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

May 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 2025-05-17.

Abstract

Modern mobile devices can make use of a wide variety of communication technologies. Besides having different applicabilities and protocols, standards like Bluetooth, Wi-Fi 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 (Statista Research Department, 2024), 27% covered by iOS (Sherif, 2025a). 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 Wi-Fi since these do not required pre-existing infrastructure and purely rely on local radio broadcast. Due to the latest advancements in these technologies, it is unclear how good direct Peer to Peer (P2P) networks work in 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 package loss. The results show that AWDL achieves the best metrics when tested in the Underground. Prototype measurements also show that most of the time User Datagram Protocol (UDP) achieves the fastest data rates with a compromise on data loss, whereas best overall metrics were measured using QUIC. 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, AWDL, iPhone

Kurzfassung

Moderne Mobilgeräte können eine Vielzahl an Kommunikationstechnologien nutzen. Neben unterschiedlichen Anwendungsmöglichkeiten und Protokollen müssen Standards wie Bluetooth, Wi-Fi oder 5G auch drahtlos sein, um Daten im modernen Kontext übertragen zu können. In den letzten Jahren hat sich die weltweite wirtschaftliche Nachfrage auf Forschung und Entwicklung ausgewirkt, sodass moderne Smartphones enorm verbessert wurden. Dies hat dazu geführt, dass im Jahr 2024 weltweit über 4 Milliarden Smartphones genutzt werden (Statista Research Department, 2024), 27 % davon benutzen das Bertriebssytem iOS (Sherif, 2025a). 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 gering der Abstand zwischen diesen sein sollte. In diesen Szenarien kann die Kommunikation zwischen zwei mobilen iOS-Geräten über Bluetooth oder ad-hoc Wi-Fi hergestellt werden, da diese keine bereits vorhandene Infrastruktur benötigen und sich ausschließlich auf lokale Broadcasts stützen. 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 AWDL verbindet und Metriken misst, welche die Qualität der Verbindung beschreiben, wie RTT, Jitter, Datengeschwindigkeit und Package Loss. Die Ergebnisse zeigen, dass AWDL die besten Werte in der U-Bahn Station gemessen wurden. Zudem erreichte UDP meist die schnellste Datenübertragungsrate, unter anderem weil der Package Loss erhöht war. Die besten übergreifenden Metriken erreichte das QUIC Transport Protokoll. Diese Arbeit 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 durchaus eingesetzt werden kann.

Contents

List of Figures

List of Tables

List of Listings

1 | Introduction

Nowadays communication between smartphones invariably relies on infrastructure. In most homes in developed countries one can find a local network advertised by Wi-Fi 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 Internet Service Provider (ISP). Wherever we go our mobile devices are connected but relying on infrastructure that is not necessarily available all the time. Besides being restricted when infrastructure is broken communicating via ad-hoc technologies when possible takes unnecessary load off the infrastructure. Different reasons can lead to restricted or broken infrastructure. 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 simultaneously or environmental disasters could take down mediators or their power supply. In all these cases people would highly profit from ad-hoc networks, that do not merely rely on any external dependencies.

Fortunately, many different technologies already exist to gain access to wide area ad-hoc connections like Long Term Evolution (LTE)-Direct, 5G 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 P2P technologies in recent years allowing devices to directly communicate via local radio link broadcasts. These improvements in technology could potentially improve connectivity in the aforementioned cases or enable scenarios in which numerous end devices connect together, to form a wide spread mesh network. Some of the well known supported technologies in smartphones nowadays include Bluetooth, NFC or some variation of direct Wi-Fi that does not make use of an access point.

As of 2024 4 billion people worldwide are carrying smartphones (Statista Research Department, 2024) with them. A large portion of them are developed and manufactured by one of the most valuable companies in the world today. Since 2007 when the first iPhone was released on 27th of June Apple has sold more than 2.8 billion devices (Sherif, 2025b). This makes iOS one of the most used mobile operating systems worldwide with a current market share of 27% (Sherif, 2025a). iOS also already utilizes several P2P technologies which are predominantly used by Apples Continuity (Apple Inc., 2024a) which bundles applications like AirDrop or AirPlay, Universal Clipboard (UC), Handoff (HO) or Wi-Fi Password Sharing (PWS) for contacts that want to join your network.

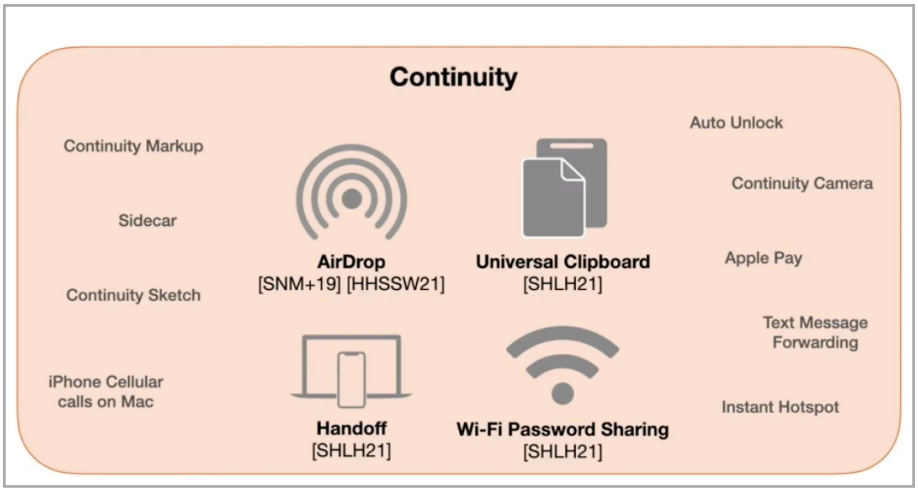

Figure 1: Summary of technologies encompassed by the Continuity platform. (Stute, Heinrich, Lorenz, *et al.*, 2021a)

Apple already makes use of P2P communications and also offers developers a way to do so with various frameworks that can be used. Under the hood of Apples Continuity platform however there mostly two P2P technologies used, Bluetooth Low Energy (BLE) and a proprietary Wi-Fi-Direct alternative AWDL. AWDL can originally only communicate with devices of the Apple ecosystem and 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 (Apple Inc., 2023a).

Moreover Apple provides the Network framework[1] which can directly access AWDL and the underlying IPS Transport Layer. The Transport Layer is composed of the protocols Transport Control Protocol (TCP), UDP and recently also QUIC which Apple also added support for starting from iOS 15.

After all the described aspects have been taken into consideration, the Network Framework, included in the iOS Software Development Kit (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. Since smartphone users tend to carry devices with them over the course of the day these tests will vary in different surroundings that represent typical locations, varying in crowd density or general ambient Radio Frequency Electro Magnetic Field (RF-EMF) levels. To be precise testing will be done in the Forest, on a free Field, in the Inner City and in an Underground station.

1.1 Research Definition

This research should help assess feasibility of applications that want to utilize AWDL connections on iPhones. 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 Network Framework is used to quantify the connection quality through metrics like RTT, Jitter, package loss and data transfer speed.

Research Questions

[1] https://developer.apple.com/documentation/network

Which aspects influence the P2P AWDL connection quality on iOS devices and what are they capable of?

