Elisabeth Salomon, BSc

Evaluating the Accuracy of WiFi-based Distance Estimation on Raspberry Pi Hardware using Channel State Information

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieurin

Master's degree programme:
Information and Computer Engineering

submitted to

Graz University of Technology

Supervisors

Assoc.Prof. Dott. Dott. mag. Dr.techn. MSc Carlo Alberto Boano

Dr.rer.nat. BSc MSc Konrad Diwold

Institute of Technical Informatics

Graz, July 2021

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date, Signature

Kurzfassung

Die steigende Anzahl an Geräten, die mit dem Internet der Dinge verbunden sind, schafft unzählige neue Möglichkeiten für die Anwendungen in intelligenten Städten und Gebäuden, intelligenter Produktion und dem Gesundheitssektor. Für solche Anwendungen ist es oft erforderlich, dass drahtlos verbundene Geräte Kenntnis über ihren Standort haben um zum Beispiel standortbezogene Dienste, Authentifizierung und Zugangskontrollen, Navigation und Tracking-Funktionalitäten zu ermöglichen. Die Positionsbestimmung muss dafür auch in Innenräumen präzise funktionieren um beispielsweise Fehler bei Authentifizierungen oder Unfälle in der Produktion zu vermeiden. Globale Satellitensysteme wie GPS, die im Freien die Position sehr genau bestimmen können, sind für die Anwendung innerhalb von Gebäuden und Gerät-zu-Gerät-Lokalisierung ungeeignet. Daher werden für solche Anwendungen in der Regel Funktechnologien wie WiFi, Bluetooth und Ultra-Wideband eingesetzt. Während sich die jüngste Forschung aufgrund der hervorragenden Positionsgenauigkeit auf Ultra-Wideband konzentriert hat, sind auch neue vielversprechende Ansätze aufgetaucht, die die flächendeckende Verfügbarkeit von WiFi nutzen und durch die Verwendung der sogenannten Channel State Information (CSI) eine Genauigkeit im Dezimeterbereich erreichen. Aufgrund der begrenzten Verfügbarkeit von CSI auf kommerzieller Standardhardware wurden sie jedoch bisher fast ausschließlich auf älteren Plattformen implementiert.

In dieser Masterarbeit präsentieren wir die erste Implementierung WiFi-basierter Entfernungsmessung mittels Frequenzsprungverfahren auf dem neuesten Raspberry Pi, einem der bekanntesten Einplatinencomputer für IoT-Anwendungen. Konkret implementieren wir *Chronos* – einen bekannten und modernen Ansatz, der die Entfernung zwischen zwei Geräten dezimetergenau bestimmt, indem er die CSI für mehrere WiFi-Kanäle ermittelt – und evaluieren dessen Performance und Genauigkeit. Ein wesentlicher Vorteil von Chronos ist, dass keine Trainingsphase, wie zum Beispiel bei modellbasierten Ansätzen oder Fingerprinting Systemen, notwendig ist. Unsere Evaluierung in Umgebungen mit und ohne Mehrwegeausbreitung zeigt, dass eine Genauigkeit im Submeterbereich möglich ist, diese jedoch aufgrund von Mehrwegeeffekten abnimmt. Daher evaluieren wir außerdem ob die Genauigkeit verbessert werden kann, wenn CSI Daten von bestimmten WiFi-Kanälen von der Berechnung der Entfernung ausgeschlossen werden. Unsere Arbeit zeigt, dass die Nutzung von CSI eine Entfernungsmessung im Submeterbereich auf modernen Plattformen ermöglicht und dass es noch Raum zur weiteren Verbesserung der Genauigkeit gibt. Wir denken, dass dies ein wichtiger Schritt für die Entwicklung innovativer Lösungen für standortbezogene IoT-Anwendungen auf Basis der bestehenden WiFi-Infrastruktur ist.

Abstract

The increasing number of devices connected to the Internet of Things creates myriad new opportunities for applications in smart cities, smart buildings, smart production, and connected health. Such applications often require that wirelessly-connected devices embed location-awareness, which is key in enabling location-based services, authentication and access control, navigation, as well as tracking functionality. To accomplish location-awareness, and avoid errors in access control or accidents during production, it is essential to accurately determine the position also in indoor environments. Global Navigation Satellite Systems such as GPS, which perform well outdoors, however, are too inaccurate for indoor localization and also unsuitable for device-to-device localization applications. Therefore, wireless technologies such as WiFi, Bluetooth, and Ultra-Wideband are typically used for these applications. While research has lately been focusing on Ultra-Wideband due to its excellent ranging accuracy, also new promising approaches that can exploit WiFi's ubiquitousness and achieve decimeter-level accuracy by leveraging Channel State Information (CSI) have emerged. However, due to the limited availability of CSI on commercial off-the-shelf hardware, they have mostly been implemented on outdated platforms.

In this thesis, we present the first implementation of WiFi-based distance estimation using a frequency hopping approach on latest Raspberry Pi hardware, one of the most popular single-board computers for IoT applications. Specifically, we implement *Chronos* – a well-known and state-of-the-art approach that determines the distance between two devices with decimeter-level accuracy by measuring the CSI on multiple WiFi channels – and evaluate its performance and accuracy. A key advantage of Chronos is that it does not require a training phase in contrast to model-based approaches or fingerprinting systems. Our experimental evaluation in multipath and multipath-free environments shows that sub-meter ranging accuracy is possible, but that the accuracy decreases due to multipath effects. Therefore, we further evaluate whether the ranging accuracy can be improved by discarding the CSI of specific WiFi channels before estimating the distance. Our work shows that exploiting CSI on state-of-the-art platforms enables ranging at sub-meter level accuracy, and that there is still room for improvements. We believe that this is a stepping stone towards the development of innovative solutions for location-aware IoT applications based on existing WiFi infrastructure.

Danksagung

Diese Diplomarbeit wurde im Jahr 2020/2021 am Institut für Technische Informatik an der Technischen Universität Graz durchgeführt.

Zuallererst möchte ich meinen Betreuern Carlo Alberto Boano und Konrad Diwold für ihre hervorragende Unterstützung danken. Ihr hilfreiches Feedback hat wesentlich zur Umsetzung dieser Arbeit im letzten Jahr beigetragen und ich bin beiden ausgesprochen dankbar für die Zeit, die sie sich dafür regelmäßig genommen haben. Weiters bedanke ich mich bei Leo Happ Botler, sowie Markus Schuß für ihre Hilfe und Unterstützung während dieser Arbeit.

Ich möchte mich außerdem bei meinem Partner Stefan dafür bedanken, dass er mir immer unterstützend zur Seite gestanden und Verständnis gezeigt hat. Abschließend danke ich meinen Eltern für ihre grenzenlose Unterstützung während meiner gesamten Ausbildungszeit und vor allem ihren Rückhalt während des Studiums: ohne sie wäre ich nicht da wo ich heute bin.

Graz, im Juli 2021 Elisabeth Salomon

Acknowledgments

This master's thesis was written during the year 2020/2021 at the Institute of Technical Informatics at Graz University of Technology.

First and foremost, I would like to thank my supervisors Carlo Alberto Boano and Konrad Diwold for their excellent support. Their helpful feedback has contributed significantly to the realization of this work over the past year, and I am deeply grateful to both of them for regularly taking the time to do so. Furthermore, I thank Leo Happ Botler and Markus Schuß for their help and support during this work.

I also want to thank my partner Stefan for always being supportive and understanding. Finally, I would like to thank my parents for their endless support throughout my education and especially for their backing during my studies: without them, I would not be where I am today.

Graz, July 2021 Elisabeth Salomon

Contents

List of Figures

List of Tables

List of Abbreviations

A-GPS Assisted GPS.

ACK Acknowledgement.

ADC Analog Digital Converter.

AGC Automatic Gain Control.

AoA Angle of Arrival.

AP Access Point.

AWGN Additive White Gaussian Noise.

BLE Bluetooth Low Energy.

BPSK Binary Phase Shift Keying.

CCA Clear Channel Assessment.

CFO Carrier Frequency Offset.

CFR Channel Frequency Response.

CIR Channel Impulse Response.

COTS Commercial Off-the-Shelf.

CPU Central Processing Unit.

CRC Cyclic Redundancy Check.

CSI Channel State Information.

DFS Dynamic Frequency Selection.

DFT Discrete Fourier Transform.

ETSI European Telecommunications Standards Institute.

FC Frame Control.

FCC Federal Communications Commission.

FHP Frequency Hopping Protocol.

FIFO First-In-First-Out.

GNSS Global Navigation Satellite System.

IEEE Institute of Electrical and Electronics Engineers.

IFFT Inverse Fourier Transform.

INDFT Inverse Non-uniform Discrete Fourier Transform.

IOCTL Input/Output Control.

IoT Internet of Things.

IP Internet Protocol.

ISM Industrial, Scientific and Medical.

LOS Line-of-Sight.

LRL Long Retry Limit.

MAC Medium Access Control layer.

MAC Media Access Control.

MCS Modulation and Coding Scheme.

MIMO Multiple-Input Multiple-Output.

MUSIC MUltiple SIgnal Classification.

NDFT Non-uniform Discrete Fourier Transform.

NFC Near Field Communication.

NIC Network Interface Card.

NLOS None-Line-of-Sight.

OFDM Orthogonal Frequency Division Multiplexing.

OS Operating System.

PAU Power Amplifier Uncertainty.

PCB Printed Circuit Board.

PDD Packet Detection Delay.

PHY Physical layer.

PLL Phase-Locked Loop.

PPO PLL Phase Offset.

QAM Quadrature Amplitude Modulation.

QPSK Quadrature Phase Shift Keying.

RAM Random Access Memory.

REQ Request.

RF radio-frequency.

RFID Radio-Frequency Identification.

ROM Read Only Memory.

RSS Received Signal Strength.

RSSI Received Signal Strength Indicator.

RToF Return Time of Flight.

RX Receiver.

SFO Sampling Frequency Offset.

SRL Short Retry Limit.

SSH Secure Shell Protocol.

TDoA Time Difference of Arrival.

ToA Time of Arrival.

ToF Time of Flight.

TX Transmitter.

UDP User Datagram Protocol.

UNII Unlicensed National Information Infrastructure.

USB Universal Serial Bus.

UWB Ultra-Wideband.

Chapter 1

Introduction

The Internet of Things (IoT) allows to interconnect billions of devices, opening up a wide range of applications. Typical application areas include smart cities, smart buildings, connected health, industrial production, smart grids, and precision agriculture [1, 2]. Location-awareness is a key requirement for many of these applications: for example, location-based authentication and access control at critical facilities like airports, tracking systems for personnel and goods, and navigation in various environments [3, 4].

Typical requirements for location-aware IoT applications encompass accuracy, energy efficiency (since IoT devices are often battery-powered), and scalability (i.e., support for several devices): these should be maximized while supporting a sufficient responsiveness (e.g., by ensuring high update rates) and while minimizing deployment costs [5].

Outdoor IoT applications can benefit from Global Navigation Satellite Systems (GNSS) to obtain an accurate estimate of a device's position. These systems are, however, not suitable for indoor applications that require fast location estimates with sub-meter level accuracy (e.g., real-time tracking of patients in healthcare facilities and tracking of firefighters in emergencies). The radio-frequency (RF) signals for satellite communication are indeed often attenuated by the presence of buildings and urban canyons, resulting in degraded localization accuracy. Assisted GPS (A-GPS), which uses additional data from cellular networks, can reduce the time-to-first-fix and the initial inaccuracy of a position estimate. Nevertheless, it is still too inaccurate and slow for many indoor localization applications [4, 6, 7].

Therefore, recent research on indoor localization has focused on radio technologies such as Bluetooth, ZigBee, Ultra-Wideband (UWB), and WiFi. Radio chips implementing some of these technologies are often embedded in various IoT devices, which can hence be used to build indoor localization systems. Each technology has its advantages and disadvantages in terms of range, throughput, power consumption, availability, and robustness to external influences. UWB, for example, can achieve an accuracy of up to 10 cm and is less prone to multipath fading than narrowband technologies thanks to a very short pulse duration. However, UWB technology is far from being ubiquitous: only recently, it started to find its way in consumer devices such as high-end smartphones [8, 9]; furthermore, only a few UWB-based development platforms are commercially available [10, 11]. For this reason, localization using narrowband technologies such as WiFi is still very appealing, given that WiFi devices are very pervasive and already deployed in many indoor environments [1].

Localization using WiFi devices can achieve decimeter-level accuracy despite multipath fading by exploiting Channel State Information (CSI) and Received Signal Strength (RSS) information [4, 2]. Several indoor localization approaches, such as those proposed in [12], [13], and [14], have successfully demonstrated this. For example, *Chronos*, which uses CSI measurements in multiple WiFi channels to estimate the Time of Flight (ToF), was shown to achieve a median ranging accuracy of 14.1 cm and a localization accuracy of 65 cm in Line-of-Sight (LOS) conditions [12].

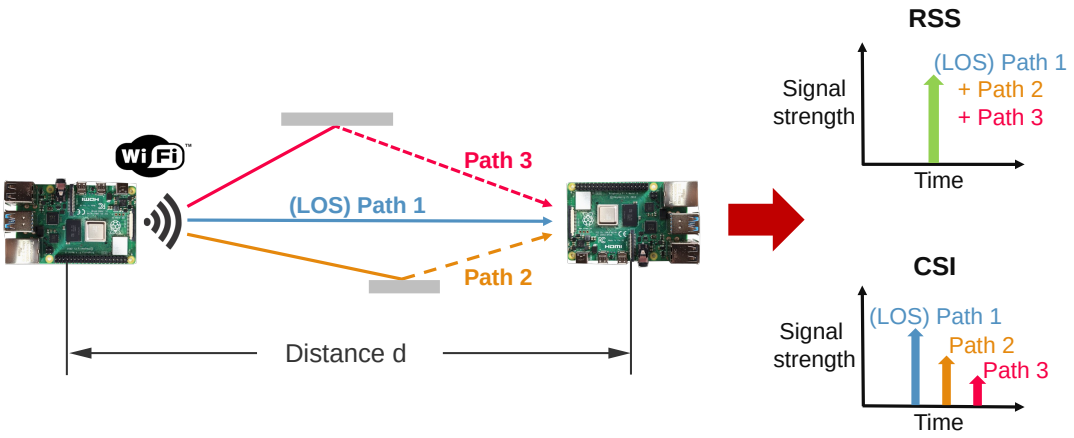


Figure 1.1: *Effects of multipath propagation on RSS and CSI.* The RSS information represents the superposition of multipath signals: therefore, it cannot be used to distinguish these signal components. In contrast, the CSI describes the frequency response of the transmission channel, which allows resolving multipath components.

1.1 Problem Statement

Multipath propagation is a major challenge for RF-based indoor localization systems. As shown in Fig. 1.1, multiple attenuated copies of the transmitted signal arrive at the receiver and affect both RSS and CSI measurements. Specifically, constructive and destructive phase superpositions lead to fluctuating measurement results in dynamic environments, which ultimately have a significant impact on localization accuracy [15, 16].

Received Signal Strength. Power-based ranging techniques assume that the power of a transmitted signal is monotonically decreasing with increasing distance. Such RSS measurements are easily and readily available on a variety of devices. For this reason, the Received Signal Strength Indicator, a vendor-specific indicator of the signal strength measured at a receiver, has been widely used for indoor localization based on fingerprinting or geometric mapping [1, 16]. RSSI-based distance estimation, however, is not very accurate, because it is easily affected by multipath effects. Small changes in the environment, indeed, can have a drastic impact on the measured power (e.g., variations can be around 5-7 dB in typical laboratory or student cubicle environments), and RSSI measurements alone cannot resolve these effects [4, 16].

Channel State Information. CSI measurements, instead, provide finer-grained information than RSS measurements, and have hence been the focus of recent research on WiFi-based indoor localization [16]. The CSI describes the channel's frequency response and consists of *amplitude* and *phase* information for each Orthogonal Frequency Division Multiplexing (OFDM) subcarrier [15]. Depending on the channel's bandwidth, CSI can be used to resolve clusters of multipath components [17]. This information is not only interesting for localization, but also for WiFi-based sensing applications, for example.

Still, environmental changes and human activities lead to changes in CSI amplitudes and phases [18]. Whilst *CSI amplitudes* are considered a reliable metric for various applications, *CSI phases* are exposed to sources of error such as Carrier Frequency Offset (CFO), Sampling Frequency Offset (SFO), Packet Detection Delay (PDD), and PLL Phase Offset (PPO) [19, 20], as further detailed in Section 2.2.3. Nevertheless, a few approaches have overcome the challenges of CSI-based localization and can achieve a sub-meter level accuracy [1, 2].

CSI extraction tools. Although the use of CSI has some advantages over RSS information, the accessibility of this metric on off-the-shelf hardware is typically rather limited. CSI measurements on Commercial Off-the-Shelf (COTS) hardware are indeed limited to certain WiFi chipsets combined with specific tools. The *Linux 802.11n CSI Tool* [21] and the *Atheros CSI Tool* [22] have long been the only CSI extraction tools available. Both tools require specific hardware that is either outdated or already deprecated by chip vendors. For example, the *IWL 5300* [23], required for the Linux CSI Tool and used in most publications about CSI-based applications so far, is discontinued by Intel [24, 25]; hardware compatible with the Atheros CSI Tool (see [26]) is also hardly available: Atheros WiFi chips embedded in Access Points (AP) and Network Interface Cards (NIC) are being replaced by new Qualcomm Atheros chipsets for which CSI extraction can no longer be guaranteed (e.g., for the *TP-Link TP-LINK WA901ND-v5*) [27]. Hence, enabling CSI extraction on state-of-the-art hardware becomes vital for future research.

Only recently, new possibilities for CSI measurements on COTS hardware have emerged. For example, the *Nexmon CSI Extraction Tool* [28], released in 2019, supports CSI extraction on Broadcom/Cypress FullMAC WiFi chips, such as those used by the Raspberry Pi 3B+/4B and Nexus smartphones. For the lightweight ESP32 microcontroller, which allows CSI extraction at low cost and low power consumption, two tools were released in 2020: the *ESP32 CSI Toolkit* [24] and *Wi-ESP* [29].

These new tools provide easy access to CSI on state-of-the-art hardware, but their use for accurate indoor localization has not yet been investigated in detail, and further research is necessary to support accurate WiFi-based localization and sensing on these platforms.

Work on CSI-based distance estimation on COTS hardware. Most of the works investigating the use of CSI extraction tools on modern COTS hardware have focused on the Nexmon CSI Extraction Tool. Several works have explored applications such as human activity and gesture recognition [30, 31, 32]. A few works have investigated the accuracy of localization approaches, but their breadth is rather limited. Gao et al. [33] have demonstrated that CRISLoc, a fingerprinting-based localization system for

smartphones, achieves a median accuracy of 29 cm. Voggu et al. [34] have shown that their fingerprinting-based solution using Raspberry Pi hardware for CSI measurements produces highly accurate localization results, i.e., a median accuracy of 93.15% for a resolution of 10 cm. Zhang et al. [35] have presented P2PLocate, a system that enables localization between single-antenna devices on off-the-shelf hardware, achieving a median accuracy of 88 cm. Whilst [33] and [34] are fingerprinting methods that use only CSI amplitudes for localization and leave the phase information unexploited, [35] uses both CSI amplitudes *and* phases, but requires additional hardware (i.e., a batteryless backscatter tag) for peer-to-peer localization. Therefore, there is clearly a gap to fill here, as no work has yet investigated the performance of CSI-based distance estimation using only phase information for unmodified COTS devices.

In this thesis, we hence aim to carry out further research on indoor localization using the Nexmon CSI Extractor and analyze whether the performance of ToF-based distance estimation using *phase information* can provide a similar accuracy to [33] and [34]. To this end, we aim to implement Chronos on Raspberry Pi 4B devices and evaluate its performance in both ideal and real-world settings.

1.2 Contributions

In this thesis, we hence focus on the implementation and experimental evaluation of Chronos on Raspberry Pi hardware. Chronos emulates a higher bandwidth using a frequency hopping approach, and it identifies and removes the errors in the CSI phases mentioned earlier, which enables precise ToF estimations. To obtain the ToF, it uses an algorithm that computes the multipath profile from CSI phases measured in multiple WiFi channels, as further explained in Section 2.3. We also analyzed whether the data of certain channels reduces the ranging accuracy and whether we can reduce ranging errors by removing these data before computing the multipath profile.

Implementing *Chronos* on the Raspberry Pi. Our first contribution is the implementation of Chronos' Frequency Hopping Protocol (FHP) on the Raspberry Pi 4B, which requires a modification of the WiFi chip's firmware. To this end, we use *Nexmon* [36], a C-based firmware patching framework for Broadcom WiFi chipsets, and the *Nexmon CSI Extraction Tool* to implement the system. Nexmon already provides some useful functions for the `BCM43455c0`, i.e., the WiFi module of the Raspberry Pi. For example, Nexmon contributes a helper function to transmit arbitrary WiFi frames and the support of implementing custom Input/Output Control (IOCTL) commands, that are required for the FHP implementation. Nevertheless, since functionalities such as frame injection in all of the 5 GHz channels and some timer functions were not available on the Raspberry Pi 4B, we had to reverse engineer parts of the firmware.

To the best of our knowledge, we are the first implementing CSI-based distance estimation using a frequency hopping approach on the Raspberry Pi. We also implemented a demonstrator that performs distance estimation with multiple devices to test the system's real-time performance. One device, acting as a server, can estimate the distance to several other Raspberry Pis (clients), and the live results are displayed on a Web-interface.

Experimental Evaluation. We conducted experiments in wired and wireless environments to evaluate the system's accuracy under various conditions. The influence of multipath propagation, different algorithm settings on the computed multipath profiles, transmission power, and frequency hopping time was further investigated. Our experiments show that sub-meter ranging errors are possible and that the selection of algorithm parameters is critical to achieving high accuracy. For the wired experiments (i.e., in a multipath-free environment), it was easier to find appropriate parameters to obtain accurate results. In contrast, the results of the wireless experiments were more sensitive to parameter changes, which we attribute to multipath effects that reduce the overall ranging accuracy in this setup.

Investigating effects of channel filtering. Due to the reduced accuracy in the wireless setup, we analyzed how discarding CSI data of some WiFi channels from the dataset before computing the multipath profiles affects the results of our experiments. We performed this analysis on the data from the wired and wireless setup to determine whether or not the observed effects were due to multipath effects. Based on these results, we investigated the use of RSSI to filter out specific channel data from the dataset to improve the accuracy of the system. Furthermore, we analyzed alternatives to RSSI-based filtering and describe options for future work.

1.3 Limitations

This thesis deals with CSI-based *distance estimation* on the Raspberry Pi 4B using the ToF estimation proposed by Chronos. Localization accuracy or ranging techniques other than Chronos (e.g., path loss based approaches or fingerprinting) are not evaluated in this work. We mainly focused on evaluating the ranging accuracy of our system. Therefore, hardware and system properties such as the Raspberry Pi's antenna quality, power consumption, or impact on the system's network traffic were also not analyzed.
Unfortunately, CSI extraction on the Raspberry Pi has a significant drawback: the payload of received WiFi packets for which CSI is measured becomes corrupted during the extraction process [37]. Thus, regular WiFi communication is no longer possible. Chronos, however, needs the CSI from both devices to estimate the distance between them. Hence, a wired connection (Ethernet cables) was used in our experiments to exchange the data and remotely control the measurements on the Raspberry Pis. This issue negatively affects our system's scalability and is a general problem for real-world applications.

1.4 Thesis Structure

This thesis is structured as follows. Chapter 2 first gives an overview of the IEEE 802.11 standard, the RSSI, and the CSI. Then, it provides information about RF-based indoor localization techniques, possible application areas, and the challenges with RSSI-based and CSI-based localization. Furthermore, Chronos [12] is explained in detail. Related work on Chronos, frequency hopping-based localization, and CSI-based applications on the Raspberry Pi are presented in Chapter 3. Chapter 4 is devoted to the implementation

of Chronos on the Raspberry Pi 4B and contains the system architecture and details on patching the WiFi chip's firmware to enable frequency hopping and the CSI extraction. At the end of this chapter, the system's limitations are discussed based on the implementation of a demonstrator. In Chapter 5, the experimental evaluation of the ranging accuracy in multipath-free and multipath environments is shown. We investigate the influence of frequency hopping speed, algorithm parameters, and transmission power on ranging accuracy in both environments. Furthermore, we evaluated whether the accuracy can be increased by discarding channel data and show the results in Chapter 6. Finally, we complete this thesis with a conclusion and an outlook on future work in Chapter 7.

Chapter 2

Background

This chapter provides an overview of WiFi-based indoor localization, its application areas, and different ranging techniques. First, the basics of the WiFi standard, including RSSI and CSI measurements, are described in Section 2.1. Then, applications and techniques of RF-based indoor localization and CSI-based approaches are explained in Section 2.2. Finally, *Chronos* [12], the ToF estimation method we implement in this work, is presented in Section 2.3.

2.1 IEEE 802.11 (WiFi)

The *IEEE 802.11 standard* is commonly known as *WiFi* and describes the Physical layer (PHY) and Medium Access Control layer (MAC) specifications for wireless connectivity. Its primary usage is for Internet Protocol (IP)-based communication networks requiring high data throughput and network coverage (e.g., to connect devices to the Internet) [1, 4]. Nowadays, it is hard to imagine life without WiFi. It plays an important role in connecting portable devices (e.g., smartphones, laptops, and tablets), smart home devices (e.g., voice assistants, surveillance cameras), and other IoT devices (e.g., Raspberry Pi, ESP32). The first version of the IEEE 802.11 standard was published in 1997 and allowed wireless communication in the 2.4 GHz band and data rates of 1 and 2 MBit/s. Since then, the standard has been adapted several times and various versions have been released [38].

IEEE 802.11 Amendments

The IEEE 802.11 standard consists of several amendments: for example, IEEE 802.11a, b, g, n, ac, and ax. These versions differ, for instance, in the highest possible data rates, operating frequency bands, supported channel bandwidth, and Multiple-Input Multiple-Output (MIMO) support [39]. Table 2.1 gives an overview of the technical specifications for IEEE 802.11/a/b/g/n/ac. Some IEEE 802.11 amendments have been developed for specific applications: e.g., IEEE 802.11s and 802.11p are designed for mesh networks and vehicular communication, respectively. In this work, we use the IEEE 802.11***n*** standard and refer to this standard when the term WiFi is used. IEEE 802.11n operates in the 2.4 GHz ISM band and the 5 GHz UNII band. It supports 20 MHz and 40 MHz channel bandwidths, a maximum of 4 MIMO streams, and data rates up to 600 MBit/s [40].

802.11 version	Released	Frequency Band	Channel Bandwidth (MHz)	Max. Data Rate	Max. supported MIMO streams	Modulation	Range (Indoor)
802.11	1997	2.4 GHz	22	1-2 MBit/s	1	DSSS, FHSS	ca. 20m
a	1999	5 GHz	20	54 MBit/s	1	OFDM	ca. 35m
b	1999	2.4 GHz	22	11 MBit/s	1	DSSS	ca. 35m
g	2003	2.4 GHz	20	54 MBit/s	1	OFDM, DSSS	ca. 38m
n	2009	2.4 GHz and 5 GHz	20	289 MBit/s	4	OFDM	ca. 70m
			40	600 MBit/s			
ac	2013	5 GHz	20	693 MBit/s	8	OFDM	ca. 35m
			40	1.6 GBit/s			
			80	3.4 GBit/s			
			160	6.9 GBit/s			

Table 2.1: Technical specifications of IEEE 802.11 amendments [39, 40, 43]. The specified data rates apply to the maximum number of supported MIMO streams.

Operating WiFi channels

The IEEE 802.11n standard supports two channel bandwidths in the 5 GHz band: 20 MHz and 40 MHz. We need the 20 MHz channels in both bands for the Chronos implementation, which are shown in Fig. 2.1. In Europe, there are 13 WiFi channels in the 2.4 GHz and 24 channels in the 5 GHz band available. Almost all channels in the 2.4 GHz band are overlapping; only channel 1, 6, and 11 are not. The channels in the 5 GHz band are all non-overlapping, but some of them (channels 52 to 140) are so-called *DFS channels* that are subject to specific regulations defined by the regulatory domain. The latter depends on the country where a device is operated: in Europe, for example, these wireless communication standards are defined by the European Telecommunications Standards Institute (ETSI) and in the United States by the Federal Communications Commission (FCC). The frequencies used by the DFS channels are shared with other radar services (e.g., military services and weather radar), and hence WiFi hardware must support Dynamic Frequency Selection (DFS) to reduce the chance of interference when a device wants to initiate transmission on such a channel (e.g., a WiFi router and a smartphone operating as AP) [41]. For further details on DFS, we refer to the corresponding ETSI regulation in [42].

Orthogonal Frequency Division Multiplexing

WiFi's PHY is based on OFDM, in which a channel is divided into N equally spaced subcarriers, each having a bandwidth of 312.5 kHz for independent data transmission. The number of subcarriers depends on the channel's bandwidth. A 20 MHz channel, for example, consists of 64 orthogonal subcarriers (subcarrier indices reach from -32 to 31) and incoming data bits are distributed among them. However, not all of the 64 subcarriers can be used to transmit user information: in the IEEE 802.11n standard, 4 of them are pilot subcarriers, 7 subcarriers at the channel borders form the guard band (subcarriers -32 to 29 and 29 to 31), and 1 is a null-subcarrier (subcarrier-0). Thus, only 52 subcarriers carry the user data. Pilot subcarriers (e.g., subcarriers ± 7 and ± 21 in a 20 MHz channel) are protocol overhead and used for channel measurements to perform channel tuning operations; guard subcarriers and the null-subcarrier are not modulated [30, 34, 43].

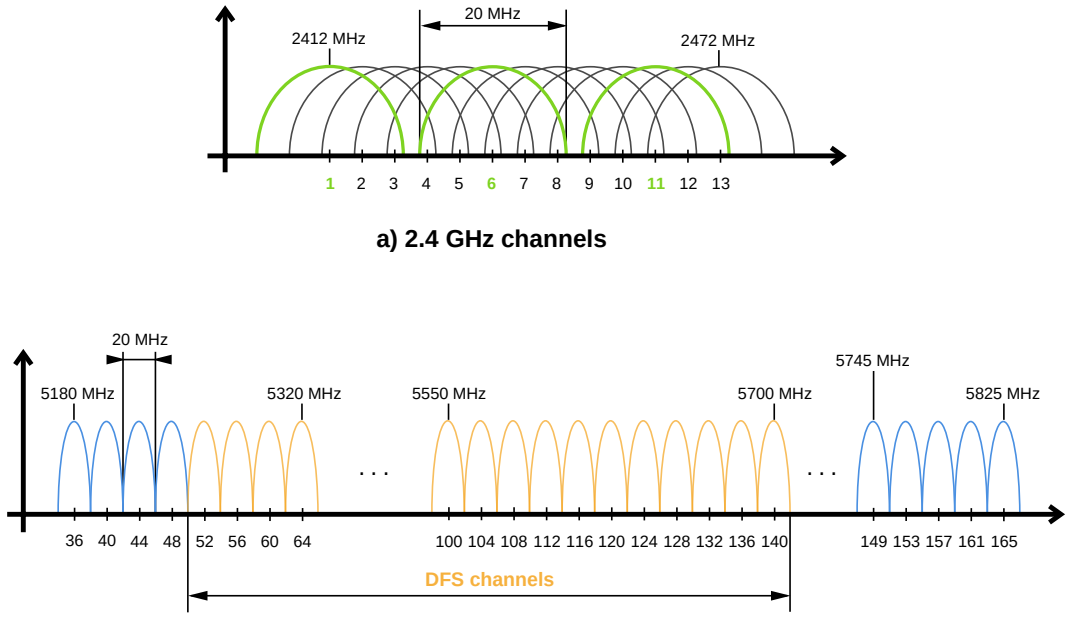


Figure 2.1: *Representation of IEEE 802.11n* 20 MHz *channels in Europe.* **a)** Channels in the 2.4 GHz band: channel 1, 6, and 11 are the only non-overlapping channels. **b)** 20 MHz channels in the 5 GHz band: channels 52 to 140 must support DFS.

