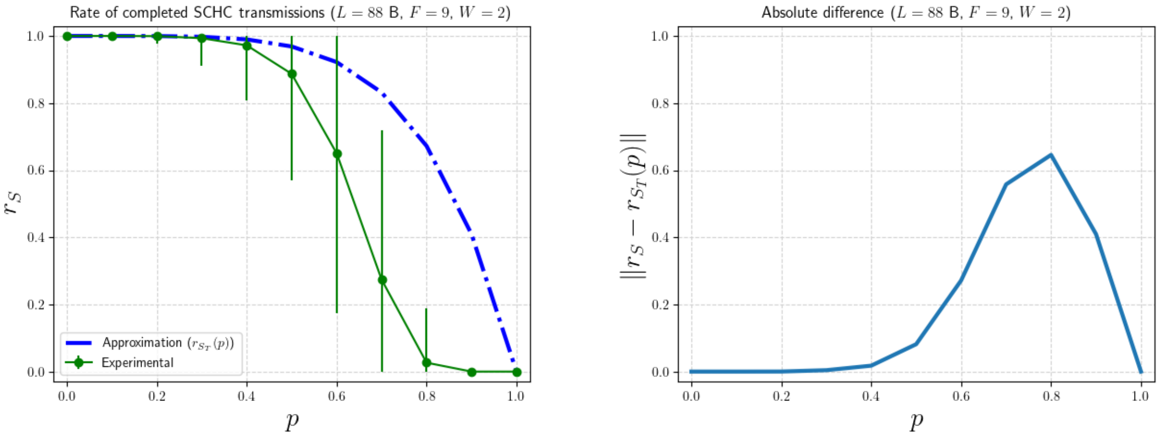
(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

Figure G.2: Simulation results for r_S and $r_{S_T(p)}$. $L = 45$ B, $F = 5$, $W = 1$.

Configuration	α	β	γ	δ
$F = 5$	2.601	0.633	3.171	7.568
$F = 9$	2.687	0.700	3.051	8.670
$F = 13$	2.130	0.580	3.626	5.873
$F = 17$	2.025	0.561	3.836	5.453
$F = 21$	2.616	0.742	2.863	9.343
$F = 24$	2.223	0.633	3.406	6.679
$F = 28$	2.164	0.627	3.407	6.519
Mean	2.350	0.640	3.337	7.158
Std. dev.	0.254	0.059	0.311	1.332
Adjusted value	2.000	0.490	4.800	3.200

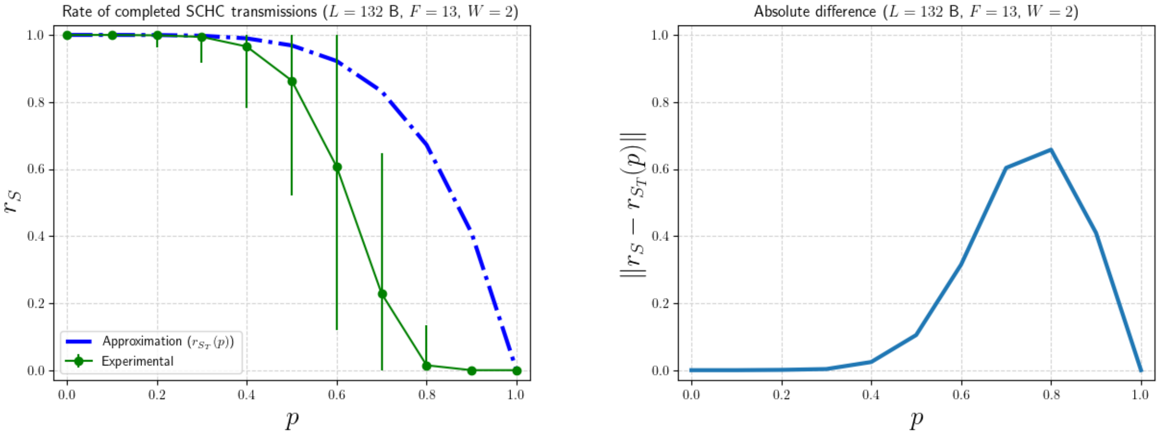
Table G.1: Values for α, β, γ and δ obtained by using LSM. The mean and standard deviation of these values are shown. The final adjustment, done by inspection, is shown in the last row.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

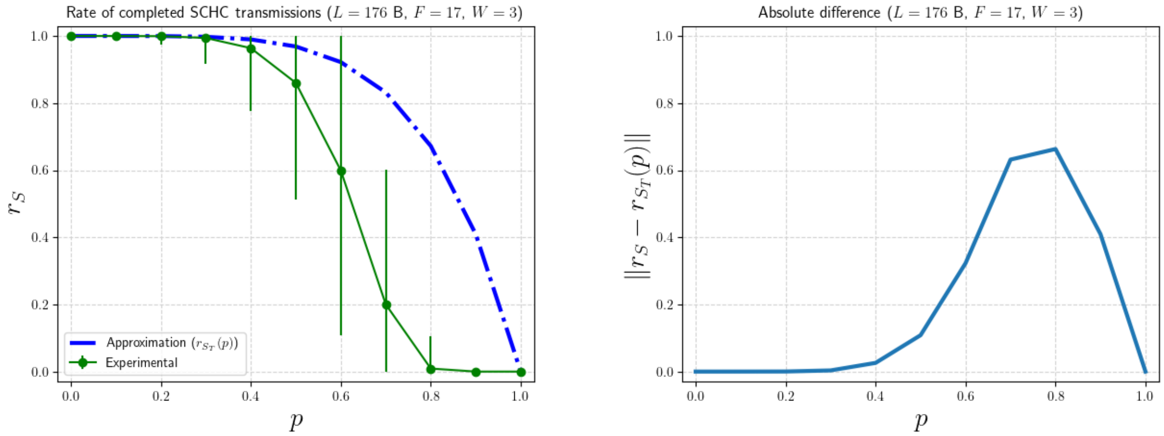
Figure G.3: Simulation results for r_S and $r_{S_T(p)}$. $L = 88$ B, $F = 9$, $W = 2$.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

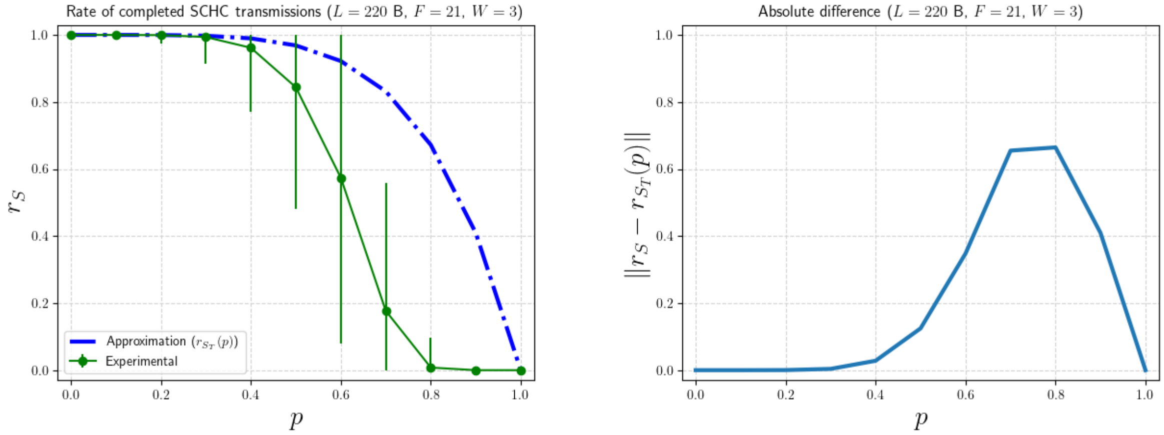
Figure G.4: Simulation results for r_S and $r_{S_T(p)}$. $L = 132$ B, $F = 13$, $W = 2$.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

