FH JOANNEUM
University of Applied Sciences

Direct Peer to Peer Communication on iOS Devices

When mediator based technology is not available

Bachelor's Thesis

submitted in conformity with the requirements for the degree of

Bachelor of Science in Engineering (BSc)

Bachelor's degree programme **Software Design and Cloud Computing**

FH JOANNEUM (University of Applied Sciences), Kapfenberg

Supervisor: DI Johannes Feiner

Submitted by: Matthias Bartholomäus

February 2025

TODO: Specify the title, subtitle, author, submission date, study, language, your name, and supervisor/advisor in the main *thesis.typ* file. Then compile with *typst compile thesis.typ*. Finally, remove all TODOs (todo marcos) within your typst ource code.

Preview printed 2025-04-25.

Abstract

Modern mobile devices can make use of a wide variety of communication technologies. Besides having different applicabilities and protocols, standards like Bluetooth, WiFi or 5G need to be wireless to seamlessly transfer data. In resent years global economic demand has impacted research and development to vastly improve data transmission and hardware on smartphones. As of 2024 this led to over 4 billion smartphones users worldwide (Department, 2024), 27% covered by iOS. Unfortunately most of the communication methods used on smartphones all rely on mediators. Be it a router in a local network or a cell tower in a cellular network, without these nodes a connection between two peers can not be established, no matter how close neighboring devices may be. However in scenarios where the required infrastructure is not available, communication between two mobile iOS devices can be established via Bluetooth or ad hoc WiFi since these do not required pre-existing infrastructure and purely rely on local radio broadcast, whereas the latter is recommended to use. Due to the latest advancements in these technologies, it is unclear how good direct Peer to Peer (P2P) networks work under different surroundings and how the choice of the transport protocol affects the connection. In this thesis, an iOS prototype is developed, that connects two peers via Apple Wireless Direct Link (AWDL) and measures metrics that describe the quality of the connection like Round Trip Time (RTT), Jitter, data speed and more. The results show that AWDL achieves the best metrics when tested in surroundings with low radio frequency pollution. Prototype measurements also show that UDP achieves the fastest data rates, with a compromise on data loss. In consideration of the findings, it can be stated that AWDL on iOS devices is not yet ready for wide area communication or building reliable mesh networks, but can be utilized for low distance applications.

Keywords: FHJ, SWD, iOS, peer-to-peer, ad-hoc, smartphone, Apple

Kurzfassung

Moderne Mobilgeräte können eine Vielzahl an Kommunikationstechnologien nutzen. Neben unterschiedlichen Anwendungsmöglichkeiten und Protokollen müssen Standards wie Bluetooth, WiFi oder 5G drahtlos sein, um Daten im modernen Kontext übertragen zu können. In den letzten Jahren hat sich die weltweite wirtschaftliche Nachfrage auf die Forschung und Entwicklung ausgewirkt, und so die Datenübertragung und die Hardware von Smartphones erheblich verbessert. Dies hat dazu geführt, dass im Jahr 2024 weltweit über 4 Milliarden Smartphones genutzt werden (Department, 2024), 27 % davon unter iOS. Leider sind die meisten der auf Smartphones verwendeten Kommunikationsmethoden auf Infrastruktur angewiesen. Ein Router in einem lokalen Netz oder ein Sendemast in einem Mobilfunknetz, ohne diese Knoten kann keine Verbindung zwischen zwei Endgeräten hergestellt werden, egal wie nah diese sein mögen. In diesen Szenarien kann die Kommunikation zwischen zwei mobilen iOS-Geräten über Bluetooth oder Ad-hoc-WiFi hergestellt werden, da diese keine bereits vorhandene Infrastruktur benötigen und sich ausschließlich auf lokale Broadcasts stützen, wobei letztere Technologie empfohlen wird. Aufgrund der jüngsten Fortschritte bei diesen Technologien ist unklar, wie gut direkte P2P-Netzwerke in verschiedenen Umgebungen funktionieren und wie die Wahl des Transportprotokolls die Verbindung beeinflusst. In dieser Arbeit wird ein iOS-Prototyp entwickelt, der zwei Endgeräte über Apple Wireless Direct Link (AWDL) verbindet und Metriken misst, die die Qualität der Verbindung beschreiben, wie Round Trip Time (RTT), Jitter, Datengeschwindigkeit und mehr. Die Ergebnisse zeigen, dass AWDL die besten Werte erzielt, wenn es in einer Umgebung mit geringer Funkfrequenzbelastung getestet wird. Messungen des Prototypen zeigen auch, dass UDP die schnellste Datenübertragungsrate erreicht, allerdings mit einem Kompromiss bei den Datenverlusten. Ein Resümee zeigt, dass AWDL auf iOS-Geräten noch nicht für die Weitbereichskommunikation oder den Aufbau zuverlässiger Mesh-Netzwerke geeignet ist, aber für Anwendungen mit geringer Entfernung eingesetzt werden kann.

Contents

List of Figures

List of Tables

List of Listings

List of Listings

Optionally, you might add an **Acknowledgements** section (in German **Danksagung**) to say thank you or give credits to someone.

1 | Introduction

Nowadays communication between smartphones or tablets invariably relies on infrastructure. In most homes in developed countries one can find a local network advertised by WiFi technology and handled by a router. Being outside and not connected to a local network, mobile devices communicate with cellular towers that give access to a big underlying network managed by an ISP. Wherever we go, our mobile devices are connected, but relying on infrastructure that is not necessarily available all the time. Different reasons can lead to restricted or broken infrastructure which, in absence of ad-hoc technologies would break all communication. In countries with governmental protests cellular networks might be restricted or monitored, infrastructure could be overwhelmed, due to a high number of people trying to access the system simultaniously or environmental disasters could take down cellular towers or their power supply. In all these cases users would highly profit from ad-hoc networks, that do not merely rely on any external dependencies.

Nowadays many different technologies exist to gain access to wide area ad-hoc connections like LTE-Direct, 5G Sidelink (SL) or LoraWan. Unfortunately these technologies are not yet supported in smartphones and there is no guarantee that they will ever be. However modern smartphones improved support for local peer-to-peer (P2P) technologies in recent years, that allow devices to directly communicate via local radio link broadcasts. This could improve connectivity in the aforementioned cases or enable scenarios in which numerous end devices connect together, to form a wide spread mesh network. The first technology that comes to mind is Bluetooth. First introduced in 1998 has already been used in mobile computing the last decades, but due to limited data transfer rates a new approach started to enter the market. These days WiFi is predominantly used to facilitate local P2P connections between smartphones, whereby unfortunately every mobile operating system has created its own version which makes communication between different vendors complex. Furthermore, the extent to which these technologies can provide stable data transfer and the quality with which they can do so, remains unclear or poorly documented. Since this information is essential for a further evaluation of the feasibility of applications based on direct P2P connections, this thesis aims to address the existing uncertainty by measuring metrics that can assist the evaluation of analogous applications.

As of 2024 4 billion people worldwide are carrying smartphones (Department, 2024) with them. Twenty seven percent of them are developed and manufactured by one of the most valuable companies in the world. Since 2007, when the first iPhone was released on 27th of June Apple has sold more than 2.3 billion devices. This makes iOS one of the most used mobile operating systems worldwide with a current market share of 27%. iOS also utilizes its own implementation of a P2P WiFi protocol which is strongly used by Apples Continuity which bundles applications like AirDrop or AirPlay, Universal Clipboard, Handoff or WiFi password sharing for contacts that want to join your network.