Hypotheses

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, e.g. the distance with a measuring tape.

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 in various scenarios to quantify P2P connection quality.

2 | 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. Other than that iOS technologies used in the process are described.

2.1 Infrastructure Networks

The National Institute of Standards and Technology (NIST) defines infrastructure networks as "a wireless network that requires the use of an infrastructure device, such as an access point or a base station, to facilitate communication between client devices" (NIST, 2018). Using underlying infrastructure that spans over wide areas let users communicate to seemingly anywhere. This is achieved due to a widespread net of connected computers called the internet, which is defined by the Internet Engineering Task Force (IETF) as "the single, interconnected, worldwide system of commercial, governmental, educational, and other computer networks that share (a) the protocol suite specified by the Internet Architecture Board (IAB) (Request For Comment (RFC) 2026) and (b) the name and address spaces managed by the Internet Corporation for Assigned Names and Numbers (ICANN) (Shirey, 2007). While the user is connected he can communicate to nearly anywhere but when out of reach of the next entry point the user can not even transfer data to nearby device, no matter how close these might be.

2.2 Ad-hoc Networks

The NIST defines ad-hoc networks as "a wireless network that allows easy connection establishment between wireless client devices in the same physical area without the use of an infrastructure device, such as an access point or a base station" (NIST, 2022). Even when the next entry point to the internet is out of reach nearby devices can communicate with each other but are limited to the nodes that form this new separate network.

2.3 Satellite Phones

While satellite phones seemingly solve the mentioned problems they are not widely spread in the day-to-day use and only used for emergency services or roadside assistance in modern iPhones (Apple Inc., 2025a). Another problem that is increasingly present is space debris which has more than doubled since 2007 and is yet to increase with more and more space missions emerging. These junks of different parts of satellites or rockets can potentially destroy more satellites which again leads to more space debris or outages in GPS or satellite phoning services (European Space Agency, 2017).

2.3.1 Starlink Direct to Cell

While Starlink operates on similar ideas they offer multiple advantages over normal geostationary satellites. While geostationary satellites orbit at 35.786 kilometers Starlink satellites orbit much closer at about 550 kilometers from the Earth reducing latency and decreasing space traffic in the geostationary field. Starlink also claims that their satellites include a collision avoidance system which notices potential collisions and actively dodges the other object (Starlink, 2025).

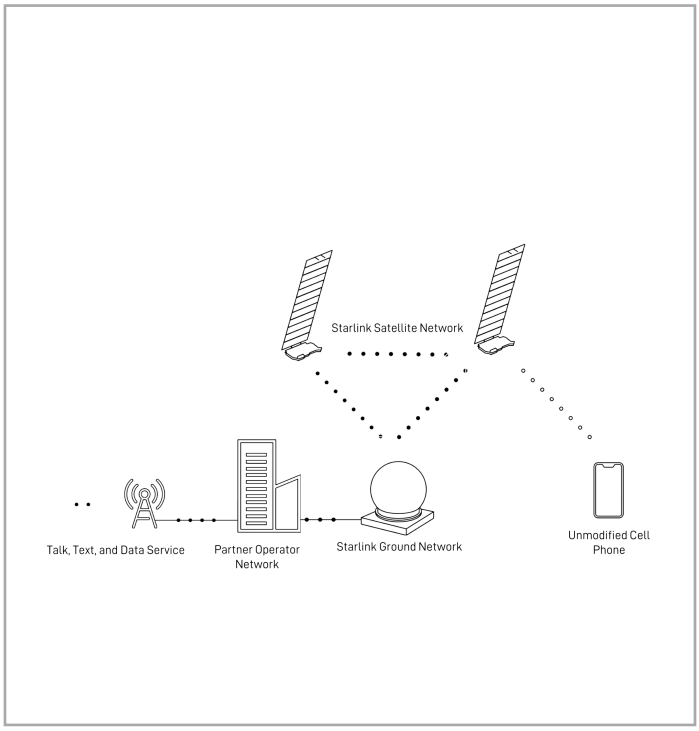

Figure 2: Abstract structure of Starlink's network. (Starlink, 2025)

2.4 Specific Ad-hoc Technologies

While and solve the issue of dead spots, they also rely on infrastructure which needs to be taken care of or can be damaged. The following is an incomprehensive list of ad hoc technologies.

2.4.1 Wi-Fi

Wi-Fi is a trademark for IEEE wireless communication standard 802.11 based technologies which already exists for over two decades. The IEEE 802.11 standard defines the protocols that are used to establish a connection with current Wi-Fi wireless nodes, including routers or access points whereas the correlating Wi-Fi versions is just used for marketing purposes and matches an underlying 802.11 specification (Wi-Fi Alliance, 2023; Cisco Systems, Inc., 2025).

Generational name	Technology supported
Wi-Fi 7	802.11be
Wi-Fi 6	802.11ax
Wi-Fi 5	802.11ac
Wi-Fi 4	802.11n

Table 1: Wi-Fi versions and the correlating IEEE 802.11 standards.

The Wi-Fi Direct trademark enables Wi-Fi devices to connect directly without underlying infrastructure. However this specification has not been widely adopted because of high energy consumption and lack of performance where establishing a connection could take four to ten seconds (Camps-Mur, Garcia-Saavedra and Serrano, 2013). Wi-Fi Direct is not available in iPhones (Quinn "The Eskimo!", 2015).

2.4.2 Bluetooth

Bluetooth a short range wireless technology enables connection between two nearby devices without relying on supporting infrastructure very similar to Wi-Fi Direct. The protocol operates on 2.4 GHz and it features two separate standards today, Bluetooth Classic and Bluetooth Low Energy (BLE) which is optimized for low energy consumption. Today Bluetooth is mostly used to connect computers to external peripherals like mice, keyboards or headphones. Using Bluetooth for data transfer is not preferred since its data rate is very limited. (Intel Corporation, 2022; Cybersecurity and Infrastructure Security Agency, 2021). The benefit of using Bluetooth is its wide spread and compatibility between different vendors.

2.4.3 LoRaWan

LoRaWan specification is a Low Power, Wide Area networking specification created to connect Internet of Things (IoT) devices to the internet. The specification features key requirements for the IoT use such as bi-directional communication, end-to-end security or location services. Also this specification is optimized to use little energy consumption since LoRaWan usually target IoT devices that are far away and not connected to any cables. It normally operates in unlicensed frequency bands and is capable of communicating up to 15 kilometers in rural areas. While this key features would also perfectly suit iOS P2P communication no support for this technology is given on iPhones (LoRa Alliance, 2024).