Multipath Fading

An RF-signal propagates in various directions and is reflected from different surfaces, which is why a receiver receives the direct signal and its delayed and attenuated copies. Changes in the environment (e.g., user movement and opening/closing doors) have a direct impact on the characteristics of a multipath channel since both amplitude and phase of the received signal are affected by multipath. Therefore, a small change can make a big difference in the received signal quality. The worst signal attenuation, also called deep fading, occurs when the phase shift between the direct and reflected signals is an odd multiple of 180° (i.e., $\pm\,\pi$) [44].

Multipath fading depends on the wavelength of a transmitted signal, i.e., it is frequency-selective: for example, a signal with frequency f_1 (S_1) and a signal with f_2 (S_2; $f_1 \approx f_2$) are likely to travel the same paths in a given environment, but their path lengths (in wavelength units) of the direct and reflected components would differ depending on the used frequency (i.e., their wavelengths). Hence, S_1 could experience fading if the phase shift of the multipath components is 180°, while S_2 is unaffected due to a different phase shift [44].

Therefore, OFDM's frequency diversity does not only have a positive effect on achievable data rates but also makes a WiFi channel more robust against multipath fading. Different subcarriers experience different fading, making the channel less susceptible to deep fades that prevent successful communication [45]. The quality of a WiFi channel is

measured for every received packet, i.e., the receiver obtains the amplitudes and phases for all OFDM subcarriers and each transmit-receive antenna pair. This information is commonly referred to as CSI and is used to adjust the transmission settings for reliable communication [13, 30]. Further details on CSI are described in Section 2.1.2.

2.1.1 Received Signal Strength Indicator

The Received Signal Strength Indicator (in short, RSSI) is a relative measurement of the RSS with arbitrary units defined by a chip vendor [1]. There is no standard that defines RSSI measurements, and it is thus up to the chip manufacturers how they measure and report this information (i.e., the range of RSSI values and the measurement accuracy), although RSSI values are typically reported in dBm [46]. Broadcom/Cypress, for example, specifies the following RSSI range for the `BCM43455`: the minimum in the 2.4 GHz band is -95 dBm and in the 5 GHz band -98 dBm [47]. The closer the RSSI is to 0, the higher is the received signal strength.

The RSS is a result of superposition of multipath components, which is denoted as:

$$V = \sum_{i=1}^{L} |A_i| e^{-j\theta_i}, \tag{2.1}$$

$$RSS = |V|^2, \tag{2.2}$$

where L is the total number of multipath components, and A_i and θ_i are the amplitude and phase of the i^{th} received signal component. Depending on the phase shifts of the multipath components, the signal can be amplified or attenuated, resulting in higher or lower RSSI values, respectively [16, 48].

The RSS follows the path loss principle, which describes that the power of a signal is monotonically decreasing with increasing distance (i.e., the RSS is inversely proportional to its traveled distance). Hence, the relation between RSSI and the distance d between a transmitter and a receiver, also known as path loss model, can be represented as:

$$RSSI = RSSI_0 - 10n \cdot log_{10}(\frac{d}{d_0}), \tag{2.3}$$

where $RSSI_0$ is the RSSI at a reference distance d_0 (usually $d_0 = 1m$) and n is the path loss exponent, which is different for each environment and compensates the influence of multipath effects [49, 50]. The path loss exponent is usually in the order of 2 to 4: the more complex the environment, the higher the path loss exponent (n can even be greater than 4) [17]. However, since multipath channel characteristics can strongly vary under small environmental changes, RSS is susceptible to high fluctuations, which can be around 5-7 dB and therefore limit the accuracy of RSSI-based ranging [16]. Furthermore, multipath is unfortunately not the only factor influencing RSSI measurements: the antenna and hardware design also have an impact, meaning that chipsets from the same manufacturer do not necessarily perform in a similar way [46].

2.1.2 IEEE 802.11 Channel State Information

WiFi receivers use CSI measurements to get the channel's description in the frequency domain, i.e., the Channel Frequency Response (CFR). Therefore, the terms CSI and CFR

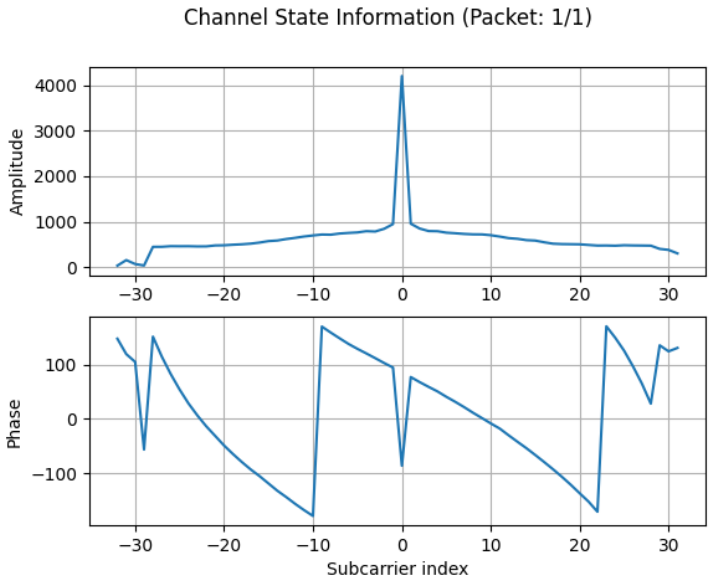


Figure 2.2: *Raw CSI of a single WiFi packet in the* 2.4 GHz *band.* The CSI was captured on a Raspberry Pi 4B using the Nexmon CSI Extractor (channel 5, bandwidth=20 MHz). The upper plot shows the amplitude-frequency response and the one on the bottom the phase-frequency response [51].

are often used interchangeably in various publications [16, 30]. The OFDM channel is modeled as:

$$\vec{y} = H\vec{x} + \vec{n}, \tag{2.4}$$

where $\vec{y}$ and $\vec{x}$ are the received and transmitted signal vector respectively, $\vec{n}$ the Additive White Gaussian Noise (AWGN) vector, and H the CSI matrix [17, 19].

The CSI per OFDM subcarrier (h_k) is then expressed as

$$h_k = |h_k| \cdot e^{j\angle h_k}, \tag{2.5}$$

where $|h_k|$ is the amplitude and $\angle h_k$ the phase of the k^{th} subcarrier [5]. An example of the CSI's *amplitude*-frequency response and *phase*-frequency response is illustrated in Fig. 2.2. This plot shows the raw CSI measured in a 20 MHz channel in the 2.4 GHz band. Since subcarrier-0 and the guard subcarriers are not involved in the data transmission (i.e., not modulated), they contain arbitrary values that can be ignored.

In the time domain, CSI can be used to describe multipath effects in the wireless channel. The channel is modeled as a linear time-invariant filter, known as Channel Impulse Response (CIR)):

$$h(\tau) = \sum_{i=1}^{L} a_i e^{-j\phi_i} \delta(\tau - \tau_i), \tag{2.6}$$

where L is the total number of multipath components and a_i, ϕ_i, and τ_i are the amplitude, phase, and time delay of the i^{th} signal, respectively. $\delta(\tau)$ is the Dirac delta function: each impulse represents a multipath component, experiencing individual attenuation and

phase shifts [16]. The CIR can be obtained from CSI by applying the Inverse Fourier Transform (IFFT). Due to the limited bandwidth of the WiFi channel, it is not possible to distinguish each multipath component, but only signal clusters. The time resolution of the CIR is inversely related to the channel bandwidth (i.e., $\tau_i - \tau_{i-1} = \frac{1}{B}$): for example, the cluster size for a 20 MHz channel is 50 ns, whereas for a 40 MHz it is channel 25 ns. Theoretically, it is possible to estimate the distance d between two devices by determining the propagation time τ of the fastest multipath signal (i.e., the direct path) from the CIR and by multiplying it with the speed of light c: $d = \tau \cdot c$. The time resolution of 20 MHz and 40 MHz WiFi channels is, however, too coarse for precise distance estimation, as the corresponding ranging resolution would only be 15 m and 7.5 m, respectively [12, 52].

2.2 Indoor Localization

In this section, the applications of and various techniques used for indoor localization are discussed. Furthermore, WiFi-based localization with RSSI and CSI measurements is explained in more detail.

2.2.1 Applications

There exists a wide range of indoor localization applications, growing with the ubiquitousness of IoT applications in modern indoor environments. Navigation, monitoring, tracking, and location-based services are just a few exemplary applications.

Navigation may be the most obvious application for an indoor localization system: for instance, it can help people to find a certain destination in a shopping mall or in public buildings, and museums and cities can offer smartphone-guided tours, with position-dependent information. Not only applications that aim to improve user experience and entertainment benefit from indoor localization: for example, disaster and risk management systems can be used to inform people in case of emergencies about evacuation plans, including navigation out of the risk zone, and help finding missing individuals.

Typical *monitoring and tracking* applications can be found in healthcare, security and access management, asset management, and industrial production. Tracking patients and staff in hospitals is an ideal application area for indoor localization: time is often a crucial factor if a patient suddenly needs medical assistance. Hence, instead of broadcasting an alarm to all staff members, only those who are close in vicinity will be notified to react as fast as possible.

Location-based services comprise applications providing entertainment, advertising, news and updates: advertisements and news about special offers can be sent to people depending on their current position and primary interests, which could be obtained from historical data and big data analytics. Another example is location-dependent WiFi connectivity (i.e., Internet access) helping café and restaurant owners to protect their WiFi against scroungers.

These three categories are not strictly separated, since indoor localization applications often combine different services like navigation and position-based information (e.g., the city guide on your smartphone and shopping mall navigation with context-aware advertising). Depending on the requirements of an indoor localization system, some RF-based technologies are better suited than others. WiFi is good for systems that want to reuse

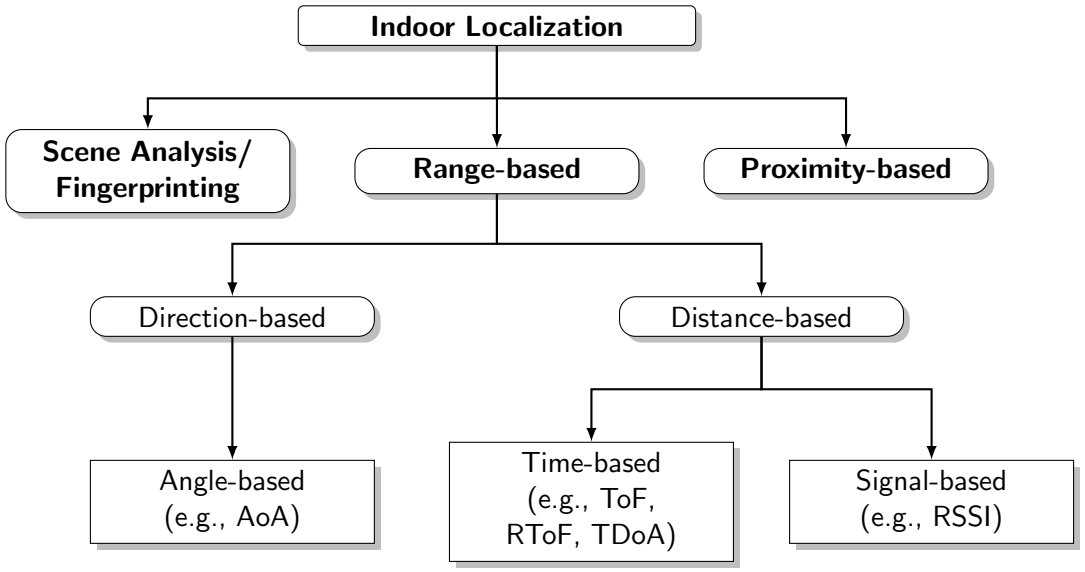


Figure 2.3: *A classification of indoor localization techniques.* The three main categories are *proximity*, *range*, and *scene analysis/fingerprinting*-based solutions. Range-based approaches can further be distinguished in direction and distance-based solutions depending on angle, timing, and signal measurements [54].

an existing infrastructure. Geo-fencing (e.g., for location-based Internet access), smart home occupancy (e.g., user tracking to adjust lighting or heating), and device-to-device localization (e.g., to find a missing friend inside a building) are some examples for such applications where the already deployed WiFi network can be exploited [1, 12, 53].

2.2.2 Classification of RF-based Indoor Localization Techniques

Indoor localization can generally be classified into the following categories: *scene analysis/fingerprinting*, *range*-based, and *proximity*-based approaches, as illustrated in Fig. 2.3. As it would go beyond the scope of this work to explain all existing methods, we will focus on the most common ones and briefly describe their basic principles.

A. Scene Analysis / Fingerprinting

Scene analysis usually consists of an *offline* (training) phase, where fingerprints (or features) are collected for a certain environment, and an *online* phase, where the system is actually in use for live measurements. Fingerprints are obtained from RSSI or CSI measurements, for instance, and then used to find a mapping between the collected information and different locations. For the latter, several algorithms can be used: e.g., use the best matching entry in a fingerprint database to obtain the most likely location or use machine learning approaches like Artificial Neural Networks (ANN), k-nearest Neighbor (kNN), or Support Vector Machines (SVM) to estimate user locations. Scene analysis can provide accurate results (depending on the spatial granularity of the collected fingerprints/features; typically below 2 m [53]); however, a large amount of data is needed to train such systems, and this data is usually strongly environment-dependent [1, 54].

B. Range-based solutions

This category can be further split into the following subcategories: solutions using angle measurements (e.g., Angle of Arrival (AoA)) are *direction*-based approaches, whereas time (e.g., Time of Flight (ToF), Time Difference of Arrival (TDoA), and Return Time of Flight (RToF)) and signal (e.g., RSSI) measurements belong to *distance*-based approaches [54].

B-1. RSSI

Besides fingerprint-based localization, RSSI can also be used for methods that deploy the path loss model for distance estimation. The basic principle of path loss-based ranging was already shown in Section 2.1.1. Eq. 2.3 does, however, not reflect random shadowing effects caused by obstacles. Shadowing leads to signal attenuation in addition to path loss due to absorption, reflection, scattering, and diffraction [16]. The Log-normal Distance Path Loss model supplements Eq. 2.3 and is denoted as:

$$RSSI = RSSI_0 - 10n \cdot log_{10}(\frac{d}{d_0}) + X_\sigma, \tag{2.7}$$

where X_σ is a zero-mean log-normally distributed random variable with standard deviation σ, representing shadowing effects [55, 56].

B-2. ToF / ToA

Time of Flight and Time of Arrival-based approaches do actually measure the same thing, namely the propagation time of a signal from a transmitter to a receiver. The difference is that ToA measurements use the transmitter as a reference point, whereas ToF the receiver: ToA uses absolute timestamps to compute the propagation time, while ToF is based on relative measurements. [57]. Since we are using a ToF-based approach in this work, we will explain further details about it. To get the distance d between two transceivers, the ToF τ is multiplied by the speed of light c [12]:

$$d = \tau \cdot c. \tag{2.8}$$

According to this equation, time-based ranging approaches are just as good as the precision of the time measurements itself: e.g., a deviation of 5 ns already causes an error of 1.5 m. Hence, ToF measurements usually require fairly accurate time synchronization [4]. Trilateration methods such as linear least squares, for example, can be used to estimate the user location. To this end, a device measures the ToF to at least three reference nodes to get a single point of intersection corresponding to the actual position [58].

B-3. RToF

The time measured by Return Time of Flight-based approaches is the round-trip time (RTT), and the principle is similar to ToF measurements. A receiver (RX) responds to the signal from a transmitter (TX), which then calculates the RTT from the timestamps of the four exchanged signals: e.g., TX sends a packet at t_1 to RX, which arrives at t_2, and RX's response, sent at t_3, is received by TX at t_4. TX can then estimate the RTT by calculating:

$$d = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \cdot c. \tag{2.9}$$

Compared to ToF measurements, RToF estimations require only a relatively moderate clock synchronization. However, the processing delay at the receiver needs to be known for short-range systems (i.e., indoor localization), as small time differences (in the order of nano-seconds) drastically impact the accuracy [1].

B-4. TDoA

The principle behind Time Difference of Arrival-based approaches is to estimate the user or device location by computing the difference in ToA at multiple ($N+1$, for $N \geq 3$) anchors with known positions. For instance, a transmitter TX sends a message to $N+1$ synchronized receivers (RX), and the difference in propagation time for each receiver pair $RX_{i,j}$ $(i, j = 1, ..., N+1)$ is calculated, resulting in N TDoA observations. The TX location (x_0, y_0) lies on the intersection of the N resulting hyperboloids with constant range difference between two receivers (x_i, y_i). The hyperbola equation can be expressed as:

$$D_{i,j} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - \sqrt{(x_j - x_0)^2 + (y_j - y_0)^2}. \tag{2.10}$$

To get the target location $D_{0,0}$, the system of N equations is either solved with numerical methods or using closed form [4].

B-5. AoA

Direction-based indoor localization systems rely on angle measurements such as the Angle of Arrival. To calculate the AoA (i.e., the direction of an incoming signal), an antenna array is needed, which usually means that more complex hardware is required. A transmitted signal impinges with different time delays (i.e., phase difference) at each receiver antenna. This phase difference and the known distance d between two receiver antennas can be exploited to estimate the AoA θ_i of the i^{th} propagation path. The phase shift $\Delta\phi$ at the m^{th} antenna, which is relative to the reference antenna, is:

$$\Delta\phi = \frac{-2\pi d(m-1)\sin(\theta_i)f}{c}, \tag{2.11}$$

where f is the frequency of the transmitted signal and c the speed of light [14]. Compared to time-based localization, AoA-based solutions already work with just two anchors. The accuracy, however, degrades as the transmitter-receiver distance increases. Small deviations in the AoA calculation are then translated into large errors in the estimated location [1, 59].

C. Proximity-based solutions

Compared to scene analysis and range-based solutions, proximity-based approaches cannot calculate the absolute or relative location of a target. The location is determined using RSSI measurements to anchor nodes at known positions. The node being in the target's proximity, i.e., the one measuring the highest signal strength, estimates its position. Typically, technologies like NFC, RFID, or BLE are applied for such systems. Proximity-based localization is rather easy to implement due to its simplicity, but accuracy suffers [57, 54].

Nevertheless, this technique is suitable for various indoor applications that do not require precise location information (e.g., user tracking in shopping malls and sending marketing alerts) [1].

2.2.3 CSI-based Indoor Localization

WiFi-based indoor localization mainly covers range and scene analysis-based approaches. The CSI amplitudes and phases provide a fine-grained description of the wireless multipath channel, which can be used, e.g., for ToF/ToA and AoA estimations and fingerprinting. Therefore, CSI-based indoor localization has increasingly displaced RSSI solutions and several approaches relying on CSI measurements have been proposed achieving sub-meter level accuracy [2]. In this section, we will describe the challenges with amplitude and phase-based localization as well as some existing approaches.

Using CSI Amplitudes

CSI amplitudes provide fine-grained received signal strength information for each OFDM subcarrier. According to the path loss model (Eq. 2.3), the signal is attenuated relative to its traveled distance. The Automatic Gain Control (AGC), a closed-loop feedback circuit, attempts to compensate for this attenuation to maintain a certain signal level. Perfect compensation, however, is not possible and hence this Power Amplifier Uncertainty (PAU) introduces an offset to the CSI amplitudes. Averaging of several measurements can help to compensate this error [60, 20]. Typical localization approaches relying on CSI amplitudes are, for instance, path loss and scene analysis-based ranging.

Path loss-based ranging

Since CSI is extracted after the AGC, the power represented in the CSI amplitudes no longer follows the path loss principle [33]. Therefore, power-based ranging cannot rely directly on CSI amplitude measurements. The Linux CSI Tool, for example, already provides the AGC value and a function to obtain scaled CSI data [61]; the Nexmon CSI Tool provides neither the AGC value nor calibration. Thus, Gao et al. [33] suggest using the RSSI to cancel out the AGC, because it is obtained before the gain control. According to Wu et al. [17], they were the first to demonstrate power-based ranging with CSI measurements. Instead of RSSI, their system (FILA) used the so-called effective CSI (CSI_{eff}), calculated as the weighted average of the subcarrier amplitudes (from a single WiFi packet). To describe the relation between the transmitter-receiver distance and CSI_{eff}, they proposed a modified version of the path loss model. Their work showed that CSI measurements were more stable over time than RSSI measurements, and the median localization accuracy could be increased from approximately 0.8m with RSSI to 0.45m with CSI.

Scene analysis with CSI amplitudes

CSI amplitudes are frequently used in scene analysis-based localization systems. Fingerprints and features are therefore computed from either CFR or CIR amplitudes, sometimes also in combination with RSSI values and CSI phases [53]. FIFS [62], for example, is a fingerprinting-based localization system that calculates, similar to FILA, an effective CSI from the amplitude values of the subcarriers to quantify the power of a received WiFi

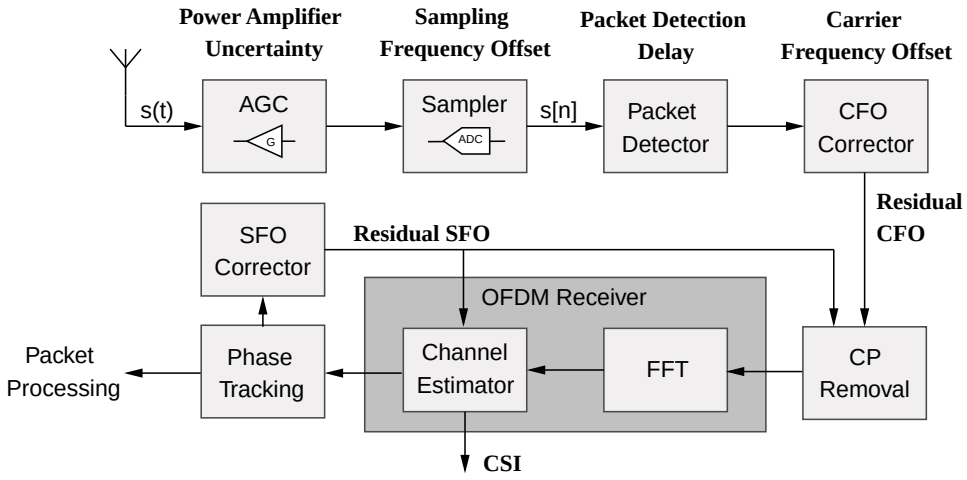


Figure 2.4: *Schematic of the signal processing on a WiFi receiver.* OFDM signal amplitudes and phases experience certain offsets and delays at different stages of the signal processing. This schematic shows where amplitude (i.e., PAU) and phase offsets (i.e., SFO, PDD, and CFO) are introduced by the receiver hardware [20, 22].

packet. A probabilistic model is applied as mapping function for the online localization, and the mean error is less than 1 m. ConFi [63] uses a convolutional neural network, trained with features combining CSI amplitudes from multiple antennas over time (so-called CSI feature images) and achieves an accuracy of 1.37 m. CRISLoc [33] is the first fingerprinting localization system deployed on off-the-shelf smartphones. They apply an edge enhanced k-nearest neighbors matching rule for the localization, and the system is able to automatically reconstruct fingerprints for altered APs. The mean error is 0.29 m in a research laboratory and 0.78 m in a more complex environment.

Using CSI Phases

Raw CSI phases are not suitable for localization approaches, as several error sources and offsets influence them. Thus, pre-processing of phase information is required to make accurate ranging with CSI phases possible [60]. Fig. 2.4 illustrates how the signal processing on a WiFi receiver influences the CSI measurement.

PLL Phase Offset (PPO)
The Phase-Locked Loop (PLL) generates the center frequency for wireless transceivers and starts at a random initial phase. Since this initial phase differs between the transmitter and the receiver, the measured CSI experiences an additional phase offset [20]. This PPO is constant for a given transmitter-receiver pair until the next reset [60].

Carrier Frequency Offset (CFO)
Perfect synchronization of transmitter and receiver center frequencies cannot be achieved, and the resulting CFO leads to a time-varying CSI phase offset affecting all subcarriers. This offset can be corrected to a certain extent, but not completely due to hardware imperfections [20].

Sampling Frequency Offset (SFO)
Missing clock synchronization causes offsets in the sampling frequencies of the transmitter and the receiver. The SFO can therefore introduce an additional time shift to the received signal after the ADC. Residual offsets after the SFO correction cause phase rotation errors, but these are almost constant over some time (in the order of minutes) [20].

Packet Detection Delay (PDD)
The time required for a transmitted packet to be recognized by the receiver is referred to as PDD. A certain energy detection threshold must be exceeded to detect an incoming packet. Hence, this delay depends on the power of the transmitted signal and is different for each packet [12].

These offsets and delays cause linear errors in the CSI phases of each subcarrier. Therefore, the measured phase $\phi_{i,k}$ for subcarrier k in channel i can be expressed as:

$$\angle h_{i,k} = \phi_{i,k} = (\theta_{i,k} + \Delta_{i,k}) \mod 2\pi, \tag{2.12}$$

$$\theta_{i,k} = -2\pi f_{i,k}\tau \mod 2\pi, \tag{2.13}$$

$$\Delta_{i,k} = -2\pi(f_{i,k} - f_{i,0})\delta_i, \tag{2.14}$$

where $\theta_{i,k}$ is the ideal phase and $\Delta_{i,k}$ the additional phase offset. δ_i represents the time shift introduced by PDD and SFO, and $(f_{i,k} - f_{i,0})$ describes the difference between subcarrier frequency $f_{i,k}$ and the center frequency $f_{i,0}$ [12].
CSI phases are typically used for AoA and ToF/ToA estimations. SpotFi [14], for example, uses the phases from different subcarriers and three antennas to estimate the AoA with the super-resolution MUSIC algorithm [64]. It combines AoA and ToF measurements and can achieve a median accuracy of 0.4 m [14]. ToneTrack [65] is a TDoA-based localization system using channel hopping to increase the bandwidth for ToA estimations. The ToA is estimated with the MUSIC algorithm, and the system has a median accuracy of 0.9 m. Chronos [12] uses a channel hopping approach for ToF estimations like ToneTrack and will be further described in the next section.

2.3 Chronos: Decimeter-level Accuracy using CSI Phases

Chronos [12] was published in 2016 by Vasisht et al., who managed to enable decimeter-level localization accuracy with a single AP using ToF estimations. The system is implemented on the well-known IWL 5300 NIC (with three antennas) combined with the Linux 802.11n CSI Tool. Their experimental evaluation showed that the median error of the estimated distances was 14.1 cm and the median localization error 65 cm, which is precise enough for applications like geo-fencing, home occupancy measurements, or finding lost devices. In this section, the different steps of Chronos' distance estimation process are described in detail.

2.3.1 Frequency Hopping Protocol

As described in Section 2.1.2, the time resolution of the CIR of a 20 MHz channel is too coarse for direct ToF estimations. Therefore, Chronos uses a frequency hopping approach to emulate a higher bandwidth. The IEEE 802.11n standard has the advantage of operating in the 2.4 GHz and the 5 GHz band, resulting in 35 usable channels in the United States (U.S.) and 37 in Europe, spanning a bandwidth of nearly 1 GHz.

Chronos lets the server and a client synchronously switch between these channels (i.e., 35 in the U.S.) using a Request (REQ)/Acknowledgement (ACK)-based Frequency Hopping Protocol (FHP), as illustrated in Fig. 2.5. The CSI is measured for each exchanged WiFi packet, i.e., in each channel and on both devices. Both server and client begin in the same channel, e.g., channel 1 in the 2.4 GHz band. The server starts the FHP by sending a REQ packet, for which the client performs the CSI measurement. After responding with an ACK, the client switches to the next channel. The server measures the CSI for the received ACK packet and tunes to the same channel as the client before sending a new REQ. The reason for measuring the CSI on both sides is explained in Section 2.3.2. If everything went fine and the server received the ACK in the last channel (e.g., channel 165 in the 5 GHz band), we can estimate the distance and restart the FHP. Otherwise, if, for example, a packet is lost and a timeout occurred, both devices automatically reset their operating channels according to the default settings. This fall-back routine is necessary because the CSI of a complete hopping sequence (i.e., the CSI in all channels) is needed to compute the ToF, as further explained in Section 2.3.3.

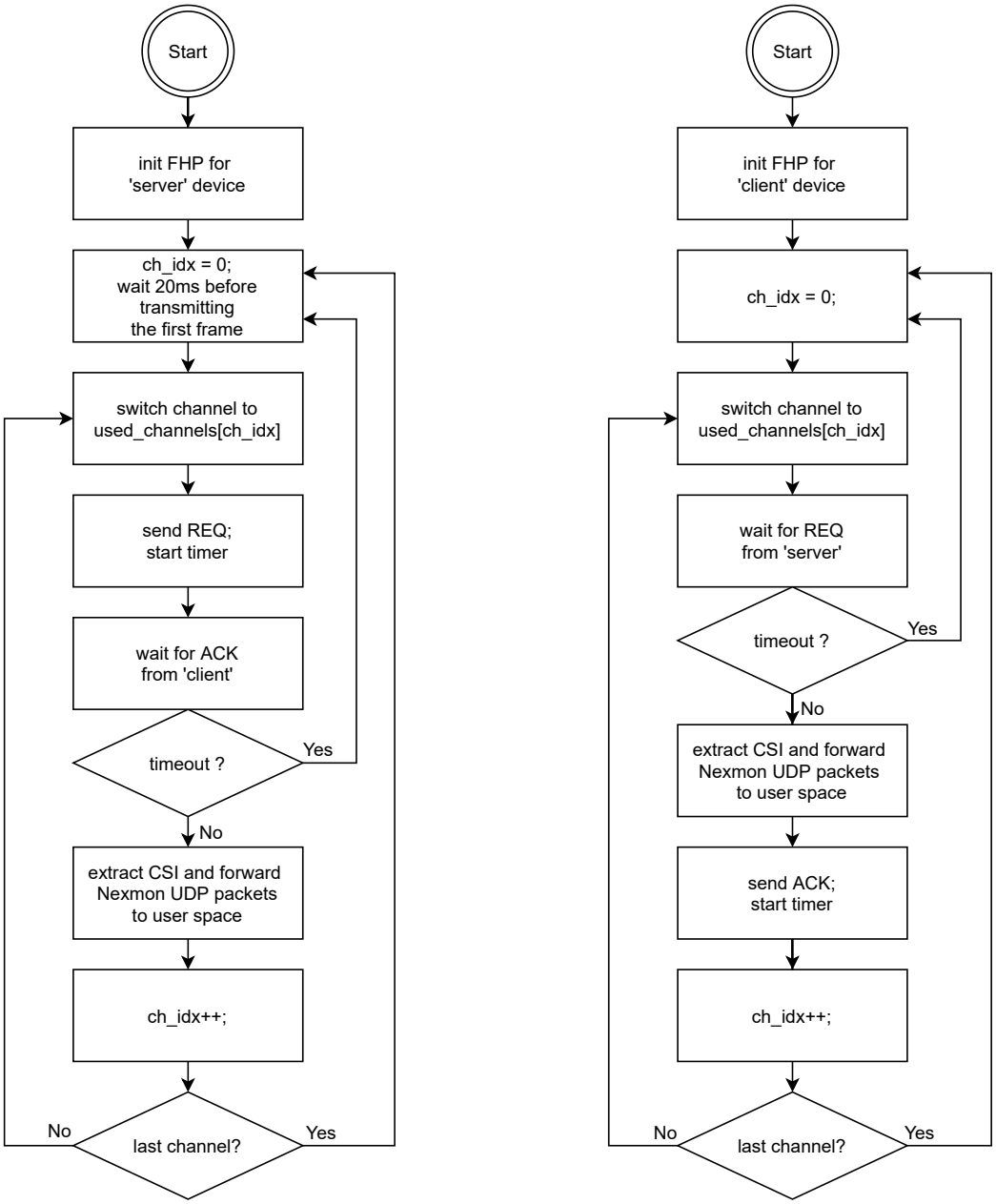
2.3.2 Eliminating Hardware Delays and Phase Offsets

Section 2.2.3 already described that the actual wireless channel h_i with center frequency f_i differs from the measured channel $\tilde{h}_i$. Hardware imperfections and packet detection result in certain offsets making raw CSI phases inapplicable for reliable and accurate ranging. Chronos, however, proposes some techniques to effectively mitigate the influence of PDD, CFO, and PPO on the ToF estimation.