Apples P2P WiFi implementation is called Apple Wireless Direct Link (AWDL) and is an undocumented protocol that is based on the 802.11 ad-hoc WiFi standard which did not live up to its expectations. This protocol is not only used internally by Continuity but is also accessible to iOS application developers via various frameworks that operate on different layers of abstraction.

Although Bluetooth is more versatile, compatible with devices from other vendors and has improved range and speed in mobile end devices in recent years, AWDL is the recommended technology to establish direct links between iOS devices.

Most of networking in mobile devices relies on the TCP/IP protocol suite. It splits transferring data into Application, Transport, Internet, and Link Layers, each playing a specific role in ensuring data transmission.

The Application Layer is responsible for encapsulating domain specific data and is the top most of layers, which passes data down to the Transport Layer. It ensures that data is transmitted reliably using protocols like TCP for error-checked delivery and congestions control via Congestive-Avoidance Algorithms (CAA) or UDP for faster and connectionless communication. This layer breaks the data into smaller segments and passes them to the Internet Layer, which is responsible for logical addressing and routing network traffic. In 2012 another Transport protocol was developed by engineers at Google called QUIC. It is built upon UDP and is supposed to obsolete TCP for applications that rely on ordered and error checked data because of its faster connection establishment and built in encryption. Apple added support for QUIC to the Networking Framework starting form iOS 15.

After all the described aspects have been taken into consideration, the Networking Framework, included in the iOS SDK is used to establish and test the quality of P2P AWDL connections between iOS devices. The tests will alternate using TCP, UDP and QUIC in different surroundings that represent typical locations, varying in crowd density or general ambient radiofrequency electromagnetic field (RF-EMF) levels.

1.1 Research Definition

This research should help assess feasibility of applications that want to utilise AWDL. It will compare different transport layer protocols (TCP, UDP and QUIC) in different locations that represent common places (City, Underground, Field and Forrest) to cover various characteristics of real life scenarios. The tests will vary in distance, size of packages and number of packages. The Networking Framework is used to quantify the connection quality through metrics like RTT, jitter, package loss and data transfer speed.

Research Questions

Which aspects influence the P2P AWDL connection quality on iOS devices and how far can it reach?

Hypothesis

H_1

P2P AWDL connection quality on iOS devices depends on the surroundings and functions worse in crowded areas.

H_2

P2P AWDL connection quality on iOS devices depends on the transport layer protocol.

Method

To measure the stated connection metrics a prototype application will be developed for the iOS platform that will serve as a tool to measure the connection quality. The metrics will be precisely defined in the test protocol in Section 5 after describing the implementation details of the prototype. Measurement of connection quality will be purely based on values captured through the Network

Framework by the prototype app itself. The characteristics of the environment will be described based on human perceive and measured with suitable methods.

1.2 Summary

Nowadays Smartphones mostly rely on infrastructure networks. As this is a strong dependency that could vastly limit access and advancements in P2P connection soft- and hardware have emerged in recent years it is unclear which connection qualities these methods can produce. Therefore a prototype application developed for the iOS platform serves as a utility for measuring selected metrics.

2 | Related Work

The following is a non-comprehensive discussion covering previous research of peer-to-peer technologies in the mobile context. After covering historic considerations of mobile ad hoc networks (MANET) research and device to device D2D communication in cellular networks via standards like LTE-Direct or 5G New radio (NR) Sidelink (SL), a deeper introduction to Apples ecosystem and AWDL is given, where a lot is based on the open wireless link (OWL) project from the TU Darmstadt. While reverse engineered the AWDL protocol, the team found several security concerns in Apple's operating systems and developed some open source applications for public use which they shared on GitHub.

2.1 History of MANET

Already back in 2001 the Proem project (Kortuem, Schneider, Preuitt, *et al.*, 2002) examined different aspects of peer-to-peer applications for mobile ad hoc networks (MANET) to enable proximity-based collaboration. In particular, Proem was an approach to provide high-level support for mobile peer-to-peer application developers and was tested by students of the University of Oregon, whilst creating an MP3 file-sharing system. They already noticed the trend for an ever-larger becoming applicability of personal mobile devices for data sharing but listed resources of mobile devices among other possible limitations. This facet has vastly changed since then and several new ideas like ShAir (Dubois, Bando, Watanabe, *et al.*, 2013), a middleware infrasructure for peer-to-peer sharing between mobile devices or mFerio (Balan, Ramasubbu, Prakobphol, *et al.*, 2009), a peer-to-peer mobile payment system have emerged.

Balan, Ramasubbu, Prakobphol, *et al.* working on mFerio already noticed the problem that mobile devices rely too heavily on static infrastructure. Back then cell phones have already become popular tools that combined calendars, address books, messaging or cameras. The increasing need to use them as a payment vehicle has become ever larger and the authors questioned the state of the art implementations back then. In particular mobile payment solution required constantly stable connections via either SMS or GSM/CDMA based technologies which were connected to a backend payment server. They noticed that these implementation, which were to heavily relying on external systems could not replace cash based systems and aimed to develop a decentraliced approach based on NFC. The goal of their larger term project aimed to create a digital wallet for cellphones which would allow users to store everything on the device which has previously been in their physical wallets, like credit cards, identification or tickets. The applicabilities of this project strongly remind of the Apple Wallet, which was introduced in iOS 6 in 2012 and also leverages NFC.

Some years later in 2013 ShAir was developed as by Dubois, Bando, Watanabe, *et al.*, a structured software engineering project written in Java that used WiFi technology on Android devices to share data between them. While Wifi-direct and Bluetooth were also accessible to the developers, they decided to use a combination of WiFi AP mode and WiFi Client mode in a random fashion to create dynamic networks and discover nearby peers because devices would not allow the former without

active user interaction. They tested the app by sharing pictures among 12 devices from several vendors using no fixed existing infrastructure. This project also strongly reminds on Apple propreitary software AirDrop which has been released by Apple in 2011.

Since then support for direct peer-to-peer connections has matured on various mobile operating systems, including iOS and its Multipeer Connectivity Framework which allows nodes to advertise itself, discover nearby advertisers and attempt to connect to detected nearby advertiser. The concept of that model motivated Newport to develop a formal definition and comparison of gossip algorithms. He describes and analyses differences in algorithm parameters and how they influence data spreading in a MANET where the goal is that messages spread to the entire network (2017). The author claims that these algorithms can help to establish peer-to-peer meshes that support infrastructure-free networking and communication. He presents the discontinued FireChat application as an example which offered group chats using smartphone peer-to-peer services such as Bluetooth, WiFi and the Multipeer Connectivity Framework. According to the author this application has been adopted in multiple governmental protest or festivals that were located out of reach of cell towers, but unfortunately did not release a new version since 2018.