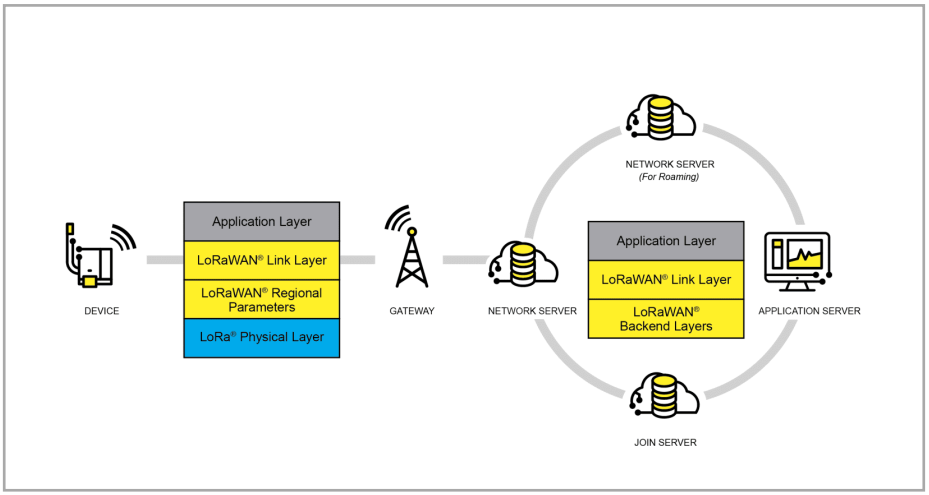

Figure 3: Abstract structure of LoRaWan network. (LoRa Alliance, 2021)

2.4.4 5G SL

5G SL the successor of LTE-Direct is capable of connecting User Equipment (UE)s directly without an intermediate base station but also to act as a relay for UEs toor far away of the next Next Generation Node B (gNB) (Vijitha, Weerackody, Kent, Benson and Sumit, Roy, 2023). This is generally designed for public safety or military operations used for Unmanned Aerial Vehicles (UVA)s (Barnes, Maheu and Kuzin, 2023) although approaches existed to introduce it into commercial markets (Qualcomm Technologies, Inc., 2014). Again iOS peer-to-peer communication would highly benefit from such technologies but unfortunately no developer support for this technology is given (Apple Inc., 2024b).

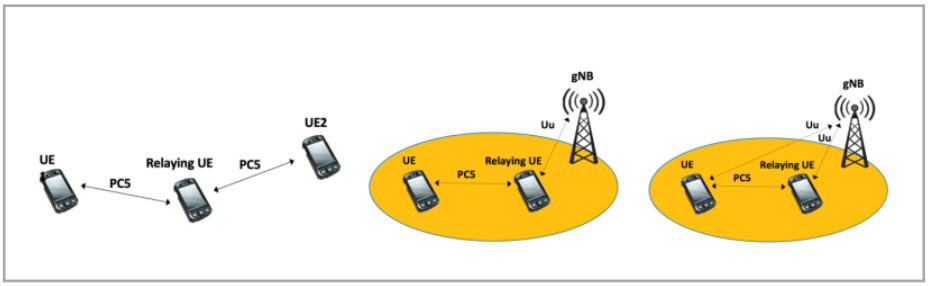

Figure 4: Abstract structure of 5G SL network. (IEEE Communication Society, 2023)

2.4.5 Near Field Communication (NFC)

NFC is a communication technology which operates at a base frequency of 13.56 Mhz. It can transfer data with a typical range of 2cm and data rates up to 1.7 Mbps. The technology is also used to connect to non powered peripherals such as bank cards[2]. Apple makes this technology also accessible to iOS and iPadOS developers but explicitely states that it is not supported in other Apple platforms[3].

2.4.6 Ultra Wide Band (UWB)

UWB in general is a radio technology focused on precise ranging and locating using a low energy density over a large radio spectrum (Android Developers, 2025). Apple allows developers to access

[2]https://nfc-forum.org/learn/nfc-technology/
[3]https://developer.apple.com/design/human-interface-guidelines/nfc
[4]https://developer.apple.com/nearby-interaction/

their UWB interface on iPhones and Apple Watches through the Nearby Interaction framework[4] which is built to locate nearby devices also using the distance and direction.

2.5 iOS

The following part tries to familiarize with technologies used in the testing process of this thesis.

2.5.1 Bonjour

Bonjour is a former proprietary zero-configuration network protocol suite over Internet Protocol (IP) that Apple has submitted to the IETF. The proposed zero-configuration solutions covers IP addressing, name-to-address translation on local networks using multicast Domain Name System (mDNS) and service discovery. Using Bonjour on Apple platforms is done via appropriate frameworks leaving the responding of mDNS queries to the mDNSResponder daemon (Apple Inc., 2013).

2.5.2 IPS

Most of networking in mobile devices relies on the TCP/IP protocol suite also called IPS. The IPS is a set of networking protocols specified by the IETF also often referred to as "TCP/IP" protocol stack. It is split into five protocol layers – Application, Transport, Internet, Network Interface and Network Hardware –, however for this thesis only the first two are relevant and listed below (Shirey, 2007).

```
 IPS Layers     Examples
--------------- ---------------
Message Format: ARPA (RFC 822)
+-------------+
| Application | SMTP (RFC 821)
|             |
|             |
|             |
+-------------+
 |  Transport  | TCP  (RFC 793)
 |             |
 +-------------+
 |  Internet   | IP   (RFC 791)
 +-------------+
 |   Network   | IP over IEEE
 |  Interface  | 802 (RFC 1042)
 +-------------+
 -   Network   - The IPS does
 -  Hardware   - not include
 - (or Network - standards for
 -  Substrate) - this layer.
 + - - - - - - +
```

Listing 1: Textual graphic showing the defined layers of the IPS (Shirey, 2007).

2.5.2.1 Application Layer

The Application Layer covers the data the application program run by the user wants to transmit and only interacts with the next lower Transport Layer. Based on the applications needs data can be transferred as a continuous stream or package based where the Transport Layer handles interaction with the next Internet Layer (Network Working Group, 1989).

2.5.2.2 Transport Layer

The Transport Layer "divides application data into packets, adds a destination address to each, and communicates them end-to-end – from one application program to another – optionally regulating the flow and ensuring reliable (error-free and sequenced) delivery" (Shirey, 2007).

TCP

TCP is an internet standard, transport layer protocol that reliably transmits data in the same order via error checked delivery and congestion control via Congestive-Avoidance Algorithms (CAA). It can be directly accessed on Apple platforms using the C based BSDSockets or the Network Framework (Apple Inc., 2023b).

Nagles Algorithm

Due to the 20 byte TCP header there has been a relatively high overhead when sending small packages which in worst case could lead to congestion collapse considering the error prevention of the TCP protocol. This algorithm therefore inhibits the sending of new TCP segments as long as no previously transmitted data stays unacknowledged. This algorithm is enabled by default on iOS systems[5] and while testing the prototype has lead to highly reduced IP package number sent compared to the data slices sent from the Application Layer (Nagle, 1984).

UDP

UDP is a Transport Layer protocol that implements the "fire-and-forget" concept. Packages are sent whenever data is received from the Application Layer without guarantee that they will be delivered or that they will be received in the same order they had been sent.

QUIC

In 2012 another transport protocol was developed by engineers at Google called QUIC. QUIC builds upon UDP and is oriented to replace TCP based applications since it also features congestion and error control features. Compared to UDP and TCP QUIC has built in Transport Layer Security (TLS) 1.3 support and does not allow non encrypted connections.