Eliminating PDD

The time it takes a receiver to detect an incoming WiFi packet can be an order of magnitude larger than the actual ToF, which is usually just a few tens of nano-seconds. Chronos measured and evaluated the PDD and observed a median delay of 177 ns with a standard deviation of 24.8 ns. A PDD of 177 ns corresponds to a distance of 53 m. Consequently, it is crucial to account for PDDs and derive the real ToF from h_i. Chronos claims that the measured channel at subcarrier-0 ($\tilde{h}_{i,0}$) is not affected by the PDD and is therefore identical to the actual channel ($h_{i,0}$).

PDD and ToF influence the subcarriers differently: while the ToF phase rotation (see Eq. 2.13) is caused when the signal propagates from a transmitter to a receiver, the PDD phase rotation (see Eq. 2.14) occurs during the digital signal processing state on the receiver once the carrier frequency has been removed. From Eq. 2.12, we can see that for $k = 0$ (i.e., subcarrier-0) the term $-2\pi(f_{i,k} - f_{i,0})\delta_i = 0$, and therefore the phase of the measured channel $\angle\tilde{h}_{i,0}$ equals the phase of the real channel $\angle h_{i,0}$. Thus, Chronos uses only the phase at subcarrier-0 for its ToF estimations.



(a) Server side FHP **(b) Client side FHP**

Figure 2.5: *REQ-ACK-based Frequency Hopping Protocol.* This flowchart describes our implementation of the server and client side FHP. The server initiates the frequency hopping and both devices hop synchronously through a given set of channels. CSI is measured for REQ and ACK packets (i.e., on both sides) for correcting phase offsets.

Since subcarrier-0 is not modulated in OFDM, CSI measurements are not available at the center frequency. Chronos overcomes this issue by interpolating the CSI at subcarrier-0 using the information from the other subcarriers. This is possible because the physical phenomena influencing an RF signal in the wireless channel are continuous over a small number of OFDM subcarriers. Vasisht et al. used the Cubic spline interpolation for their implementation.

Correcting PPO and CFO

Every time the channel is switched during the FHP, the PLL starts at a new random initial phase which is different at the transmitter ($\phi^{tx}_{i,0}$) and at the receiver ($\phi^{rx}_{i,0}$). This offset ($\phi^{tx}_{i,0} - \phi^{rx}_{i,0}$) negatively impacts the measured phase and thus the ToF estimation. The carrier frequency difference between the transmitter ($f^{tx}_{i,0}$) and the receiver ($f^{rx}_{i,0}$) also causes a phase offset that accumulates quickly over time. Hence, the CFO correction needs to be applied for each WiFi packet. Taking PPO and CFO into account, the CSI of subcarrier-0 measured in channel i at the receiver can be expressed as:

$$CSI^{rx}_{i,0}(t) = \tilde{h}_{i,0} e^{j(f^{tx}_{i,0} - f^{rx}_{i,0})t + j(\phi^{tx}_{i,0} - \phi^{rx}_{i,0})}. \tag{2.15}$$

If the CSI is measured for a WiFi packet sent in the other direction (i.e., from the receiver to the transmitter), the signs of the phase and frequency offsets change with respect to the second device (i.e., the receiver):

$$CSI^{tx}_{i,0}(t) = \tilde{h}_{i,0} e^{j(f^{rx}_{i,0} - f^{tx}_{i,0})t + j(\phi^{rx}_{i,0} - \phi^{tx}_{i,0})}. \tag{2.16}$$

According to this knowledge, the transmitter and receiver CSI can simply be multiplied to cancel out the CFO and PPO, and what remains is the square of the wireless channel:

$$\tilde{h}^2_{i,0} = CSI^{rx}_{i,0}(t) \cdot CSI^{tx}_{i,0}(t). \tag{2.17}$$

Therefore, CSI measurements are necessary in the forward and the reverse channel (i.e., on the server and the client side) while hopping through all the WiFi channels. The CSI multiplication results in the ToF being doubled, which is not a problem for Chronos' ToF estimation.

2.3.3 Time-of-Flight Computation with Multipath Profiles

From Eq. 2.13 we know that the phase of a received signal is depending on its ToF. Hence, we can use this relation to estimate the ToF τ from the measured phase ϕ as:

$$\tau = -\frac{\phi}{2\pi f} \mod \frac{1}{f}. \tag{2.18}$$

The modulo operation is needed because the phase repeats every 2π leading to several possible solutions for the ToF. For example, if $f = 5\,\text{GHz}$, the ToF is obtained modulo $0.2\,\text{ns}$ since the phase is identical for signal propagation times $\tau + x \cdot 0.2\,\text{ns}$ (for $x \in \mathbb{N}$), i.e., $\tau_1 = 0.15\,\text{ns}$ and $\tau_2 = 0.35\,\text{ns}$ lead to the same phase shift $\phi = 4.712\,\text{rad}$ at a receiver.

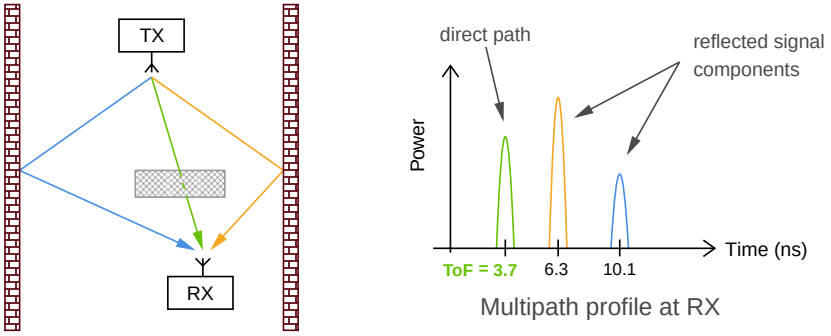


Figure 2.6: *Illustration of a multipath profile with three signal components.* The transmitted signal propagates over different paths from the transmitter (TX) to the receiver (RX). It is possible to distinguish the direct and the reflected signals components by their propagation time. The ToF of the direct path can be used to estimate the distance between RX and TX.

If the phase of a signal is known for a set of different frequencies f_i, the Chinese remainder theorem can be applied to obtain the correct ToF among all possible solutions:

$$\tau = -\frac{\phi_i}{2\pi f_i} \mod \frac{1}{f_i}, \qquad \forall i \in \{1, 2, ...n\}. \tag{2.19}$$

The FHP and the associated CSI measurements in each channel provide us this system of equations and the correct ToF will satisfy all of them. Chronos, however, also takes into account the issue of multipath propagation and therefore finds the correct solution by calculating the Inverse Non-uniform Discrete Fourier Transform (Inverse NDFT).

The Inverse NDFT

Multipath propagation is an already discussed issue for indoor localization. CSI, however, can help to disentangle multipath components and make ToF estimations more accurate. Chronos computes the multipath profile from the CSI at subcarrier-0 ($\tilde{h}_{i,0}$) measured in several channels, which allows to distinguish the ToF of the direct signal from its reflections: an example with three multipath components and the resulting multipath profile is shown in Fig. 2.6. The measured channel $\tilde{h}_{i,0}$ can be written as the sum of multipath components:

$$\tilde{h}_{i,0} = \sum_{l=1}^{L} a_i e^{-j2\pi f_{i,0}\tau_l}, \qquad \forall i \in \{1, 2, ...n\} \tag{2.20}$$

As this equation has the form of a Discrete Fourier Transform (DFT), the inverse DFT could be used to get the multipath profile. However, the inverse DFT requires samples at equidistant frequencies and cannot be used in our case as WiFi channels are not equally spaced. Therefore, Chronos is inverting the NDFT to compute multipath profiles.

The Inverse Non-uniform Discrete Fourier Transform (INDFT) is an under-determined system that has not only a single closed-form solution, but several possible ones. To get

Algorithm 1: Inverse NDFT Computation

Data: $\tilde{\mathbf{h}}$, measured channels,
$\mathcal{F}$: non-uniform DFT matrix ($\mathcal{F}_{i,k} = e^{-j2\pi f_{i,0}\tau_k}$),
α: sparsity parameter,
ϵ: convergence parameter
Result: $\mathbf{p}$, inverse NDFT (multipath profile)
Initialize: $t = 0$, $\mathbf{p_0}$ to a random value, $\gamma = \frac{1}{\|\mathcal{F}\|_2^2}$;
while *converged = FALSE* **do**
  $\mathbf{p_{t+1}} = \texttt{sparsify}(\mathbf{p_t} - \gamma\mathcal{F}^*(\mathcal{F}\mathbf{p_t} - \tilde{\mathbf{h}}), \gamma\alpha)$;
  if $\|\mathbf{p_{t+1}} - \mathbf{p_t}\|_2 < \epsilon$ **then**
    converged = TRUE;
    $\mathbf{p_t} = \mathbf{p_{t+1}}$;
  else
    $t = t + 1$;
  end
end
Function `sparsify(`$\mathbf{p}$, t`):`
  for $i = 1, 2, ..., length(\mathbf{p})$ **do**
    if $|\mathbf{p_i}| < t$ **then**
      $\mathbf{p_i} = 0$;
    else
      $\mathbf{p_i} = \mathbf{p_i}\frac{|\mathbf{p_i}|-t}{|\mathbf{p_i}|}$;
    end
  end
  return $\mathbf{p}$;

the best solution among them, Chronos adds the sparsity constraint. It can be formulated as an L-1 minimization problem and guarantees that the sparse multipath profile $\mathbf{p}$ is an INDFT solution of $\tilde{\mathbf{h}}$ which can be obtained by solving:

$$\min \|\mathbf{p}\|_1 \quad s.t. \quad \|\tilde{\mathbf{h}} - \mathcal{F}\mathbf{p}\|_2^2 = 0, \tag{2.21}$$

where $\tilde{\mathbf{h}} = [\tilde{h}_{1,0}, ..., \tilde{h}_{n,0}]^T$ is the $n \times 1$ vector of wireless channel measurements (n is the number of channels), $\mathcal{F}$ is the $n \times m$ Fourier matrix with m being the number of discrete time samples $\tau \in \{\tau_1, ..., \tau_m\}$ of vector $\mathbf{p}$, and $\|\cdot\|_1$ and $\|\cdot\|_2$ are the L-1 norm and the L-2 norm respectively. Chronos favors sparse solutions (i.e., multipath profiles with a few dominant signal components) because in indoor environments usually some minimal attenuated paths dominate weaker ones.

The optimization problem in Eq. 2.21 can be reformulated using Lagrange multipliers:

$$\min_{\mathbf{p}} \|\tilde{\mathbf{h}} - \mathcal{F}\mathbf{p}\|_2^2 + \alpha\|\mathbf{p}\|_1, \tag{2.22}$$

with α being the sparsity parameter; the higher the sparsity parameter, the less dominant signal components in the multipath profile. Chronos uses Algorithm 1, a proximal

gradient-descent style algorithm with step size γ, to compute the INDFT and get the sparse multipath profile $\mathbf{p}$ with respect to Eq. 2.22. The `sparsify()` function thereby obtains the sparse solutions for $\mathbf{p}$ in every iteration step: elements in $\mathbf{p}$ are set to zero depending on a certain threshold t, computed from the step size and the sparsity parameter. Suitable parameter values will be discussed in Chapter 5.

Distance Estimation with Corrected Data

Now that we have the multipath profile, which gives the propagation times of some dominant paths, the time at which the first peak appears is selected as the ToF; the direct path is the shortest, and thus fastest. We can estimate the distance between the transmitter-receiver pair by multiplying the ToF with the speed of light (see Eq. 2.8).

Since we are multiplying the CSI for CFO and PPO correction, the algorithm input is $\tilde{\mathbf{h}}^{\mathbf{2}} = [\tilde{h}^2_{1,0}, ..., \tilde{h}^2_{n,0}]^T$ instead of $\tilde{\mathbf{h}}$. Using the squared input, however, only leads to the first peak appearing at twice the ToF. Thus, the obtained ToF can simply be divided by two to get the correct result.

For an ideal CFO and PPO correction, the CSI needs to be measured on the server and the client simultaneously. As this is not possible in practice, there is a certain delay between the measurements introducing an additional phase error. Nevertheless, compared to uncompensated CFO and PPO errors, this error is significantly smaller, and Chronos suggests compensating it by averaging several packets. Hardware delays also distort the ToF estimation. According to Vasisht et al., such delays lead to a constant additive value, which could be calibrated by measuring the ToF at a known distance.

Another limitation is that the Fourier transform leads to an ambiguity depending on the *minimum* frequency separation Δf of the input vector. Vasisht et al. state that Δf is 5 MHz for their system (i.e., the time ambiguity is 200 ns corresponding to 60 m) as the frequency separation in the 2.4 GHz band is less than the channel bandwidth because of overlapping channels. For indoor localization, however, 60 m is usually enough.

Chapter 3

Related Work

This chapter introduces some of the existing work related to frequency hopping-based indoor localization approaches and CSI-based applications on the Raspberry Pi. Section 3.1 summarizes the findings from a work that re-evaluated Chronos on the original hardware, i.e., the IWL 5300. Next, WiFi-based localization solutions that exploit frequency hopping techniques are described in Section 3.2. Finally, Section 3.3 outlines other works using the Raspberry Pi for localization and sensing with CSI measurements.

3.1 Chronos Evaluation on the IWL 5300

In 2019, Patricia García Ferrín [27] investigated the accuracy of Chronos on the original hardware, i.e., the IWL 5300 NIC. Her insights were helpful in implementing our work, and the most important ones are briefly described in this section.

Frequency Hopping

Chronos obtains the CSI samples on 35 channels in the 2.4 GHz and 5 GHz band using the FHP implemented at kernel level. Ferrín's FHP is implemented in userspace and works only in the 5 GHz band, i.e., with 24 channels. The packet loss in the 2.4 GHz band was about 90 %, resulting in the FHP being restarted too often. According to Ferrín, packet injection with the Linux 802.11n CSI tool in 2.4 GHz channels is an issue that has been reported several times, and factors such as interference with other signals or antenna positions could be ruled out. This limitation restricts the bandwidth that can be achieved with frequency hopping and thus the time resolution of the multipath profile.

INDFT Algorithm Settings

The gradient descent algorithm used to compute the multipath profile (see Algorithm 1) requires a proper step size γ to converge. A step size that is too big causes the minimum we want to find for the problem described in Eq. 2.22 to be missed and prevents the algorithm from converging, while a too small value increases the time and computational resources needed to find the minimum. The specified step size for Chronos is $\gamma = \frac{1}{\|\mathcal{F}\|_2}$. Ferrín, however, figured out that this step size is too big and proposed to set it to:

$$\gamma = \left(\frac{1}{\|\mathcal{F}\|_2} \right)^2, \tag{3.1}$$

where $\mathcal{F}$ is the complex Fourier matrix of the matrix formulation of the DFT. The author of Chronos confirmed this typo, and therefore it is suggested to use the step size in Eq. 3.1.

The size of the Fourier matrix $\mathcal{F}$ depends on the number of used channels and the length and resolution of the multipath profile: the length defines the largest ToF (i.e., distance) that can be obtained from a multipath profile and its resolution the granularity at which the ToF can be estimated. Vasisht et al. [12] state that they measure the ToF with sub-nanosecond accuracy, but they do not specify a concrete value for the time resolution or how to select it accordingly. Therefore, Ferrín calculates the resolution from the bandwidth spanned by the 24 channels in the 5 GHz band (i.e., 645 MHz); the resulting time resolution of 1.55 ns corresponds to a spatial resolution of 46.5 cm. Since using the squared CSI as input for the INDFT algorithm results in a halving of the values on the multipath profile's time axis, the resolution increases to 0.78 ns (i.e., 23.25 cm).

Experimental Evaluation

According to Vasisht et al. [12], there is a constant additive offset to the ToF introduced by the hardware that can be fixed with a one-time calibration of a WiFi card. In an experimental evaluation, Ferrín identified that such an additional delay could be attributed to the hardware. Nevertheless, it is neither constant nor does it fit linear regression or any other pattern. Since it is unknown where this delay exactly comes from and how it can be removed, the distance estimation accuracy is limited. Overall, Ferrín achieved a median distance accuracy of 96.9 cm in her experiments: some results were close to Chronos' accuracy of 14.1 cm; others had an error of almost 2 m.

3.2 Frequency Hopping Approaches

Chronos is not the only system that exploits CSI measurements across multiple frequencies to increase the accuracy of WiFi-based indoor localization. In this section, we describe three different approaches that benefit from combining measurements in various channels. Furthermore, we also discuss the limitations of frequency hopping for localization applications.

ToneTrack. Xiong et al. [65] combine the CSI of several WiFi channels to obtain the ToA for a signal sent from a client (transmitter) to a pair of fully synchronized APs (receivers). In order to get the ToA, the combined CSI is fed into the super-resolution MUSIC algorithm, and the TDoA is calculated across pairs of APs to finally get the client location. ToneTrack was implemented on the Rice WARP [66] radio platform, and it was demonstrated that channel combinations could be used to increase the localization accuracy. With the CSI of a single 20 MHz channel in the 2.4 GHz band, the median error in an office environment was about 1.9 m; with the data of three channels, the median error was reduced to 0.9 m.

Splicer. Xie et al. [22] use CSI measurements across multiple channels to obtain an accurate power delay profile describing the multipath dynamics in the wireless channel. CSI from adjacent channels is spliced together to extend the bandwidth and increase the power delay profile's time resolution. The challenge here is to account for various hardware errors (e.g., phase offsets like the CFO) before splicing the data. Splicer is implemented on an Atheros 9580 NIC connected to a laptop that acts as AP. The localization itself is based on CUPID [67], which can locate a target with a single AP. Therefore, the power of the LOS signal retrieved from the power delay profile is used to estimate the distance, and the direction is obtained by applying the MUSIC algorithm on the CSI. Using the spliced CSI instead of the CSI of a single channel, the accuracy could be increased by 71 %, and the median localization error is 0.95 m.

System by Chen et al. [68]. Chen et al. exploit CSI measurements at a fixed number of frequencies for a fingerprinting-based indoor positioning system. CSI samples are sanitized before they are saved in the fingerprint database to avoid errors due to synchronization issues and inference. To locate a target, the authors calculate the Time-Reversal Resonating Strength (TRRS), which describes the similarity between the live measurements and the stored fingerprints. The system is tested on two Universal Software Radio Peripherals (USRPs) N210, each equipped with one antenna. Instead of using standardized WiFi channels, they perform their measurements at frequencies ranging from 4.9 GHz to 5.9 GHz with a fixed step size of 8.28 MHz (i.e., spanning an effective bandwidth of 1 GHz). The system was tested in an office, and the grid points at which the fingerprints were obtained were 5 cm apart. Such a fine-grained grid resolution enables localization results with centimeter-level accuracy even in NLOS environments.

Limitations. Frequency hopping is very effective in emulating a higher channel bandwidth for precise localization. Nevertheless, regular network communication is limited. It has to be interrupted to switch between the different channels: the more devices that need to be located, the less time for normal network traffic. Hence, reusing existing infrastructure for such systems is only suitable to a limited extent. Xie et al., therefore, suggest using less active or dedicated APs to perform frequency hopping-based localization. Another limitation is that wireless channels are time-varying; the CSI is only stable within a certain time, known as coherence time. This coherence time is related to the Doppler shift, i.e., the higher a transmitter's moving speed, the smaller the coherence time [22]. Therefore, when designing an application based on frequency hopping, one has to consider the coherence time. Vasisht et al. state that Chronos hops through all channels in about 84 ms and that this is within the coherence time in typical indoor environments: according to [69], the coherence time is several hundred milliseconds.

3.3 CSI-based Applications on the Raspberry Pi

The Raspberry Pi is a low-cost single-board computer and a widely used IoT device. Since CSI extraction on the Raspberry Pi has only recently become possible, there are not that many works that have used this hardware for CSI-based localization or sensing. In this section, we summarize some of the existing publications in this area.

Path loss-based ranging. In prior work [51], we evaluated path loss-based distance estimation using CSI amplitudes on the Raspberry Pi 4B. We calculated the effective CSI (i.e., the weighted sum of CSI amplitudes for OFDM data subcarriers) and trained the path loss-based model proposed by Wu et al. [17] to estimate the distance between a transmitter-receiver pair. Since CSI amplitudes are affected by the AGC, we use the RSSI to calibrate the amplitudes as suggested by Gao et al. [33]. The system was tested outdoors for a range of 1-40 m, using 20 MHz channels in the 2.4 GHz band, and the resulting median error was about 2 m. In comparison, the median error for the RSSI-based path loss model (see Eq. 2.3) was 2.87 m. Due to the RSSI-based calibration, the distribution of our effective CSI training samples was similar to that of the RSSI samples, including measurement uncertainties. Hence, RSSI-based calibration, while necessary, is not ideal for this approach; CSI amplitudes measured on the Raspberry Pi are better suited for approaches that do not rely solely on the path loss model.

Fingerprinting-based localization. Voggu et al. [34] used the Raspberry Pi 3B+ for fingerprinting-based localization using CSI amplitudes. Fingerprints are obtained for LOS and NLOS scenarios, and before storing the data in a database, the CSI samples are de-noised. The Raspberry Pi is used to obtain the CSI for WiFi frames transmitted by an AP in a 40 MHz channel. They trained four classifiers (Gaussian Naive Bayes Classifier, Support Vector Classifier, Decision Tree Classifier, and Random Forest Classifier), each with three different resolutions (10 cm, 20 cm, and 40 cm) to compare their accuracy. In LOS scenarios and with a resolution of 10 cm, the Support Vector Classifier and the Random Forest Classifier provided the best accuracy with a mean error of 3.43 cm and 3.18 cm, respectively. For NLOS experiments, they observed that the accuracy was even higher than for LOS measurements (i.e., 0.42 cm and 0.65 cm, respectively). Voggu et al. argue that NLOS fingerprints are more distinct than those in LOS scenarios; therefore, it is easier to distinguish between different locations.

Human activity recognition. Forbes et al. [31] evaluated the suitability of the Raspberry Pi 4B for human activity recognition. To do so, they applied a deep learning approach (i.e., the DeepConvLSTM model) that uses CSI amplitudes of 80 MHz channels in the 5 GHz band as features. Since the wireless channel is affected differently by different activities, the CSI amplitudes effectively reflect these disturbances. Their system is capable of distinguishing typical home activities like drinking, cooking, and washing dishes. The Raspberry Pi is not directly involved in the wireless communication, as it passively measures the CSI for WiFi packets sent by an AP. The experimental evaluation showed that the activity recognition accuracy is 92 % and thus that the Raspberry Pi is suitable for CSI-based activity recognition applications.

Limitations. Previous work already successfully demonstrated that the Raspberry Pi is capable of CSI-based ranging and sensing despite having only a single PCB antenna. However, further testing and research is necessary to find certain bottlenecks and limitations. For instance, to the best of our knowledge, power consumption or hardware imperfections related to CSI measurements on the Raspberry Pi have not yet been investigated. Since enabling CSI extraction on the Raspberry Pi has the drawback of not allowing regular WiFi communication, it is better suited for localization and sensing applications that rely on passive CSI measurements. More details on obtaining the CSI using the Nexmon firmware patch are described in the next section.

Chapter 4

Enabling Chronos on the Raspberry Pi 4B

In this chapter, we describe our implementation of Chronos on the Raspberry Pi 4B. We begin with an introduction of the Nexmon CSI Extraction Tool in Section 4.1, followed by an overview of our system architecture in Section 4.2. In Section 4.3 and 4.4, we summarize the implementation of frequency hopping and frame injection in restricted 5 GHz channels in the WiFi chip firmware, respectively. Furthermore, we describe the implementation of a demonstrator and discuss its limitations in Section 4.5.

4.1 Nexmon CSI Extraction Tool

The Nexmon project [36] is a C-based firmware patching framework for Broadcom/Cypress FullMAC WiFi chips that allows writing WiFi firmware patches for various devices (e.g., smartphones, routers, and IoT devices). A complete list of supported chipsets and devices can be found in the Nexmon GitHub repository [70]. Firmware patches can be used, for instance, to put devices into monitor mode, implement frame injection, use the WiFi chip as software-defined radio, and access the CSI for received WiFi frames; the latter already exists as Nexmon CSI Extraction Tool [28].

Overview

The Nexmon CSI Extractor is an extension to the Nexmon framework that supports CSI extraction on state-of-the-art hardware, such as the Raspberry Pi 3B+/4B, the Nexus 5/6P, and the Asus RT-AC86U. Table 4.1 provides a comparison of the Nexmon CSI Extractor to the Linux 802.11n CSI Tool and Atheros CSI Tool. Besides supporting modern IoT devices, the Nexmon CSI Extractor has the advantage of making the CSI of all data subcarriers available compared to the Linux CSI Tool; the latter provides only 30 subcarrier values. Furthermore, the highest supported channel bandwidth and the resolution of numbers (i.e., of the real and imaginary part of the CSI values) are also higher than for the other tools.

Tool	Devices	supp. Chipsets	max. BW	MIMO support (TX×RX antennas)	highest supp. Standard	max. number of SC	Resolution
Linux 802.11n CSI Tool	PCI	IWL5300	40MHz	3x3	11n	30	8 bit {int}
Atheros CSI Tool	Router, PCIE	AR9580, AR9590 AR9344, QCA9558	40MHz	3x3	11n	114	10 bit {int}
Nexmon CSI Extraction Tool	Router, PCIE (e.g., Asus RT-AC86U)	bcm43[65, 66]	80MHz	4x4	11ac	242	12 bit {float}
	Smartphone, IoT (e.g., Nexus 5/6P, Raspberry Pi 3B+/4B)	bcm43[39, 58, 455]	80MHz	1x1	11ac	242	14 bit {int} or 10 bit {float}

Table 4.1: Comparison of CSI extraction tools, showing the supported WiFi chipsets, the maximum channel bandwidth (BW), the number of supported MIMO streams, the highest supported IEEE 802.11 standard, the maximum number of available subcarrier (SC) values, and the resolution of numbers (int = signed integer, float = signed floating mantissa) [28].

On the Raspberry Pi 4B, for example, CSI extraction is possible for channel bandwidths of 20, 40, and 80 MHz, and the real and imaginary parts of the CSI are each represented as signed 14 bit integer values. However, the Raspberry Pi supports only one spatial stream due to the single PCB antenna (i.e., TX×RX streams is 1×1).

CSI Extraction Patch

We briefly describe the receive path of an incoming WiFi frame and how the CSI for that frame is extracted to give the reader an overview of the CSI extraction process. For details, we refer to Gringoli et al.'s paper [28], where the Nexmon CSI Extraction Tool is introduced.

The receive path. The supported FullMAC WiFi chipsets (e.g., the `BCM43455c0` and `BCM4339`) consist of a radio front-end (i.e., the physical interface (PHY)), the D11 MAC core which executes the microcode (i.e., ucode), and the FullMAC ARM CPU which runs the firmware. When the reception of a WiFi frame is completed, the D11 MAC core reads information about this frame (e.g., bandwidth and number of spatial streams) from PHY registers in the receiver's radio front-end and stores it in a so-called receive header. This information is forwarded to the FullMAC core, which decides what to do depending on the frame type. If it is a data frame, the FullMAC core checks if the frame can be accepted (e.g., if it was sent by an accepted AP) and performs actions to submit it to the host (e.g., the Raspberry Pi's host CPU running a Linux kernel) via the kernel module.

Bringing CSI to the userspace. To make CSI available at the host, the Nexmon CSI Extractor patches the ucode and firmware of the WiFi chip. The ucode patch reads the CSI from the PHY tables and embeds it into multiple receive headers, since it does not fit into a single one. These packets are collected by the FullMAC core, which runs the patched firmware. When extracting the CSI, the payload of the received WiFi frame gets corrupted, and, so far, there is no solution that fixes this issue. Therefore, enabling

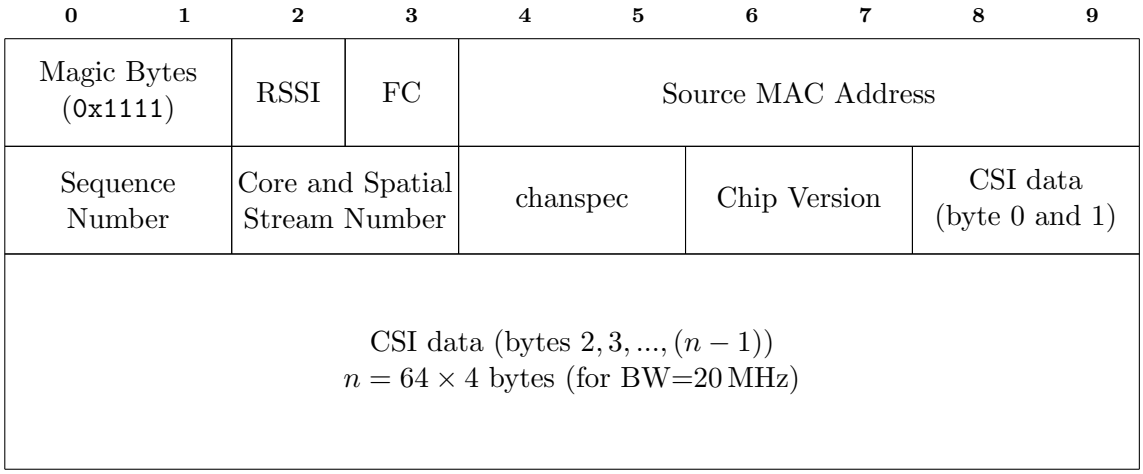


Figure 4.1: *The modified Nexmon UDP packet payload.* It starts with two magic bytes, followed by the RSSI (1 byte) and the FC field value (1 byte). After that, the source MAC address (6 bytes) and the sequence number (2 bytes) of the received WiFi frame are added. The core and spatial stream number are encoded in 2 bytes and the 2 byte long chanspec contains the channel number and bandwidth. The number of CSI values depends on the channel bandwidth and they are added after the chip version (2 bytes) [51].

CSI extraction comes with the drawback that regular WiFi communication is no longer possible. When the FullMAC core has received all parts of the CSI, the data is forwarded in a UDP packet to the host's userspace. There, it can be captured with any program that can analyze the network traffic (e.g., *tcpdump* [71]) – the source address of the UDP packet is `10.10.10.10`, the destination address `255.255.255.255`, and the destination port 5500.

Usage and UDP Packet Format

The detailed setup instructions and usage description can be found in the GitHub repository of the Nexmon CSI Extraction Tool [72]. For the implementation of our work, we also wanted to have the RSSI available for each received frame, which the Nexmon CSI Extractor patch does not provide. Therefore, we use a slightly different firmware patch [73] that extends the Nexmon CSI Extractor by obtaining RSSI and FC field values and adding them to the UDP packet payload. Fig. 4.1 shows the structure of this modified payload, which also contains the source MAC address of the WiFi frame, the channel specification (short *chanspec*), the chip version, and the CSI. The channel specification consists of the number and bandwidth of the WiFi channel in which the frame was received. For the Raspberry Pi 4B (i.e., the `BCM43455c0`), the CSI of each subcarrier is a complex value consisting of an int16 real and int16 imaginary part. The number of CSI data in the UDP payload depends on the channel bandwidth (i.e., the number of available subcarrier values). The GitHub repository of the Nexmon CSI Extractor contains MATLAB scripts demonstrating the correct conversion of the values in the payload.

4.2 System Architecture

This section gives an overview of our system architecture, which is shown in Fig. 4.2. The figure shows the different modules that we used to implement Chronos on the Raspberry Pi 4B. Each module is described separately; the implementation of the Chronos firmware patch is explained in detail in the next section (Section 4.3).