2.2 D2D in cellular networks

Although a lot of research exists on D2D communication in cellular networks, most of it is done in a military use case (Gamboa, Ben Mosbah, Garey, *et al.*, 2023), like unmanned aerial vehicles (UVA) or public safety networks (Gamboa, Henderson, Garey, *et al.*, 2024), like vehicle to everything (V2X). Most of this research builds upon 5G New Radio (NR) Sidelink (SL) which has implemented protocol support for (V2X) and Proximity Services (ProSe) for public safety networks which allows user equipment (UE) to directly talk to each other without the interference of a base station (gNB). Although approaches existed to also introduce D2D communication to the commercial markets back in 2014 by Qualcomm and Condoluci, Militano, Orsino, *et al.* back in 2015 proved that LTE-Direct, a predecessor of 5G SL, has some energy and scaling benefits over WiFi-Direct, according to Apple engineers no support for this technology is given on mobile smartphones for third party developers. This is also pointed out by the authors of this critical review of mobile device-to-device communications (Desauw, Luxey-Bitri, Raes, *et al.*, 2023).

2.3 Apple Ecosystem and TU Darmstadt

From 2018 on the OWL project by Secure Mobile Networking Lab (SEEMOO) at TU darmstadt contributed several papers to research on Apple's wireless ecosystem (Stute, Kreitschmann and Hollick, 2018a). Their goal was to assess security and privacy concerns as well as enable cross-platform compatibility with other vendors. They started to investigate AWDL which is heavily used in Apple's Continuity platform. While reverse engineering the 802.11 Wifi based protocol the authors stumbled across various security concerns which they mainly focused on next to Apple's Bluetooth LE usage in following papers until 2021.

On the projects first conference the authors presented the operations of the undocumented AWDL protocol. They used binary and runtime analyses to reconstruct the daemons and frameworks involved in communicating via AWDL and found that each AWDL node announces a sequence of Availability Windows indicating that it is ready to communicate with other AWDL nodes. In the process they also detected that AWDL connections do not feature any security mechanisms leaving authentication or

encryption to the transport and application layers, which the authors claim to be an informed decision by Apple (Stute, Kreitschmann and Hollick, 2018b).

Following the initial findings on missing security considerations, the authors dedicated a separate paper on researching security and privacy issues in the AWDL protocol. The study uncovers multiple vulnerabilities related to both design flaws and implementation bugs. One of the major findings is the possibility of a man-in-the-middle (MitM) attack, which would allow an attacker to stealthily modify files transferred via AirDrop. Additionally, the study identifies denial-of-service (DoS) vulnerabilities that can disrupt communication or force the sudden crash of all nearby devices. The research also reveals privacy weaknesses that allow attackers to track users over extended periods, effectively bypassing MAC address randomization. The authors included a demonstration showing the feasibility of these attacks where the researchers developed proof-of-concept implementations using inexpensive hardware like a 20 dollar micro:bit device. Although following responsible disclosure Apple addressed one of the DoS attack vulnerabilities, the researchers also highlight that several of the identified security and privacy risks require fundamental redesigns of some of Apple's services to be fully mitigated. Overall the study highlights critical security flaws in AWDL design and implementation demonstrating a potential impact on over a billion Apple devices and emphasizing the need for stronger security measures and protocol improvements. Onebillionopeninterfaces Disrupting continuity.

In 2020 Ian Beer a british computer securtiy expert and white hat hacker, inspired by the initial work of TU Darmstadt found another severe security issue in AWDL which could remotely trigger an unauthenticated kernel memory corruption that lead to all iOS devices in radio proximity to reboot. Further he describes how this issue could lead to a system state that lets an attacker run any code on nearby iOS devices and steal all user information. In his demos he forced the former flagship iPhone 11 Pro to activate the AWDL interface which is successfully exploited to steal sensitive information like emails, photos, message or even the keychain. atIanBeer_2020

In 2021 the authors of the OWL project dedicated another paper to the analysis of Apple's offline file sharing service AirDrop. The authors discovered two design flows in the underlying protocol which would allow an attacker to sniff vulnerable hashes of contact information such as the phone number or email address. These hashes are particularly vulnerable to brute force attacks because of the small input number space. For example, phone numbers in Austria with exempt of the mobile operator prefix consist of only seven digits where a hash would be easily cracked within seconds on modern PCs according to the authors. After presenting security issues and their effects the authors propose an optimized private set intersection (PSI) based protocol called PrivateDrop that solves the problem of privacy preserving authentication between nearby offline devices. While preventing potential attackers to steal private user data, users still remain trackable via their UUID in the TLS certificate used during the initial handshake. Finally, the authors also claim that their proposed approach is not limited to the Apple ecosystem and could help Googles similar platform "Nearby" for Android. They also open sourced this implementation as part of their OWL project.

While focusing merely on AirDrop in the previous paper the authors of the OWL project dedicated another paper to a broader range of Apple's Continuity services. The authors described a guide to approach a structured analysis of the protocols involved in these services and developed a toolkit to automate parts of this mostly manual process. Based on the created tools the authors analyzed the full protocol stacks involved in various Continuity services like Handoff (HO), Universal Clipboard (UC) and Wi-Fi Password Sharing (PWS). During this process they again found several security issues which could lead to possible denial-of-service (DoS) or machine-in-the-middle (MitM) attacks. To demonstrate their findings the authors implemented a PoC using an affordable off-the-shelf Wi-Fi card and urge readers to share similar findings with Apple to help make widely used devices more secure.

In yet another paper about Apples local broadcasting platform the authors of the OWL project examine worlds biggest offline finding (OF) system. In short lost devices advertise rolling public keys via Bluetooth Low-Energy (BLE) that are captured by nearby “finder devices” and sent to an Apple server with the corresponding location of the finder device. The owner of the lost device can the query the Apple server for entries sent by these finder devices to get an estimated location of his lost device. While the authors claim that this design mostly achieves Apple’s specific security goals they also share two distinct design and implementation faults. These would allow Apple to correlate different users location if their locations are reported by the same finder device to let Apple form social user graphs. Moreover a malicious macOS application could retrieve and decrypt location reports of the last week as teh rolling advertisement keys are cached and stored on the filesystem as clear text. Quelle:Who Can Find My Devices?

2.4 Summary

Although considerations about ad-hoc networks were dealt with very early in the history of mobile computing, it has only been recently that widespread features using local radio broadcasting were adopted. With great power comes great responsibility which Apple tends to have underestimated with respect to the great work and great findings of the OWL project by SEMOO at TU Darmstadt. The group noticed the need for research on mobile local ad hoc networks since these have vastly improved and found several new applications over the last few years especially in the Apple ecosystem, but focused mainly on demistifying the underlying protocols and analysing those with regard to security concerns.

3 | Background

This section tries to describe and familiarize with concepts of networking topologies for mobile devices. From an abstract perspective networking can be categorized into infrastructure and ad-hoc networks, one relying on mediators while the other works without intermediary infrastructure letting the participants itself form the network.

3.1 Infrastructure Networks

Infrastructure networks are defined as wireless networks that require the use of an infrastructure device, like an access point or cellular base station to enable communication between clients no matter how close they might be (NIST).

3.2 Ad-hoc Networks

This should better be part of introduction Describe that all of mobile connections use mediators and why this happend like it and why it is beneficial (bigger antennas that can handle weaker signals and send stronger signals) describe what has changes in the last years on apples platform and why non-mediator communication got so important for apples ecosystem, apple watch pairing, clone app to mac (when opened on iphone, eg. calendar), cmd+c and cmd+v via across iphone and mac what is lorawan and how could it be used one day in smartphones and mobile computing describe why peertopeer wlan or bluetooth is not yet suitable for long distance communications, also mention the longest wlan connection, 273km with bigger antennas

describe what was used on ios device what frameworks exist. how bluetooth was used and how wifi is now used for ptp connections

3.3 Satellite phones

ESA organisation claims that more and more space junk destroys modern satellites and this can lead to outages too.