2.5.3 AWDL

AWDL was developed by Apple due to concerns regarding Wi-Fi Alliance's Wi-Fi Direct specification and eventually got adopted by the Wi-Fi Alliance as the basis for Neighbor Awareness Networking (NAN) (Cheshire, 2018). It is undocumented (Stute, Kreitschmann and Hollick, 2018a), based on IEEE 802.11 ad hoc protocol and built to let mobile devices communicate directly with each other without utilizing an intermediary access point. It is heavily used in Apple's Continuity platform (Stute, Heinrich, Lorenz, *et al.*, 2021b).

2.6 Summary

Although all wireless communication technologies serve the same purpose a lot of variants have emerged that are fine tuned for more precise use cases. Since infrastructure networking has nowadays proved its worth upcoming ad-hoc solutions tend to be compatible with the overlaying IPS like AWDL.

[5] https://developer.apple.com/documentation/network/nwprotocoltcp/options/nodelay

3 | Related Work

The following is a non-comprehensive discussion covering previous research of P2P technologies in the mobile context. After covering historic considerations of MANET research and device to device D2D communication in cellular networks via standards like LTE-Direct or 5G New Radio (NR) SL, a deeper introduction to Apples P2P 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.

3.1 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 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 technical 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 infrastructure 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 Short Message Service (SMS) or Global System for Mobile Communications (GSM)/Code Division Multiple Access (CDMA) based technologies which were connected to a backend payment server. They noticed that these implementation, which were just too heavily relying on external systems could not replace cash based systems and aimed to develop a decentralized 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 Wi-Fi technology on Android devices to share data between them. While Wi-Fi-Direct and Bluetooth were also accessible to the developers, they decided to use a combination of Wi-Fi AP mode and Wi-Fi 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 application by sharing pictures among twelve devices from several vendors using no fixed existing infrastructure. This project also strongly reminds on Apple proprietary software AirDrop which has been released by Apple in 2011.

Figure 5: 12 Devices sharing images using local radio broadcasting. (Dubois, Bando, Watanabe, *et al.*, 2013)

Since then support for direct P2P 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, Wi-Fi 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.

3.2 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 UVA or public safety networks (Gamboa, Henderson, Garey, *et al.*, 2024), like Vehicle To Everything (V2X). Most of this research builds upon 5G NR SL which has implemented protocol support for V2X and Proximity Services (ProSe) for public safety networks which allows UE to directly talk to each other without the interference of a gNB.

Although approaches existed to also introduce D2D communication to the commercial markets back in 2014 by Qualcomm (Qualcomm Technologies, Inc., 2014) 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 Wi-Fi-Direct, according to Apple engineers no support for this technology is given on mobile smartphones for third party developers (Quinn "The Eskimo!", 2023). This is also pointed out by the authors of this critical review of mobile D2D communications (Desauw, Luxey-Bitri, Raes, *et al.*, 2023).

3.3 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 Wi-Fi based protocol the authors stumbled across various security concerns which they mainly focused on next to Apple's BLE 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 by Apple, the authors dedicated a separate paper on researching security and privacy issues of 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 Media Access Control (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, however 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 (Stute, Narain, Mariotto, *et al.*, 2019).

In 2020 Ian Beer a british computer security 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 then successfully exploited to steal sensitive information like emails, photos, message or even the keychain (Beer, 2020).

In 2021 the authors of the OWL project dedicated another paper to the analysis of Apple's offline file sharing service AirDrop which uses AWDL under the hood. The authors discovered two design flaws 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 Personal Computer (PC)s 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 Universal Unique Identifier (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 Google's similar platform "Nearby" for Android. They also open sourced this implementation as part of their OWL project (Heinrich, Hollick, Schneider, *et al.*, 2021).

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 HO, UC and Wi-Fi PWS. During this process they again found several security issues which could lead to possible DoS or MitM attacks. To demonstrate their findings the authors implemented a Proof of Concept (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 (Stute, Heinrich, Lorenz, *et al.*, 2021b).

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 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 then query the Apple server for entries sent by these finder devices to get an estimated location of his/her 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 the rolling advertisement keys are cached and stored on the filesystem as clear text (Heinrich, Stute, Kornhuber, *et al.*, 2021).

3.4 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 SEEMOO 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 demystifying the underlying protocols and analyzing those with regard to security concerns.

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 analyzed. 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 analyzing 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 Interface (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 also utilizes 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 (AO Kaspersky Lab, 2025). This however violates Apple's iOS Software License Agreement and testing could potentially disturb restricted frequency bands that are licensed (Apple Inc., 2025b). 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 possible. 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 and store 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

After discussing the broad approach to testing in Section 4 Figure 6 nicely shows the things already covered. The following section tries to get more narrow on defining the tools used and the testing done to fill the below image with more information.

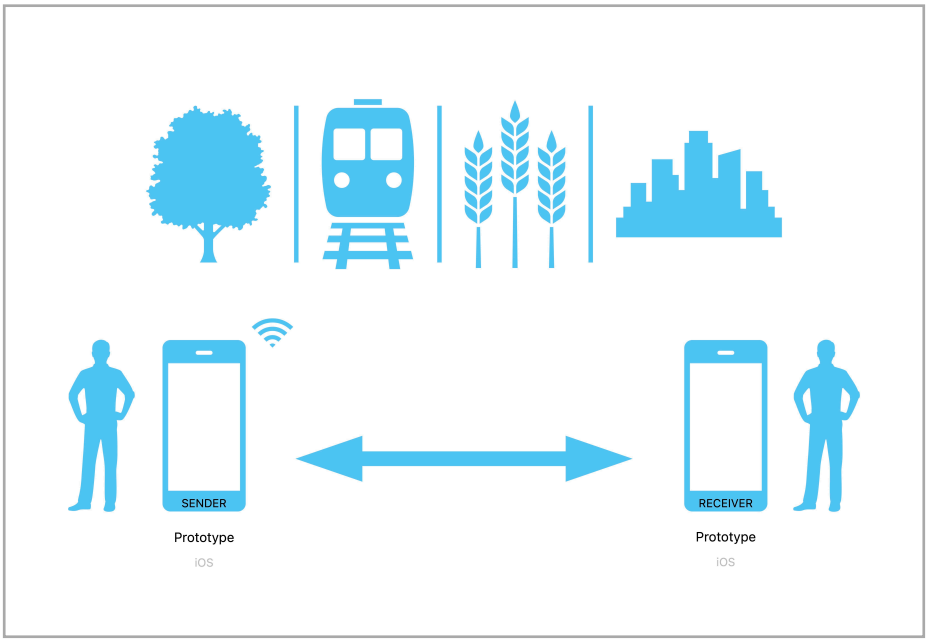

Figure 6: Abstract representation of testing concept without implementation details.

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 Universal Resource Locator (URL) or directly installed by the developer machine. The application is written using a modified version of the Model View View Model (MVVM) Graphical User Interface (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.

[6]https://developer.apple.com/xcode/swiftui/

5.1.1 User Interface

The GUI is developed using SwiftUI[6], 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.