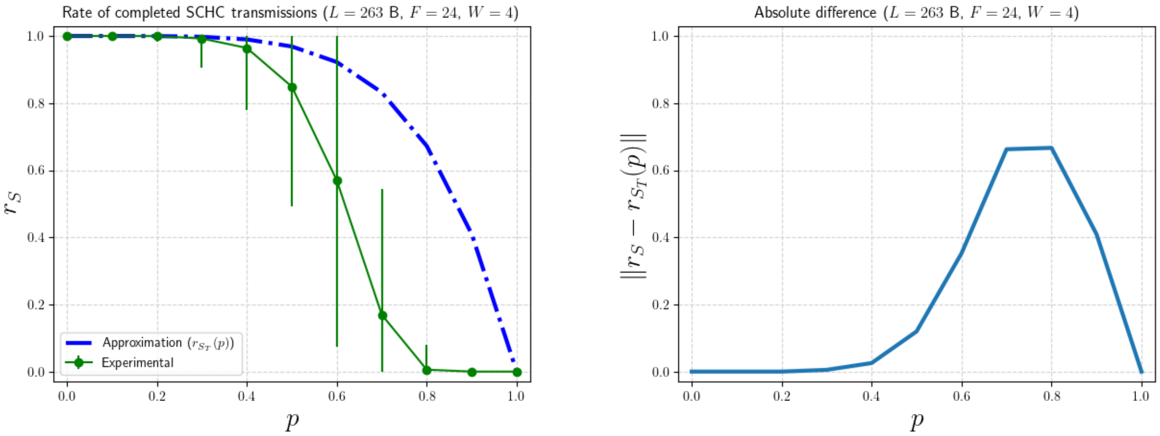
Figure G.5: Simulation results for r_S and $r_{S_T(p)}$. $L = 176$ B, $F = 17$, $W = 3$.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

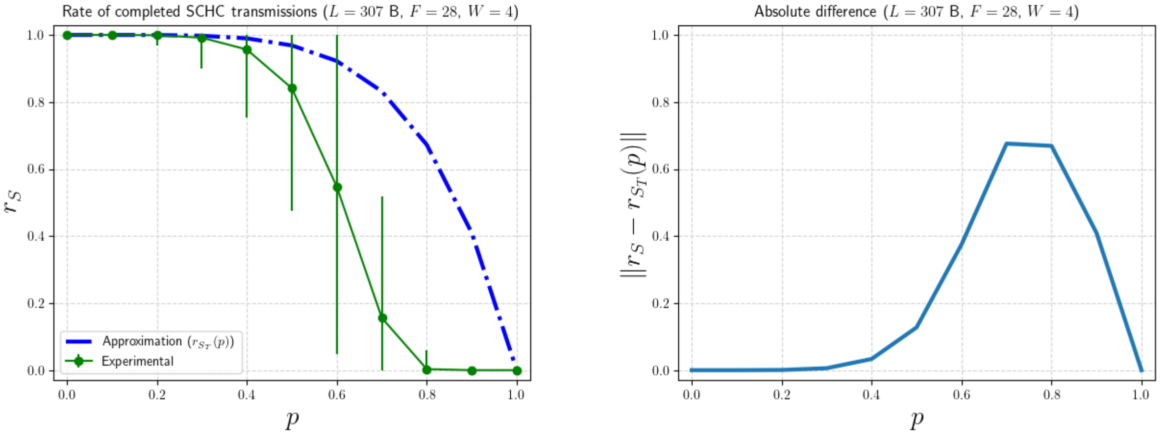
Figure G.6: Simulation results for r_S and $r_{S_T(p)}$. $L = 220$ B, $F = 21$, $W = 3$.



(a) Comparison between $r_{S_T}(p)$ and r_S.

(b) Difference between simulations and the theoretical model.

Figure G.7: Simulation results for r_S and $r_{S_T(p)}$. $L = 263$ B, $F = 24$, $W = 4$.



(a) Comparison between $r_{S_T}(p)$ and r_S.

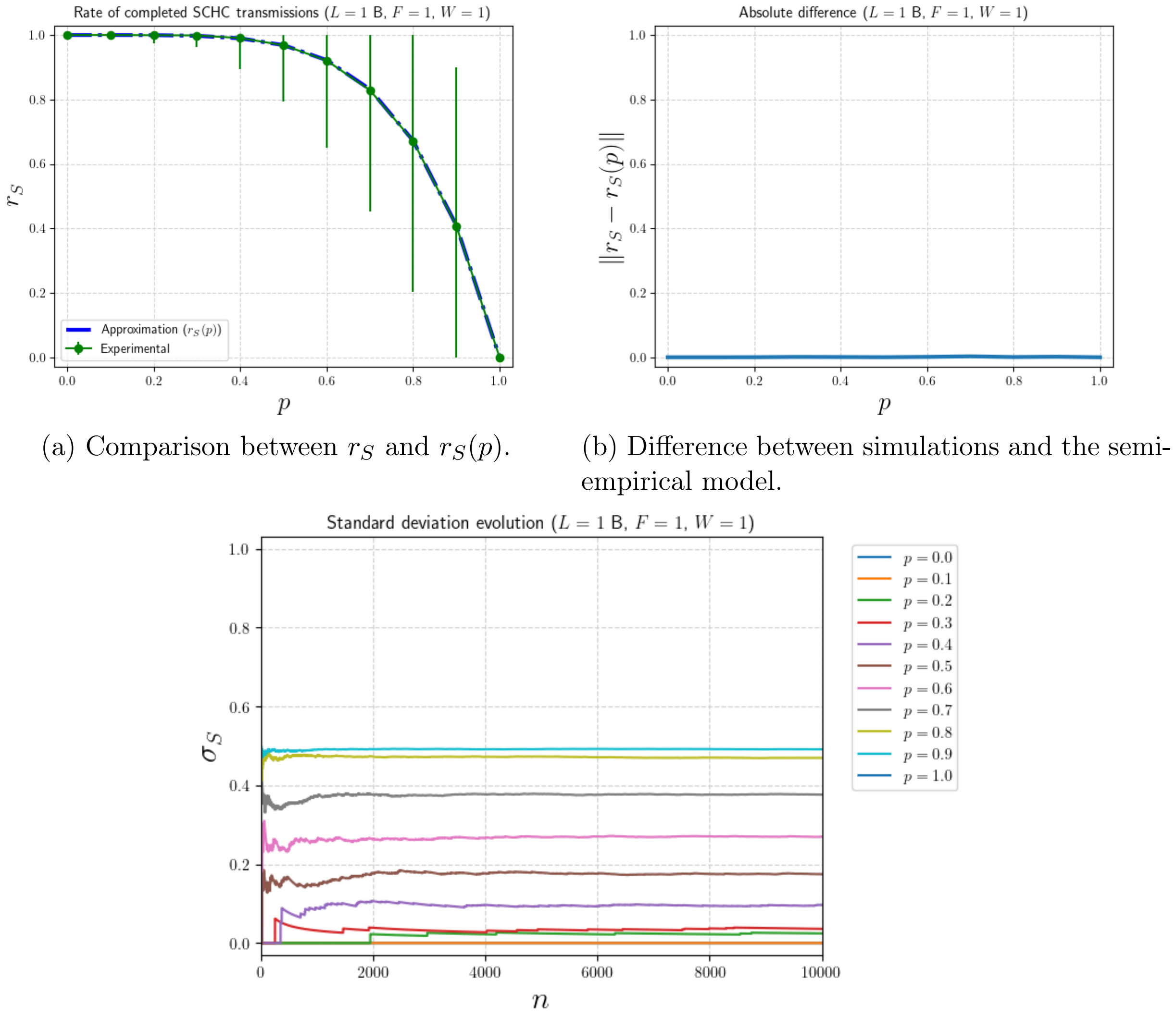
(b) Difference between simulations and the theoretical model.