Raspberry Pi 4B

The Raspberry Pi 4 Model B [74], the latest model in the Raspberry Pi series (released 2019), uses a 1.5 GHz 64 bit quad-core Cortex-A72 CPU and is available with 1, 2, 4, or 8 GB of RAM. As mentioned earlier in this work, it embeds a `BCM43455c0` Broadcom FullMAC WiFi chipset, supporting 2.4 and 5 GHz IEEE 802.11b/g/n/ac standards, with an integrated Bluetooth transceiver for wireless communication. For wired communication, Gigabit Ethernet and USB (2.0 and 3.0) interfaces are available. The single-board computer provides a micro SD card slot for loading the OS and for data storage. In our work, we use devices with 4 GB of RAM, on which we run the Raspberry Pi OS (kernel 4.19), a Debian-based operating system. The OS kernel version is important as the Nexmon CSI Extractor is currently only compatible with versions 4.19 and 5.4.

nexutil

The Nexmon project provides a program called *nexutil* which can be used to configure settings like the operating channel and its bandwidth for the CSI extraction at runtime. *nexutil* uses Input/Output Control (IOCTL) system calls to communicate with the firmware. An IOCTL contains a number to identify the command, a pointer to a data buffer, and the length of this data; it can be used to either set or get data in/from the firmware. The `wlc_ioctl_hook` function is provided by Nexmon and allows the handling of custom IOCTLs. The Nexmon CSI Extractor uses these system calls, for example, to set filters such as a MAC address filter to extract CSI only for WiFi frames containing a specific source MAC address and a WiFi frame filter to obtain the CSI only for defined frame types (i.e., frames with a specific Frame Control (FC) field). Additional IOCTL commands with different functionalities can be added by modifying the source code of the firmware patch accordingly.

Chronos Firmware Patch

The Chronos firmware patch consists of two parts: the FHP functionality and the implementation of custom IOCTL commands to control the FHP. We used the Nexmon firmware patching framework to implement this module and changed the modified Nexmon CSI Extractor (i.e., [73]) accordingly. With our custom IOCTLs, we can control the FHP and adjust specific settings in the firmware (e.g., start/stop the channel hopping and set the transmission power). These IOCTL commands are communicated from the host's userspace to the WiFi chip's firmware using *nexutil*. A detailed description of this module is provided in Section 4.3.

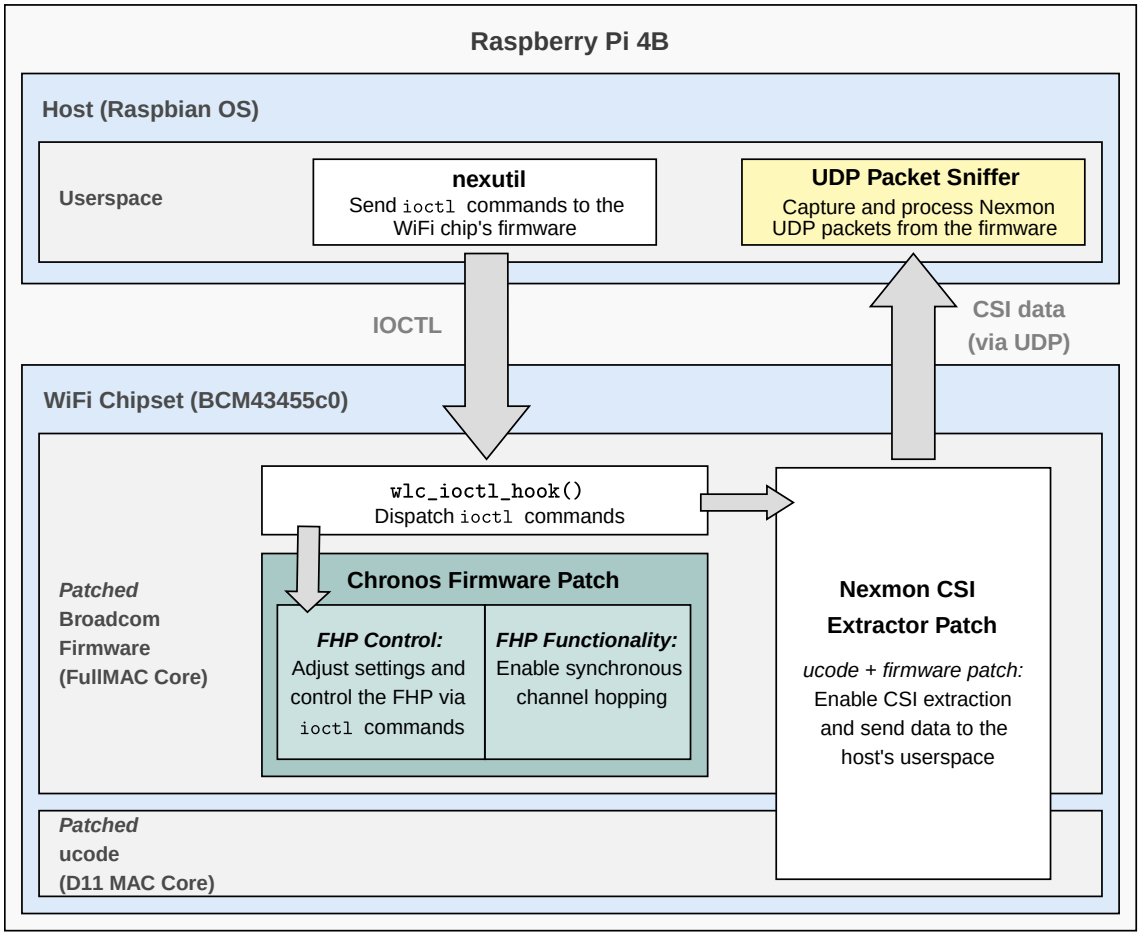


Figure 4.2: *System architecture of our Chronos implementation on the Rasbperry Pi 4B.* The Chronos firmware patch represents the implementation of the FHP in the FullMAC core. Custom IOCTL commands are used to configure and control the FHP and the CSI extraction (i.e., the Nexmon CSI Extraction Tool); *nexutil* is used to send these IOCTLs from the host to the WiFi chip. The UDP packet sniffer captures and processes the UDP packets containing the CSI.

Figure 4.3: *Generic IEEE 802.11 data frame format.* The FC field defines the frame type; the number and function of MAC addresses depends on the deployed network type. The frame body can have a maximum size of 2312 bytes and after that a 32 bit CRC is added in the FCS field [77, 78].

UDP Packet Sniffer

The UDP packet sniffer can be any program capable of capturing network traffic (e.g., tcpdump or Wireshark [75]). We implemented such a packet sniffer in Python that uses raw sockets to capture Nexmon UDP packets and process them as required. For example, in our experimental setup, we used the Python script to forward the UDP packets from the server and the client Raspberry Pis to a laptop, where we collected the data from both devices using Wireshark. Our demonstrator, described in Section 4.5, also uses this Python script to obtain the UDP packets.

4.3 Chronos Firmware Patch

In this section, we focus on the FHP implementation on the Raspberry Pi; the frequency hopping procedure has already been described in Section 2.3.1. We divided the description of our Chronos firmware patch (i.e., the FHP implementation) into two parts: the FHP functionality and the FHP control. We fully integrated the FHP in the firmware of the FullMAC core; it was implemented in C and is illustrated in Fig. 2.5.

The work of [28, 37, 76] helped us to find useful *firmware functions* (i.e., those already available in the unpatched Broadcom firmware) and *helper functions* (i.e., those provided by the Nexmon framework) that we could use for our implementation. However, some of the firmware functions were not directly accessible on the Raspberry Pi, and therefore we had to reverse engineer their address information from the firmware binary. More information about the reverse engineering process and a list of found functions, including their addresses, can be found in Appendix A.

FHP Functionality

In order to implement the FHP, we need to consider three things: the transmission of REQ and ACK packets inside the firmware, setting appropriate transmission parameters, and using timers to check for timeouts during the hopping.

Transmission of REQ/ACK packets

REQ and ACK packets sent by the server and the client, respectively, are both data frames. The generic format of such a data frame is shown in Fig. 4.3. The FC field indicates the WiFi frame type (i.e., whether it is a control, management, or data frame) and provides control information (e.g., fragmentation and privacy information). The number of addresses depends on the context: address 1 contains the receiver MAC address, address 2 that of the transmitter, address 3 is used for filtering by APs or distribution system, and address 4 is optional as only wireless bridges use it. The body of a WiFi data frame can contain up to 2312 bytes, and a Cyclic Redundancy Check (CRC) is added at the end for error detection reasons [77, 78]. In our implementation, the data transmitted in the frame body is not relevant since it gets destroyed during the CSI extraction. However, we need to set the receiver and transmitter MAC addresses (i.e., address 1 and 2) in the WiFi frames correctly since we apply the Nexmon CSI Extractor MAC address filter; we want to extract CSI only for frames from one specific device (i.e., the server or client) while the FHP runs. Nevertheless, the most important thing is that we can trigger the transmission of WiFi frames in the correct channel within the firmware.

Frame injection. Nexmon provides helper functions that we used for our implementation. For frame injection at the firmware level, we use their `sendframe` function. Two versions of this function are available: one calls the firmware function `wlc_sendctl`, which appends the required header structure to the frame, enqueues it in the TX FIFO buffer, and triggers its transmission; the other calls lower-level firmware functions to achieve the same functionality with reduced complexity. Since time is a critical factor for frequency hopping-based measurements, we tested both versions to select the fastest option. However, while hopping through a set of channels for a certain amount of time, we could not observe that one is faster than the other. For our experiments, we kept the second version of the `sendframe` function.

Switching channels. In order to keep track of the current state of the FHP, we use a structure that contains information like the role of the device (i.e., server or client), a list of channels that defines the hopping order, and the channel currently in use. If we want to hop from one channel to another, we need to change the channel specification (i.e., channel number and bandwidth) accordingly. Therefore, we use Nexmon's `set_chanspec` function, which does precisely that, and we update the parameters in our structure. Both server and client know when they reached the last channel in the list and switch back to the first.

Transmission Settings

If a REQ or ACK packet gets lost or its transmission is delayed for too long, the server and client experience a timeout, and the FHP is restarted, as shown in Fig. 2.5. Thus, both reset their operating channel to the first one specified in the hopping order and must retransmit all packets to obtain the CSI of a complete (i.e., successful) frequency hopping round. Packet loss occurs, for example, due to interference with other signals, too many errors in the CRC, or too weak signal strength. To guarantee reliable server-client communication (i.e., a correct operation of the FHP with as few restarts as possible), we

can adjust some transmission parameters: the Modulation and Coding Scheme (MCS), retransmission limits for transmitted frames, and the transmission power.

MCS. The IEEE 802.11n standard specifies Modulation and Coding Schemes defining different combinations of modulation types (i.e., BPSK, QPSK, 16-QAM, or 64-QAM), coding rates (i.e., 1/2, 3/4, 2/3, or 5/6), and number of MIMO (spatial) streams. Each combination is assigned to a unique MCS index (counting starts at zero): MCS index 3, for instance, indicates that there is one spatial stream and that 16-QAM modulation and a coding rate of 1/2 are used. The selected MCS and the channel bandwidth impact the achievable data rate; the higher the MCS index, the higher the number of spatial streams, and thus the higher the data rate [38]. As we do not have much data to transmit, we do not need the highest data rates. Therefore, we decided to use the same MCS as Ferrín [27] in her work, i.e., MCS index 3. We also tested other MCS settings, but the higher the MCS index, the less reliable the communication (i.e., the higher the packet loss due to CRC errors), which caused restarting the FHP too often.

Retransmission limits. If a WiFi transmitter requires an acknowledgement for its transmitted frame but does not receive one, it retransmits this frame a certain number of times. This number is defined by the Short Retry Limit (SRL) for short and the Long Retry Limit (LRL) for long frames (i.e., frames shorter or longer than the request-to-send threshold), which by default are 6 and 7, respectively. Since the FHP uses data frames to acknowledge the reception of REQ and ACK packets and not the standardized IEEE 802.11 ACK frame, both server and client transmit their packets multiple times. Using the standardized ACK frame format did not work in combination with CSI extraction and frequency hopping. The retransmissions were interfering with the FHP, so we disabled them by setting the SRL and LRL to 1 using IOCTLs: the `WLC_SET_SRL` respectively `WLC_SET_LRL` command is provided by the firmware (i.e., no custom commands). To trigger these IOCTLs within the firmware, we called the `set_intioctl` helper function.

Transmission Power. We have tried to adjust the transmission power settings using the `qtxpower` iovar as described in [37]. The value of `qtxpower` (i.e., the transmission power in qdBm) is internally translated into a power index used by the hardware to set the amplifier gains accordingly. However, in our tests, we did not observe any changes in RSSI measured for frames transmitted with different `qtxpower` values. Schulz et al. [76] showed that the transmission power could also be adjusted by directly setting the power index, e.g., with the `exp_set_gains_by_index` helper function. This way, we do not know which power index leads to which transmission power (in mW or dBm), only that the lower the power index, the higher the transmission power. As we do not need to know the transmission power value in, e.g., milliwatts, we can use this option to vary the transmission power, which worked well.

Timing

The server and the client must be able to detect communication timeouts so that both can restart the FHP and switch back to the first channel. For this purpose, we use the hardware timer of the D11 core (`hwtimer` or `hrt`), which counts in microseconds steps.

We need to deactivate the minimum power consumption, as this timer is only available when the D11 core is active. The timer is initialized with the `wlc_hwtimer_alloc_timeout` function, and timeouts are added with `wlc_hwtimer_add_timeout`, resulting in the execution of a callback function after a defined delay. To delete and free a timeout, we call `wlc_hwtimer_del_timeout` and `wlc_hwtimer_free_timeout`, respectively. We use three timeouts for the FHP implementation: one delaying the transmission of the first REQ sent by the server, another to schedule the transmission of REQ and ACK frames, and the last one is used to detect a timeout in case of an error during the frequency hopping (i.e., when the server or client does not respond). In our experiments, we evaluate the impact of the frequency hopping speed (i.e., different delays to schedule frame transmission). Therefore, the selection of appropriate timeouts is further discussed in Chapter 5.

FHP Control

We implemented several custom IOCTL commands in the firmware to allow controlling the FHP (e.g., starting and stopping the FHP) and the transmission settings (e.g., transmission power) from the host's userspace. The most relevant functionalities of these IOCTLs are listed below:

- **Role definition:** define a device either as a server or client;
- **Start/Stop:** start and stop the frequency hopping;
- **Channel selection:** select a set of channels for the frequency hopping: either all 2.4 GHz channels (i.e., 13), all 5 GHz channels (i.e., 24), or the channels of both bands (i.e., 37);
- **Max. number of hops:** if specified, the server stops the FHP automatically after the defined number of successful frequency hopping rounds;
- **Max. number of retries:** if specified, the server stops to restart the FHP after the defined number of retries;
- **MCS setting:** set the MCS to be used by its index (the Raspberry Pi supports MCS index 0 to 7);
- **Transmission power adjustment:** set the power index to a value between 10 and 125 to adjust the transmission power.

Limitations

Besides the limitation that regular WiFi communication is not possible on the Raspberry Pi when CSI extraction is active (see Section 4.1), we encountered two further critical limitations while implementing and testing the Chronos firmware patch. First, we could not inject frames in the DFS channels in the 5 GHz band (i.e., channel 52 to 140), and second, using the 2.4 GHz channels for the frequency hopping led to too many restarts of the FHP. Hence, we could not make reliable measurements to test Chronos. The problem and the solution regarding frame injection in arbitrary 5 GHz channels are explained in the next section (Section 4.4); the second is briefly described here.

If the FHP restarts too often, the problem is an unreliable communication between the server and the client, and the reasons can be manifold. Since the 2.4 GHz band is relatively crowded, interference with other signals could be one of them. However, we could rule this out since we observed the same behavior in the wireless and wired environments. We were able to attribute this issue to the time it takes from enqueuing to the final transmission of the packet. There is an arbitrary delay that we cannot control, and that seems to be higher when frames are transmitted in the 2.4 GHz band. Since we could not find a solution for this problem, we decided to use only 5 GHz channels for our experiments.

4.4 Enabling Transmissions in 5 GHz Channels

As mentioned in the previous section, we could not inject frames in some of the 5 GHz channels, namely the DFS channels. Without the DFS and the 2.4 GHz channels, we have only 9 channels available to measure the CSI, too few to obtain accurate distance estimations.

Firmware analysis. To overcome this problem, we analyzed the source code of the firmware to understand where the restrictions are set and how we could remove them. The firmware stores information about channel regulations depending on the regulatory domain in a structure: this structure contains, for example, the current country code, a list of radar-sensitive channels for the current locale, and a list of quiet channels (i.e., channels on which we cannot transmit because of radar sensitivity). Before transmitting a frame, the firmware checks if the quiet bit is set for the currently used channel, i.e., whether the channel can be used for transmission or not. If the quiet bit is set, the channel is muted, and transmission fails.

Firmware patching. We found two ways to bypass this limitation: the first is to unmute the currently used channel. The second is to not set the channels as radar sensitive during the initialization process at system startup. To apply the first solution, we call the function `wlc_bmac_mute` each time before enqueuing a frame in the TX FIFO. This function can either be used to mute or unmute a channel according to the provided parameters. Nevertheless, we applied the second solution in our FHP implementation by patching the function that tells the system at startup if a channel should be marked quiet (i.e., by setting the quiet bit) or not. The `wlc_radar_chanspec` function returns 1 (i.e., true) if the channel is included in the list of radar channels stored in the structure mentioned earlier, and 0 (i.e., false) otherwise. As shown in Listing 4.1, our patch overwrites the branch instruction that calls the original function with a jump to our target function (i.e., `wlc_radar_chanspec_hook`). `wlc_clr_quiet_chanspec` clears the quiet bit of the channel, and by returning 0, we indicate that the channel is not radar sensitive. We used the software reverse engineering tool Ghidra [79] to find the addresses of these functions. For more information about reverse engineering the firmware binary, see Appendix A.

```
1 int
2 wlc_radar_chanspec_hook(void* wlc_cmi, unsigned short chanspec)
3 {
4     wlc_clr_quiet_chanspec(wlc_cmi, chanspec);
5     return 0;
6 }
7 __attribute__((at(0x58a0e, "flashpatch", CHIP_VER_BCM43455c0,
      FW_VER_7_45_189)))
8 BLPatch(wlc_valid_chanspec_db, wlc_radar_chanspec_hook);
```

Listing 4.1: This firmware patch disables setting channels as radar sensitive during the initialization process at system startup.

4.5 Demonstrator

We implemented a demonstrator to test Chronos' performance for live measurements (i.e., online distance estimation). In this section, we briefly describe the demonstrator implementation and the bottlenecks we have found. The accuracy of Chronos on the Raspberry Pi will be evaluated in the next chapter (Chapter 5).

Implementation

The demonstrator is implemented according to the system architecture shown in Fig. 4.2, where the UDP packet sniffer is the userspace application of our demonstrator in this case. This application is implemented in Python, and its functionality depends on whether a device (i.e., the Raspberry Pi) is used as a server or a client; the role can be specified in a configuration file. As a client, the application must forward the Nexmon UDP packets containing the CSI of the received REQ to the server via an Ethernet connection. The server collects the client data and the UDP packets from its WiFi chip, and performs the distance estimation with the CSI of a complete frequency hopping round as described in Section 2.3. The userspace application is also responsible for initializing the FHP in the firmware with our custom IOCTLs using *nexutil*. We implemented a graphical interface that shows the estimated distances between a server and several clients. It is a simple web-application hosted by the server and shown in Fig. 4.4. The corrected distance shown in the table is the estimated distance after subtracting the constant additive offset introduced by the hardware, which can be defined in the configuration file.

Limitations

The time to compute the distance between a server and a client was about 1-2 s; too slow for a real-time tracking system, but acceptable for WiFi geo-fencing, home occupancy measurements, or finding devices. The bottlenecks in distance estimation are that we average the CSI of 5 frequency hopping rounds to resolve the error we get because simultaneous CSI measurements at the server and the client are not possible, and that the algorithm computing the multipath profiles is also not very fast. The latter is not an issue of our implementation but of Algorithm 1 itself. Vasisht et al. [12] mention that their implementation takes approximately 3.1 s until it converges. There are other algorithms available that can solve the L-1 minimization problem in Eq. 2.21 eventually faster (e.g.,

the SPGL-1 [80] or the in-crowd [81] algorithm). However, we did not evaluate that, as this was out of the scope of our work.

The most significant limitation, however, is that we cannot use regular WiFi communication (see Section 4.1) to exchange the CSI directly in the REQ and ACK frames. Therefore, we always need Ethernet cables to connect the devices, making Chronos on the Raspberry Pi (or any other device supported by the Nexmon CSI Extractor) unsuitable for real applications. This problem will remain until CSI extraction and regular communication are possible at the same time. However, this is something that is up to the Nexmon developers and also beyond the scope of this work.

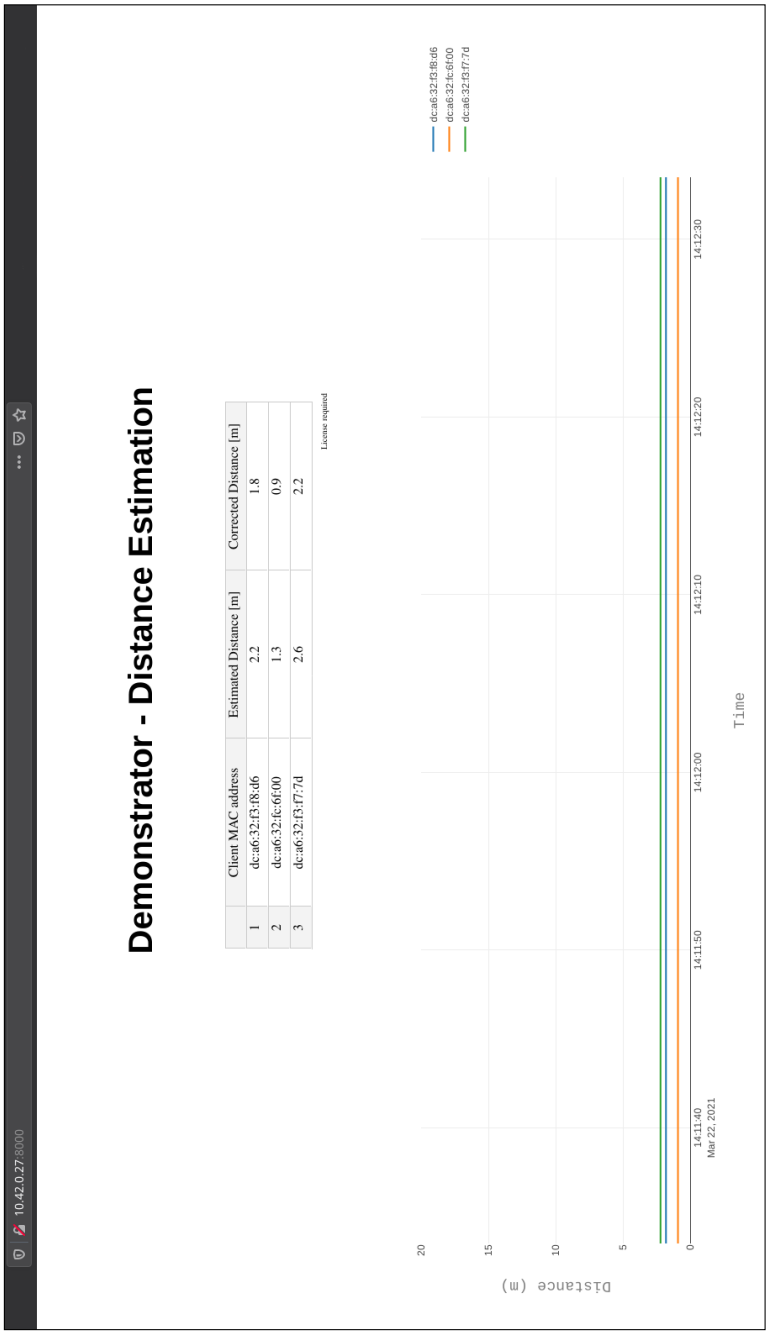


Figure 4.4: *User interface of the demonstrator.* A Raspberry Pi configured as server calculates its distance to several clients (three in this case) and hosts a web-application that displays these distances in real-time. The corrected distance is the estimated distance after subtracting a constant offset (0.4 m in this case).

Chapter 5

Experimental Evaluation

This chapter summarizes the experimental evaluation of Chronos on the Raspberry Pi. First, we describe our experimental setup for evaluating the distance estimation accuracy in Section 5.1. Then, we present our results from measurements in wired and wireless environments in Section 5.2 and 5.3, respectively. We evaluate the impact of the frequency hopping speed (i.e., the time it takes to hop through all the channels), the time resolution of the multipath profiles, and the transmission power for each environment. Finally, we discuss our results in Section 5.4.

5.1 Experimental Setup

In this section, we describe the setup we used in our experiments. We start with the description of the hardware setup and briefly summarize the data processing required to perform the distance estimation with the CSI measurements. Furthermore, we demonstrate the impact of the algorithm settings (i.e., resolution of the time vector and sparsity parameter) on ideal data.

5.1.1 Hardware Setup

We evaluated the distance estimation accuracy of Chronos on Raspberry Pis by measuring the CSI according to the FHP (see Fig. 2.5) between two devices; a server and a client. The system architecture of the Raspberry Pis is shown in Fig. 4.2. The UDP packet sniffer is, in this case, a Python script that uses raw sockets to capture the Nexmon UDP packets and forward them from the Raspberry Pis to a laptop. We use the laptop to control the measurements via SSH and collect the packets containing the CSI with Wireshark. The captured traffic is saved in *pcap*-files to evaluate the experiments offline. Since we cannot use regular WiFi communication to control the Raspberry Pis and exchange the CSI (see Section 4.1), we connect the devices with Ethernet cables. To test the effects of multipath on the ranging results, we conduct the experiments for WiFi communication in wired (multipath-free) and wireless (multipath) environments.

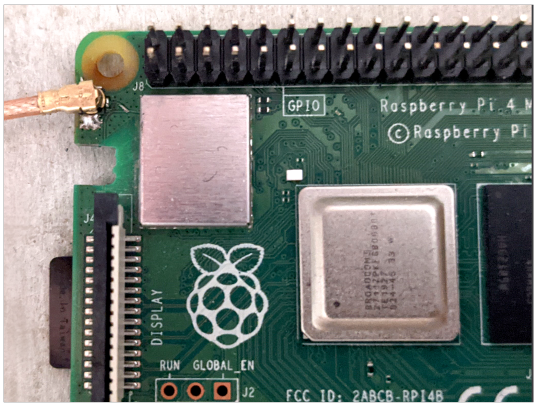


Figure 5.1: *Raspberry Pi with U.FL connector and removed PCB antenna.* We used this modification to enable CSI measurements in a wired (i.e., multipath-free) environment.

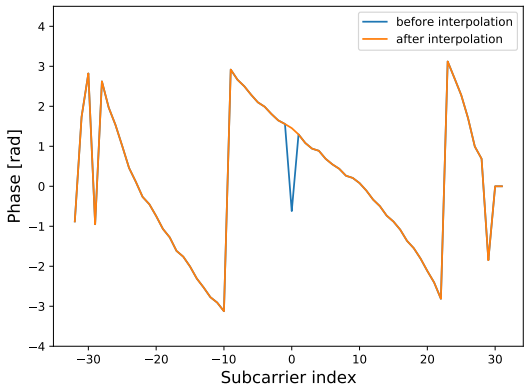


Figure 5.2: *CSI interpolation example.* The blue line shows the CSI phase of a 20 MHz channel before and the orange after the interpolation of subcarrier-0.

Wired setup. Instead of transmitting WiFi frames over the air using the PCB antenna, we want to send them via coaxial cables from one device to another. Therefore, we soldered U.FL connectors on two Raspberry Pis so we can connect them with the cables. We placed each Raspberry Pi in a box that we shielded with aluminum foil and cut away the PCB antenna to avoid interference. The Raspberry Pi modifications are visible in Fig. 5.1. To perform the experiments for different distances, we connected cables of different lengths. We also put two 20 dB attenuators [82] between the devices to avoid damaging the hardware with too high transmission power.

Wireless setup. The wireless experiments were conducted in a 5.8×8.8 m seminar room and a 3 m wide corridor, where we mounted two (unmodified) Raspberry Pis on tripods at the height of 1.5 m. The server was placed at a fixed position and the client was moved to vary the distance. We tested only LOS scenarios, i.e., there were no obstacles between the two devices, and the devices were placed facing each other. Except for one person who performed the experiments, no one else was in the rooms. Therefore, our results represent the accuracy in (almost) static environments.

As described in Section 4.3, we could not use the 2.4 GHz channels for the frequency hopping. Therefore, our set of used channels comprises only the 24 channels in the 5 GHz band (see Fig. 2.1b); we start the FHP at channel 36 and hop from one channel to the other in ascending order (i.e., 36, 40, ..., 165). We restart the FHP after the CSI measurements in channel 165 are complete. If hopping through all channels and thus CSI measurements were successful, we call it a *complete frequency hopping round*. Hence, for each complete frequency hopping round, we get 24×2 UDP packets (i.e., 24 from the server and 24 from the client) containing the CSI. Since we use 20 MHz channels, we get the CSI of 64 subcarriers in each of them.

5.1.2 Data Processing

The data processing and evaluation are implemented in Python according to the description of Chronos in Section 2.3. Therefore, we only briefly summarize the main steps for our offline evaluation. First, we read the CSI from the UDP packets stored in the *pcap*-files and use the Cubic spline interpolation to get the CSI of subcarrier-0 ($CSI_{i,0}$) for each of them. Then, we identify the CSI of a complete frequency hopping round and multiply $CSI_{i,0}$ of channel i from the server and the client as shown in Eq. 2.17 to obtain the vector $\tilde{\mathbf{h}}^{\mathbf{2}}$ (i.e., the input vector for the INDFT algorithm). Instead of directly using $\tilde{\mathbf{h}}^{\mathbf{2}}$ of a single frequency hopping round to compute the multipath profile, we average the data per channel for 5 of these vectors, i.e., we average the data of 5 frequency hopping rounds. We do this to resolve the phase error introduced by the delayed CSI measurements at the server and the client side (see Section 2.3.3). Finally, using the averaged data as input to Algorithm 1, we compute the multipath profile to get the ToF and calculate the distance with Eq. 2.8. In our evaluation, we consider as wave velocity the speed of light c and $\frac{2}{3} \cdot c$ for the wireless and wired environments, respectively. We also divide the results by two since we use the squared CSI as input.

5.1.3 Evaluation

To investigate the accuracy of Chronos on the Raspberry Pi, we evaluate the impact of three factors: the frequency hopping speed, time resolution settings for the multipath computation, and the transmission power. We evaluate the accuracy based on the distance errors (or ranging errors) in *absolute* numbers.

The Frequency Hopping Speed

By frequency hopping speed, we mean the time it takes to hop through all 24 channels in a complete frequency hopping round. Since non-simultaneous CSI measurements on the server and client side introduce an additional phase error, we want to investigate how it impacts the accuracy. In our experiments, we control the time it takes to measure the forward and reverse channel by adjusting the transmission delays of REQ and ACK packets with timeouts, as described in Section 4.3. We tested two delay settings for scheduling packet transmissions: one at millisecond level (i.e., 5 ms delay for REQ and 1 ms for ACK packets) and the other at microsecond level (i.e., 0.2 ms for both packets). We refer to the first as *slow hopping* and the second as *fast hopping*. The different transmission delays for the slow hopping result from the attempt to make the FHP also work reliably in the 2.4 GHz band (see Section 4.3). After the first round of experiments with slow hopping, however, we decided to use equal transmission delays for both packets for the fast hopping.

We measured the approximate time needed to hop through all channels 1000 times. With the *slow hopping* settings, a complete frequency hopping round takes approximately 240 ms, i.e., channels are switched every 10 ms. *Fast hopping* reduces the time to about 100 ms per hopping round, i.e., to 4.2 ms per channel; this is close to the original Chronos' implementation, which switches channel every 2-3 ms. Faster frequency hopping has unfortunately not been successful so far, i.e., we could not achieve faster channel switching than the 4.2 ms; the arbitrary delay from queuing the packet to its transmission could be the reason.

The Time Resolution

The time resolution is the granularity of the multipath profile's time vector. It is directly related to the resolution at which distances can be estimated and thus influences the ranging accuracy. Vasisht et al. [12] do not specify what time resolution they used. Therefore, we evaluate whether the resolution proposed by Ferrín (i.e., 1.55 ns [27]; see Section 3.1), who also used only 5 GHz channels, leads to accurate results or whether we should better choose a different time resolution. The *sparsity parameter* (we will continue to call it simply sparsity), which defines how many dominant paths we can observe in the multipath profiles (see Section 2.3.3), is also a crucial factor in obtaining accurate results in this context.