3.4 Starlink Direct to Cell

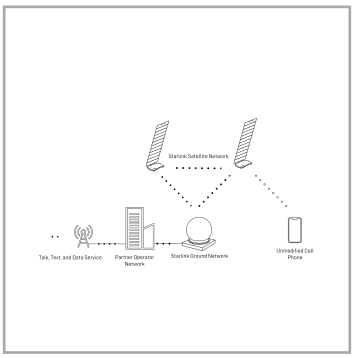

Figure 1: Abstract structure of Starlink's network.

3.5 OSI Transport layer

3.5.1 TCP

3.5.1.1 Nagle's algorithm

3.5.2 UDP

3.5.3 QUIC

3.6 Ad-hoc technologies

While and solve the issue of dead spots, they also rely on infrastructure. The following is an incomprehensive list of ad hoc technologies.

3.6.1 WiFi

3.6.2 Bluetooth

3.6.3 LoraWan

3.6.4 LTE-Direct and 5G Sidelink

3.6.4.1 gNB

3.6.5 NFC

3.7 Ad-hoc on iPhones

3.7.1 Bonjour

Bonjour is a mDNS protocol, Bonjour services are registered and handled by the mDNSdaemon from the underlying operating system so no message parsing and response handling must be conformed by the developer using this method.

3.7.2 Multipeer Connectivity

3.7.3 Networking Framework

3.7.4 AWDL

biggest kernel module in iOS

3.7.5 Websockets

3.8 Wireless communication on iPhones

this heading will focus more on wireless communication in general not per se ad hoc so software here will include

3.8.1 Hardware

Differences of wifi antennas and cellular antennas, where they are located and how they operate power of cellular antennas vs wifi antennas of iphones, frequency spectrum of these technologies, what is the real difference which makes the one so much wider than the other, wifi and bluetooth go over the same chip/antenna on iphone

3.8.2 Software

packages like Networking framework, BSD Sockets or URL Session…

In the background section you might give explanations which are necessary to read the remainder of the thesis. For example define and/or explain the terms used. Optionally, you might provide a glossary (index of terms used with/without explanations).

Hints for equations in Typst:

Mathematical formulas are (embedded in `$`) in Typst. For example:

The notation used for **calculating** of *code performance* might typically look like shown in Equation 1, i.e. the first one for **slow** in Eq. I and the other one for **very slow** in Eq. II.

$$O(n) = n^2 \tag{I}$$

$$O(n) = 2^n \tag{II}$$

Equation 1: Equations calculate the performance.

In the text we refer multiple times to ϕ. We define it to be calcultated as shown here:

$$\begin{aligned} d &= 24 - 10 - 7 - \sqrt{3} \\ d &= 14 - 7 - \sqrt{3} \\ d &= 7 - \sqrt{3} \end{aligned} \tag{III}$$

$$\phi := \frac{d}{3} \tag{IV}$$

Equation 2: A custom definition of ϕ allows to shorten upcoming equations.

The Equation 2 explains (for the single steps see Eq. III and Eq. IV) how the overall ϕ is calculated to be used in the upcoming formulas of this thesis.

4 | Concept

The following introduces the abstract design of how direct P2P communication between iOS devices is tested and measured for evaluation in this thesis. The measurement setup should try to systematically quantify how the performance of P2P iOS connection is and how it changes under different surroundings. Different approaches to solve this problem do exist, which are briefly evaluated and compared among each other.

4.1 Approaches

4.1.1 Continuity Black Box Testing

Using Apple's Continuity features to send and receive data on different iOS devices could be tested and analysed. In the simplest form this would involve selecting a particular file with a particular size and measuring the time it takes to transport this file from one device to another. The data transfer speed could be approximated using the file size and the time it took to transfer the file. Another approach to this black box testing could involve using a network sniffer to monitor connection establishment like local mDNS and security handshakes including recording and analysing the transmission process like congestion control and packet loss recovery. This could further be applied on different abstraction layers like measuring the physical radio frequency energy used or how many IP packages needed to be sent.

4.1.2 iOS Application

iOS provides several application programming interfaces (API) that allow a third party developer to access various underlying technologies to establish P2P connections. The software could directly record how much data is sent and received mitigating overhead of measurement logic. Using the frameworks provided by Apple is also an interface available to any third party developer and can therefore be implemented in any iOS application without the need to bypass any restrictions.

4.1.3 Jailbreaking

Jailbreaking is a term used to describe the bypassing of the security mechanism in iOS. This allows the user to install arbitrary third party software and gain full access to the operating system. This would allow to also access interfaces like the cellular antenna that is restricted and not accessible to a third party developer or turning off services that would interfere with testing. This however violates Apple's iOS Software License Agreement and testing could potentially disturb restricted frequency bands that are licensed. Additionally considering the current use cases of iOS devices and restrictions of the operating system it seems unlikely that developers other than Apple could have similar interfaces in near future.

4.2 Experiment Design

After evaluating the aforementioned concepts the decision was taken to build an iOS Application establishing and intercepting the P2P connection to measure connection metrics. It is the most practicable considering the use for a wide mass, because staying in the boundaries imposed by Apple and using only first party frameworks makes developing and distributing in the App Store easier. The application needs to be installed on two nearby devices to establish a connection and transfer data. Furthermore since iPhones are typically used under various circumstances and surroundings testing should also cover representable scenarios for common places visited by iPhone users.

4.2.1 Prototype

The prototype should be installable on arbitrary iOS devices and should serve as both client and server. The client must be capable of discovering nearby peers and sending a connection request after user instruction. The server must be capable of advertising a service that clients can find and handling incoming connection requests. Both must be able to display the metrics that the applications measured to the user and should support a method to abort ongoing connections to start new advertisers/ discoverers.

4.2.2 Testing

Capturing the data of interest is done by the prototype itself. However general conditions for environmental and data variations have to be defined. Testing must be done in different surroundings distinguishing each other in the density of obstacles, radio frequency emissions or both to cover most real life scenarios. Moreover data size must vary to represent use cases for small payloads like simple message transfer to bigger payloads like sharing files. Another factor to consider is the distance between the two testing devices. All these three external factors must be tested in each possible variation forming a three dimensional matrix.