Figure G.8: Simulation results for r_S and $r_{S_T(p)}$. $L = 307$ B, $F = 28$, $W = 4$.

Annex H

Validation of $r_S(p)$

This appendix provides a comparison between $r_S(p)$ and empirical results for r_S, with 10,000 samples per combination of F and p. Vertical error bars of Figures H.1, H.2, H.3, H.4, H.5, H.6, H.7, and H.8 correspond to $\pm\sigma_S$, capped not to exceed the $[0, 1]$ interval.

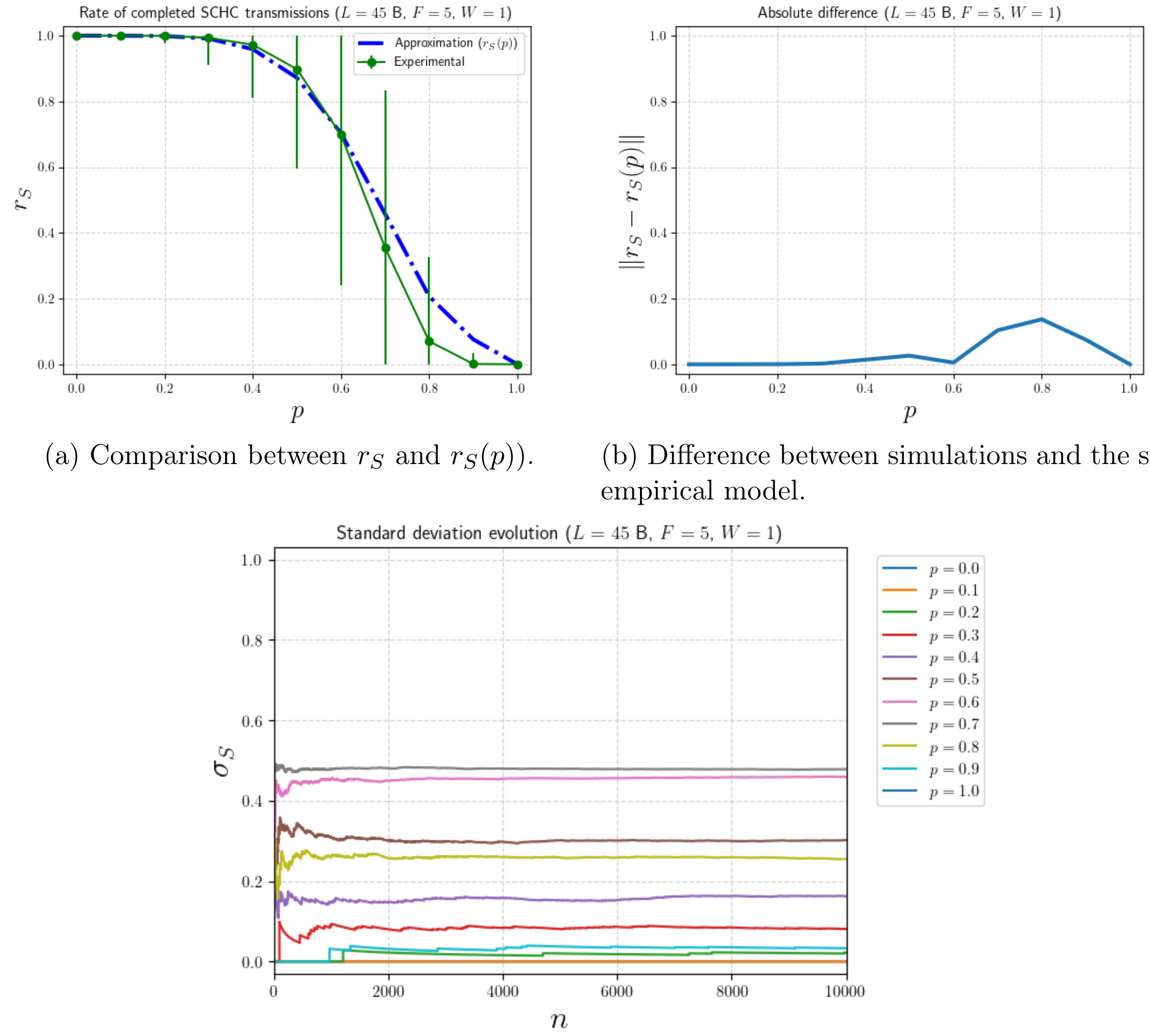


(a) Comparison between r_S and $r_S(p)$.

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.1: Simulation results for r_S and $r_S(p)$. $L = 1$ B, $F = 1$, $W = 1$.