Testing ideal data. We tested the impact of time resolution and sparsity settings for ideal data generated for a distance of 12.01 m. The ideal phase values are calculated according to Eq. 2.13 for all the 5 GHz channels. With the ideal data, we computed the multipath profiles for 5 different time resolutions and 4 sparsity values. These multipath profiles are shown in Fig. 5.3; instead of the time, we directly plot the corresponding distances on the x-axis. We can observe that despite having ideal data, the selected time resolution matters. In the plot that shows the multipath profile for a time resolution of 1.33 ns (i.e., a distance resolution of 0.2 m), the first peak only appears at the correct position for the highest sparsity setting. For lower sparsity values, it appears too early, namely, 7.5 m before the expected distance; this comes from the frequency ambiguity mentioned in Section 2.3.3. Since the minimum frequency separation is 20 MHz in the 5 GHz band, the ambiguity corresponds to 15 m ($\frac{1}{20\,\text{MHz}} \cdot c$), which is twice the offset that we observe (i.e., $2 \cdot 7.5\,\text{m}$). We attribute this difference to the division of the estimated ToF by two. Using a higher time resolution (e.g., 0.67 ns) and an appropriate sparsity reduces the impact of this problem, at least for the ideal data. If the sparsity value is too high, no peaks would be visible, and thus the estimated distance would be 0 m.

The Transmission Power

As described in Section 2.2.3, the Packet Detection Delay (PDD) depends on the transmission power. Hence, a WiFi frame sent with higher power should be detected earlier than one sent with lower power, and thus the PDD is also expected to be smaller. The phase error introduced by the PDD should not influence Chronos' ranging accuracy since only the CSI of subcarrier-0 is used for the ToF estimation, which is not affected by the PDD (see Section 2.3.2). Nevertheless, we want to investigate Chronos' resistance against changes in transmission power. Therefore, we take measurements for different 5 power index settings (i.e., 20, 40, 60, 80, and 100) in the wired and wireless environment. Since we do not use RSSI or CSI amplitude information to obtain the ToF, our results directly represent the influence of the transmission power on the measured CSI phases.

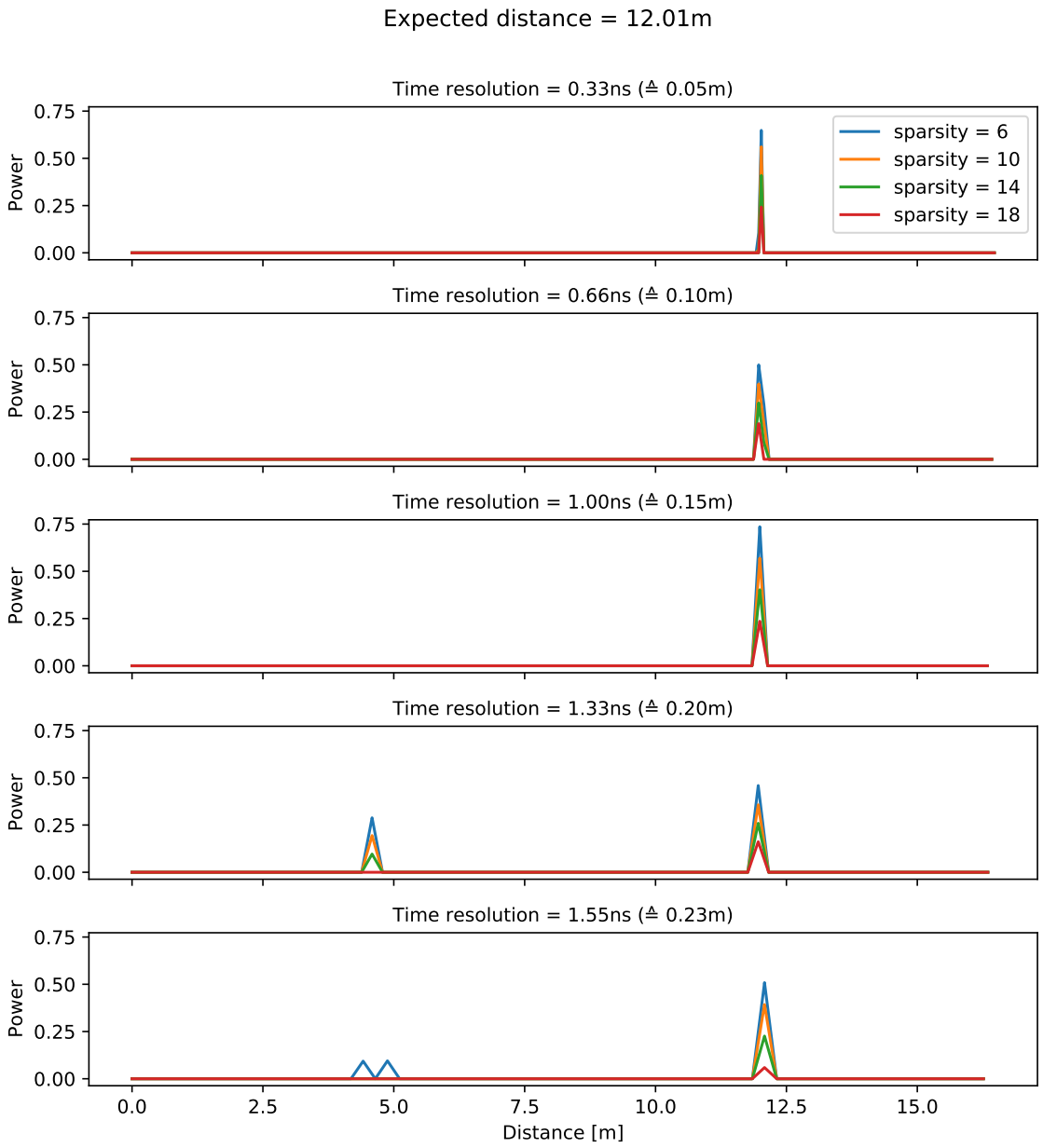


Figure 5.3: *Multipath profiles computed with different time resolution and sparsity settings for ideal phase values at a distance of* 12.01 m. Instead of time, the corresponding distances are visible on the x-axis of the multipath profiles. Non-ideal settings for time resolution and sparsity have a negative effect on the accuracy; for example, the first peak might appear too early (e.g., as in the plot with time resolution 1.33 ns) or there might be no peaks at all.

5.2 Evaluation Results for a Wired Environment

This section contains the evaluation results for our measurements in the wired (i.e., multipath-free environment). First, we evaluate the impact of the frequency hopping speed, as this is a crucial factor in obtaining accurate ranging results. Then, we test how the multipath profile's time resolution influences the accuracy, and last, we investigate whether the transmission power affects the distance estimation or not.

For all experiments in the wired setup, we measure the CSI for 2500 complete frequency hopping rounds. To this end, we used a custom IOCTL command that communicates this number to the Chronos firmware patch, which stops the FHP after reaching this number of complete hopping rounds. As described in Section 5.1.2, we average the data from 5 rounds, and thus, we can compute 500 multipath profiles in total. In this context, we call the distances estimated from the multipath profiles *samples*. Hence, we get $N = 500$ samples for each distance. While forwarding the UDP packets from the Raspberry Pis to the laptop, packets may get lost. Therefore, the data of an actually successful frequency hopping round becomes incomplete; this is the reason why, for example, in Fig. 5.4a the number of samples is not always exactly 500 (i.e., $N \approx 500$). However, it is not critical if one or two packets are lost, resulting in fewer samples.

Impact of Frequency Hopping Speed

The impact of the frequency hopping speed is investigated for measurements taken at three different distances: 1.91 m, 3.41 m, and 5.46 m. We tested both slow and fast hopping under the same conditions. For the multipath computation, we set the time resolution coherently to what proposed by Ferrín [27], i.e., to 1.55 ns. After empirically determining a good sparsity value for the data we use in this evaluation, we decided to set it to 5. This time resolution results in estimated distances that are a multiple of 15.49 cm.

Representation of results. In Fig. 5.4, we show histograms representing the distribution of estimated distances for slow and fast hopping. They show where and how often (i.e., in percent) first peaks appear at the same position in N multipath profiles. Instead of time, we directly plot the corresponding distances on the x-axis. For example, in Fig. 5.4d, the first peak appears at the same position in all of the 499 multipath profiles, i.e., 100 % of the estimated distances are 3.87 m; in Fig. 5.4b, approximately 70 % of the first peaks are at 2.17 m and 30 % at 2.32 m. If the histogram shows an entry at $d = 0$ m, the multipath profile was empty (i.e., there are no peaks). The figures on the left (i.e., subfigures a, c, and e) show the results for slow hopping and the figures on the right (i.e., subfigures b, d, and f) for fast hopping. Since we can only obtain discrete values for the estimated distances defined by the time vector, we call them classification results in the figures. The median and percentile ranging errors (i.e., median, 25th, 75th, and 90th percentile errors in absolute values) for both measurements are shown in Table 5.1.

Evaluation. From the histograms, we can see that the results for slow hopping are not as consistent as for fast hopping; the distribution of the estimated distances is wider, and the results less accurate. If we compare the ranging errors in the table, we can see

Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.91	0.72	0.57	0.88	1.91
3.41	0.77	0.46	0.77	3.26
5.46	4.14	0.74	4.68	5.46
Total	0.77	0.57	3.41	4.53

(a) *Slow hopping*

Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.91	0.26	0.26	0.41	0.41
3.41	0.46	0.46	0.46	0.46
5.46	0.28	0.28	0.28	5.00
Total	0.28	0.28	0.46	0.46

(b) *Fast hopping*

Table 5.1: Ranging errors for measurements in a *wired* environment for *(a) slow* and *(b) fast* hopping settings.

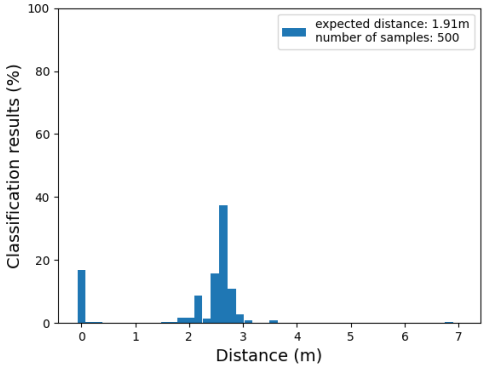
that the accuracy is much higher with fast hopping. The median error for the expected distance of 5.46 m, for example, decreased from 4.14 m to 0.28 m; the total median error calculated for the results of all 3 distances could be reduced from 0.77 m to 0.28 m, and all other percentile errors could be reduced as well. These results meet our expectations, as a faster hopping speed should lead to a smaller additional phase error and thus to more accurate ranging results. We continue to use the fast hopping settings for further experiments.

Impact of Time Resolution Settings

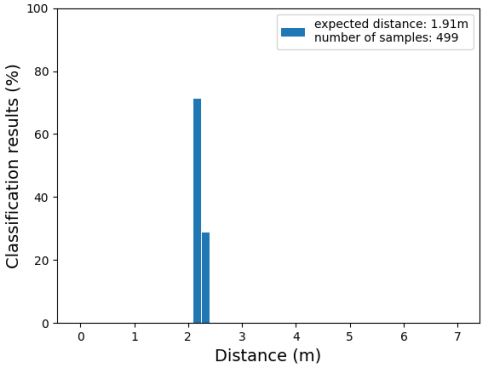
For ideal phase values (see Section 5.1.3), we could already see that unsuitable time resolution settings negatively impact the accuracy of the ranging results. To test its influence also for real data, we take measurements for 5 distances (1.91 m, 3.41 m, 5.46 m, 8.46 m, and 12.01 m) and compute the multipath profiles for a time resolution of 1.55 ns ($\hat{=}$ 15.49 cm) and 0.67 ns ($\hat{=}$ 6.6 cm); we set the sparsity parameter to 14 in both evaluations. In this work, we will also refer to the 0.67 ns time resolution as *finer-grained* resolution.

Representation of results. The results of this evaluation are presented in Fig. 5.5. Fig. 5.5a shows a violin plot, a graphical representation of the ranging error distribution, and a table containing the corresponding percentile errors for a time resolution of 1.55 ns. Fig. 5.5b shows the ranging errors for the 0.67 ns time resolution.

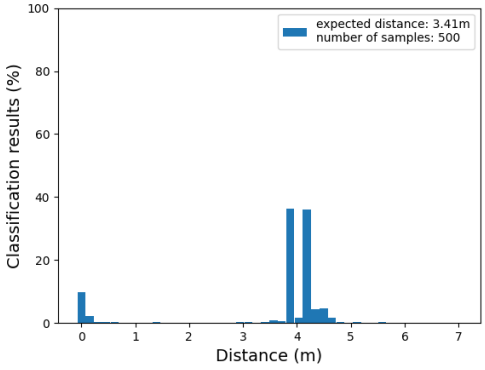
Evaluation. Minor differences in the distance errors between the two evaluations, such as at 8.46 m, where the median error decreased by 4 cm, are not of our interest. They result from the different time resolutions, which also change the granularity at which distances are estimated. The most interesting difference we can see is the one for the expected distance of 1.91 m: the percentile errors for the 1.55 ns resolution are quite high (e.g.,



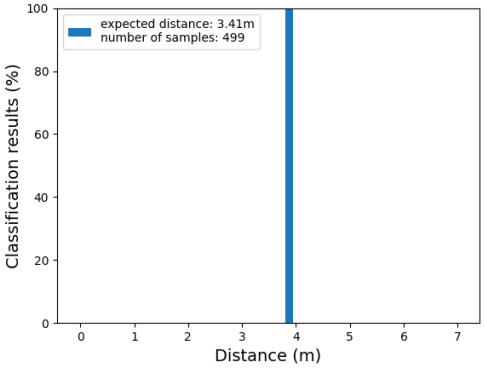
(a) Slow hopping: $d = 1.91\,\mathrm{m}$



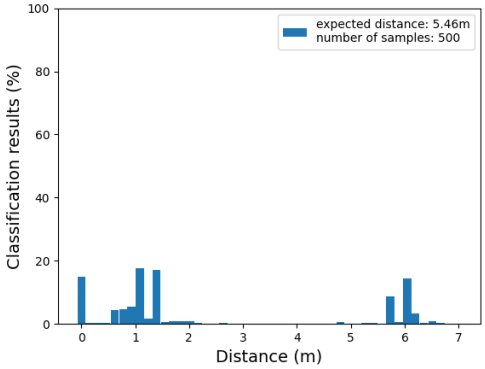
(b) Fast hopping: $d = 1.91\,\mathrm{m}$



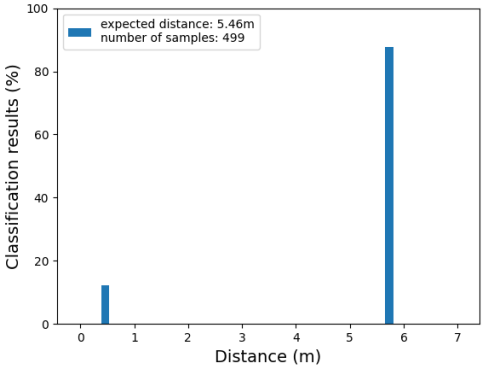
(c) Slow hopping: $d = 3.41\,\mathrm{m}$



(d) Fast hopping: $d = 3.41\,\mathrm{m}$



(e) Slow hopping: $d = 5.46\,\mathrm{m}$



(f) Fast hopping: $d = 5.46\,\mathrm{m}$

Figure 5.4: *Estimated distances for slow and fast hopping in a **wired** environment.* The histograms show where and how often (in percent) the first peak appears at the same position in N (number of samples) multipath profiles. The figures on the left show the results for *slow* and the figures on the right for *fast* hopping.

1.91 m median error), whereas for 0.67 ns, we get much better results (e.g., 0.36 m median error). The higher distance errors for 1.55 ns result from the combination of sparsity and time resolution setting; the median error is exactly as high as the expected distance, which is an indicator that the estimated distances were 0 m for some of the samples (i.e., the multipath profiles were empty) . We cannot say in general terms what the best parameters are for computing multipath profiles, but we know from this evaluation that the balance between time resolution and sparsity is crucial for obtaining accurate ranging results.

Impact of Transmission Power

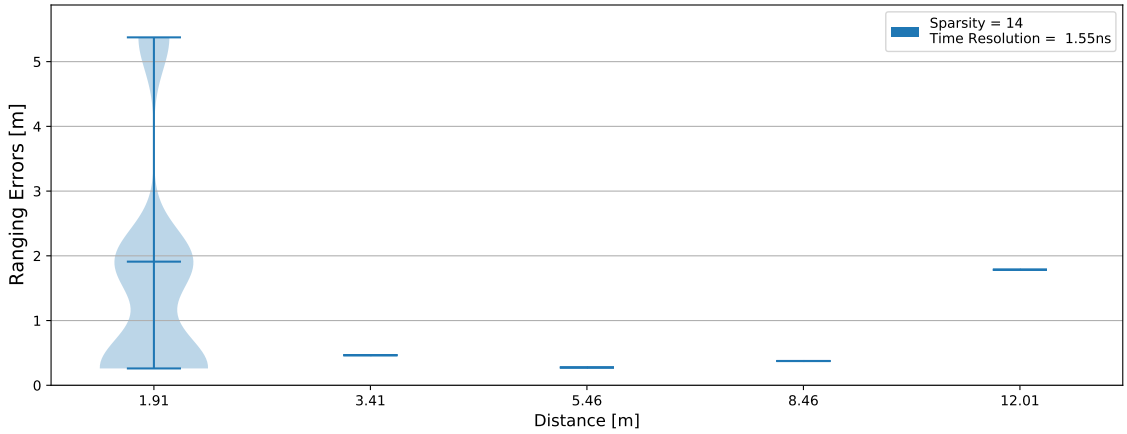
The last thing we evaluate for the wired environment is the impact of the transmission power. As you can see in Fig. 5.6, with increasing power index values the RSSI is linearly decreasing; the RSSI also decreases for longer distances. Each box plot in this figure shows the distribution of RSSI from measurements in all channels we used for the frequency hopping, i.e., the RSSI is not only from a single channel. Since we are in a multipath-free environment, the RSSI variations cannot come from frequency-selective multipath fading (see Section 2.1) but from hardware imperfections.

In this experiment, we performed measurements with different power index settings for 5 distances (i.e., 1.91 m, 3.41 m, 5.46 m, 8.46 m, and 12.01 m). We set the multipath profile's time resolution to 0.67 ns and used 3 different sparsity values for the evaluation to avoid getting worse results just because of non-ideal sparsity settings.

Representation of results. Fig. 5.7 shows the results for 1.91 m and 5.46 m in a heatmap: in each row, we show the results per power index, and the columns represent clusters with a size of 0.5 m to which the estimated distances are assigned: e.g., $x = 1.5$ m contains estimated distances that are in the range of 1-1.5 m). The color indicates how many of the calculated distances (in percent) appear in the same cluster; it is in principle just another representation of the histograms in Fig 5.4. We only want to determine if we get a significant difference for the ranging results when the transmission power changes. Therefore, we do not compare the percentile errors in a table as we did in the previous experiments.

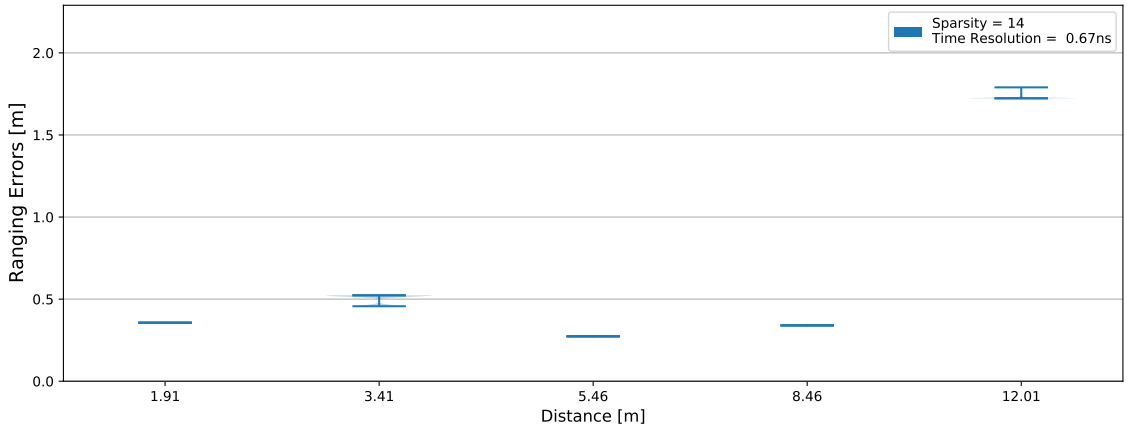
Evaluation. For an expected distance of 1.91 m (Fig. 5.7a), we can see that 100 % of the estimated distances appear in the same cluster (i.e., between 2-2.5 m) and that the results stay the same when the transmission power changes. We observed the same for our measurements at 3.41 m and 12.01 m; for 12.01 m, we could only evaluate the results for power indices 20, 40, and 60 since for power indices greater than 60, the received signal strength was too weak to detect the WiFi frames.

In Fig. 5.7b, for a power index of 60, we can see that a small number (i.e., less than 10 %) of the estimated distances are not in line with the other results. The first peaks in the multipath profiles appear approximately 5 m too early; this corresponds to the time ambiguity we get for signal transmission over the wire (i.e., $\frac{2}{3} \cdot 7.5$ m). At 8.46 m, we could observe something similar: for power index 20, 100 % of the results differed from results of the other power indices: i.e., the estimated distances for power index 20 were all in cluster $x = 9$, whereas for power index 40, 50, 80, and 100 the results were in cluster $x = 4$. Thus, the difference is again 5 m.



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.91	1.91	0.26	1.91	5.38
3.41	0.46	0.46	0.46	0.46
5.46	0.28	0.28	0.28	0.28
8.46	0.38	0.38	0.38	0.38
12.01	1.78	1.78	1.78	1.78
Total	0.46	0.28	1.78	1.91

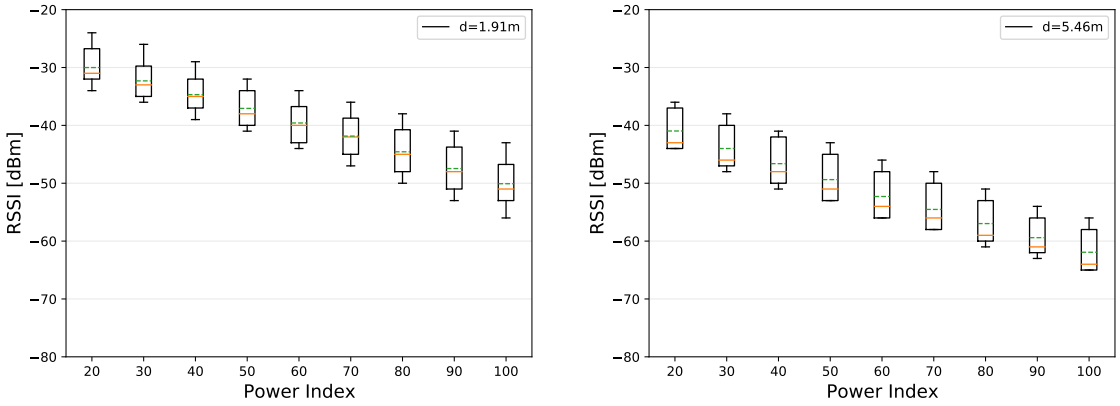
(a) Time resolution = 1.55 ns



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.91	0.36	0.36	0.36	0.36
3.41	0.52	0.46	0.52	0.52
5.46	0.27	0.27	0.27	0.27
8.46	0.34	0.34	0.34	0.34
12.01	1.72	1.72	1.72	1.72
Total	0.36	0.34	0.52	1.72

(b) Time resolution = 0.67 ns

Figure 5.5: *Ranging errors for **wired** measurements for a time resolution of **(a)*** 1.55 ns *and **(b)*** 0.67 ns. The violin plots show the graphical representation of the ranging error distribution and the tables the percentile errors in *m*.



(a) $d = 1.91\,\text{m}$ (b) $d = 5.46\,\text{m}$

Figure 5.6: *RSSI for different power index settings in the* ***wired*** *environment.* The box plots show the distribution of the measured RSSI per power index at a distance of *(a)* 1.91 m and *(b)* 5.46 m. The orange line represents the median and the green the mean.

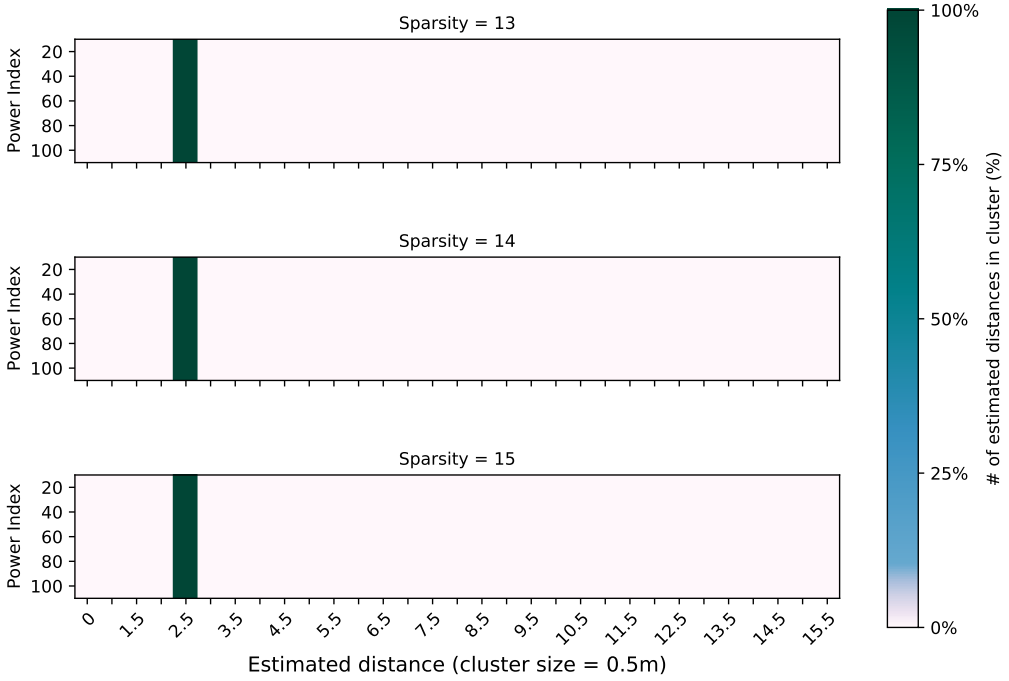
Based on this evaluation in a multipath-free environment, we conclude that Chronos' PDD mitigation, and thus resistance to changes in the received signal strength, works. Nevertheless, we cannot confirm complete resistance against phase errors introduced by PDDs due to the results for 5.46 m and 8.46 m.

5.3 Evaluation Results for Wireless Environments

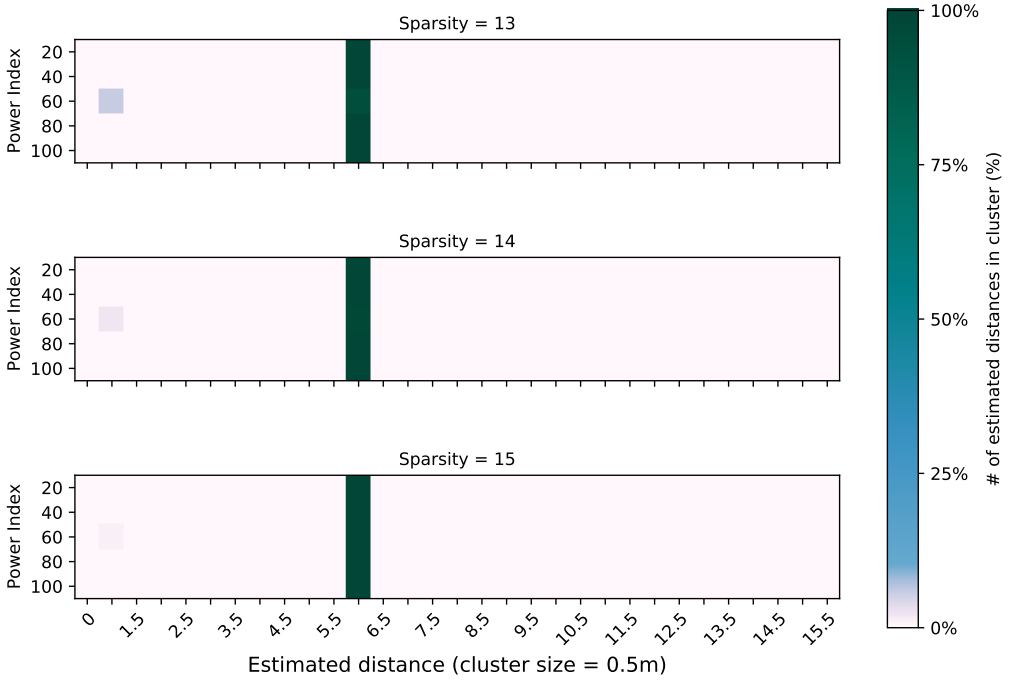
In this section, we present our results for measurements in wireless environments. As for the wired environment, we demonstrate the effects of frequency hopping speed, time resolution settings, and transmission power in the same order.

Impact of Frequency Hopping Speed

We investigate the impact of the frequency hopping speed by evaluating the results for two rounds of measurements in the seminar room. The experiments were conducted independently from each other, which means that we performed the measurements on different days and in a non-identical environment (i.e., the furniture was not placed at exactly the same positions). The ground truth distances for which we obtained the data are also slightly different. Therefore, we compare the results from both experiments where the ground truth was closest to each other. In the first round, we obtained $N \approx 1000$ samples at distances between 1.0 m and 5.5 m with a step size of 0.5 m, and in the second round, $N \approx 500$ samples at 1.0 m, 1.91 m, 3.41 m, 4.5 m, and 5.46 m. For the former, we used slow hopping, and for the latter fast hopping (see Section 5.1.3). For the multipath computation, we set the time resolution to 1.55 ns ($\hat{=}$ 23.25 cm). Since we did not have the exactly identical environment in both experiments, we used two different sparsity values (i.e., 5 for the slow and 8 for the fast hopping) to obtain the best results for each of them.



(a) $d = 1.91\,\mathrm{m}$



(b) $d = 5.46\,\mathrm{m}$

Figure 5.7: *Heatmaps representing the impact of transmission power on the distance estimation in a* ***wired*** *environment for* ***(a)*** 1.91 m *and* ***(a)*** 5.46 m. Rows contain the results per power index; columns represent clusters with a size of 0.5 m. The color indicates the number of estimated distances per cluster.

Representation of results. The ranging errors per distance are shown in Fig. 5.8: the upper figure (5.8a) shows the results for slow hopping and the lower (5.8b) for fast hopping. The violin plots represent the distribution of the distance errors, and the tables show the percentile errors.

Evaluation. Fast hopping reduced the median error at 1.0 m and 2.0 m (compared to 1.91 m) from 0.63 m resp. 0.56 m to 0.4 m resp. 0.42 m. The median error at 3.5 m (compared to 3.41 m) stayed almost the same, but the 75th and 95th percentile errors are also reduced. For distances $d > 3.5$ m, however, we cannot observe any improvement of accuracy. In the case of 4.5 m, the median error even worsened, i.e., it increased from 0.85 m to 3.8 m. Therefore, different from what we expected, we could not see the improvement of ranging accuracy for fast hopping we had hoped for. Since we observed something different in the multipath-free environment, we believe that multipath effects impact the accuracy in the wireless setup, which we cannot resolve with our fast hopping implementation. In the next section, we investigate whether the accuracy of the fast hopping results gets better with different settings for time resolution and sparsity.

Impact of Time Resolution Settings

Since fast hopping alone could not improve the accuracy in the wireless environment, we adjust the algorithm parameters and reevaluate the ranging errors for fast hopping. For the seminar room environment, we set the sparsity to 8 for a time resolution of 1.55 ns ($\hat{=}$ 23.25 cm) and 9 for 0.67 ns ($\hat{=}$ 10 cm); for the corridor, we use sparsity 8.5 and 9.5, respectively. The number of samples is $N \approx 500$ per distance, and we increased the time vector length from 50 ns to 110 ns ($\hat{=}$ 16.5 m).