5 | Implementation

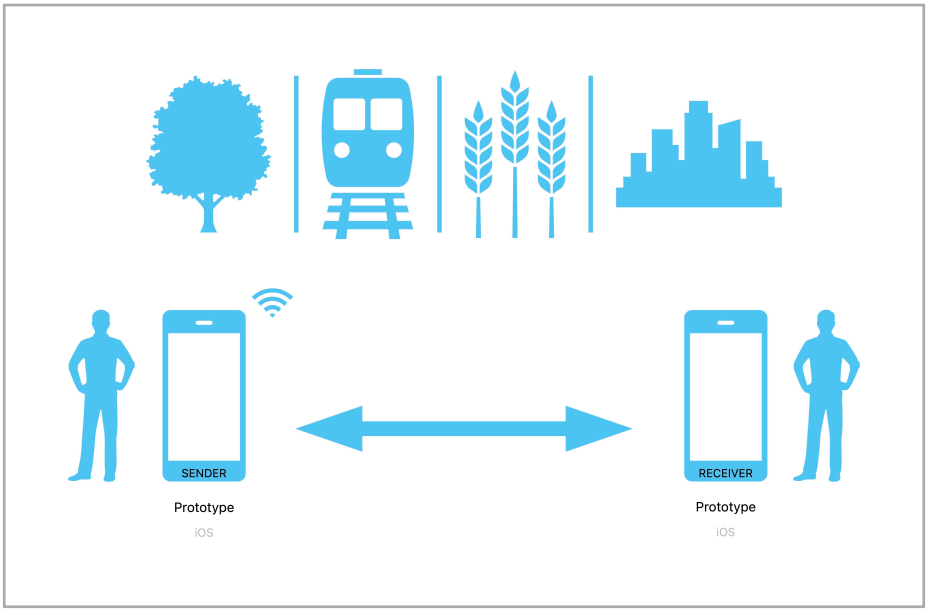

Figure 2: Abstract representation of scientific concept

5.1 Prototype

The application is written in `Swift` using the integrated development environment (IDE) `XCode`, which is the suggested way to build iOS application by Apple. The built artifact is distributed via TestFlight, an online service for installing and testing apps for Apple devices and can be downloaded via an URL or directly installed by the developer machine. The application is written using a modified version of the MVVM GUI design pattern. The application must feature a mechanism to find local peers, connect them and intercept data transfer to measure metrics. The technologies used to achieve these features are described in the following sections.

5.1.1 User Interface

The User Interface (UI) is developed using SwiftUI, a declarative way to build applications across all Apple platforms. Considering the aforementioned features the application is split into five different screens each one serving a specific purpose in the process of establishing the connection and transferring data. The screens are listed below in the order the user would walk through during a testing procedure and are called views to match terminology of the MVVM pattern.

5.1.1.1 Decision View

This view is the first a tester sees when opening the application and is responsible for configuring the next steps of testing. Depending on the decision the tester makes the application is initialized as a client that browses for nearby services or as a server that advertises a service to nearby clients.

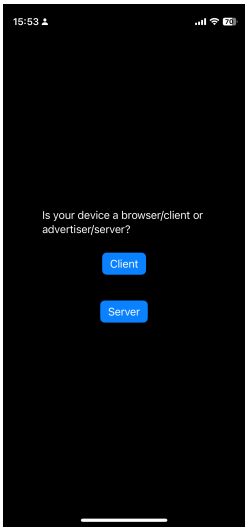

Figure 3: Screenshot of decision screen.

5.1.1.2 Server View

The server views purpose is to present the user the state of each connection and the test results which were measured on the server. For this, the user has to tap the `Get Test Results` button in the right corner of the top tool bar. On the server view, the user can also tap the `Reload` button which aborts all ongoing connections, destructs the server objects and creates new ones which immediately start advertising again.

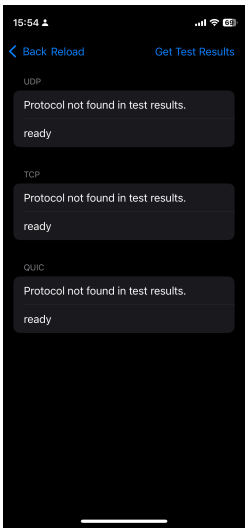

Figure 4: Screenshot of server screen.

5.1.1.3 Client Views

The client view is split into to seperate steps since the client needs to find nearby peers and connect to them. Only after a successful connection was established the testing can begin.

5.1.1.3.1 Browser View

The browser view lists all nearby servers found. Through a tap on an item the client tries to connect to the advertiser and navigates to the testing view.

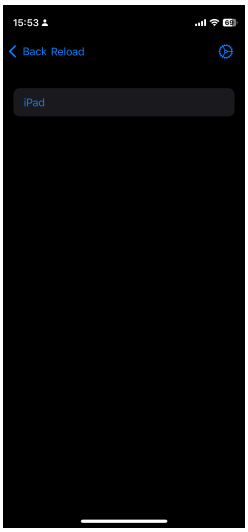

Figure 5: Screenshot of browser screen.

5.1.1.3.2 Testing View

The testing views purpose is to present the user the state of each connection and the test results which were measured on the client. Furthermore the buttons `Start Test` initiate the sending of the test data for the associated connection. The testing view, as well as the server view features a `Reload` button that aborts all ongoing connections, destructs and reintializes all client objects and navigates the user back to the browser view. There the user can select another server to connect to.

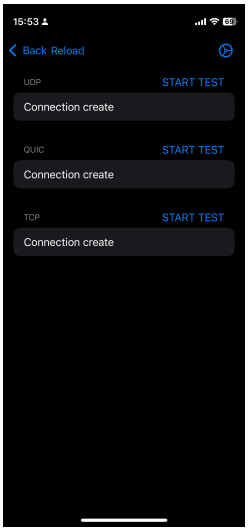

Figure 6: Screenshot of testing screen.

5.1.2 Networking Frameworks

Apple provides different frameworks for P2P connections using different layers of abstraction or different underlying technolgies. One of these frameworks is called Multipeer Connectivity. Newport describes it as an implementation of the mobile telephone model 2017 in his article about gossip in smartphone P2P networks. Apple states, the framework “supports the discovery of services provided by nearby devices and supports communicating with those services through message-based data, streaming data, and resources (such as files). In iOS, the framework uses infrastructure Wi-Fi networks, peer-to-peer Wi-Fi, and Bluetooth personal area networks for the underlying transport. In macOS and tvOS, it uses infrastructure Wi-Fi, peer-to-peer Wi-Fi, and Ethernet.” Contrary to this excerpt of the documentation, tests and information gathered from Apple’s developer forum conclude that Mulipeer

Connectivty does not support Bluetooth for P2P networking anymore and got disabled with the release of iOS 11.

In an approach to give a brief overview about Apples networking APIs, Apple describes Multipeer Connectivity as a high-level interface to Apples P2P WiFi support and also introduces the Network Framework, which is considered a low-level interface by Apple engineers. Apples Documentation states developers should use this framework when they need direct access to protocols like TLS, TCP, and UDP for their custom application protocols. The Network framework features opt-in support for P2P connection establishment via AWDL and also does not support connecting via Bluetooth, which is accessible through the Core Bluetooth Framework.

Nearby Interaction is yet another framework to establish P2P connections. It uses the iPhones ultra wideband (UWB) chip to "locate and interact with nearby devices using identifiers, distance, and direction." These chips are usually used in smaller distances to precisely locate compatible hardware, so in examples from Apples world wide developer conference (WWDC) distances of one and a half to three meters are shown which does not meet the requirements for this experiments.

Following Apples recommendations documented in a technote about choosing the right networking API, the Networking framework is used for establishing a connection and transferring data.