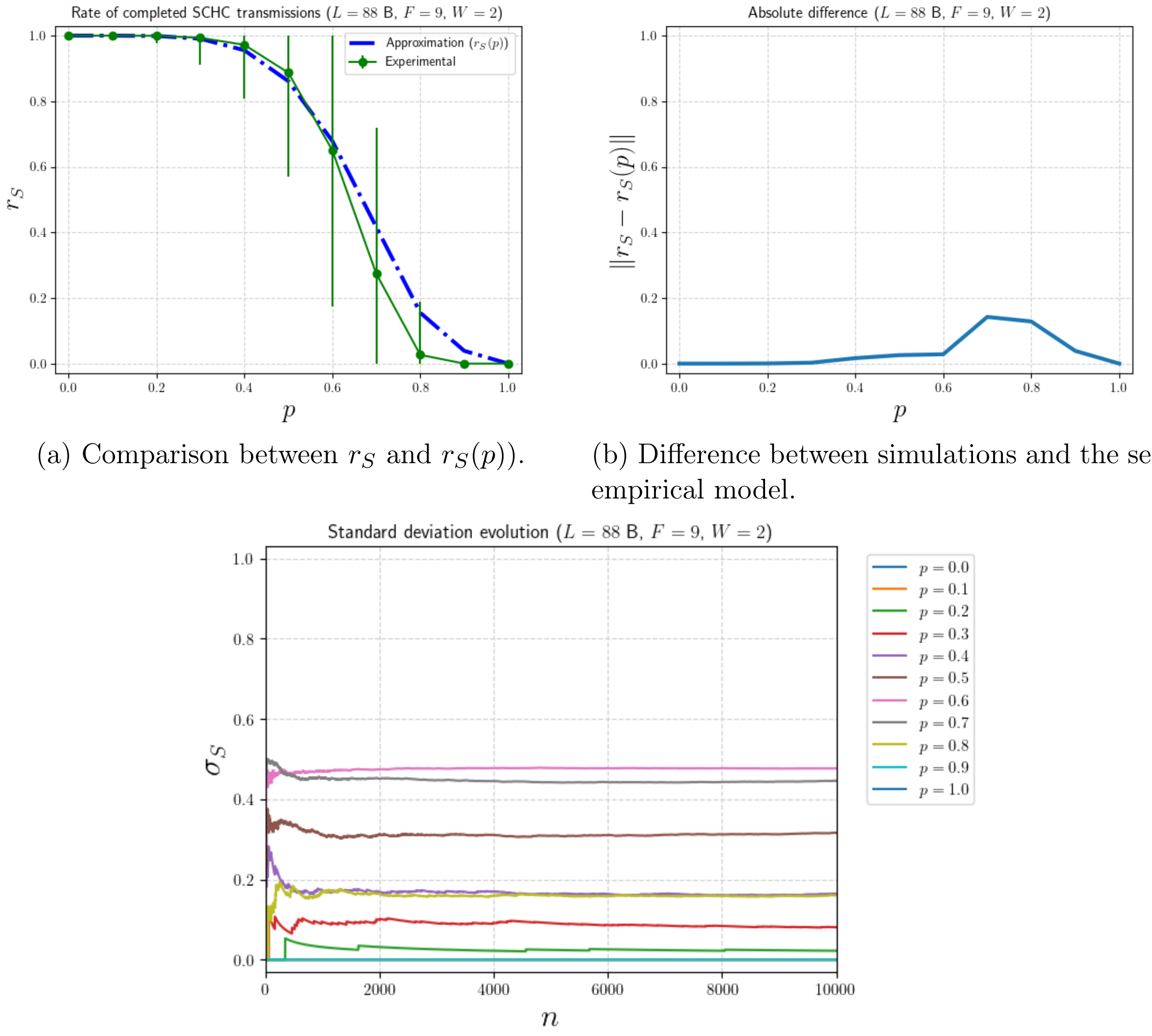


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.2: Simulation results for r_S and $r_S(p)$. $L = 45$ B, $F = 5$, $W = 1$.

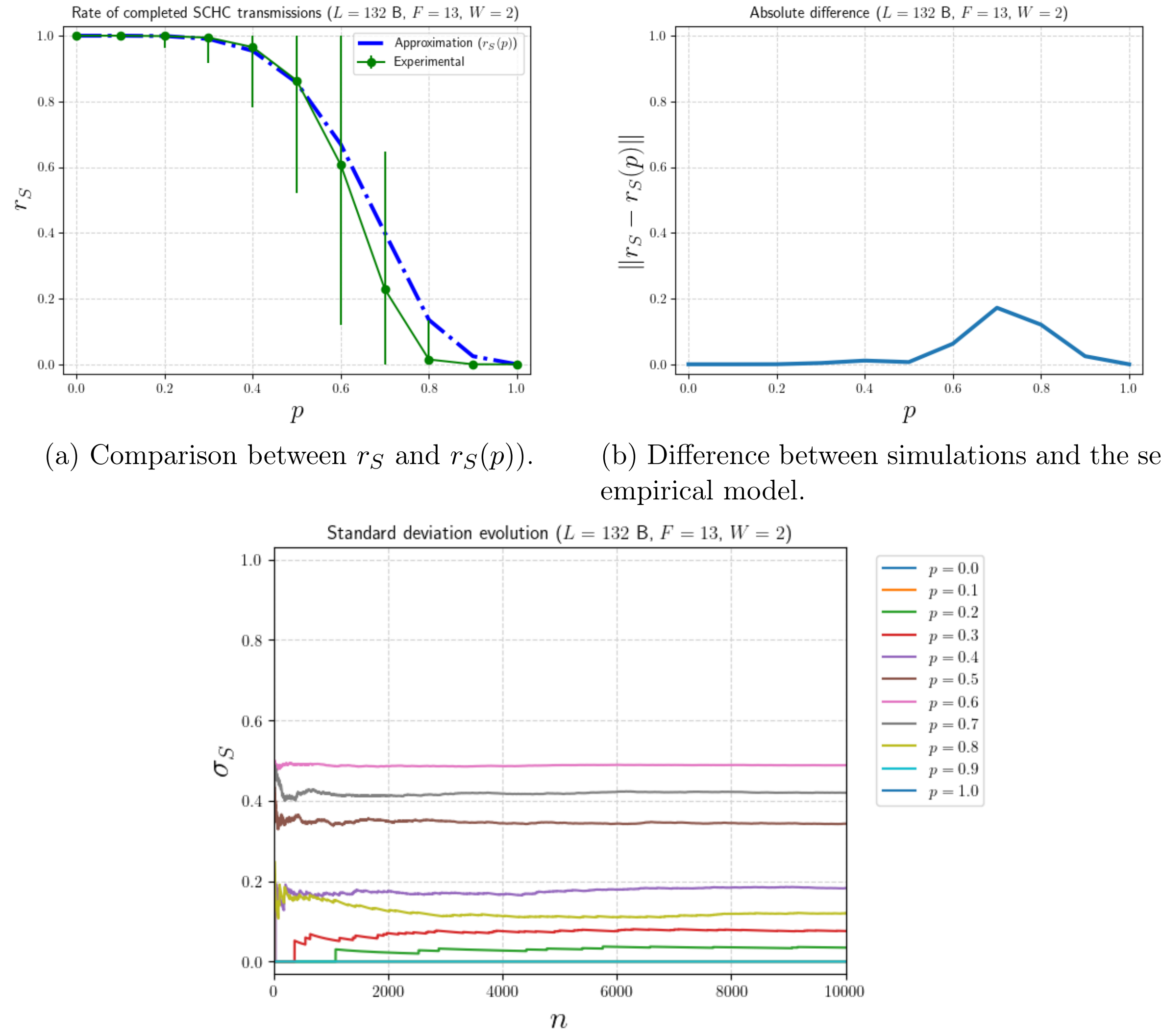


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.3: Simulation results for r_S and $r_S(p)$. $L = 88$ B, $F = 9$, $W = 2$.