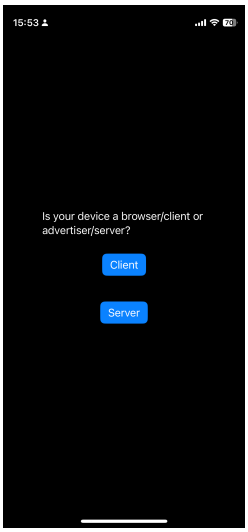

Figure 7: Screenshot of decision screen to select server/client configuration.

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.

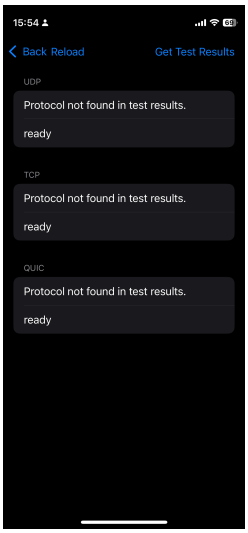

Figure 8: Screenshot of server screen showing underlying protocols and associated metrics.

5.1.1.3 Client Views

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

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.

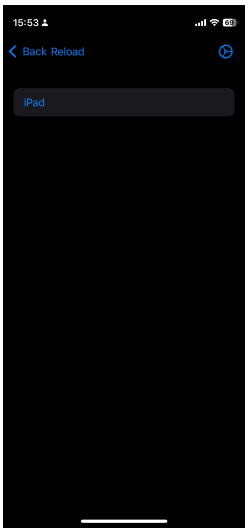

Figure 9: Screenshot of browser screen to select nearby advertisers.

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 reinitializes all client objects and navigates the user back to the browser view. There the user can select another server to connect to.

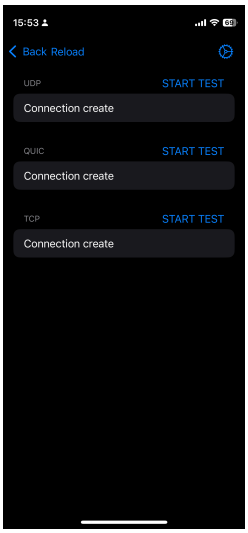

Figure 10: Screenshot of testing screen to start testing and display associated information.

5.1.2 Networking Frameworks

Apple provides different frameworks for P2P connections using different layers of abstraction or different underlying technologies. 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, P2P Wi-Fi, and Bluetooth personal area networks for the underlying transport. In macOS and tvOS, it uses infrastructure Wi-Fi, P2P Wi-Fi, and Ethernet"[7]. 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 (Quinn "The Eskimo!", 2017).

In an approach to give a brief overview about Apples networking APIs, Apple describes Multipeer Connectivity as a high-level interface to Apples P2P Wi-Fi support and also introduces the Network Framework, which is considered a low-level interface by Apple engineers (Quinn "The Eskimo!", 2024). 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. (Apple Inc., 2023b)

Nearby Interaction is yet another framework to establish P2P connections. It uses the iPhones UWB chip to "locate and interact with nearby devices using identifiers, distance, and direction"[8]. 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 (Apple Inc., 2021). However in Apples article about the advanced ranging capabilities of second generation UWB chips which are included in iPhone 15 and above they use a maximum distance of 50 meters (Apple Inc., 2025c).

[7] https://developer.apple.com/documentation/multipeerconnectivity
[8] https://developer.apple.com/nearby-interaction/

Nevertheless following Apples recommendations documented in a tech note about choosing the right networking API, the Networking framework is suggested for establishing a connection and transferring data. (Apple Inc., 2023b)

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 selected 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)
         }
     }
}
```

Listing 2: Configuration of transport protocols used while testing.

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

Listing 3: Injection of configured servers to view models.

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 in both `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"
    }
}
```

Listing 4: Configuration of transport protocol parameters that can be used for testing.

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.

After adding it to the bundle[9] 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)!
)
```

Listing 5: Setting the local identity used while TLS 1.3 handshake on server.

[9] https://developer.apple.com/documentation/foundation/bundle

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

Listing 6: Setting the verify block used while TLS 1.3 handshake on client.

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 object 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())
}
```

Listing 7: Initialization and starting of network listener for injected transport protocol.

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 browser result in the `BrowserView` this instance 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.

```
1  func createConnection(with browserResult: NWBrowser.Result) -> Error? {
2      let nwConnection = NWConnection(
3          to: browserResult.endpoint,
4          using: transportProtocol.parameters
5      )
6
7      self.connection?.cancel()
8      self.connection = nil
9      self.connection = C(nwConnection)
10
11      if case let .failed(error) = self.connection?.state {
12          return error
13      }
14      return nil
15  }
```

Listing 8: Initialization of connection on client using a browser result.

5.1.4.1 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[10].

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 simultaneously. 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.

5.1.4.2 Displaying and Selecting Advertisers

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

[10]https://developer.apple.com/documentation/bundleresources/information-property-list

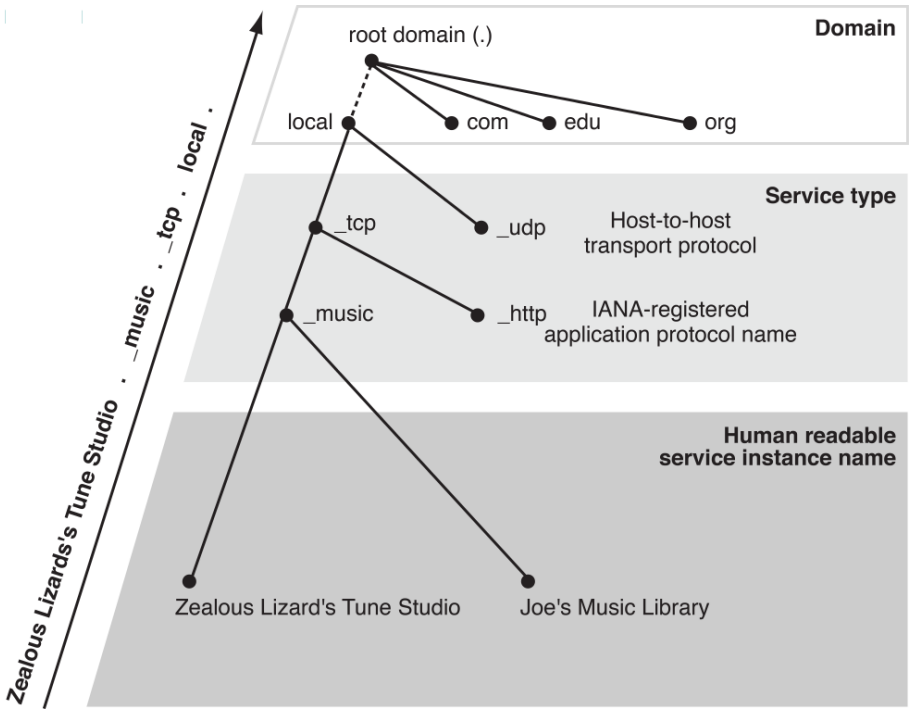

Figure 11: Graphic showing the Bonjour naming convention. (Apple Inc., 2013)

In case of this test application the Bonjour service name od the server device is configured using the `UIDevice.current.name` which represents a generic device name like “iPad” or “iPhone”[11]. This name is extracted from the Bonjour `NWEndpoint` on the client side and listed in the Browser View Figure 9 so the user can choose the server device he wants to test.