5.1.3 Configuration

Different transport protocols can be used to establish a connection. Following the single responsibility principle the transport protocols and their configurations are injected into the server and client implementations to create the corresponding connections. The transport protocols for which a server and client will be created are listed in a central singleton responsible for creating the server and client objects for each protocol which are injected to the view models.

```
struct Config {
    static let serviceProtocols: [TransportProtocol] = [.udp, .tcp, .quic]

    static var clients: [any Client] {
        serviceProtocols.map { ClientImpl<ConnectionImpl>(transportProtocol: $0) }
    }

    static var servers: [any Server] {
      serviceProtocols.compactMap { try? ServerImpl<ConnectionImpl>(transportProtocol:
$0) }
    }
}

...
    init(state: State = .init(), servers: [any Server] = Config.servers) {
        self.state = state
        self.servers = servers
    }
...
```

The `TransportProtocol` itself is an enum, where each case is representing a transport protocol used while testing. The enumeration consists of TCP, UDP and QUIC cases and their configurations are accessed through the `parameters` and `type` computed properties. The `type` property delivers the local mDNS name used to advertise and browse for the service via Bonjour. The `parameter` property delivers the `NWParameters` configurations used both in `NWListener` and `NWBrowser` to configure the network stack for these objects. It is important to set the `includePeerToPeer` property to enable local AWDL broadcasting.

```
    var parameters: NWParameters {
        switch self {
        case .udp:
            let udpOptions = NWProtocolUDP.Options()
            let parameters = NWParameters(dtls: nil, udp: udpOptions)
            parameters.includePeerToPeer = true
            return parameters

        case .tcp:
            let tcpOptions = NWProtocolTCP.Options()
            tcpOptions.enableKeepalive = true
            tcpOptions.keepaliveIdle = 2

            let parameters = NWParameters(tls: nil, tcp: tcpOptions)
            parameters.includePeerToPeer = true
            return parameters

        case .quic:
                    ...
        }
    }

    var type: String {
        switch self {
        case .udp:
            "_txtchat._udp"
        case .tcp:
            "_txtchat._tcp"
        case .quic:
            "_txtchatquic._udp"
        }
    }
```

5.1.3.1 Secure Connection Establishment for QUIC

Since QUIC has built in support for secure connections and requires TLS v1.3 a secure identity composed of a certificate and a private key has been created and added to the applications bundle.

`openssl command`

After adding it to the bundle the application must read the secure identity and add it to QUIC's `NWParameters` `securityProtocolOptions` for client and server.

```
        sec_protocol_options_set_local_identity(quicOptions.securityProtocolOptions,
sec_identity_create(identity)!)
       sec_protocol_options_set_verify_block(quicOptions.securityProtocolOptions, { _,
sec_trust, completion in
          ... //check validity of sec_trust
          completion(isTrusting)
        }, .global())
```

This enables the application to establish a secure QUIC connection.

5.1.4 Connection Establishment

Connection Establishment is done via Bonjour using the Network Framework. The servers register a listener using the `NWListener` class and the `NWParameters` and local mDNS record from the injected transport protocols to listen for incoming network connections to that service. Once an inbound

connection is received the listener object calls the `newConnectionHandler` method previously set when configuring the listener object. When the method is invoked it cancels the previous connection, creates a new one and posts a message to the connection state subject, indicating that a connection has been established. When a listener is created a service object which represents the Bonjour service to advertise the endpoint on the local network is initialized and passed to the listener.

```
    init(transportProtocol: TransportProtocol) throws {
        self.transportProtocol = transportProtocol

        listener = try NWListener(
            service: .init(
                name: UIDevice.current.name,
                type: transportProtocol.type,
                domain: nil
            ),
            using: transportProtocol.parameters
        )
    }

    func startAdvertising() {
        listener.newConnectionHandler = { [weak self] connection in
            self?.connection?.cancel()
            self?.connection = nil
            self?.connection = C(connection)
            self?.connectionStatus.value = "Connection established"
        }

        listener.stateUpdateHandler = { [weak self] state in
            self?.connectionStatus.value = "\(state)"
        }

        listener.start(queue: .global())
    }
```

The clients instantiate a `NWBrowser` object used to browse for available network services. When the browser object finds new Bonjour services it calls the `browseResultsChangedHandler` method. This method is previously configured to write the results to `browserResults` subject which can be observed by the view model. Once the user selects a browse result in the `BrowserView` this instanz is passed to the `createConnection` method which cancels the old connections, sets the new one for further use and reports an error in case the connection failed.

```
    func createConnection(with browserResult: NWBrowser.Result) -> Error? {
        let nwConnection = NWConnection(to: browserResult.endpoint, using:
transportProtocol.parameters)

        self.connection?.cancel()
        self.connection = nil
        self.connection = C(nwConnection)

        if case let .failed(error) = self.connection?.state {
            return error
        }
        return nil
    }
```

5.1.4.1 Injecting a concrete Implementation of a Protocol

injecting ConnectionImpl to Client and serverimpl so i can create a new object inside

5.1.4.2 Adding local domains to Info.plist

Bonjour services which are browsed for must be listed in the Info.plist using the `NSBonjourServices` key. The format is similar to the ones used to configure the `NWBrowser` and `NWListener` objects, composed of the application and transport protocol like "_myservice._tcp". The Info.plist file is an information property list file that contains information and configuration about the application bundle.

In the case of the test application the aforementioned key contains the following entries.

- "_txtchat._udp"
- "_txtchat._tcp"
- "_txtchatquic._udp"

One can notice that two entries using UDP as the transport protocol exist. This is because the application should advertise and browse for UDP and the UDP based QUIC protocol simultaniously. Without using a second service entry Bonjour would automatically rename the service entry which would break the logic for displaying and selecting the browser results.

5.1.4.3 Displaying and Selecting Advertisers

Local advertisers are displayed based on their human readable service instance name.

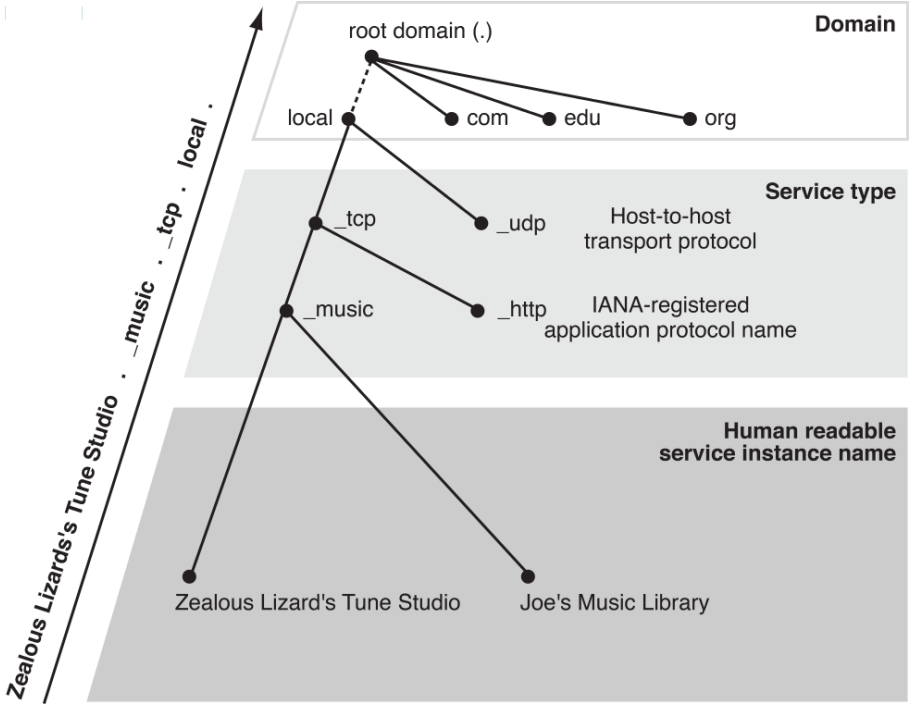

Figure 7: Graphic showing the Bonjour naming convention.