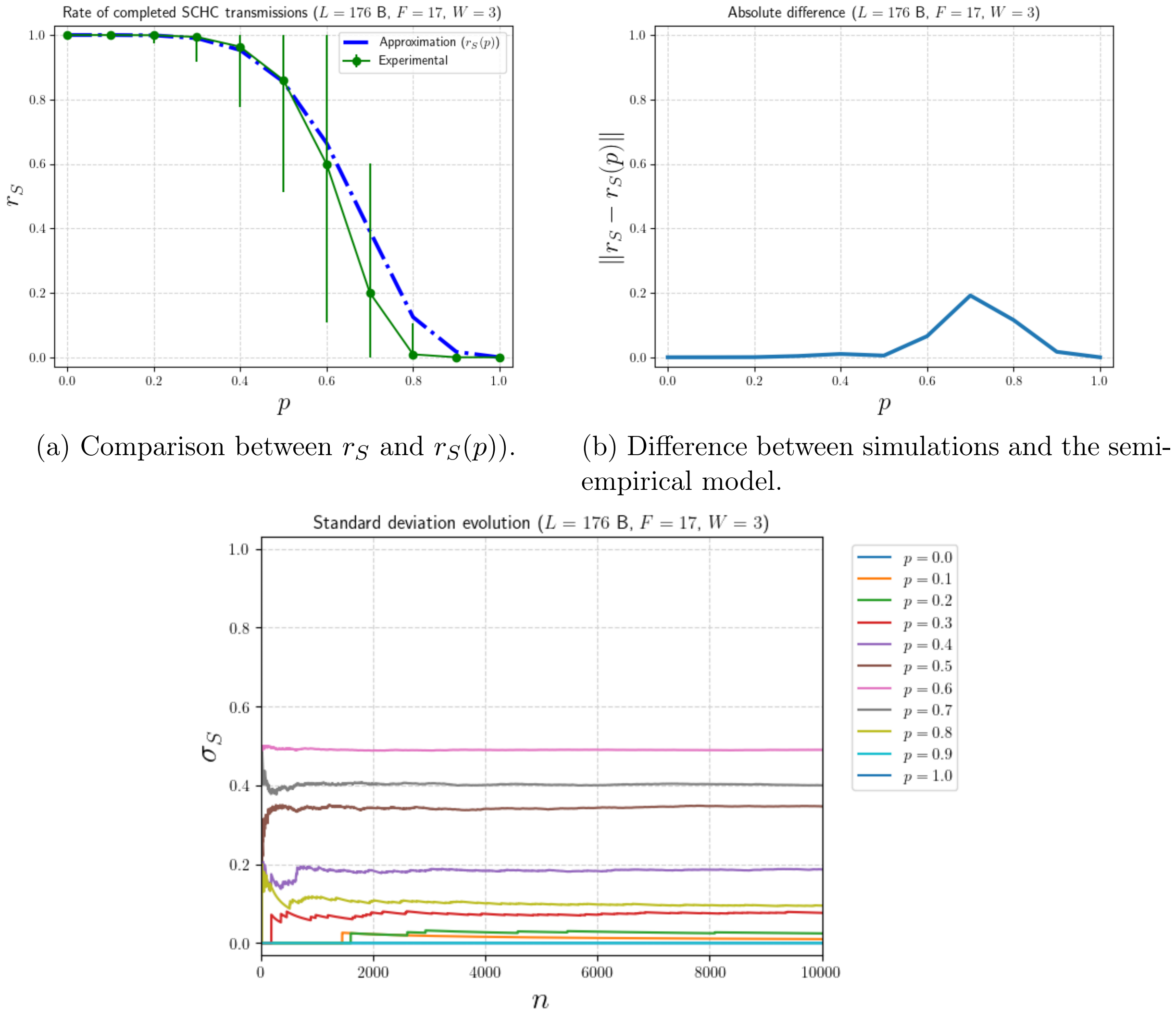


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.4: Simulation results for r_S and $r_S(p)$. $L = 132$ B, $F = 13$, $W = 2$.

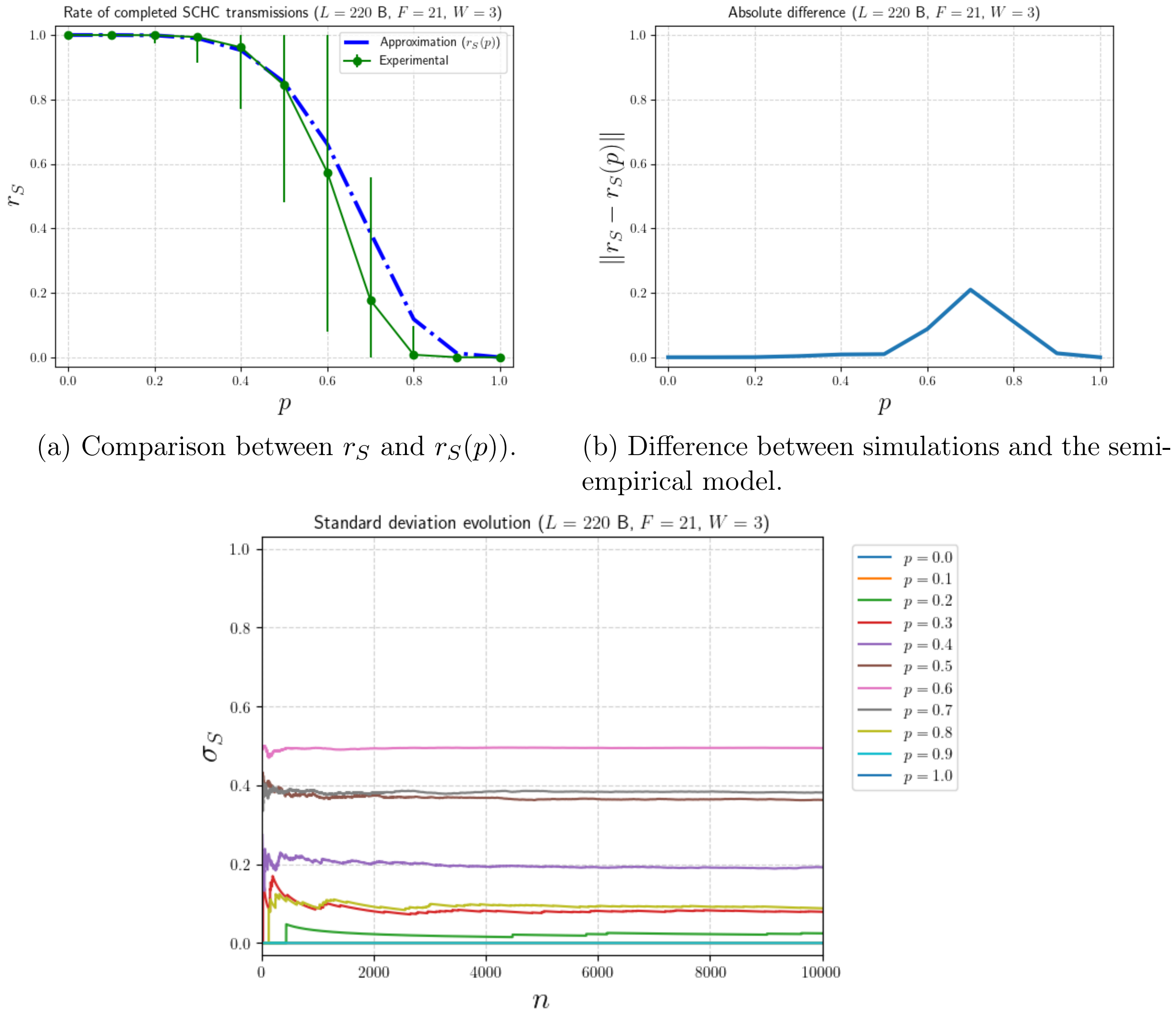


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.5: Simulation results for r_S and $r_S(p)$. $L = 176$ B, $F = 17$, $W = 3$.