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

Listing 9: Extension for extracting the name of the Bonjour service string.

[11]https://developer.apple.com/documentation/uikit/uidevice/name

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

Listing 10: How the view model listens to the browser result changes of the client object.

Using the same service type like mentioned in the previous section Section 5.1.4.1 Bonjour would automatically rename the service if it detects the same service on the local network (Apple Inc., 2013).

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

Listing 11: Comparison of Bonjour services that tried to use the same name and transport protocol.

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

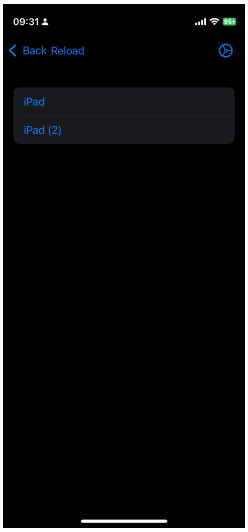

Figure 12: Screenshot of UDP and QUIC services using the same Bonjour service type.

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 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 are sent to measure transfer speed 1000 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)
```

Listing 12: How the time span is precisely calculated using kernel functions that return tick count.

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
```

Listing 13: Signature of the `startTesting` method that is provided by client objects.

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, using the iPhone 12 mini as sender/client and the iPhone 15 Pro as receiver/server.

5.2.1 Places

Testing is done in four different places representing typical places for iPhone users. One testing environment will be an 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 as the inner city or the underground but also like the field minimal in RF-EMF disturbances that could negatively impact the connection. (Loizeau, Zahner, Schindler, *et al.*, 2023)

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
1 000	4 096	5
10 000	9 216	10

Table 2: 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 RTT, Jitter, data transfer speed. These metrics will be measured in all combinations of the above mentioned scenarios.

Protocols	TCP	UDP	QUIC

Table 3: Transport Protocols used for testing.

RTT and Jitter are measured on the client and tested separately from the transfer speed. The variations listed in Section 5.2.2 are only relevant for transfer speed since the other two metrics are tested before using 1000 distinct packages including the kernel time. The RTT used to evaluate the findings will be the average RTT of these 1000 packages that are sent per cycle.

Metrics	Calculation
RTT (ms)	$\overline{RTT} = \frac{1}{N}\sum_{i=0}^{N} RTT_i$
Jitter (ms)	$Jitter = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(RTT_i - \overline{RTT}\right)^2}$
Data Rate (Mbps)	$DataRate = \frac{count*size}{duration}$

Table 4: 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) and the metrics are displayed to the user ready for recording and saving as Comma Separated Values (CSV). 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. For every combination of package size, number of packages, distance and transport protocol two distinct rounds will be done resulting in 162 individual tests per scenario. Before these tests to measure data transfer speed are executed RTT and Jitter are tested 5 times with 1000 packages for each distance and transport protocol resulting in 45 individual tests per scenario.

Metrics	Scenarios	Protocols	Distances in Meters	Number of Packages	Package size in Bytes
RTT	Underground	TCP	1	100	128
Jitter	Inner City	UDP	5	1 000	4 096
Data Rate	Free Field	QUIC	10	10 000	9 216
	Forest				

Table 5: Definition of testing scenarios and variations.

After getting more precise on how testing will be done the aforementioned Figure 6 can be filled out as follows adding the specific testing environments, measured data and protocol stack.

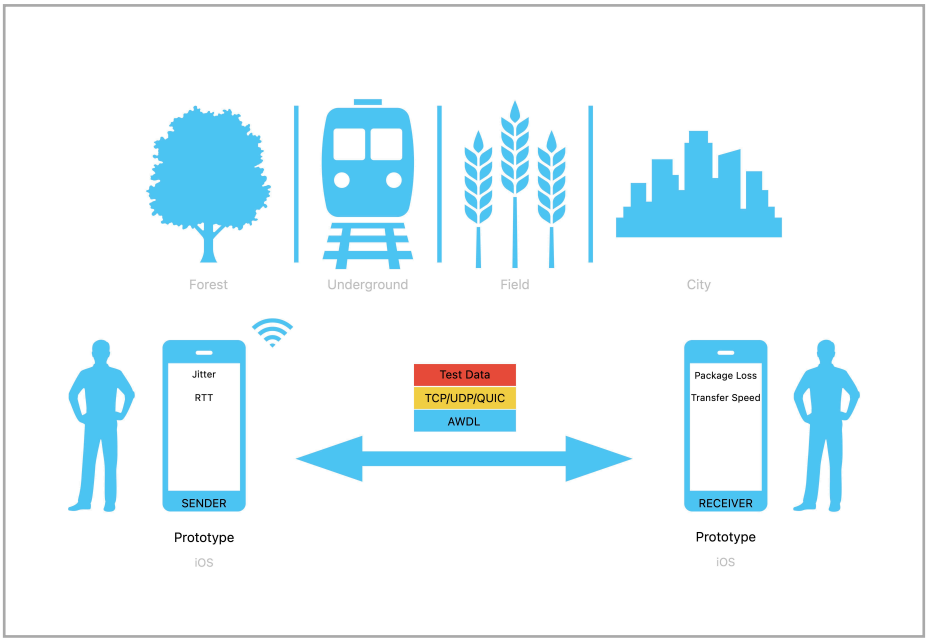

Figure 13: Abstract representation of testing concept including implementation details and protocol stack.

After collecting the described data like shown above they will be visualized and used to test the hypotheses in the following Section 6.

6 | Results and Evaluation

The following section shows the scenarios the testings were executed at and interprets the tested metrics. Testing has been done on 1.Mai 2025 in Vienna around noon. The weather was sunny and it had around 25 degrees celsius.

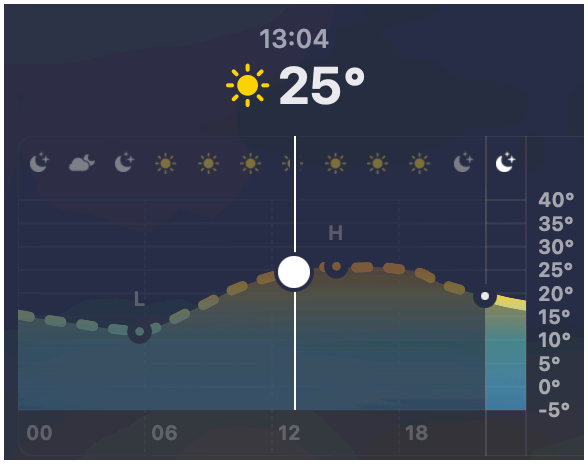

Figure 14: Weather at day of testing.

6.1 Field

Testing on the field started at around 11:38 and ended at around 12:15 at the coordinates `48,22695° N, 16,24221° E`.

Figure 15: Field at time of testing.

6.2 Forest

Testing in the forest started at around 12:21 and ended at around 12:37 at the coordinates `48,22605° N, 16,24379° E`.

Figure 16: Forest at time of testing.

6.3 Inner City of Vienna

Testing in the inner city of Vienna started at around 13:31 and ended at around 13:47 at the coordinates `48,20837° N, 16,37043° E`.