In case of this test application the Bonjour service name is configured using the `UIDevice.current.name` which represents a generic device name like "iPad" or "iPhone" which can be seen in Listing NWListener. This name is extracted from the bonjour `NWEndpoint` on the client side and listed in the Browser View Figure 5.

```
extension NWBrowser.Result {
    var name: String? {
        if case let NWEndpoint.service(name: name, type: _, domain: _, interface: _) =
self.endpoint {
            return name
```

```
        }
        return nil
    }
}
                Task { @MainActor in
                    for await browseResults in client.browserResults.values {
                      state.advertiserNames.append(contentsOf: browseResults.compactMap
{ $0.name })
                        state.advertiserNames = state.advertiserNames.removingDuplicates()
                    }
                }
```

Using the same service type like mentioned in the previous Section, Bonjour would automatically rename the service if it detects the same service on the local network.

```
iPad\\032(2)._txtchat._udp.local.
iPad._txtchat._udp.local.
```

The human readable service instance name is not identical and users could not choose a single testing server for all advertised transport services anymore.

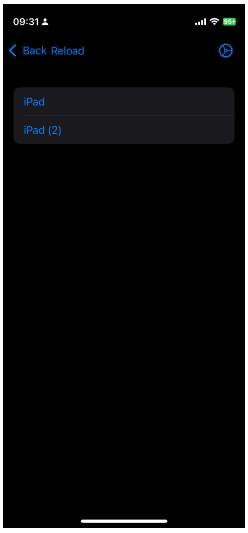

Figure 8: Screenshot of UDP and QUIC services using the same Bonjour service type.

5.1.5 Measuring and Networking

Whenever a connection is ready a `DataTransferReport` is started which provides metrics about data being sent and received on a connection like data size in bytes, the number of IP packages or round trip time (RTT).

Besides that the application also measures the testing start and end time and implements an own approach to measure RTT since DataTransferReport only takes TCP's control packages into account. Before the configured number of packages with the configured number of bytes get sent 100 separate packages to measure RTT and jitter are emitted. These packages contain the time the package was emitted and is recognized and sent back from the testing server. When received again on the testing client, the time in the package and the new local current time are compared and the difference is stored to later calculate the average RTT and the Jitter. To get precise timing measurements the kernel function `mach_absolute_time` is used which returns current value of a clock that increments monotonically in tick units. This value needs to be converted to nanoseconds using a time base containing information about the duration of a tick.

```
                    var now = mach_absolute_time()
                    var elapsed = now - date
                    var timebase: mach_timebase_info_data_t = .init()
                    mach_timebase_info(&timebase)
                    let latencyNanoSeconds = elapsed * UInt64(timebase.numer) /
UInt64(timebase.denom)
```

To transfer testing data the `NWConnection` class and its synchronous `send` and `receive` methods are used. These methods are wrapped in an asynchronous `withCheckedContinuation` method to support Swift's `async await` concurrency. This testing application contains a wrapper class for the `NWConnection` which features an asynchronous `startTesting` that utilizes the aforementioned `send` method which is called from the client defining the number of packages to send and its size.

```
    func startTesting(numberOfPackages: Int, packageSizeInByte: Int) async
```

5.2 Testing

Testing is done using an iPhone 12 mini and an iPhone 15 Pro both using the current iOS version 18.4.1. It is tested in various scenarios, which are defined below.

5.2.1 Places

Testing is done in four different places representing typical places for iPhone users. One testing environment will be the underground station which is dense in people on a small space. Another environment will be the inner city of vienna which is also dense in people but more open than the underground. The next environment will be a free field with perfect conditions for radio broadcasting since minimal other signals or objects like persons could disturb the signal. The last place that will be tested is the forest since it may have the same density of obstacles but also like the filed minimal radio frequency disturbances.

5.2.2 Data sizes and distance

Data size of the whole testing process is composed of the number of packages that are sent and the size of each package. Both can vary as well as the distance between the connected devices. Testing will be done with the following values each combined with all values of the other categories.

Number of Packages	Size of Package in Bytes	Distance in Meters
100	128	1
1000	4096	10
10 000	16 384	30
100 000		max

Table 1: Definition of test scenarios.

5.2.3 Protocols and Metrics

Data transfer metrics of three different transport protocols will be tested. TCP, UDP and QUIC will be compared in the average round trip time (RTT), Jitter, data transfer speed. These metrics will be measured in all combinations of the above mentioned scenarios.

Protocols
TCP
UDP
QUIC

Table 2: Transport Protocols used for testing.

Metrics
RTT
Jitter
Data Rate

Table 3: Metrics used to evaluate protocols.

5.3 Summary

Testing is done using a prototype application written in SwiftUI enabling the user to browse and advertise nearby services via Bonjour and connect to them. Once two devices are connected testing can be started for each transport protocol (TCP, UDP and QUIC) separately and the metrics are displayed to the user ready for recording. Testing will be done in four different scenarios each representing a typical place for iPhone users including the underground, the inner city of vienna, a free field and the forest. Moreover different numbers of packages and package sizes will be sent in varying distances between the devices.

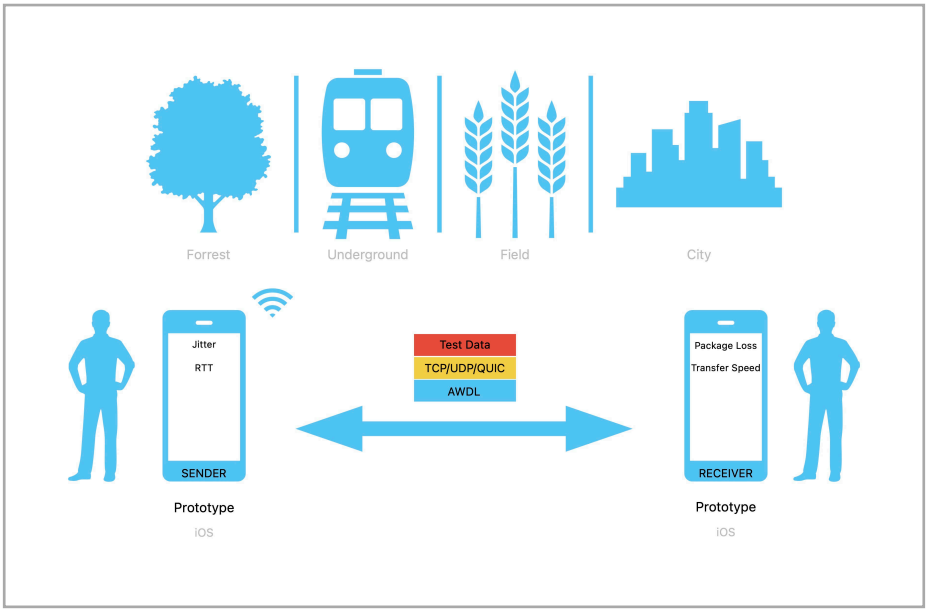

Figure 9: Abstract representation of scientific concept

Metrics	Scenarios	Protocols	Distances in Meters	Number of Packages	Package size in Bytes
RTT	Underground	TCP	1	100	128
Jitter	Inner City	UDP	10	1000	4 096
Data Rate	Free Field	QUIC	30	10 000	16 384
	Forest		max	100 000	

Table 4: Definition of testing scenarios and variations.