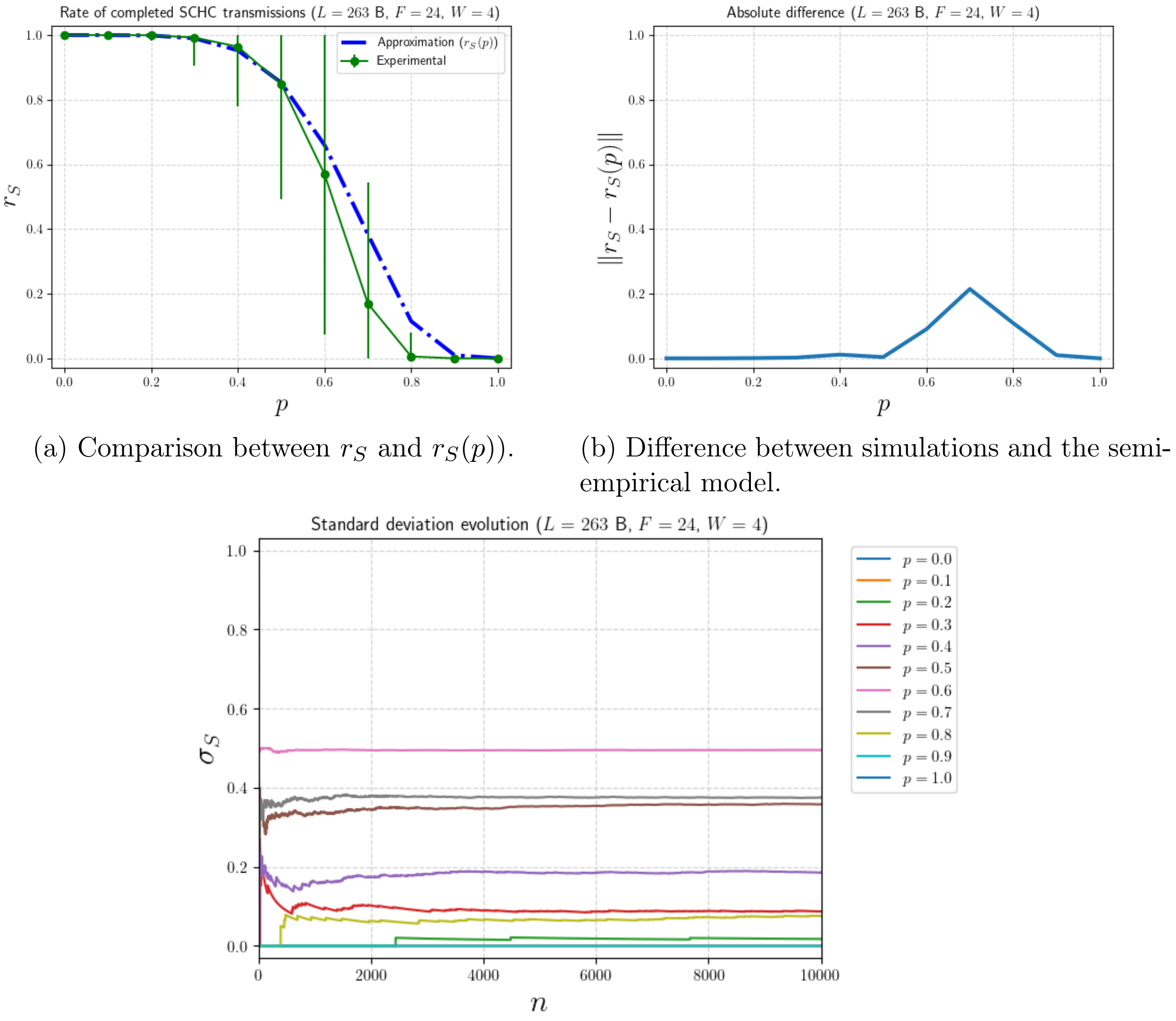


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.6: Simulation results for r_S and $r_S(p)$. $L = 220$ B, $F = 21$, $W = 3$.

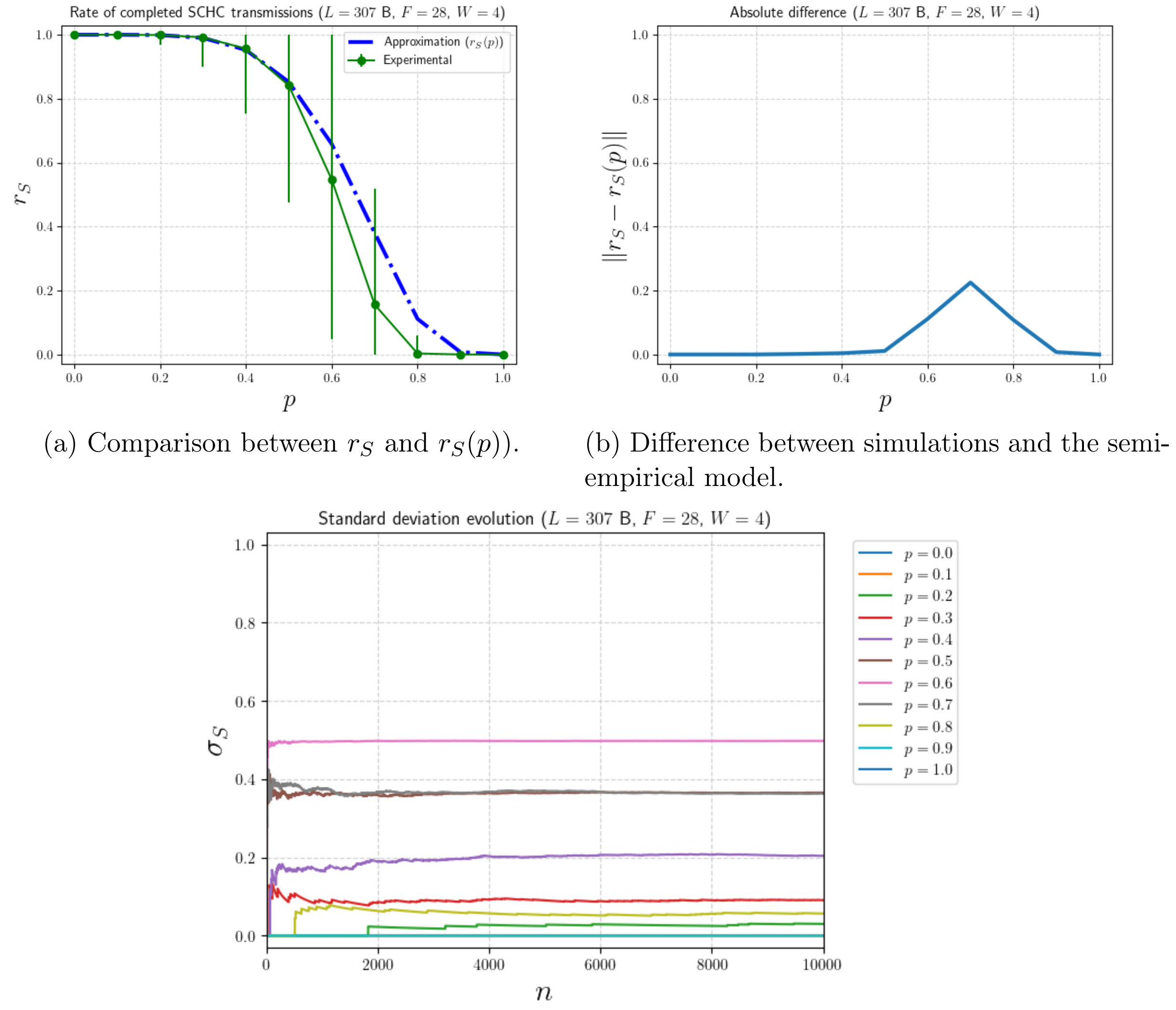


(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

(c) Standard deviation evolution per value of p.

Figure H.7: Simulation results for r_S and $r_S(p)$. $L = 263$ B, $F = 24$, $W = 4$.



(a) Comparison between r_S and $r_S(p)$).

(b) Difference between simulations and the semi-empirical model.