Figure 17: Inner city of vienna at time of testing.

6.4 Underground

Testing in the Underground started at around 14:02 and ended at around 14:12 at the coordinates `48,20815° N, 16,37162° E`.

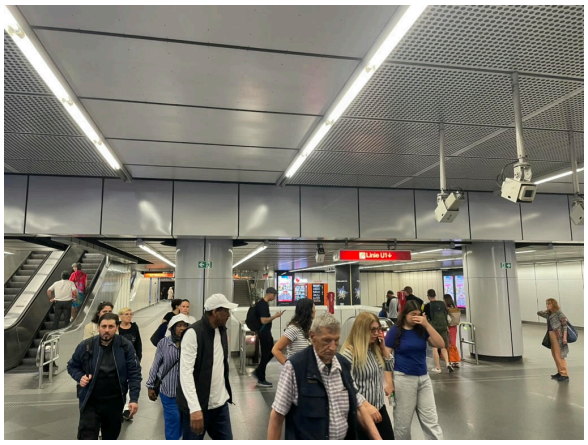

Figure 18: Underground at time of testing.

6.5 Interpretation

6.5.1 Effects of the surroundings

The first hypothesis claims that direct P2P AWDL communication between iOS devices depends on the surroundings and functions worse in crowded areas. Data shows that this is not the case in general, although when a pack of people walked between the devices transfer speed got seemingly worse which can also be seen at increased outliers of RTT and Jitter in the inner city.

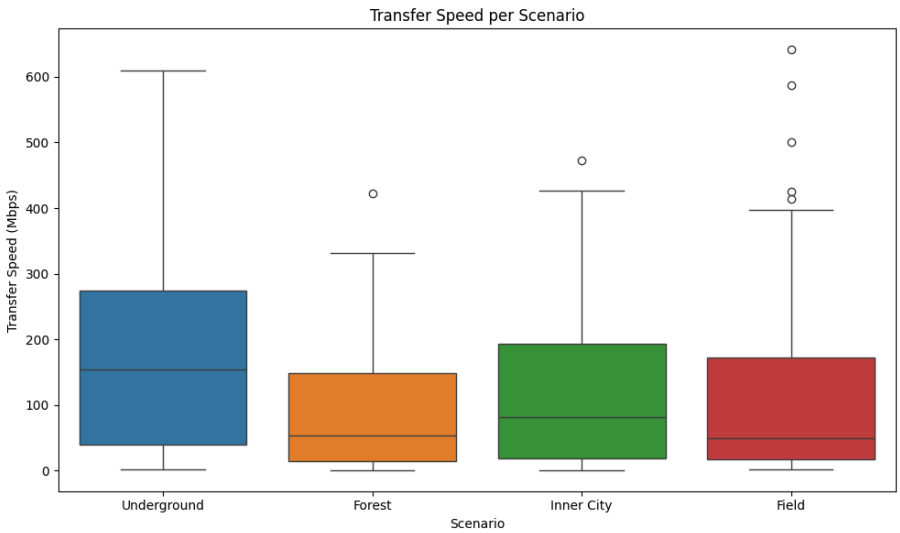

Figure 19: Transfer speed per scenario.

Measurements of the average RTT shows massive outliers in the inner city which again is a hint at the loss of connection quality when people blocked the direct view. Other than that one can see that the median of the measurements in the underground is the lowest.

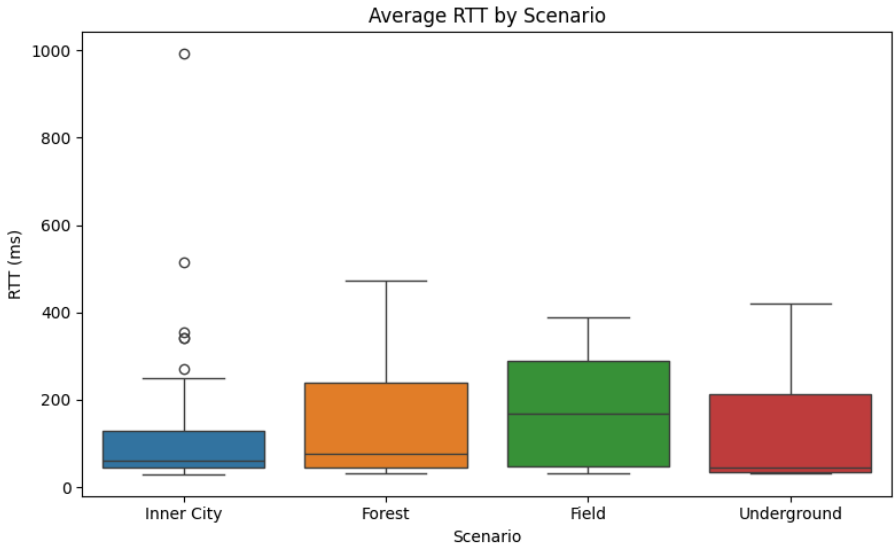

Figure 20: RTT per scenario.

Measurements of the Jitter do not show any reliabilities on the surroundings. Although the Jitter might seem worse in the Forest on the first sight, comparing it to the outliers and the median of the other scenarios it seems that this assumptions is based on how the data is represented and some randomly increased density in upper outliers.

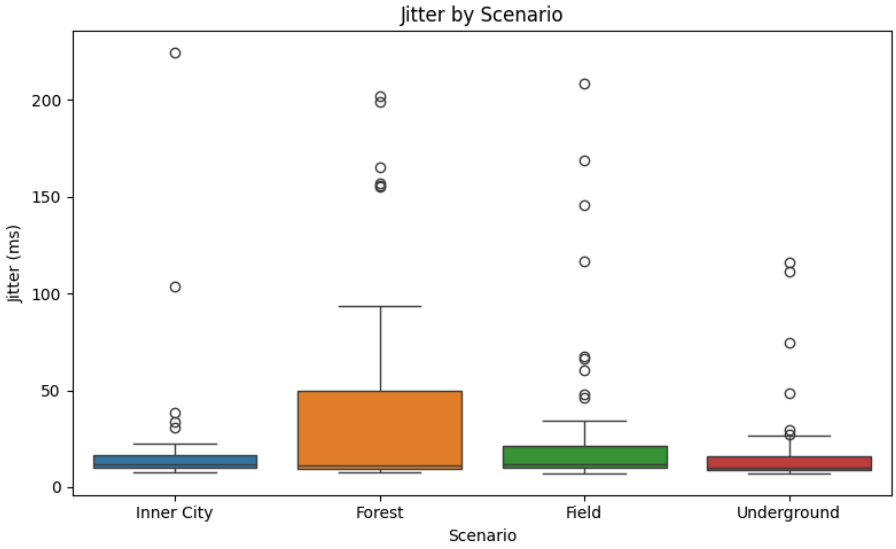

Figure 21: Jitter per scenario.

This is also supported by a comparison of the statistical values of the Jitter in the different scenarios. It shows that besides some accumulations in the higher sector of the measurements in the forest leading to a higher 0.75 quantile, no differences to the other scenarios can explicitly pointed out in the other quantiles.