Representation of results. We show the distribution of ranging errors in violin plots and the percentile errors in the tables below these plots. Fig. 5.9 shows the results for the measurements in the seminar room, and Fig. 5.10 in the corridor. In both, the upper one contains the data for the 1.55 ns time resolution and the lower for 0.67 ns.

Evaluation. With the finer-grained time resolution (i.e., 0.67 ns), we can see a significant improvement in accuracy for both environments, seminar room and corridor. For example, in the seminar room, the median error at 5.46 m decreased from 3.83 m to 0.64 m, and the total median error reduced from 1.01 m to 0.69 m. Only the median error at 1.0 m doubled from 0.4 m to 0.8 m. The increase of the median error at 1.91 m by 3 cm results from the different granularity at which distances are estimated. In the corridor, we see the most considerable improvement of accuracy at 8.46 m, where the median error has decreased from 4.97 m to 1.44 m; this also causes the total median error to decrease from 4.97 m to 2.16 m. However, we do not see a significant difference for the other two distances (i.e., 5.46 m and 12.01 m).

Similar to the results in the wired setup, we showed that the time resolution and sparsity settings are crucial parameters to obtain accurate results as accuracy can suffer drastically if those values are not set correctly.

Impact of Transmission Power

We evaluate the impact of transmission power for wireless measurements in the seminar room and use the same power indices as in the wired environment (i.e., 20, 40, 60, 80, and 100). Measurements were taken at 1.0 m, 1.91 m, 3.41 m, 4.5 m, and 5.46 m, and the number of samples per distance is $N \approx 500$. For the multipath computation, we set the time resolution to 0.67 ns, and the selected sparsity values are 8, 8.5, and 9.

Representation of results. The results for 1.91 m and 5.46 m are shown in the heatmaps in Fig. 5.11; these plots represent our results in the same way as described in Section 5.2, namely, the number of estimated distances in 0.5 m clusters per power index.

Evaluation. In Fig. 5.11a, the estimated distances for different power indices are almost all in the same cluster (i.e., 2.5 m). For power indices greater than or equal to 60, however, we observe that some results (in the worst case approximately 25 %) appear in a cluster that is 7 m away from the expected (i.e., 9.5 m), which is close to the 7.5 m ambiguity discussed in Section 5.1.3. At 5.46 m, we do not see this ambiguity, but we see that the results for different transmission powers differ significantly. The results at this distance are generally not very accurate, as also shown in Section 5.3. For the other distances, i.e., 1.0 m, 3.41 m, and 4.5 m, we could also see the influence of the transmission power on the distance estimation. Since we observed the same for the wired setup, we know that these differences do not only result from multipath effects, and thus the phase error introduced by the variation of PDD is the cause.

5.4 Discussion

This section summarizes and concludes our results from the experiments in the wired and wireless setup. First, the effects of frequency hopping speed, time resolution, and transmit power are discussed separately, and a more general conclusion is provided at the end.

Impact of Frequency Hopping Speed. In the wired environment, we could see that the accuracy for the fast hopping improved compared to slow hopping. We already expected this because the shorter the time between the CSI measurements on the server and client side, the smaller the introduced phase error. Therefore, it was surprising that the results in the wireless environment did only get better for distances $d \leq 3.5$ m. Since the main difference between the wired and the wireless environment is that we experience multipath propagation in the latter, we attribute the poorer accuracy to multipath effects. Based on our experiments, the impact of multipath effects on the ranging results seem to be more critical for longer distances. Nevertheless, it is also possible that the time between the CSI measurements on the server and the client is yet too high, i.e., phase errors caused by insufficient PPO and CFO correction may still be a problem. Unfortunately, we were not able to verify this because switching channels faster than every 4.2 ms was not possible.

Impact of Time Resolution Settings. We know that sparsity and time resolution need to be carefully selected to avoid the first peak in the multipath profile appearing too early or too late from the evaluation of ideal data. In our experimental evaluation, we

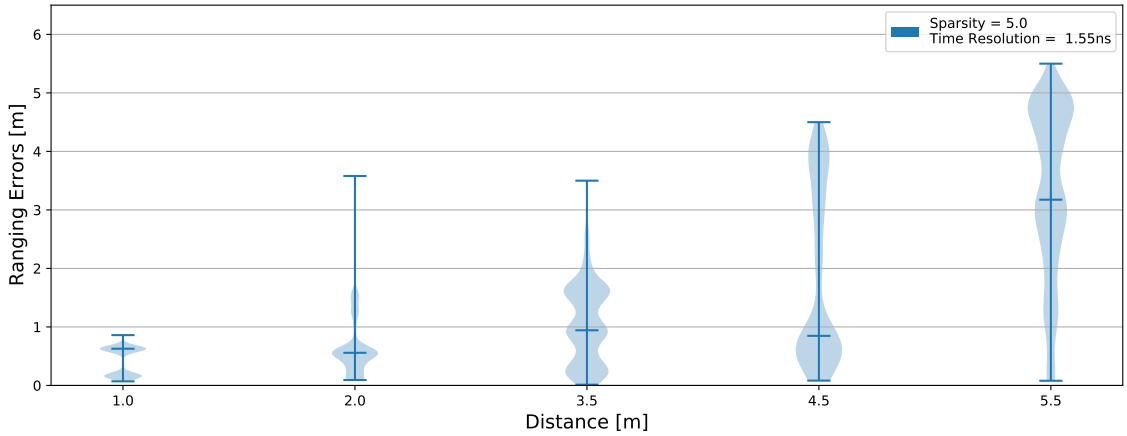
improved the accuracy of the estimated distances by using the higher time resolution (i.e., 0.67 ns) instead of the 1.55 ns proposed by Ferrín [27]. Although the resolution of 1.55 ns corresponds to the effective bandwidth we obtain by hopping through the 24 channels in the 5 GHz band (i.e., 645 MHz), selecting a finer-grained time resolution for the multipath profiles is better. The sparsity value needs to be evaluated empirically depending on the time resolution setting and the environment.

Impact of Transmission Power. We evaluated whether the estimated distances remain approximately the same (i.e., in the same cluster) when we perform the frequency hopping with different transmission powers. We expected that the transmission power and, conversely, the received signal strength would not significantly affect the distance estimation; however, we observed something different. The transmission power influences the distance estimation in both the wired and wireless setup. Therefore, multipath effects alone cannot be the reason, and we conclude that the differences in the results come from the phase error introduced by the PDD. Hence, Chronos' PDD mitigation strategy only works to some degree.

Summary. In this chapter, we evaluated the distance estimation accuracy of Chronos on the Raspberry Pi 4B. The median error for 5 distances (in the range of 1.91-12.01 m) in the multipath-free environment is 0.36 m. In a seminar room, we get a median error of 0.69 m for measurements at 5 distances (between 1.0 and 5.46 m), and the smallest median error, i.e., 0.39 m, could be observed at 1.91 m. In our measurements in a corridor (at 5.46 m, 8.46 m, and 12.01 m) the median error is 2.16 m, which is significantly higher than in the seminar room. The reason could be the longer distances, and that multipath reflections are stronger in the smaller corridor than in the broader seminar room.

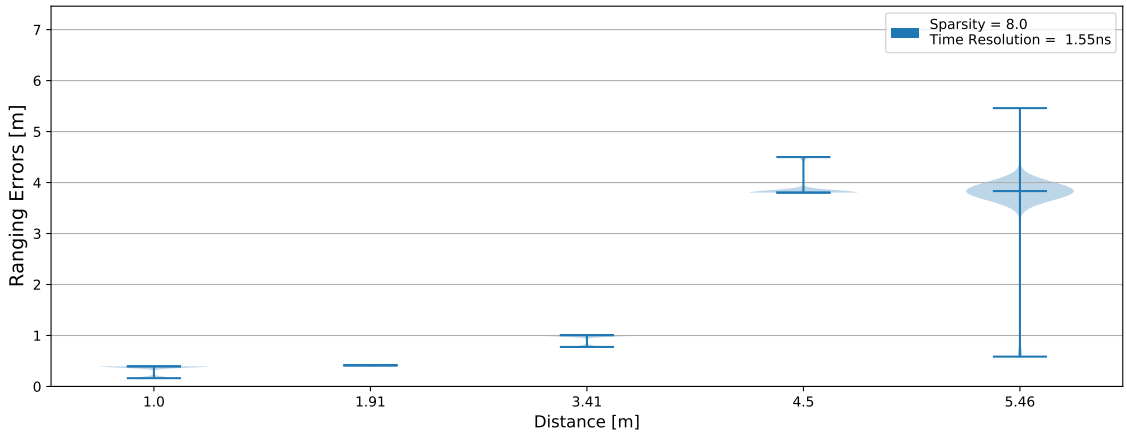
Although we achieved sub-meter level accuracy in the wired setup and the measurements in the seminar room, our results do not get close Chronos' median distance error of 14.1 cm. Chronos benefits from the additional channel hopping in the 2.4 GHz band, which increases the effective bandwidth to almost 1 GHz. Its frequency hopping speed (i.e., switching channels every 2-3 ms) is also faster than our implementation, which takes about 4.2 ms. Vasisht et al. [12] also mention hardware delays that introduce a constant additive offset to the estimated ToF. We did, however, not investigate this further and could not observe a trend that the distance errors would follow (e.g., a linear regression) based on our results. Also Ferrín [27] was unable to observe a pattern in the hardware delay.

Since our ranging errors in the wireless environment increase significantly at distances greater than 3.5 m, we next investigate whether we can further increase the accuracy of our system with additional data processing. In the next chapter, we describe and evaluate our attempt to improve the distance estimation accuracy of Chronos on the Raspberry Pi.



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.00	0.63	0.16	0.63	0.63
2.00	0.56	0.33	0.56	1.30
3.50	0.94	0.45	1.64	1.85
4.50	0.85	0.62	3.10	4.04
5.50	3.17	2.25	4.80	4.80
Total	0.85	0.56	2.71	4.11

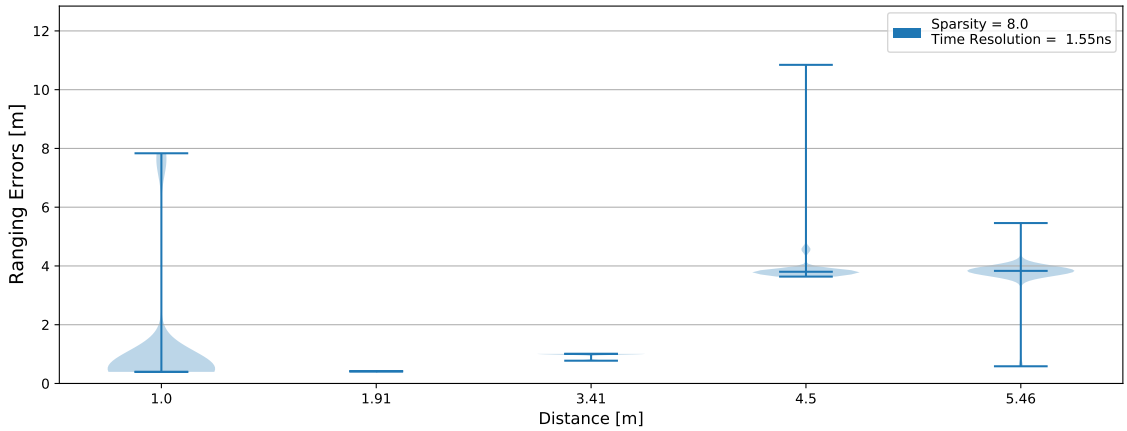
(a) *Slow hopping*



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.00	0.40	0.40	0.40	0.40
1.91	0.42	0.42	0.42	0.42
3.41	1.01	1.01	1.01	1.01
4.50	3.80	3.80	3.80	3.80
5.46	3.83	3.83	3.83	3.83
Total	1.01	0.42	3.80	3.83

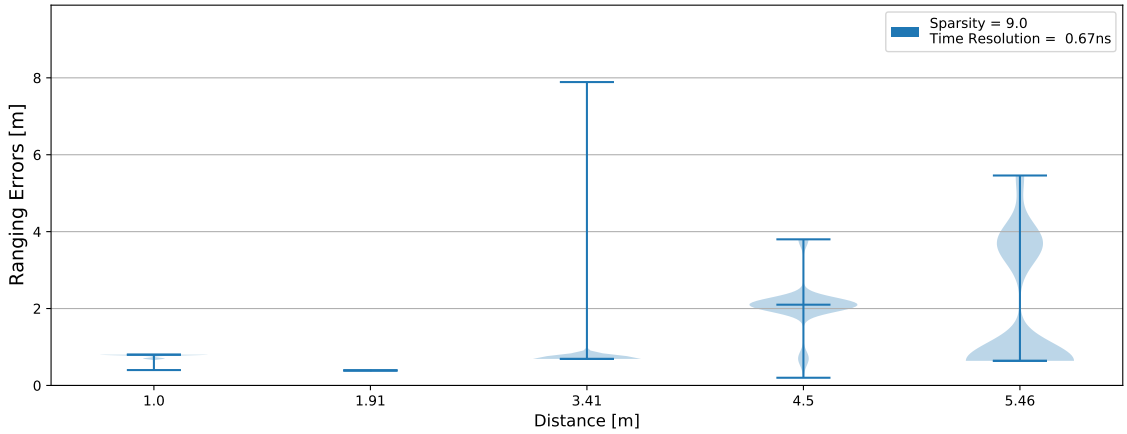
(b) *Fast hopping*

Figure 5.8: *Ranging errors for **wireless** measurements in a **seminar room** for **(a)** slow and **(b)** fast hopping settings.* The violin plots show the graphical representation of the ranging error distribution and the tables the percentile errors in *m*.



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.00	0.40	0.40	0.86	0.86
1.91	0.42	0.42	0.42	0.42
3.41	1.01	1.01	1.01	1.01
4.50	3.80	3.80	3.80	3.80
5.46	3.83	3.83	3.83	3.83
Total	1.01	0.42	3.80	3.83

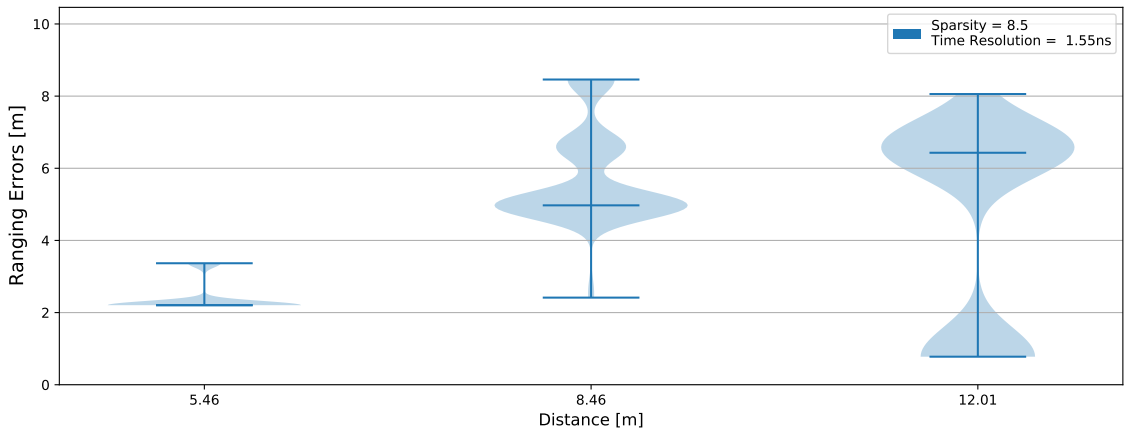
(a) Seminar room: time resolution = 1.55 ns



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
1.00	0.80	0.80	0.80	0.80
1.91	0.39	0.39	0.39	0.39
3.41	0.69	0.69	0.69	0.69
4.50	2.10	2.10	2.10	2.10
5.46	0.64	0.64	3.46	3.86
Total	0.69	0.64	2.10	2.10

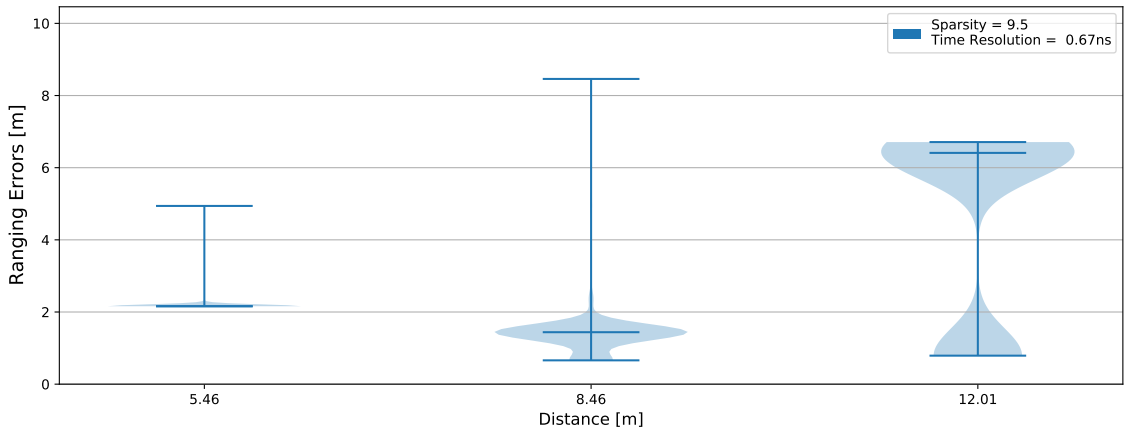
(b) Seminar room: time resolution = 0.67 ns

Figure 5.9: *Ranging errors for* ***wireless*** *measurements in a* ***seminar room*** *for a time resolution of* ***(a)*** 1.55 ns *and* ***(b)*** 0.67 ns. The violin plots show the graphical representation of the ranging error distribution and the tables the percentile errors in *m*.



Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
5.46	2.20	2.20	2.20	3.37
8.46	4.97	4.97	6.60	8.46
12.01	6.43	0.78	6.66	6.66
Total	4.97	2.20	6.60	6.66

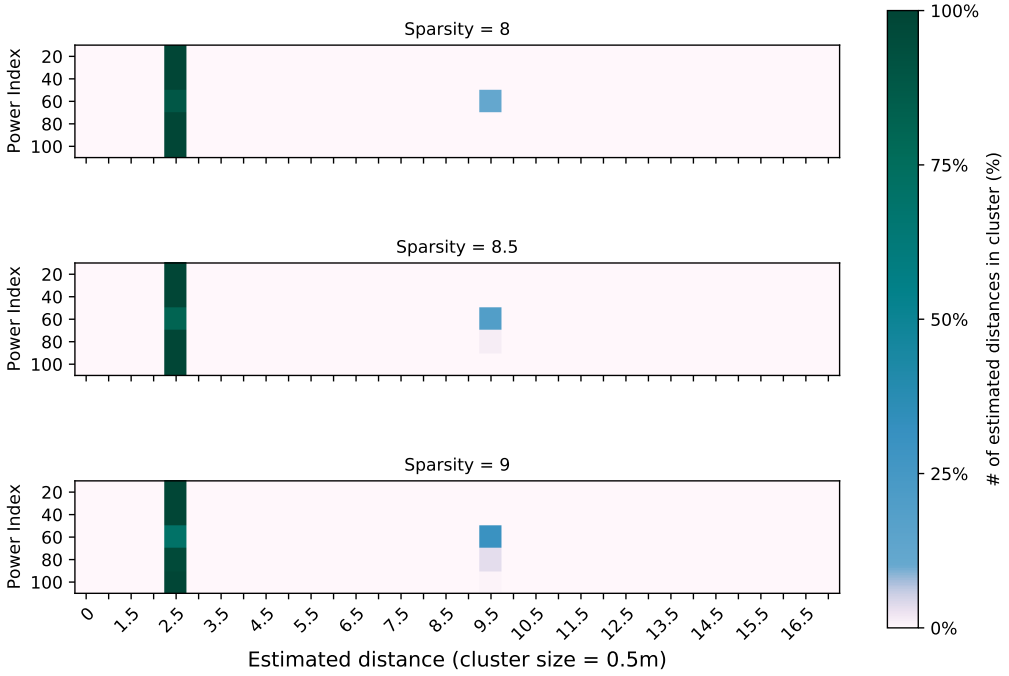
(a) Corridor: time resolution = 1.55 ns



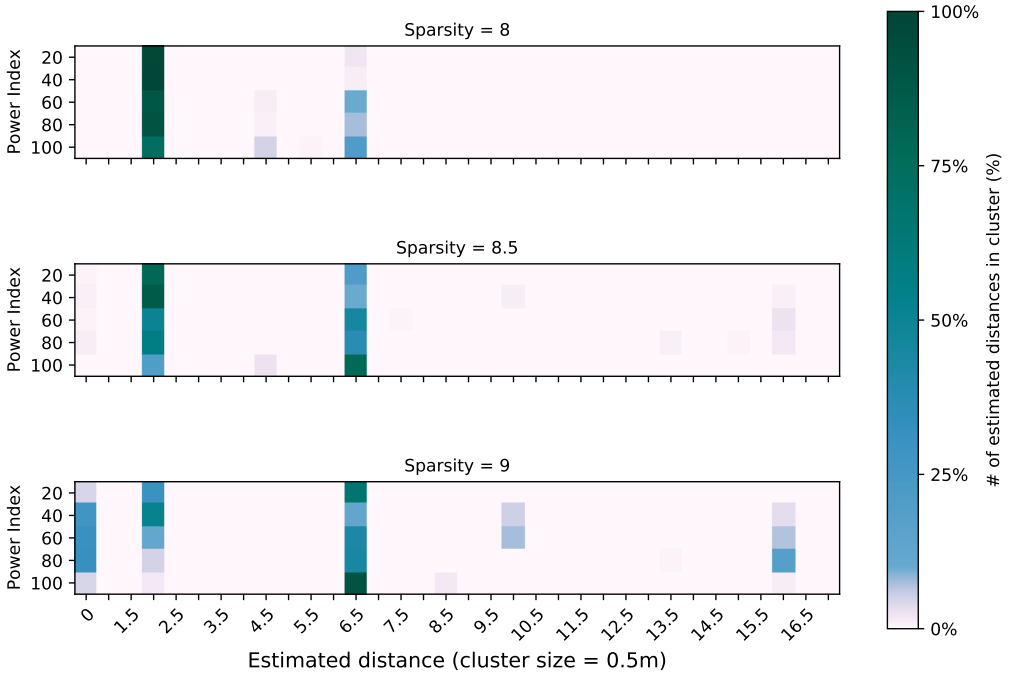
Distance in m	Median Error in m	25th percentile Error in m	75th percentile Error in m	90th percentile Error in m
5.46	2.16	2.16	2.16	2.16
8.46	1.44	1.44	1.44	1.44
12.01	6.41	0.79	6.41	6.41
Total	2.16	1.44	2.16	6.41

(b) Corridor: time resolution = 0.67 ns

Figure 5.10: *Ranging errors for **wireless** measurements in a **corridor** for a time resolution of **(a)*** 1.55 ns *and **(b)*** 0.67 ns. The violin plots show the graphical representation of the ranging error distribution and the tables the percentile errors in *m*.



(a) $d = 1.91\,\mathrm{m}$



(b) $d = 5.46\,\mathrm{m}$

Figure 5.11: *Heatmaps representing the impact of transmission power on the distance estimation in a* ***wireless*** *environment for* ***(a)*** 1.91 m *and* ***(a)*** 5.46 m. Rows contain the results per power index; columns represent clusters with a size of 0.5 m. The color indicates the number of estimated distances per cluster.

Chapter 6

Investigating the Impact of Channel Filtering

In this chapter, we focus on improving the ranging accuracy of Chronos on Raspberry Pi 4B hardware by filtering channel data before computing the multipath profiles. Since we noticed a decreasing ranging accuracy when taking measurements in a typical multipath environment instead of a multipath-free environment, we investigate next whether there are ways to improve the accuracy of Chronos. We do this by analyzing the effects of multipath fading per channel since they are frequency selective (see Section 2.1). Therefore, we remove the data of a single channel i (i.e., $CSI^2_{i,0}$) from the vector containing the data for computing the multipath profile (i.e., $\tilde{\mathbf{h}}^{\mathbf{2}}$): for instance, to find out which influence the CSI in channel 36 has on the ranging accuracy, we discard $CSI^2_{36,0}$ from $\tilde{\mathbf{h}}^{\mathbf{2}}$ (i.e., the length of $\tilde{\mathbf{h}}^{\mathbf{2}}$ is now 23 instead of 24) and perform the distance estimation again with the reduced dataset. We do this for all 24 channels, i.e., $i = 36, 40, ..., 165$, and analyze how much each channel's data affects the distance estimation; to ensure that our observations are effects of multipath propagation, we perform this evaluation for the data from both the wired and wireless setup. The results of this evaluation are presented in Section 6.1. In Section 6.2, we then investigate whether we can find a way to automatically detect channels that cause an increase of the ranging errors.

6.1 Analyzing the Effects of discarding Channel Data

We present the results of discarding channel data for the multipath profile computation in violin plots like the one shown in Fig. 6.1a. The first violin plot in the figures (labeled 'None') shows the ranging error distribution of the *complete dataset* (i.e., where we used the data of all 24 channels); the other violin plots show the errors for the *reduced dataset* (i.e., where we removed the data of a single channel). The values on the x-axis indicate the channel for which the data was discarded from $\tilde{\mathbf{h}}^{\mathbf{2}}$.

To evaluate whether or not accuracy has improved, we compare the 25th, 50th (i.e., median), 75th, and 90th percentile errors from the reduced datasets to those of the *reference dataset* (i.e., the complete dataset). Therefore, we calculate the relative error $\delta_{i,p}$ (i.e., the change) between the p-th percentile error of the reference dataset r_p and the reduced dataset $e_{i,p}$ as

$$\delta_{i,p} = \frac{e_{i,p} - r_p}{r_p} \cdot 100\,\%, \tag{6.1}$$

where i indicates the channel for which the data was removed. Thus, a negative value for $\delta_{i,p}$ means that the p-th percentile of our estimated distances has become more accurate: for example, if the reference median error r_{50} is 0.5 m and the new median error $e_{128,50}$ is 0.4 m, the change in the median error $\delta_{128,50}$ is $-20\,\%$ when we discard the data for channel 128. The values for $\delta_{i,p}$ are shown in tables such as in Table 6.1a.

6.1.1 Results in a Wired Environment

This evaluation is performed for the data from the wired setup described in Section 5.2 (i.e., for 1.91 m, 3.41 m, 5.46 m, 8.46 m, and 12.01 m). We set the sparsity to 14 and the time resolution to 0.67 ns and evaluate $N = 100$ samples for the reference dataset and each reduced dataset. Since we are in a multipath-free environment, we expect that removing the data from a particular channel will not affect the ranging accuracy. We show the results for the distances 3.41 m and 12.01 m in Fig. 6.1 and Table 6.1.

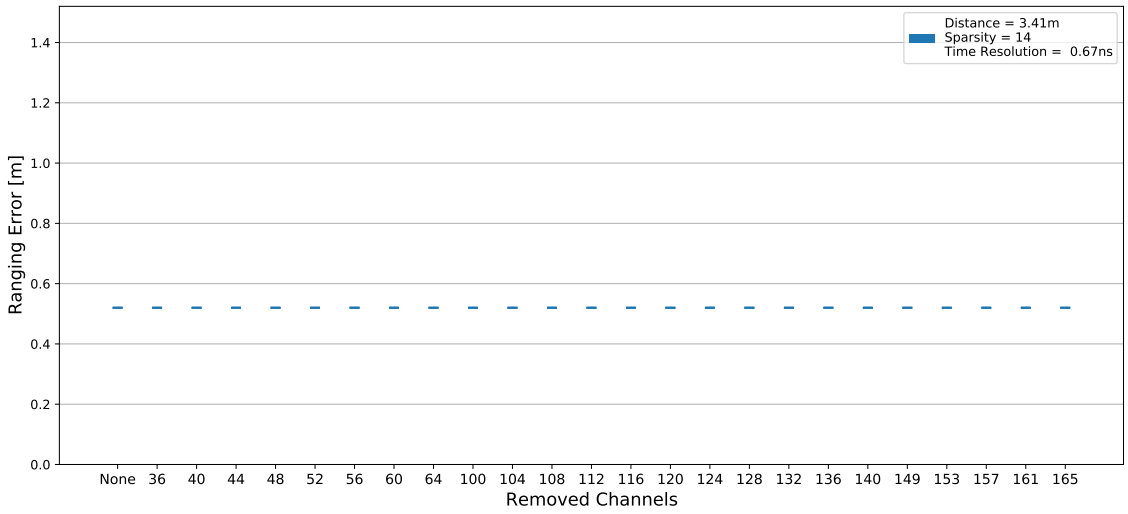
Evaluation. In Fig. 6.1a, we can see that the ranging errors at 3.41 m remain unchanged when we discard the data of a single channel. Therefore, the values for $\delta_{i,p}$ in Table 6.1a are all 0 %, indicating that the accuracy did not change. We observe the same at the other distances (i.e., 1.91 m, 5.46 m, and 8.46 m), except 12.01 m; Fig. 6.1b shows that the distance errors there are slightly different. The most apparent difference is visible when we remove the data of channel 149 or 153. Looking at the results in Table 6.1b, we can see a bit more details: for example, discarding the data of channel 44 or 128 reduces the 90th percentile error by 3.91 %. On the other hand, if the data of channel 149 is removed, the median error and the 25th and 75th percentile errors increase by 4.07 %. The difference visible in the violin plot when discarding the data of channel 153 is only caused by outliers, as the percentile errors neither increase nor decrease.

The results of this evaluation are in line with our expectations, i.e., that the accuracy remains constant if we remove the data of a single channel; the minor deviations observed at 12.01 m can be attributed to non-ideal hardware, i.e., the cables, connectors, and attenuators.

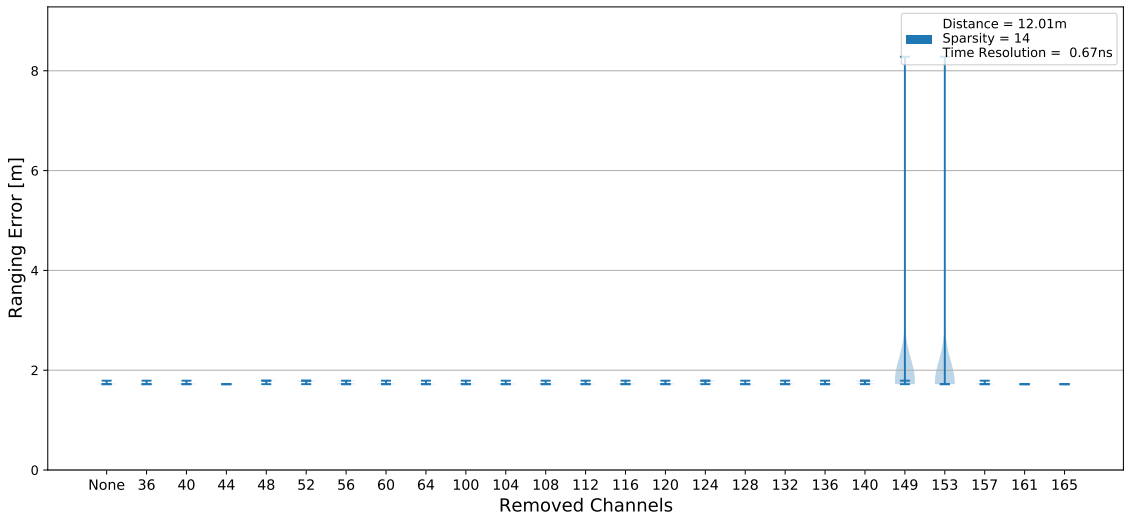
6.1.2 Results in Wireless Environments

For the wireless environment, we hope to see a difference, at best an improvement, in ranging errors when discarding the data of single channels. We evaluate the data from the seminar room and the corridor for a time resolution of 0.67 ns and a sparsity of 9 and 9.5, respectively. The results for the seminar room are shown in Fig. 6.2 and Table 6.2; the results for the corridor are reported in Fig. 6.3 and Table 6.3; the number of samples per violin plot is $N = 100$.

Evaluation. In Fig. 6.2a, which shows the results at 3.41 m in the seminar room, we can see something similar to what we have seen for the wired environment: discarding channel data does not cause the ranging errors to change, i.e., all entries for $\delta_{i,p}$ in Table 6.2a are 0 %. We can also observe the same at a distance of 1.91 m.