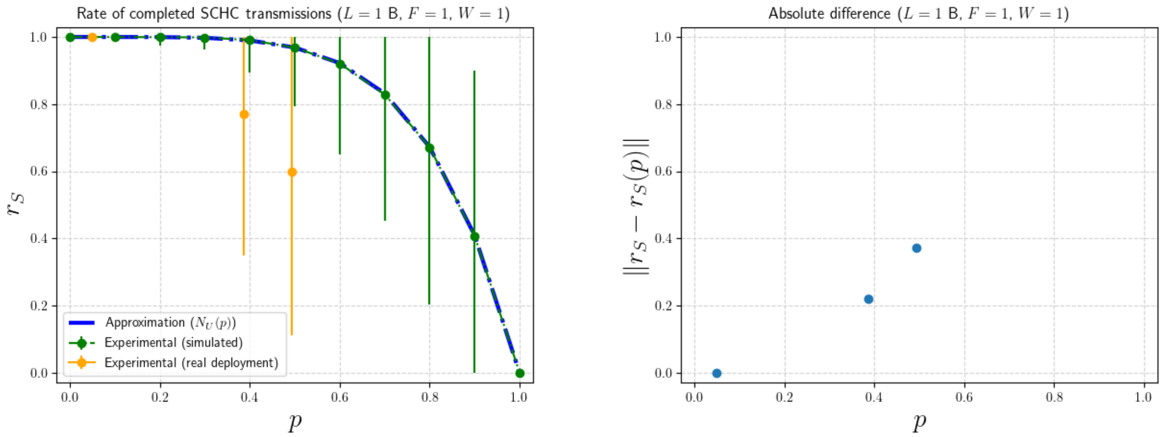
(c) Standard deviation evolution per value of p.

Figure H.8: Simulation results for r_S and $r_S(p)$. $L = 307$ B, $F = 28$, $W = 4$.

Annex I

Real deployment results of r_S

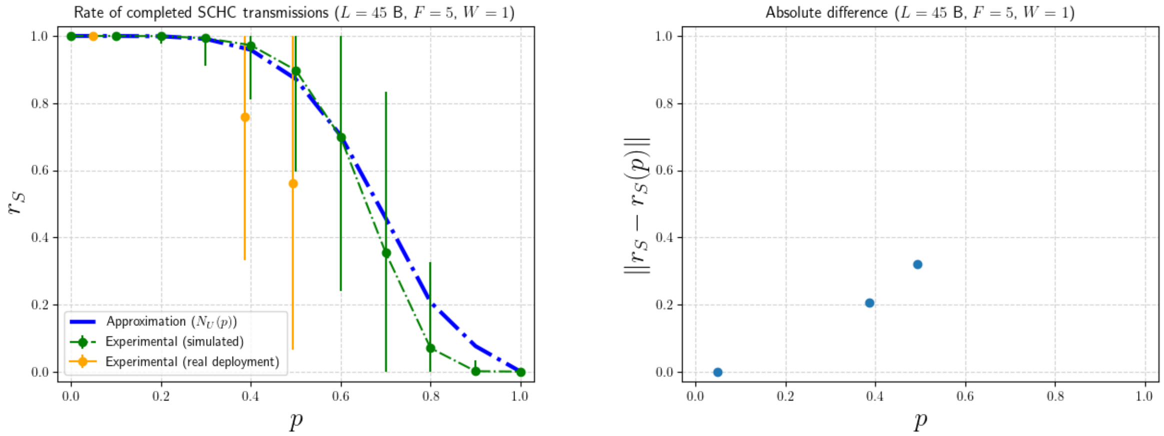
This appendix provides a comparison between $r_S(p)$ and the results for r_S obtained in the real deployment scenario by performing 100 SCHC/Sigfox transmissions per value of F in three separate runs. Vertical error bars of Figures I.1, I.2, I.3, I.4, I.5, I.6, I.7, and I.8 correspond to $\pm\sigma_S$, capped not to exceed the $[0, 1]$ interval.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

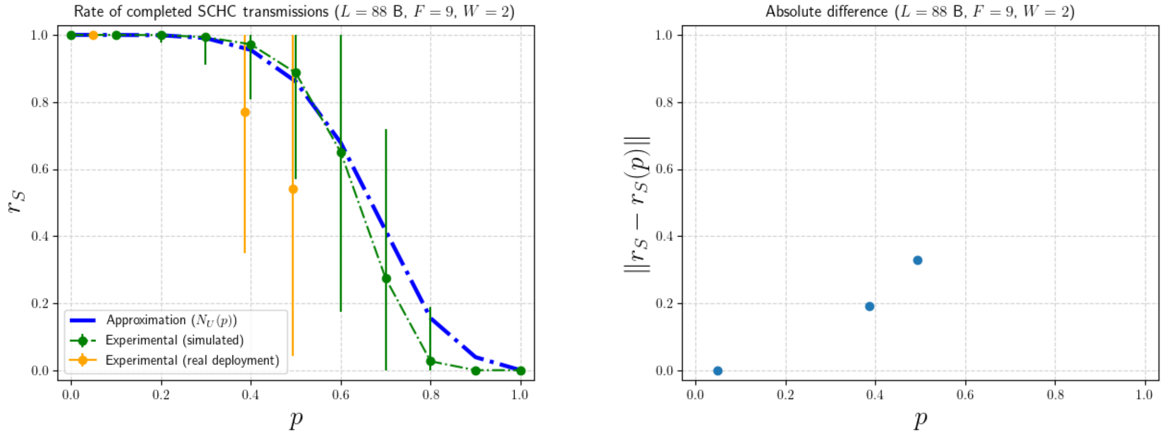
Figure I.1: Real deployment results for r_S and $r_S(p)$. $L = 1$ B, $F = 1$, $W = 1$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

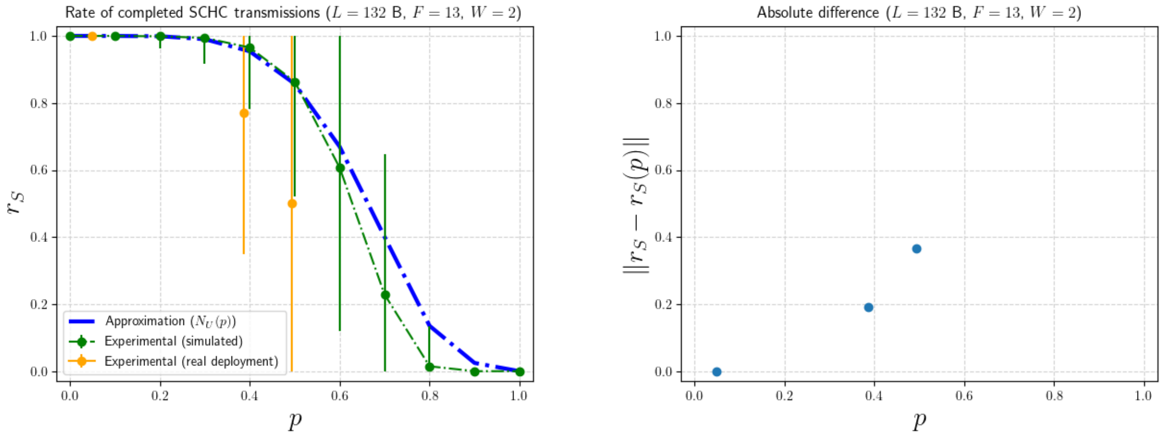
Figure I.2: Real deployment results for r_S and $r_S(p)$. $L = 45$ B, $F = 5$, $W = 1$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