6 | Results and Evaluation

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

Describe (proof) how your implementation really solved the stated problem. I.e. accept or reject your hypotheses. Provide a range of input data sets. Run experiments and gather the output (of tools) to meter your prototype. For the analysis, collect the measurement-data, process (e.g. filter) data and interpret the data. Include an interpretation of the work. What do the results mean to you? State current limitations of your solution. Give (personal) interpretation where suitable. Your own opinion is relevant, but must be marked clearly as such.

6.1 Setup Experiment

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si.

For example: During the setup the Garbage Collection (GC) was configured for the parallel version using the value `+UseParallelGC` for the command line argument `-XX` (`java -XX:+UseParallelGC`).

Hints on dynamically reading in external data for tables in Typst:

Using the custom macro `fhjtable` it is possible to include data dynamically for table generation. The data has to be specified in Comma-separated Values (CSV) as shown below:

Name	Profession	Experience (in years)
Max	Student	3
Mia	UX-Designer	7
Helga	Programmer	9

Table 5: Professional experience of the test users with databases.

Find in Table 5 the years a user has worked with different relational or nosql databases in a professional context.

6.2 Measurement

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si.

Hints on using tables in Typst:

Somewhere in the normal text of the thesis the interpretation of data and information shown in a table must be discussed. Explain to the readers which numbers are important. Possibly, highlight unexpected or special data points.

	Min	**Max**	∅	σ
Network roundtrip time	34.6s	42.5s	38.1s	2.3s
Time for single request	2.4s	13.5s	**7.1s**	4.3s

Table 6: The numbers in the table above show the minimum, maximum, average ∅ , and standard deviation σ of the 273 measured network times in seconds.

For example: … Table 6 shows some calculated results on the roundtrip and request times measured in the experiment. The average, the minimum, the maximum and the standard deviations hint to a dramatic increase (> 13%) in performance in comparison to the old solution of 2003.

6.3 Interpretation of the Data

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si.

For example: The customisation of the GC seem to have following positive and negative consequences....

Hints on dynamic calculation in Typst:

We might calculate, e.g. `#calc.max(...)`, within our document, such as max of three and seven times two is: 14.

Hints on using logic in Typst:

For example, we might use **for loop** to arrange a few images in a grid box, as shown below.

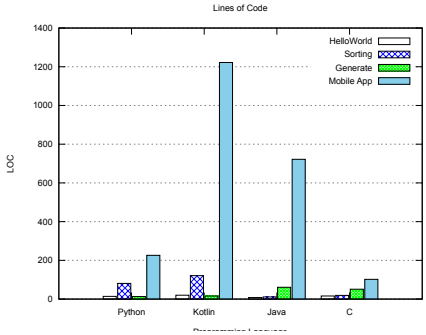

Figure 10: Compared source code by metric 1.

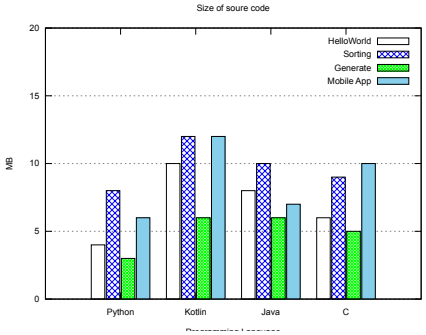

Figure 11: Compared source code by metric 2.

Hints on Charts:

Note: the charts (**vector**! images) shown have been created from raw data using the tool **gnuplot** on the command line. With `gnuplot` you can create charts by use of a textual command language. This is great for automation and it is also great for managing the source code in `git`.

7 | Conclusion and Outlook

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit.

Sum up the results achieved and give an outlook by suggesting further research by explaining how others could built on your results.

Glossary

CSV – **Comma-separated Values** 26

GC – **Garbage Collection** 26, 28

P2P – **Peer to Peer** i, ii

Bibliography

Balan, R. K., Ramasubbu, N., Prakobphol, K., Christin, N., et al. (2009) mFerio: the design and evaluation of a peer-to-peer mobile payment system. In: *Proceedings of the 7th international conference on Mobile systems, applications, and services.* [Online]. June 2009 Kraków Poland, ACM. pp. 291–304. DOI:10.1145/1555816.1555846.

Condoluci, M., Militano, L., Orsino, A., Alonso-Zarate, J., et al. (2015) LTE-direct vs. WiFi-direct for machine-type communications over LTE-A systems. In: *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).* [Online]. August 2015 Hong Kong, China, IEEE. pp. 2298–2302. DOI:10.1109/PIMRC.2015.7343681.

Department, S. R. (2024) *Number of smartphone users worldwide from 2014 to 2029 (in millions).* [Online]. December 2024. https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world.

Desauw, L., Luxey-Bitri, A., Raes, R., Rouvoy, R., et al. (2023) *A critical review of mobile device-to-device communication.* [Online]. https://inria.hal.science/hal-04198528.

Dubois, D. J., Bando, Y., Watanabe, K. & Holtzman, H. (2013) ShAir: extensible middleware for mobile peer-to-peer resource sharing. In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering.* [Online]. August 2013 Saint Petersburg Russia, ACM. pp. 687–690. DOI:10.1145/2491411.2494573.

Gamboa, S., Henderson, T. R., Garey, W., Liu, C., et al. (2024) Towards System Level Simulations of Public Safety Applications over 5G NR Sidelink. In: *2024 IEEE World Forum on Public Safety Technology (WFPST).* [Online]. May 2024 Herndon, VA, USA, IEEE. pp. 1–6. DOI:10.1109/WFPST58552.2024.00043.

Gamboa, S., Ben Mosbah, A., Garey, W., Liu, C., et al. (2023) System-Level Evaluation of 5G NR UE-Based Relays. In: *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM).* [Online]. October 2023 Boston, MA, USA, IEEE. pp. 807–814. DOI:10.1109/MILCOM58377.2023.10356291.

Kortuem, G., Schneider, J., Preuitt, D., Thompson, T., et al. (2002) When peer-to-peer comes face-to-face: collaborative peer-to-peer computing in mobile ad-hoc networks. In: *Proceedings First International Conference on Peer-to-Peer Computing.* [Online]. 2002 Linkoping, Sweden, IEEE Comput. Soc. pp. 75–91. DOI:10.1109/P2P.2001.990429.

Newport, C. (2017) Gossip in a Smartphone Peer-to-Peer Network. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing.* [Online]. July 2017 Washington DC USA, ACM. pp. 43–52. DOI:10.1145/3087801.3087813.

Stute, M., Kreitschmann, D. & Hollick, M. (2018b) One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol. In: *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. [Online]. October 2018 New Delhi India, ACM. pp. 529–543. DOI:10.1145/3241539.3241566.

Stute, M., Kreitschmann, D. & Hollick, M. (2018a) *The Open Wireless Link Project*. [Online]. 2018. https://owlink.org/.