(a) $d = 3.41\,\mathrm{m}$



(b) $d = 12.01\,\mathrm{m}$

Figure 6.1: *Distribution of ranging errors in the* ***wired*** *environment for the complete and the reduced datasets at* ***(a)*** 3.41 m *and* ***(b)*** 12.01 m. The x-axis shows the channel for which the data was discarded before estimating the distances; the violin plot labeled 'None' shows the reference errors.

Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	0.00%	0.00%	0.00%	0.00%
40	0.00%	0.00%	0.00%	0.00%
44	0.00%	0.00%	0.00%	0.00%
48	0.00%	0.00%	0.00%	0.00%
52	0.00%	0.00%	0.00%	0.00%
56	0.00%	0.00%	0.00%	0.00%
60	0.00%	0.00%	0.00%	0.00%
64	0.00%	0.00%	0.00%	0.00%
100	0.00%	0.00%	0.00%	0.00%
104	0.00%	0.00%	0.00%	0.00%
108	0.00%	0.00%	0.00%	0.00%
112	0.00%	0.00%	0.00%	0.00%
116	0.00%	0.00%	0.00%	0.00%
120	0.00%	0.00%	0.00%	0.00%
124	0.00%	0.00%	0.00%	0.00%
128	0.00%	0.00%	0.00%	0.00%
132	0.00%	0.00%	0.00%	0.00%
136	0.00%	0.00%	0.00%	0.00%
140	0.00%	0.00%	0.00%	0.00%
149	0.00%	0.00%	0.00%	0.00%
153	0.00%	0.00%	0.00%	0.00%
157	0.00%	0.00%	0.00%	0.00%
161	0.00%	0.00%	0.00%	0.00%
165	0.00%	0.00%	0.00%	0.00%

(a) $d = 3.41\,\mathrm{m}$

Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	0.00%	0.00%	0.00%	-3.91%
40	0.00%	0.00%	0.00%	0.00%
44	0.00%	0.00%	0.00%	-3.91%
48	4.07%	4.07%	4.07%	0.00%
52	4.07%	0.00%	4.07%	0.00%
56	0.00%	0.00%	4.07%	0.00%
60	0.00%	0.00%	1.02%	0.00%
64	0.00%	0.00%	0.00%	-3.52%
100	0.00%	0.00%	0.00%	0.00%
104	0.00%	0.00%	0.00%	0.00%
108	0.00%	0.00%	0.00%	0.00%
112	0.00%	0.00%	0.00%	0.00%
116	0.00%	0.00%	0.00%	0.00%
120	0.00%	0.00%	4.07%	0.00%
124	4.07%	0.00%	4.07%	0.00%
128	0.00%	0.00%	0.00%	-3.91%
132	0.00%	0.00%	0.00%	0.00%
136	0.00%	0.00%	0.00%	0.00%
140	4.07%	0.00%	4.07%	0.00%
149	4.07%	4.07%	4.07%	0.00%
153	0.00%	0.00%	0.00%	0.00%
157	0.00%	0.00%	0.00%	0.00%
161	0.00%	0.00%	0.00%	-3.91%
165	0.00%	0.00%	0.00%	-3.91%

(b) $d = 12.01\,\mathrm{m}$

Table 6.1: *Wired environment:* Ranging error changes (in percent) from the complete to the reduced dataset at **(a)** 3.41 m and **(b)** 12.01 m.

Nevertheless, we got the results we hoped for the other distances in the seminar room (i.e., 1.0 m, 4.5 m, and 5.46 m). Fig. 6.2b shows the ranging errors at 4.5 m, and we can see that removing the data of channel 104 or 157, for example, reduces them by over 1 m. We will refer to the channels for which discarding the data leads to improved accuracy as *removable channels* from now on. We note that the removable channels differ per distance: this is expected, as multipath effects vary depending on the position of the devices.

In the corridor, we can see these effects as well. At 12.01 m (see Fig. 6.3b and Table 6.3b), we observed the most considerable accuracy improvement compared to all other distances in the wireless environment: the median error could be reduced by 87.68 %. The number of removable channels is relatively high at this distance; if we count only those that improve the median error, there are 11. Fig. 6.3a and Table 6.3a show the results at 5.46 m. There, we see no improvement but a reduction in accuracy when we remove channel data and that the ranging errors are already above 2 m for the reference dataset. For the latter, we have discussed possible reasons in Section 5.4.

Based on the evaluation of wireless distance estimation, we conclude that discarding data for certain channels can, although not always, improve the ranging accuracy. We are particularly interested in finding removable channels such as channel 104 at 4.5 m in the seminar room that can significantly reduce the ranging errors if their data is not used for distance estimation. In the next section, we investigate whether it is possible to find removable channels using RSS-based information.

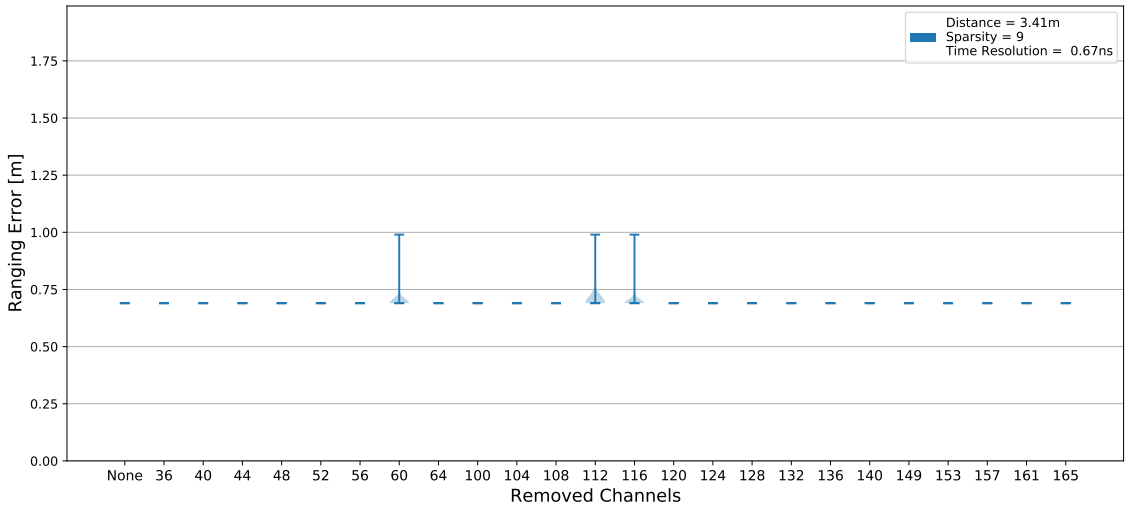
6.2 Discarding Channel Data based on RSS Information

We now want to find a method to detect removable channels during live measurements, based on our previous observations in Section 6.1. The goal is to find channels for which we can remove data from the dataset without increasing the ranging errors. We investigate whether the RSSI is a suitable metric for this purpose, as it is very susceptible to multipath effects (see Section 2.1.1). Furthermore, as discussed in Chapter 5, the transmission power used, and thus the received signal strength, affects the distance estimation. If multipath fading causes the RSSI for each channel to be different, the PDD will be different as well. Therefore, the RSSI might indicate channels that experience higher phase errors than others and negatively influence the accuracy.

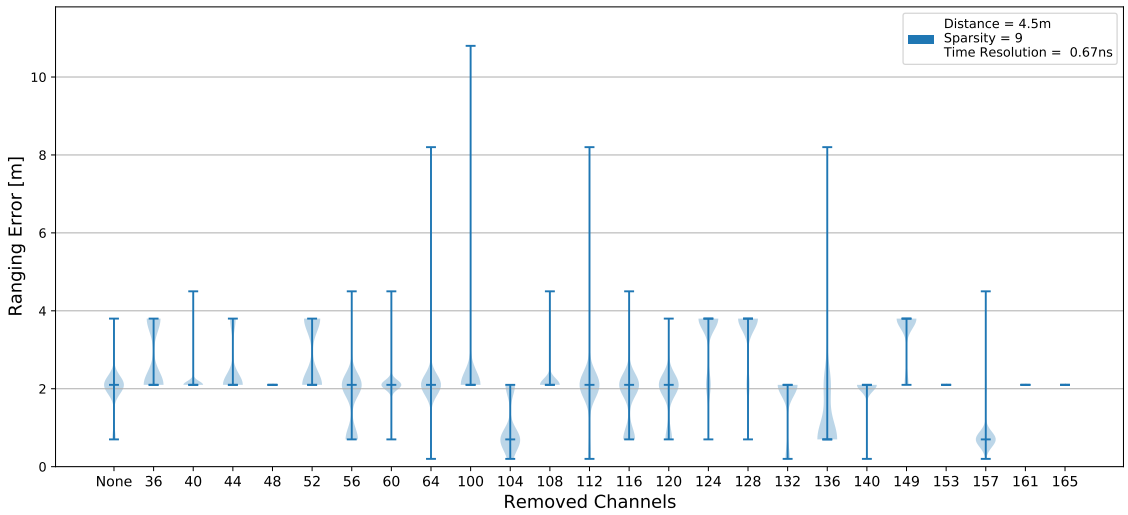
First, we analyze the RSSI measurements from our experiments (i.e., for 2500 complete frequency hopping rounds) in the wired and in the wireless setup to detect channels affected more by multipath effects than others. Then, we investigate whether there is a correlation between removable channels and the RSSI to improve accuracy. Last, we discuss alternatives to RSSI and options for future work.

Investigating RSSI per channel

We use scatter plots to illustrate the RSSI per channel and its fluctuations over time. These plots show the RSSI measured at one device, as the plots from the server and the client are almost identical.



(a) Seminar room: $d = 3.41\,\mathrm{m}$



(b) Seminar room: $d = 4.50\,\mathrm{m}$

Figure 6.2: *Distribution of ranging errors for* ***wireless*** *measurements in a* ***seminar room*** *for the complete and the reduced datasets at* ***(a)*** 3.41 m *and* ***(b)*** 4.50 m. The x-axis shows the channel for which the data was discarded before estimating the distances; the violin plot labeled 'None' shows the reference errors.

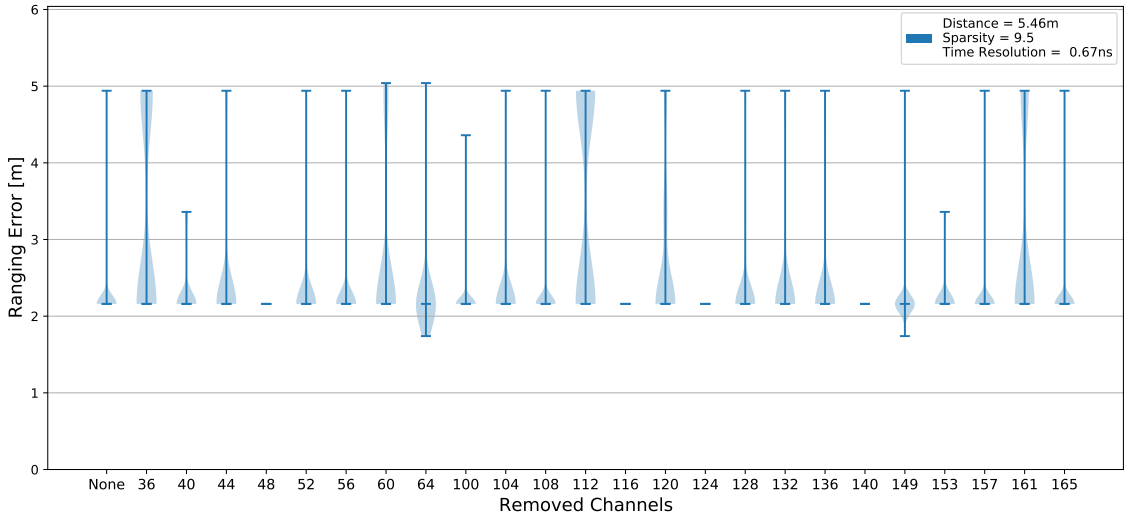
Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	0.00%	0.00%	0.00%	0.00%
40	0.00%	0.00%	0.00%	0.00%
44	0.00%	0.00%	0.00%	0.00%
48	0.00%	0.00%	0.00%	0.00%
52	0.00%	0.00%	0.00%	0.00%
56	0.00%	0.00%	0.00%	0.00%
60	0.00%	0.00%	0.00%	0.00%
64	0.00%	0.00%	0.00%	0.00%
100	0.00%	0.00%	0.00%	0.00%
104	0.00%	0.00%	0.00%	0.00%
108	0.00%	0.00%	0.00%	0.00%
112	0.00%	0.00%	0.00%	0.00%
116	0.00%	0.00%	0.00%	0.00%
120	0.00%	0.00%	0.00%	0.00%
124	0.00%	0.00%	0.00%	0.00%
128	0.00%	0.00%	0.00%	0.00%
132	0.00%	0.00%	0.00%	0.00%
136	0.00%	0.00%	0.00%	0.00%
140	0.00%	0.00%	0.00%	0.00%
149	0.00%	0.00%	0.00%	0.00%
153	0.00%	0.00%	0.00%	0.00%
157	0.00%	0.00%	0.00%	0.00%
161	0.00%	0.00%	0.00%	0.00%
165	0.00%	0.00%	0.00%	0.00%

(a) Seminar room: $d = 3.41\,\mathrm{m}$

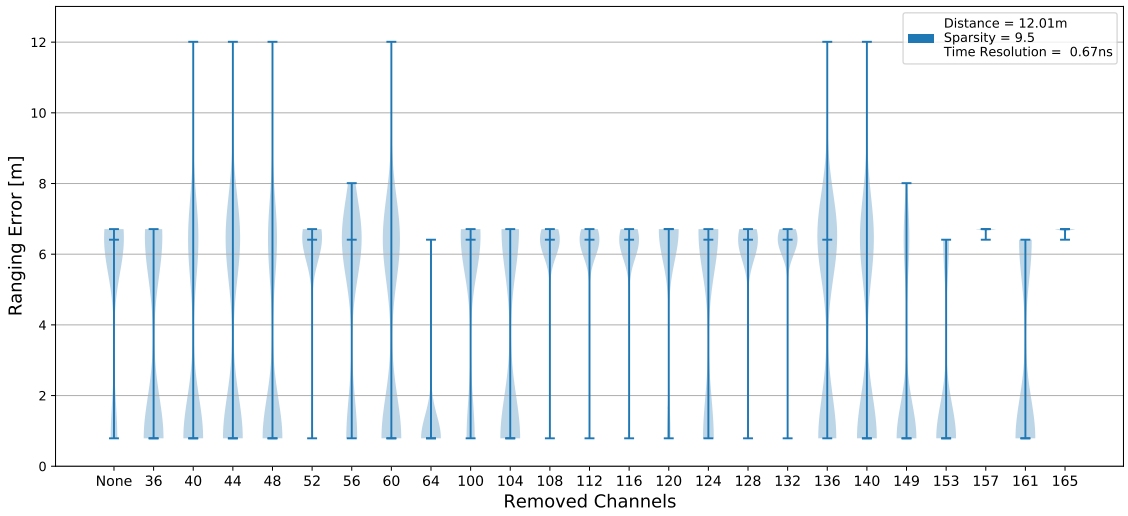
Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	0.00%	0.00%	80.95%	80.95%
40	0.00%	0.00%	0.00%	0.00%
44	0.00%	0.00%	0.00%	80.95%
48	0.00%	0.00%	0.00%	0.00%
52	0.00%	0.00%	80.95%	80.95%
56	0.00%	-66.67%	0.00%	0.00%
60	0.00%	0.00%	0.00%	0.00%
64	0.00%	0.00%	0.00%	0.00%
100	0.00%	0.00%	0.00%	0.00%
104	-66.67%	-66.67%	0.00%	0.00%
108	0.00%	0.00%	0.00%	0.00%
112	0.00%	0.00%	0.00%	0.00%
116	0.00%	-66.67%	0.00%	0.00%
120	0.00%	0.00%	0.00%	0.00%
124	80.95%	80.95%	80.95%	80.95%
128	80.95%	80.95%	80.95%	80.95%
132	0.00%	0.00%	0.00%	0.00%
136	-66.67%	-66.67%	0.00%	0.00%
140	0.00%	0.00%	0.00%	0.00%
149	80.95%	80.95%	80.95%	80.95%
153	0.00%	0.00%	0.00%	0.00%
157	-66.67%	-66.67%	-66.67%	-66.67%
161	0.00%	0.00%	0.00%	0.00%
165	0.00%	0.00%	0.00%	0.00%

(b) Seminar room: $d = 4.50\,\mathrm{m}$

Table 6.2: *Wireless environment (seminar room):* Ranging error changes (in percent) from the complete to the reduced dataset at **(a)** 3.41 m and **(b)** 4.50 m.



(a) Corridor: $d = 5.46\,\mathrm{m}$



(b) Corridor: $d = 12.01\,\mathrm{m}$

Figure 6.3: *Distribution of ranging errors for* ***wireless*** *measurements in a* ***corridor*** *for the complete and the reduced datasets at* ***(a)*** 5.46 m *and* ***(b)*** 12.01 m. The x-axis shows the channel for which the data was discarded before estimating the distances; the violin plot labeled 'None' shows the reference errors.

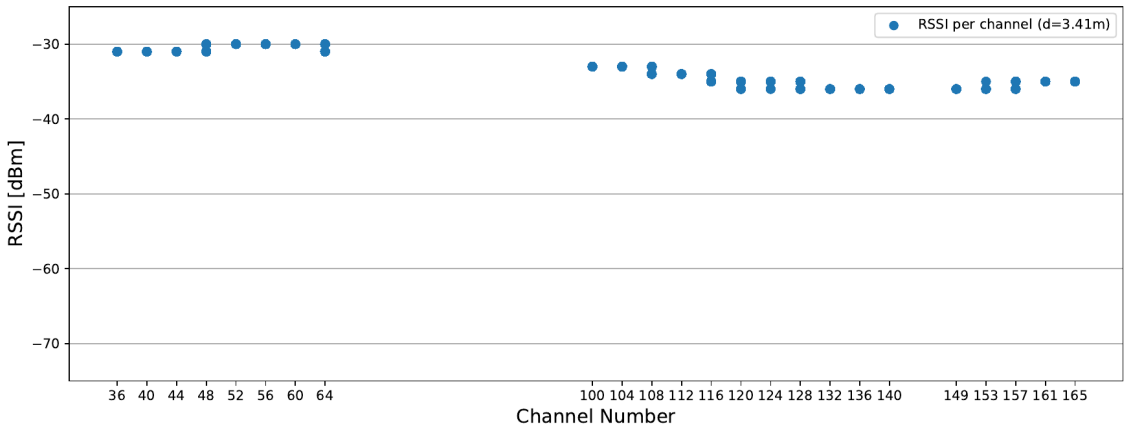
Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	0.00%	0.00%	128.70%	128.70%
40	0.00%	0.00%	0.00%	55.56%
44	0.00%	0.00%	0.00%	0.46%
48	0.00%	0.00%	0.00%	0.00%
52	0.00%	0.00%	0.00%	0.00%
56	0.00%	0.00%	0.00%	0.00%
60	0.00%	0.00%	0.00%	128.70%
64	0.00%	0.00%	0.00%	0.00%
100	0.00%	0.00%	0.00%	0.00%
104	0.00%	0.00%	0.00%	0.00%
108	0.00%	0.00%	0.00%	0.00%
112	0.00%	0.00%	128.70%	128.70%
116	0.00%	0.00%	0.00%	0.00%
120	0.00%	0.00%	55.56%	128.70%
124	0.00%	0.00%	0.00%	0.00%
128	0.00%	0.00%	0.00%	0.00%
132	0.00%	0.00%	0.00%	0.00%
136	0.00%	0.00%	0.00%	0.00%
140	0.00%	0.00%	0.00%	0.00%
149	0.00%	0.00%	0.00%	0.00%
153	0.00%	0.00%	0.00%	0.00%
157	0.00%	0.00%	0.00%	0.00%
161	0.00%	0.00%	128.70%	128.70%
165	0.00%	0.00%	0.00%	0.00%

(a) Corridor: $d = 5.46\,\mathrm{m}$

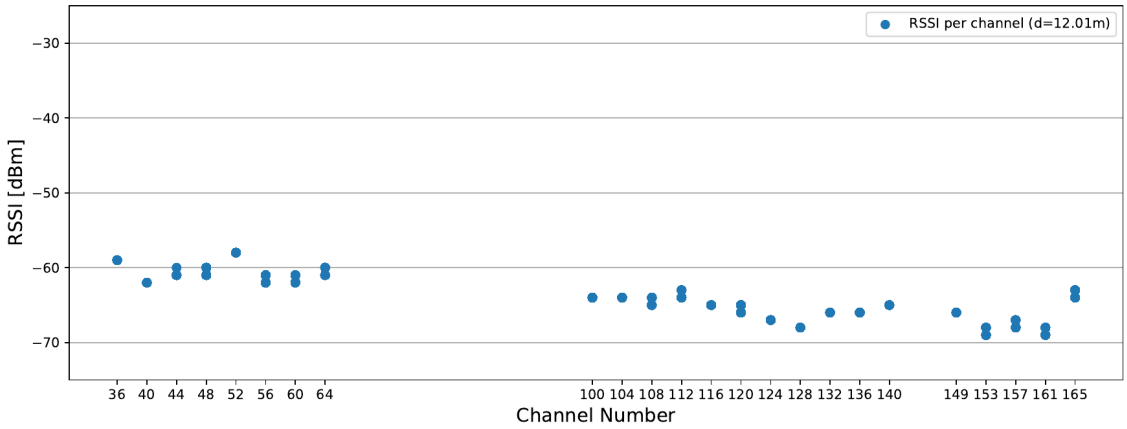
Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	-87.68%	-84.22%	0.00%	0.00%
40	-87.68%	-84.22%	0.00%	-4.47%
44	-87.68%	-84.22%	0.00%	-4.47%
48	-87.68%	-84.22%	0.00%	-4.47%
52	0.00%	28.07%	0.00%	0.00%
56	0.00%	-84.22%	0.00%	-4.47%
60	-87.68%	-84.22%	0.00%	-4.47%
64	-87.68%	-84.22%	-87.68%	-88.23%
100	0.00%	-84.22%	1.17%	0.00%
104	-87.68%	-84.22%	4.68%	0.00%
108	0.00%	28.07%	0.00%	-4.47%
112	0.00%	28.07%	0.00%	-4.47%
116	0.00%	28.07%	0.00%	-4.47%
120	4.68%	28.07%	4.68%	0.00%
124	0.00%	-84.22%	4.68%	0.00%
128	0.00%	28.07%	0.00%	-4.47%
132	0.00%	28.07%	0.00%	-4.47%
136	0.00%	-84.22%	0.00%	0.00%
140	-87.68%	-84.22%	0.00%	-4.47%
149	-87.68%	-84.22%	-87.68%	-4.47%
153	-87.68%	-84.22%	-87.68%	-4.47%
157	4.68%	34.07%	4.68%	0.00%
161	-87.68%	-84.22%	0.00%	-4.47%
165	4.68%	34.07%	4.68%	0.00%

(b) Corridor: $d = 12.01\,\mathrm{m}$

Table 6.3: *Wireless environment (corridor):* Ranging error changes (in percent) from the complete to the reduced dataset at **(a)** 5.46 m and **(b)** 12.01 m.



(a) $d = 3.41\,\text{m}$



(b) $d = 12.01\,\text{m}$

Figure 6.4: *Scatter plots of RSSI per channel in the* ***wired*** *environment at* ***(a)*** 3.41 m *and* ***(b)*** 12.01 m.

Wired environment. The plots for 3.41 m and 12.01 m in the wired environment are shown in Fig. 6.4. We can see that the RSSI per channel is relatively stable, i.e., only small variations less than 5 dBm are visible. Comparing the RSSI over all channels in Fig. 6.4a (at 3.41 m) shows that the difference between the highest and the lowest RSSI is less than 10 dBm. Nevertheless, there is no sudden change of RSSI from one channel to another; it is a smooth transition from higher to lower RSSI values. We also noticed this at 1.91 m, 5.46 m, and 8.46 m. Only at 12.01 m (Fig. 6.4b), where we could observe changes when discarding certain channel data, the RSSI shows higher fluctuations, i.e., several local minima are visible.

Wireless environment. Fig. 6.5a and Fig. 6.5b present the RSSI measured in the seminar room at 3.41 m and 4.5 m, respectively. These plots show a notable difference to the RSSI measured in the multipath-free environment, as we observe higher fluctuations

of RSSI over all channels. At 3.41 m, for instance, we see a rapid transition of RSSI from about −45 dBm at channel 36 to −30 dBm at channel 44, and from this point back to about −40 dBm at channel 52. Nevertheless, the RSSI is not always changing as fast as in the mentioned example (i.e., ±15 dBm over 3 channels), which is for example visible in the plot showing the RSSI at 4.5 m: the RSSI decreases from approximately −37 dBm at channel 100 to −47 dBm at channel 120, i.e., it changes by about −10 dBm over 6 channels. In the corridor (see Fig. 6.6), we observe these high variations of RSSI across all channels as well. For example, at 5.46 m, an even higher change of almost −20 dBm from channel 149 to channel 157 is visible.

The analysis of the scatter plots showed that the RSSI per channel shows no significant fluctuations over time in both wired and wireless environments. However, this is because our experiments were conducted in an almost static environment. We expect that in more realistic localization scenarios where we have a non-static environment, e.g., due to the movement of people or machines, the variations of RSSI per channel will be higher over time. If we compare the RSSI from wireless communications in multiple channels at a given position, some channels are always more affected by multipath than others. Consequently, we can see several local minima and maxima in the RSSI plots caused by the frequency-selective multipath fading. Based on this knowledge, we investigate whether there is a correlation between the RSSI changes across multiple channels and the removable channels.

Finding a Correlation between RSSI and removable Channels

Comparing the removable channels from our evaluation in Section 6.1 with the RSSI plots in Section 6.2, we see that these are *not* channels *at* local minima (i.e., where the multipath fading is strongest) but channels *next to* a local minimum. In Fig. 6.7, we have marked the channels with the smallest RSSI in orange and removable channels in green. We observe that removable channels often appear to the left or right of a channel with the lowest RSSI (i.e., a minimum), but not always: for example, at 4.5 m, we reduce the median error by 66.67 % when removing the data of channel 104 but its RSSI is not adjacent to a local minimum. Discarding the data for channel 116, which is to the left of the local minimum of RSSI at channel 120, reduces the 25th percentile error by 66.67 %; however, removing the data for channel 124, which is to the right of channel 120, increases the median and all other percentile errors by 80.95 %.

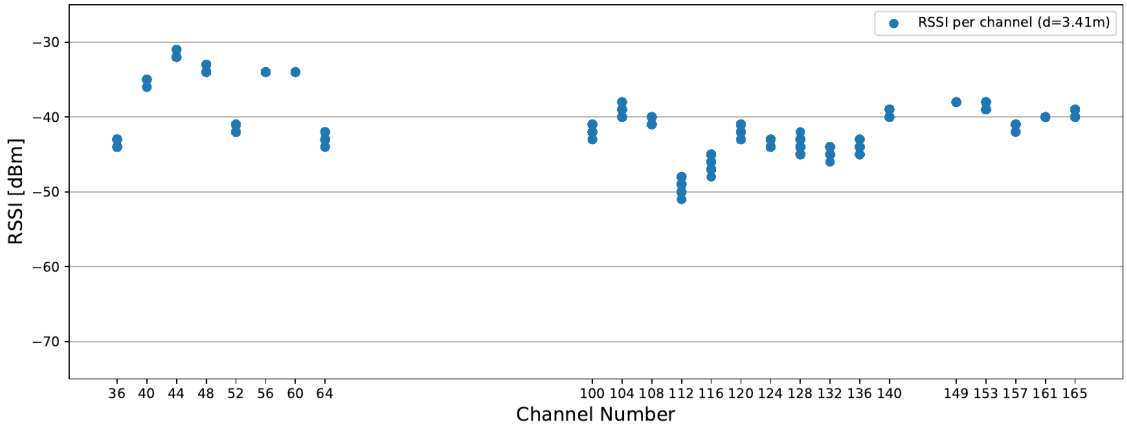
From all our experimental results of the seminar room and the corridor, we could not find a clear correlation between RSSI and removable channels that would guarantee successful discarding of data during live measurements. Although there is a high probability that a removable channel is next to a local minimum, we cannot distinguish whether it is the one on the left or the right side. Additionally, we also have cases where discarding channel data in the wireless environment does not affect the accuracy at all (e.g., at 3.41 m in the seminar room) and where the accuracy only worsens (e.g., at 5.46 m in the corridor). We assume that the combination of multipath effects, our chosen algorithm parameters for the multipath computation, and the implementation of the FHP as discussed in Chapter 5 are too strongly coupled for this simple analysis based on RSSI. Therefore, we conclude that RSSI alone is not a reliable metric for discarding channel data to improve accuracy.

Alternatives and Future Options

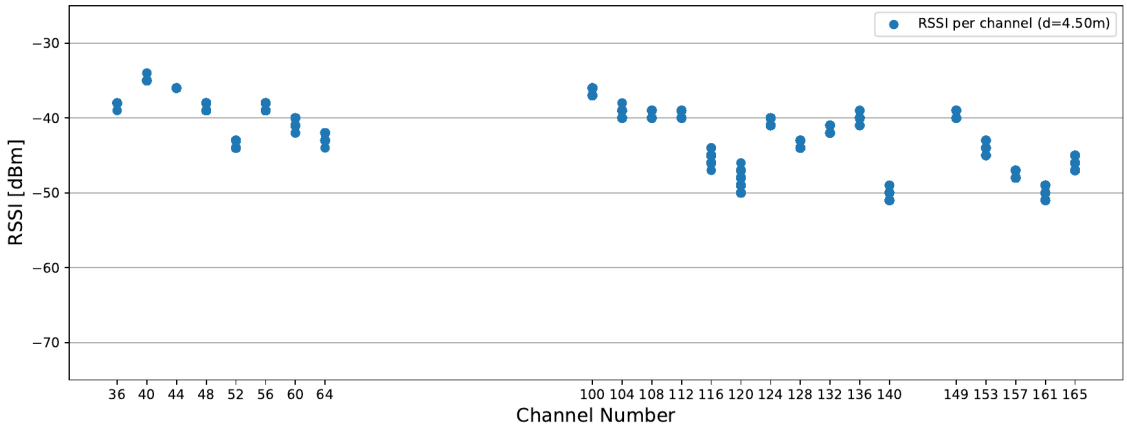
Since discarding channel data based on RSSI was not shown to be a good option for finding removable channels, we also looked for possible alternatives. One obvious alternative would be to use CSI amplitudes for this purpose as they describe multipath fading at a finer-grained level than RSSI, i.e., for each OFDM subcarrier per channel. Therefore, the CSI amplitudes could be used to identify which channels are more affected by multipath effects and thus experience higher phase errors than others. However, we investigated only superficially whether there is a correlation between CSI amplitudes and removable channels, but again, we could not observe one. Hence, we decided not to dive deeper into evaluating the suitability of CSI amplitudes for this purpose, as it would also require a more complex analysis due to the increased number of data (i.e., 56 values per channel compared to 1).

Instead, we investigated whether the removable channels stay the same when evaluating the data for a different time resolution. To this end, we performed the distance estimation again on the data from the seminar room ($N = 100$) with a time resolution of 0.33 ns ($\hat{=}$ 5 cm) and a sparsity of 9. We observed two things: first, the median ranging errors for the complete dataset could be further reduced in some cases, e.g., for 4.5 m, the median error decreased from 2.1 m to 0.75 m and for 1.0 m from 0.7 m to 0.45 m. For 3.41 m and 5.46 m, the median error stayed the same; for the latter, however, the 75th percentile error increased from 0.64 m to 3.51 m. For a distance of 1.91 m, we see a minor increase of the median error by 5 cm, i.e., from 0.39 m to 0.44 m. Second, we observed that the removable channels are different from those where we used a time resolution of 0.67 ns for the evaluation because the results were in general more accurate; there are still similarities in removable channels, like for the distances 4.5 m and 5.46 m, but none of them could reduce the median error, only the 75th or 90th percentile errors. When comparing the values in Table 6.2b and 6.4 removing data of channel 104, for instance, still increases the ranging accuracy, but if we discard channel 56, the median error increases by 180 %. For the other 3 distances, discarding channel data had almost no effect anymore, i.e., removing channel data did not increase the accuracy any further.