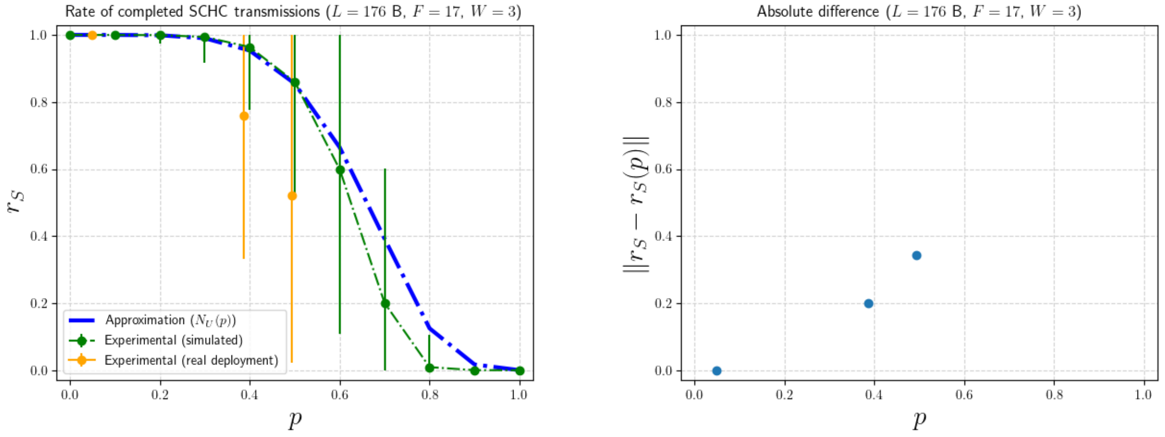
Figure I.3: Real deployment results for r_S and $r_S(p)$. $L = 88$ B, $F = 9$, $W = 2$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

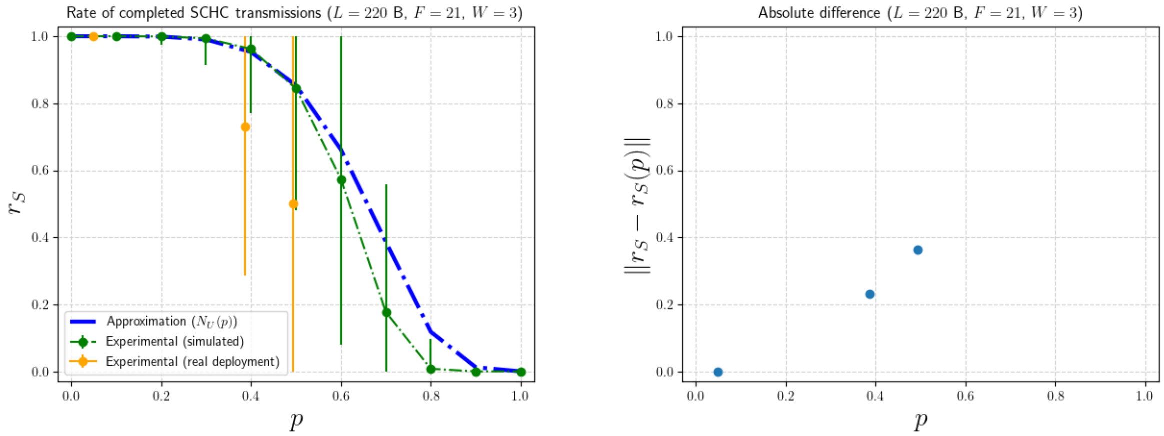
Figure I.4: Real deployment results for r_S and $r_S(p)$. $L = 132$ B, $F = 13$, $W = 2$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

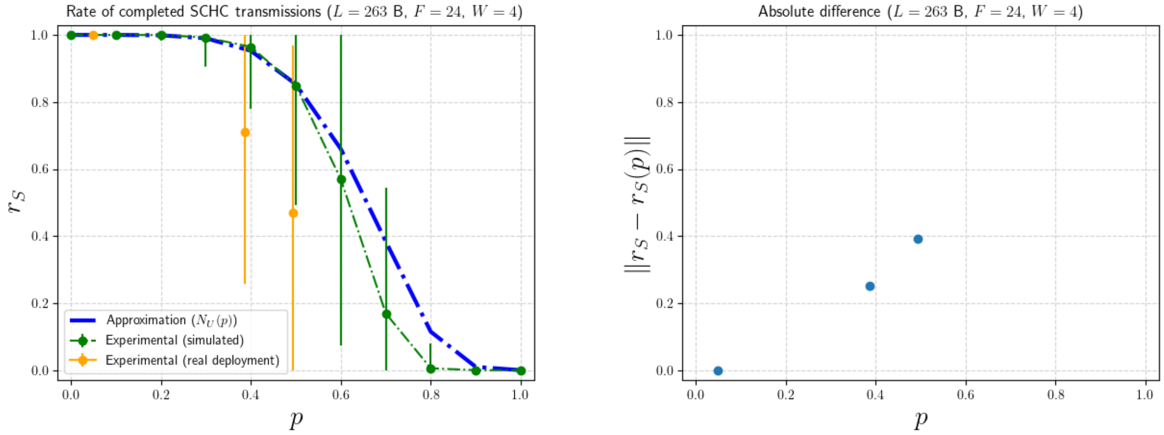
Figure I.5: Real deployment results for r_S and $r_S(p)$. $L = 176$ B, $F = 17$, $W = 3$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

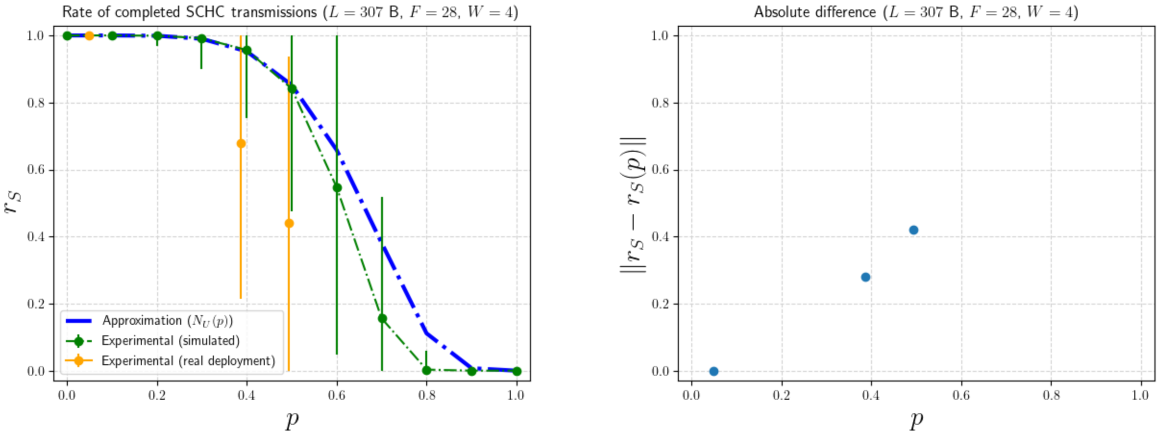
Figure I.6: Real deployment results for r_S and $r_S(p)$. $L = 220$ B, $F = 21$, $W = 3$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

Figure I.7: Real deployment results for r_S and $r_S(p)$. $L = 263$ B, $F = 24$, $W = 4$.



(a) Comparison between $r_S(p)$ and r_S.

(b) Difference between real deployment and semi-empirical model.

Figure I.8: Real deployment results for r_S and $r_S(p)$. $L = 307$ B, $F = 28$, $W = 4$.