Therefore, we expect that even CSI amplitudes will not help to determine whether channel data is removable or not. For future work, we suggest not to concentrate on discarding channel data but on finding the best combination of time resolution and sparsity settings since this is crucial for obtaining accurate ranging results, as discussed in Chapter 5. Further ideas on future work are presented in Section 7.2.

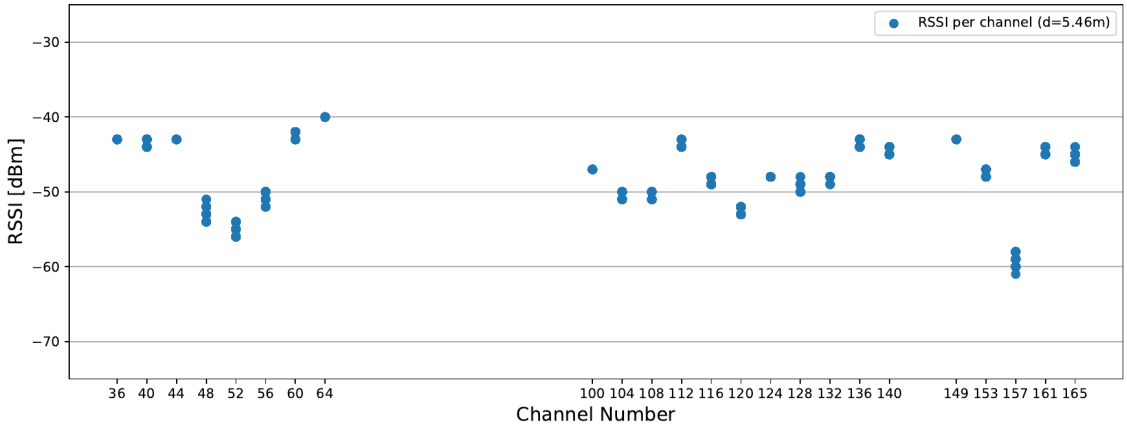


(a) Seminar room: $d = 3.41\,\mathrm{m}$

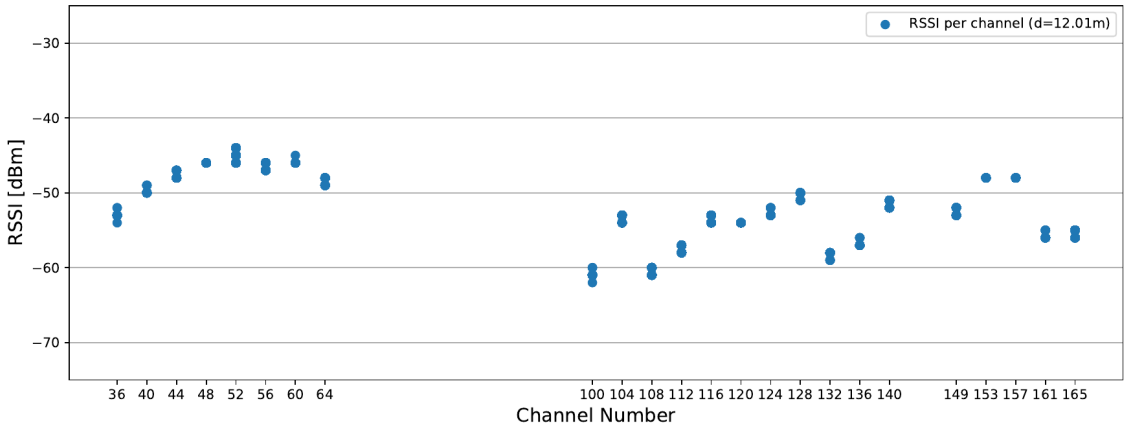


(b) Seminar room: $d = 4.50\,\mathrm{m}$

Figure 6.5: *Scatter plots of RSSI per channel in the* ***wireless*** *environment (****seminar room****) at* ***(a)*** 3.41 m *and* ***(b)*** 4.50 m.



(a) Corridor: $d = 5.46\,\mathrm{m}$



(b) Corridor: $d = 12.01\,\mathrm{m}$

Figure 6.6: *Scatter plots of RSSI per channel in the* ***wireless*** *environment (****corridor****) at* ***(a)*** 5.46 m *and* ***(b)*** 12.01 m.

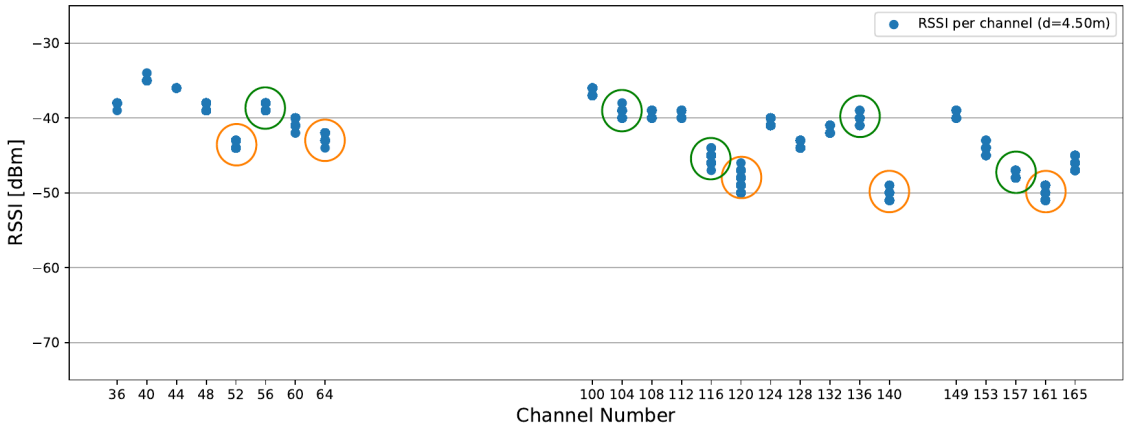


Figure 6.7: *RSSI plot with removable channels in the* ***seminar room*** *at* $d = 4.50\,\mathrm{m}$. Removable channels are marked green and the local minima with the smallest RSSI orange.

Discarded Channel i	$\delta_{i,50}$ **(Median)**	$\delta_{i,25}$ **(25th Percentile)**	$\delta_{i,75}$ **(75th Percentile)**	$\delta_{i,90}$ **(90th Percentile)**
None	0.00%	0.00%	0.00%	0.00%
36	400.00%	180.00%	78.57%	80.95%
40	180.00%	180.00%	0.00%	0.00%
44	180.00%	0.00%	0.00%	80.95%
48	180.00%	180.00%	0.00%	0.00%
52	180.00%	180.00%	80.95%	80.95%
56	180.00%	0.00%	0.00%	0.00%
60	180.00%	180.00%	0.00%	0.00%
64	0.00%	0.00%	-64.29%	-64.29%
100	180.00%	180.00%	0.00%	0.00%
104	0.00%	0.00%	-64.29%	-64.29%
108	180.00%	180.00%	0.00%	0.00%
112	180.00%	180.00%	0.00%	0.00%
116	0.00%	0.00%	-64.29%	-64.29%
120	0.00%	0.00%	-64.29%	-64.29%
124	406.67%	406.67%	80.95%	80.95%
128	406.67%	406.67%	80.95%	80.95%
132	180.00%	180.00%	0.00%	0.00%
136	0.00%	0.00%	-64.29%	-64.29%
140	180.00%	180.00%	0.00%	0.00%
149	406.67%	180.00%	80.95%	80.95%
153	180.00%	180.00%	0.00%	2.38%
157	0.00%	0.00%	-64.29%	-64.29%
161	180.00%	180.00%	0.00%	0.00%
165	180.00%	180.00%	0.00%	0.00%

Table 6.4: *Wireless environment (seminar room):* Ranging error changes (in percent) from the complete to the reduced dataset at 4.50 m for a time resolution of 0.33 ns

Chapter 7

Conclusion and Outlook

This chapter summarizes the contributions of this thesis and the results of our experimental evaluation in Section 7.1. We finally give an outlook and list possible future work in Section 7.2.

7.1 Conclusion

In this thesis, we brought Chronos [12], a well-known and state-of-the-art approach for WiFi-based distance estimation, to the latest Raspberry Pi hardware. The distance between two devices is estimated from the Time of Flight, which is obtained from sparse multipath profiles computed from CSI measurements in multiple channels. To this end, the availability of 24 channels in the 5 GHz band in the IEEE 802.11n standard is exploited by using a REQ-ACK-based frequency hopping protocol to perform bidirectional CSI measurements in each channel. Phase errors caused by Packet Detection Delay, Carrier Frequency Offset, and PLL Phase Offset are also addressed in order to enable accurate ranging with the CSI phase information.

The frequency hopping functionality is implemented in the firmware of the Raspberry Pi 4B's WiFi chip and CSI measurements are enabled by the Nexmon CSI Extraction Tool. In addition, our work also provides a firmware patch that allows frame injection in restricted 5 GHz channels, and we built a demonstrator that performs live distance estimation for multiple devices. The experimental evaluation in a multipath-free (wired) and typical multipath (wireless) environments showed that the time it takes to hop across all channels, the parameter settings for the multipath profile computation, and the transmission power, largely influence ranging accuracy. In particular, the combination of time resolution and sparsity is critical to obtain accurate results. Although the number of positions in each test setup was rather limited, the experiments that we carried out were sufficient to draw good conclusions regarding the effects of the aforementioned properties. Our system achieved a median ranging error of 0.36 m in the multipath-free environment (range: 1.91-12.01 m), 0.69 m in a seminar room (range: 1.0-5.46 m) and 2.16 m in a corridor (range: 5.46-12.01 m) in case the resolution of the multipath profile's time axis is set to 0.67 ns. Frequency-selective multipath effects in the wireless environment cause a reduction of accuracy compared to the multipath-free environment. Discarding the data of single channels, i.e., removable channels, before computing the multipath profiles can

increase the accuracy. However, our evaluation showed that is depends on the time resolution settings whether a channel is removable or not and thus RSSI and CSI amplitudes are no good indicators to identify them. We expect that the time it takes to exchange the REQ and ACK packets and the unavailability of 2.4 GHz channels for the frequency hopping, currently also limit the accuracy of our system. The most significant limitation that affects the applicability of the system for real world applications is that the CSI extraction breaks regular WiFi communication.

7.2 Future Work

Multipath profile computation. The combination of time resolution and sparsity is crucial for obtaining accurate ranging results. Analyzing further combinations of these settings could result in even more accurate results for the given data. Additionally, it makes sense to speed up the multipath profile computation by using, e.g., the SPGL-1 [80] or in-crowd [81] algorithm since the currently used algorithm is not very fast. The longer and finer-grained the time vector, the longer it takes the algorithm to converge.

Analyzing hardware properties. Since we did not yet investigate the impact of constant hardware delays on ToF estimations that were mentioned by Vasisht et al. [12], analyzing them and removing the offset from the results could further increase the ranging accuracy. Furthermore, finding a way to enable faster frame injection (i.e., faster channel hopping) and the usage of 2.4 GHz is also expected to be beneficial to decrease distance errors. Disabling the Clear Channel Assessment (CCA) might help to achieve these goals.

Performance in dynamic and NLOS environments. Currently we conducted our experiments only in LOS scenarios. Real indoor localization systems, however, are typically used in dynamic environments where we also face NLOS conditions. Movement of devices and NLOS communication usually degrade the localization performance. Therefore, investigating the accuracy for realistic scenarios is necessary to evaluate the capability of our system for real world applications.

Localization. In case the ranging accuracy can be increased for multipath environments, testing the localization accuracy of our system, e.g., by using four Raspberry Pis as anchor nodes that localize a target using trilateration, is also of our interest.

Interoperability with other devices. The availability of COTS hardware that allows CSI to be extracted and used for WiFi-based localization or sensing is currently rather limited. We think that the list of devices supported by the Nexmon CSI Extraction Tool should be extended in the future and therefore it could be interesting to bring the frequency hopping approach also on smartphones and routers. Using routers with multiple antennas, for example, would allow to evaluate Chronos' performance for single-AP localization.

Nevertheless, as long as the CSI extraction with Nexmon breaks the regular WiFi communication, it makes sense to focus on indoor localization systems performing passive CSI measurement, i.e., where the device on which the CSI is extracted is not participating in the direct WiFi communication.

Appendix A

Reverse Engineering

For our work we needed some firmware functions that were not directly accessible on the Raspberry Pi 4B (i.e., the `BCM43455c0`). Therefore, we had to reverse engineer the Broadcom FullMAC firmware binary to get the addresses of these functions. The information provided by Nexmon [37] and Hugues Anguelkov's [83] step-by-step guide about reverse engineering Broadcom WiFi chips, helped us with that.

The firmware consists of two parts: the RAM and ROM part. The former is already contained in the Nexmon framework and can also be found in `/lib/firmware/brcm` on a Linux system (e.g., the Raspberry Pi OS). This part of the firmware is writable and loaded to the WiFi chip's RAM by the driver (e.g., the *brcmfmac* on Linux). To get the second part of the firmware, we used Nexmon's ROM extraction patch[1] which we modified to make it work for firmware version `7_45_189`. The complete firmware binary is obtained by placing the ROM and RAM contents at their defined base addresses; for the `BCM43455c0` these addresses are `0x000000` and `0x198000`, respectively.

We use the software reverse engineering tool Ghidra [79] to get a disassembled version of the firmware and, with the help of existing code, we were able to find the addresses of all firmware functions that we needed in order to implement the Chronos firmware patch. An overview of these functions and a short description for each of them is shown in Table A.1.

[1] `https://github.com/seemoo-lab/nexmon/tree/master/patches/bcm43455c0/7_45_154/rom_extraction`

Firmware Function	Description	Address
wlc_radar_chanspec	Used by the firmware to figure out whether a channel is radar sensitive or not. For radar sensitive channels it returns 1, otherwise 0.	0x58a04
wlc_quiet_chanspec	Used by the firmware to figure out if the quiet bit is set for the channel (i.e., if it can be used for transmission or not). Returns 1 if the quiet bit is set, otherwise 0.	0x58950
wlc_clr_quiet_chanspec	Clears the quiet bit for a channel.	0x57b70
wlc_set_quiet_chanspec	Sets the quiet bit for a channel.	0x58ce0
wlc_bmac_mute	Mute or unmute a channel; quiet channels are always muted.	0x49874
wlc_hwtimer_free_timeout	Frees the allocated timer in the D11 core.	0x5daa4
hndrte_del_timer	Deletes timeouts for the FullMAC timer.	0x19a468
wlc_d11hdrs	Appends the D11 headers to the WiFi frame before it is enqueued in the TX FIFO.	0x9c2fc
wlc_get_txh_info	Required to get the TX header information for frame injection.	0x1a9880
wlc_txfifo	Enqueues the WiFi frames in the TX FIFO.	0x1b1318
wlc_valid_chanspec	Validates the chanspec for the current locale by calling `wlc_valid_chanspec_ext` (dualband = False)	0x85eac
wlc_valid_chanspec_db	Validates the chanspec for the current locale by calling `wlc_valid_chanspec_ext` (dualband = True)	0x58eb4

Table A.1: Firmware functions and their reverse engineered addresses.

References

[1] F. Zafari, A. Gkelias, and K. K. Leung. "A Survey of Indoor Localization Systems and Technologies". In: *IEEE Communications Surveys and Tutorials* vol. 21, no. 3, pages (Apr. 2019), pp. 2568–2599.

[2] Wen Liu et al. "Survey on CSI-based Indoor Positioning Systems and Recent Advances". In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2019, pp. 1–8.

[3] Mehdi Mohammadi et al. "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services". In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 624–635. DOI: 10.1109/JIOT.2017.2712560.

[4] Ahmed Makki et al. "Survey of WiFi Positioning Using Time-Based Techniques". In: *Comput. Netw.* 88.C (Sept. 2015), 218–233. DOI: 10.1016/j.comnet.2015.06.015.

[5] Jiang Xiao et al. "A Survey on Wireless Indoor Localization from the Device Perspective". In: *ACM Comput. Surv.* 49.2 (June 2016). URL: https://doi.org/10.1145/2933232.

[6] Hui Liu et al. "Survey of Wireless Indoor Positioning Techniques and Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), pp. 1067–1080. DOI: 10.1109/TSMCC.2007.905750.

[7] Mikkel Baun Kjærgaard et al. "Indoor Positioning Using GPS Revisited". In: *Pervasive Computing*. Ed. by Patrik Floréen, Antonio Krüger, and Mirjana Spasojevic. Springer Berlin Heidelberg, 2010, pp. 38–56.

[8] Apple iPhone 12. https://www.apple.com/at/iphone-12/specs/ – Last accessed: 2021-05-14.

[9] Samsung Galaxy S21 Ultra 5G. https://www.samsung.com/at/smartphones/galaxy-s21-ultra-5g/ – Last accessed: 2021-05-14.

[10] Qorvo UWB Transceiver Evaluation Kit for the DW1000. https://www.qorvo.com/products/p/EVK1000 – Last accessed: 2021-05-14.

[11] NXP Trimension UWB Development Kit. https://www.nxp.com/products/wireless/secure-ultra-wideband-uwb/trimension-uwb-development-kit:MK-UWB-DEV-KIT – Last accessed: 2021-05-14.

[12] Deepak Vasisht, Swarun Kumar, and Dina Katabi. "Decimeter-Level Localization with a Single WiFi Access Point". In: *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, Mar. 2016, pp. 165–178.

[13] Navid Tadayon et al. "Decimeter Ranging With Channel State Information". In: *IEEE Transactions on Wireless Communications* 18.7 (2019), pp. 3453–3468. DOI: 10.1109/TWC.2019.2914194.

[14] Manikanta Kotaru et al. "SpotFi: Decimeter Level Localization Using WiFi". In: *SIGCOMM Comput. Commun. Rev.* 45.4 (Aug. 2015), 269–282. URL: https://doi.org/10.1145/2829988.2787487.

[15] Jiao Liu, Guanlong Teng, and Feng Hong. "Human Activity Sensing with Wireless Signals: A Survey". In: *Sensors* 20.4 (2020). DOI: 10.3390/s20041210.

[16] Zheng Yang, Zimu Zhou, and Yunhao Liu. "From RSSI to CSI: Indoor Localization via Channel Response". In: *ACM Comput. Surv.* 46.2 (Dec. 2013). URL: https://doi.org/10.1145/2543581.2543592.

[17] K. Wu et al. "FILA: Fine-grained indoor localization". In: *2012 Proceedings IEEE INFOCOM*. 2012, pp. 2210–2218.

[18] Zhengjie Wang et al. "CSI-based human sensing using model-based approaches: a survey". In: *Journal of Computational Design and Engineering* 8.2 (2021), pp. 510–523.

[19] Yongsen Ma, Gang Zhou, and Shuangquan Wang. "WiFi Sensing with Channel State Information: A Survey". In: *ACM Comput. Surv.* 52.3 (June 2019). URL: https://doi.org/10.1145/3310194.

[20] Hongzi Zhu et al. "π-Splicer: Perceiving Accurate CSI Phases with Commodity WiFi Devices". In: *IEEE Transactions on Mobile Computing* 17.9 (2018), pp. 2155–2165. DOI: 10.1109/TMC.2018.2793222.

[21] Daniel Halperin et al. "Tool Release: Gathering 802.11n Traces with Channel State Information". In: *ACM SIGCOMM CCR* 41.1 (Jan. 2011), p. 53.

[22] Yaxiong Xie, Zhenjiang Li, and Mo Li. "Precise Power Delay Profiling with Commodity WiFi". In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. MobiCom '15. ACM, 2015, 53–64. URL: http://doi.acm.org/10.1145/2789168.2790124.

[23] Intel WiFi Link 5300. *Datasheet*. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ultimate-n-wifi-link-5300-brief.pdf – Last accessed: 2021-05-16.

[24] Steven M Hernandez and Eyuphan Bulut. "Lightweight and Standalone IoT based WiFi Sensing for Active Repositioning and Mobility". In: *21st International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM) (WoWMoM 2020)*. June 2020.

[25] Intel WiFi Link 5300. *Support Information*. https://www.intel.com/content/www/us/en/support/products/70971/wireless/legacy-intel-wireless-products/intel-wireless-series/intel-wifi-link-5300.html – Last accessed: 2021-05-16.

[26] Atheros CSI Tool. *Compatible Hardware*. https://wands.sg/research/wifi/AtherosCSI/Hardware.html – Last accessed: 2021-05-16.

[27] Patricia García Ferrín. *WiFi Indoor Localization Using Channel State Information*. http://resolver.tudelft.nl/uuid:6d5a3afd-1966-4357-b063-7a82c0fdb0ab – Last accessed: 2021-05-16. Apr. 2019.

[28] Francesco Gringoli et al. "Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets". In: *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*. WiNTECH '19. 2019, 21–28. URL: https://doi.org/10.1145/3349623.3355477.

[29] Muhammad Atif et al. "Wi-ESP—A tool for CSI-based Device-Free Wi-Fi Sensing (DFWS)". In: *Journal of Computational Design and Engineering* 7.5 (May 2020), pp. 644–656. URL: https://doi.org/10.1093/jcde/qwaa048.

[30] Francesca Meneghello et al. *Environment and Person Independent Activity Recognition with a Commodity IEEE 802.11ac Access Point*. 2021. URL: {https://arxiv.org/abs/2103.09924}.

[31] Glenn Forbes, Stewart Massie, and Susan Craw. "Wifi-based human activity recognition using raspberry pi." In: Institute of Electrical and Electronics Engineers. 2020.

[32] Tao Li et al. "A Novel Gesture Recognition System Based on CSI Extracted from a Smartphone with Nexmon Firmware". In: *Sensors* 21.1 (2021), p. 222.

[33] Zhihui Gao et al. "CRISLoc: Reconstructable CSI Fingerprintingfor Indoor Smartphone Localization". In: *arXiv e-prints* (Oct. 2019).

[34] Aravind Reddy Voggu, Vikas Vazhayily, and Madhav Ra. "Decimeter Level Indoor Localisation with a Single WiFi Router using CSI Fingerprinting". In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. 2021, pp. 1–5. DOI: 10.1109/WCNC49053.2021.9417483.

[35] Xianan Zhang et al. "Peer-to-Peer Localization for Single-Antenna Devices". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4.3 (2020), pp. 1–25.

[36] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. *Nexmon: The C-based Firmware Patching Framework*. 2017. URL: \url{https://nexmon.org}.

[37] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. "The Nexmon firmware analysis and modification framework: Empowering researchers to enhance Wi-Fi devices". In: *Computer Communications* 129 (2018), pp. 269–285.

[38] Elena Lopez-Aguilera, Eduard Garcia-Villegas, and Jordi Casademont. "Evaluation of IEEE 802.11 coexistence in WLAN deployments". In: *Wireless Networks* 25.1 (2019), pp. 87–104.

[39] Ramia Babiker Mohammed Abdelrahman, Amin Babiker A Mustafa, and Ashraf A Osman. "A Comparison between IEEE 802.11 a, b, g, n and ac Standards". In: *IOSR Journal of Computer Engineering (IOSR-JEC)* 17 (2015), pp. 26–29.

[40] Priya Sharma and Gurpreet Singh. "Comparison of Wi-Fi IEEE 802.11 Standards Relating to Media Access Control Protocols". In: *International Journal of Computer Science and Information Security,* 14 (Oct. 2016), pp. 856–862.

[41] DrayTek. *The 5 Ghz Wireless Band*. `https://www.draytek.co.uk/support/guides/the-5ghz-wireless-band` – Last accessed: 2021-06-03.

[42] ETSI. *5 GHz RLAN; Harmonised Standard covering the essential requirements of article 3.2 of Directive 2014/53/EU (2017-05)*. ETSI EN 301 893 V2.1.1. `https://www.etsi.org/deliver/etsi_en/301800_301899/301893/02.01.01_60/en_301893v020101p.pdf` – Last accessed: 2021-05-24.

[43] Matthew S. Gast. *802.11ac: A Survival Guide*. O'Reilly Media, Inc., 2013. Chap. 2. The PHY, Table 2-1. Channel description attributes. ISBN: 9781449343149. URL: `oreilly.com/library/view/80211ac-a-survival/9781449357702/ch02.html#table-channel-description`.

[44] K. McClaning. *Wireless Receiver Design for Digital Communications*. Telecommunications Series. Institution of Engineering and Technology, 2012. Chap. 3, pp. 189–206. ISBN: 9781891121807. URL: `https://m.eet.com/media/1116127/mcclaning_3_pt2.pdf`.

[45] Daniel Halperin et al. "802.11 with multiple antennas for dummies". In: *ACM SIGCOMM Computer Communication Review* 40.1 (2010), pp. 19–25.

[46] Gough Lui et al. "Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of WLAN localization". In: *2011 International Conference on Localization and GNSS (ICL-GNSS)*. 2011, pp. 53–57. DOI: `10.1109/ICL-GNSS.2011.5955283`.

[47] Cypress. *CYW43455*. Single-Chip 5G WiFi IEEE 802.11n/ac MAC/Baseband/ Radio with Integrated Bluetooth 5.0 `https://www.cypress.com/file/358916/download` – Last accessed: 2021-05-25.

[48] Neal Patwari and Joey Wilson. "Spatial models for human motion-induced signal strength variance on static links". In: *IEEE Transactions on Information Forensics and Security* 6.3 (2011), pp. 791–802.

[49] Praveen Kumar, Lohith Reddy, and Shirshu Varma. "Distance measurement and error estimation scheme for RSSI based localization in Wireless Sensor Networks". In: *2009 Fifth international conference on wireless communication and sensor networks (WCSN)*. IEEE. 2009, pp. 1–4.

[50] Fasih Ullah Khan et al. "A Comparison of Wireless Standards in IoT for Indoor Localization Using LoPy". In: *IEEE Access* (2021).

[51] Elisabeth Salomon. *FILA based Distance Estimation with Raspberry Pis using Nexmon CSI Extractor*. Seminar Project. 2020.

[52] Z. Li, T. Braun, and D. C. Dimitrova. "A passive WiFi source localization system based on fine-grained power-based trilateration". In: *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2015, pp. 1–9.

[53] Wan-Mai Mak Josyl Mariela Rocamora Ivan Wang-Hei Ho and Alan Pak-Tao Lau. "Survey of CSI fingerprinting-based indoor positioning and mobility tracking systems". English. In: *IET Signal Processing* 14 (7 Sept. 2020), 407–419(12).

[54] Tom Van Haute et al. "Performance analysis of multiple Indoor Positioning Systems in a healthcare environment". In: *International journal of health geographics* 15.1 (2016), pp. 1–15.

[55] Scott Y Seidel and Theodore S Rappaport. "914 MHz path loss prediction models for indoor wireless communications in multifloored buildings". In: *IEEE transactions on Antennas and Propagation* 40.2 (1992), pp. 207–217.

[56] Amir Guidara et al. "Impacts of Temperature and Humidity variations on RSSI in indoor Wireless Sensor Networks". In: *Procedia Computer Science* 126 (2018), pp. 1072–1081.

[57] Wilson Sakpere, Michael Adeyeye-Oshin, and Nhlanhla BW Mlitwa. "A state-of-the-art survey of indoor positioning and navigation systems and technologies". In: *South African Computer Journal* 29.3 (2017), pp. 145–197.

[58] Ahmed Azeez Khudhair et al. "Wireless indoor localization systems and techniques: survey and comparative study". In: *Indonesian Journal of Electrical Engineering and Computer Science* 3.2 (2016), pp. 392–409.

[59] Ali Yassin et al. "Recent advances in indoor localization: A survey on theoretical approaches and applications". In: *IEEE Communications Surveys & Tutorials* 19.2 (2016), pp. 1327–1346.

[60] Laxima Niure Kandel and Shucheng Yu. "Indoor localization using commodity wi-fi aps: Techniques and challenges". In: *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE. 2019, pp. 526–530.

[61] Daniel Halperin et al. *Linux 802.11n CSI Tool: FAQ, Things to Know, and Troubleshooting*. See FAQ, Question 2 - B. Inspecting the CSI: `https://dhalperi.github.io/linux-80211n-csitool/faq.html` – Last accessed: 2021-05-17.

[62] Jiang Xiao et al. "FIFS: Fine-grained indoor fingerprinting system". In: *2012 21st international conference on computer communications and networks (ICCCN)*. IEEE. 2012, pp. 1–7.

[63] Hao Chen et al. "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information". In: *IEEE Access* 5 (2017), pp. 18066–18074.

[64] Ralph Schmidt. "Multiple emitter location and signal parameter estimation". In: *IEEE transactions on antennas and propagation* 34.3 (1986), pp. 276–280.

[65] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. "ToneTrack: Leveraging frequency-agile radios for time-based indoor wireless localization". In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 2015, pp. 537–549.

[66] Rice Univ. Wireless Open Access Research Platform (WARP). *WARP Project*. `http://warpproject.org` – Last accessed: 2021-06-11.

[67] Souvik Sen et al. "Avoiding multipath to revive inbuilding WiFi localization". In: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. 2013, pp. 249–262.

[68] Chen Chen et al. "Achieving centimeter-accuracy indoor localization on WiFi platforms: A frequency hopping approach". In: *IEEE Internet of Things Journal* 4.1 (2016), pp. 111–121.

[69] H. Rahul, S. Kumar, and D. Katabi. "MegaMIMO: Scaling Wireless Capacity with User Demands". In: *ACM SIGCOMM* (2012).

[70] *Nexmon - GitHub repository.* `https://github.com/seemoo-lab/nexmon` – Last accessed: 2021-06-14.

[71] *tcpdump - command-line packet analyzer.* `https://www.tcpdump.org/` – Last accessed: 2021-06-15.

[72] *Nexmon CSI Extraction Tool - GitHub repository.* `https://github.com/seemoo-lab/nexmon_csi` – Last accessed: 2021-06-14.

[73] Mikhail Zakharov. *Modified Nexmon CSI Extraction Tool: adds RSSI and Frame Control to the UDP packet).* `https://github.com/mzakharo/nexmon_csi` – Last accessed: 2021-06-14.

[74] *Raspberry Pi 4 Model B.* `https://www.raspberrypi.org/products/raspberry-pi-4-model-b` – Last accessed: 2021-06-16.

[75] *Wireshark - network protocol analyzer.* `https://www.wireshark.org/` – Last accessed: 2021-06-15.

[76] Matthias Schulz et al. "Massive reactive smartphone-based jamming using arbitrary waveforms and adaptive power control". In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2017, pp. 111–121.

[77] Fraida Fund. *Understanding the 802.11 Wireless LAN MAC frame format.* `https://witestlab.poly.edu/blog/802-11-wireless-lan-2/` – Last accessed: 2021-06-15. 2017.

[78] Matthew Gast. *802.11 wireless networks: the definitive guide.* O'Reilly Media, Inc., 2005. ISBN: 9780596100520.

[79] *Ghidra - A software reverse engineering (SRE) suite of tools developed by NSA's Research Directorate in support of the Cybersecurity mission.* `https://ghidra-sre.org/` – Last accessed: 2021-06-15.

[80] E. van den Berg and M. P. Friedlander. "Probing the Pareto frontier for basis pursuit solutions". In: *SIAM Journal on Scientific Computing* 31.2 (2008), pp. 890–912. DOI: 10.1137/080714488. URL: http://link.aip.org/link/?SCE/31/890.

[81] Patrick R Gill, Albert Wang, and Alyosha Molnar. "The in-crowd algorithm for fast basis pursuit denoising". In: *IEEE Transactions on Signal Processing* 59.10 (2011), pp. 4595–4605.

[82] Mini-Circuits. *15542 VAT-20+, SMA Fixed Attenuator,* 20 dB*, DC*-6000 MHz*,* 50 Ω. `https://www.minicircuits.com/pdfs/VAT-20+.pdf` – Last accessed: 2021-06-20.

[83] Hugues Anguelkov. *Reverse-engineering Broadcom wireless chipsets.* `https://blog.quarkslab.com/reverse-engineering-broadcom-wireless-chipsets.html` – Last accessed: 2021-06-19. 2019.