Scenario	count	mean	std	min	25%	50%	75%	max
Field	45.0	30.55262	44.80011	6.896699	10.279047	11.665953	21.19959	208.395091
Forest	45.0	40.57642	56.34134	7.429105	9.504099	11.492499	49.980442	201.616895
Inner City	45.0	20.65895	34.49646	7.86939	10.144808	11.731033	16.857606	224.581483
Under-ground	45.0	19.01231	23.86936	6.923351	8.880072	9.80344	16.157669	116.316138

Table 6: Statistical metrics of Jitter in different scenarios.

Measurements of the package loss showed that the connection worked best in the underground while not seeing big differences comparing the other scenarios.

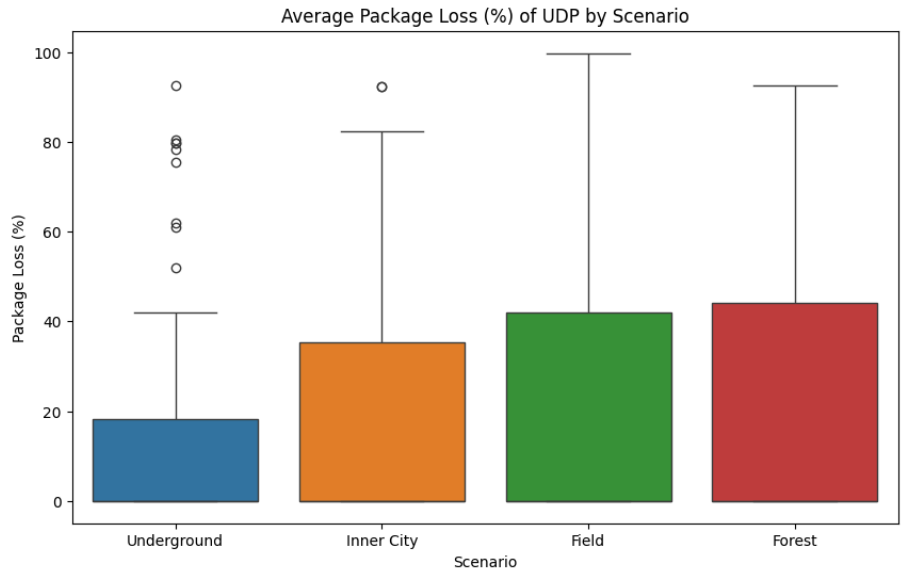

Figure 22: Package loss per scenario.

6.5.1.1 Conclusion

Data shows that connection quality indeed depends on the surroundings but cannot be linked directly to function worse in crowded areas. While RF-EMF seems to have no effect at all, blocking the direct sight of view severely decreased connection quality. This can be seen in the extreme outliers of RTT and Jitter in the inner city which can be attributed to a group of people walking between the testing devices. This did not happen in the forest because trees and bushes did not move to block view of sight completely during testing. Contrary to initial assumptions data also shows that connection quality was best in the underground. This could be attributed to all the metal panels installed on the roof and walls in the underground stations which might better reflect electro magnetic signals.

6.5.2 Effects of the Transport Protocol

The second hypothesis claims that P2P AWDL connection quality depends on the Transport Layer protocol used. Considering the measured metrics this hypothesis can be proven correct.

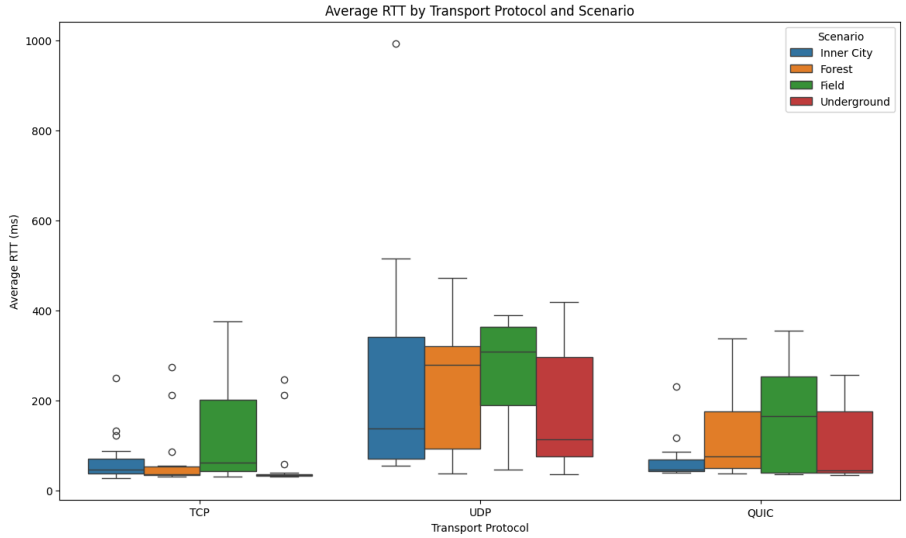

Figure 23: RTT per protocol and scenario.

TCP has significant lower RTT compared to UDP or QUIC which is attributed to Nagle's Algorithm. What is fascinating however is that QUIC achieves lower RTT than UDP even though QUIC is based on UDP and on top includes error correction and reliable delivery. Also previous tests using the testing prototype have shown that QUIC does not implement comparable algorithms like Nagle's Algorithm and always sends the same number of IP packages as UDP does.

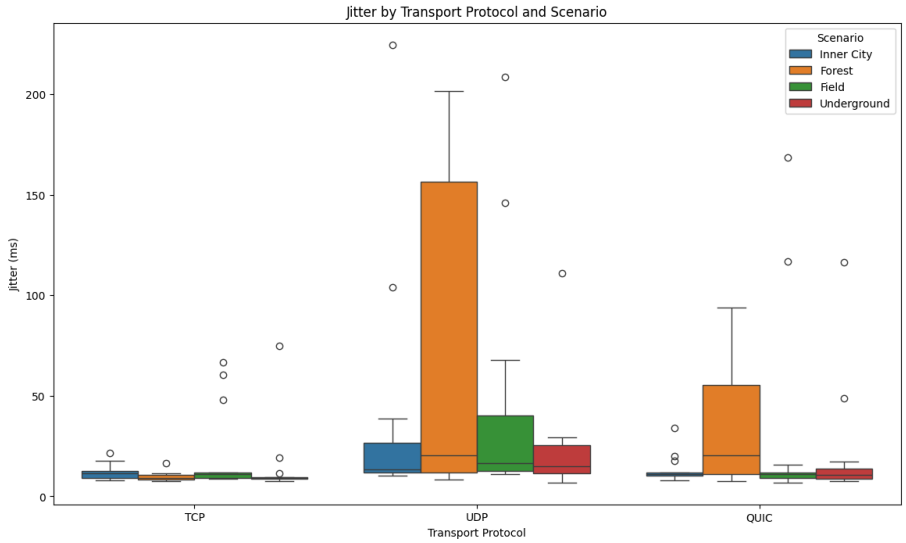

Figure 24: Jitter per protocol and scenario.

Again Jitter, like RTT is relatively low compared to the other Transport Layer protocols due to Nagle's Algorithm. But what again sticks out is that QUIC has generally lower Jitter than UDP.

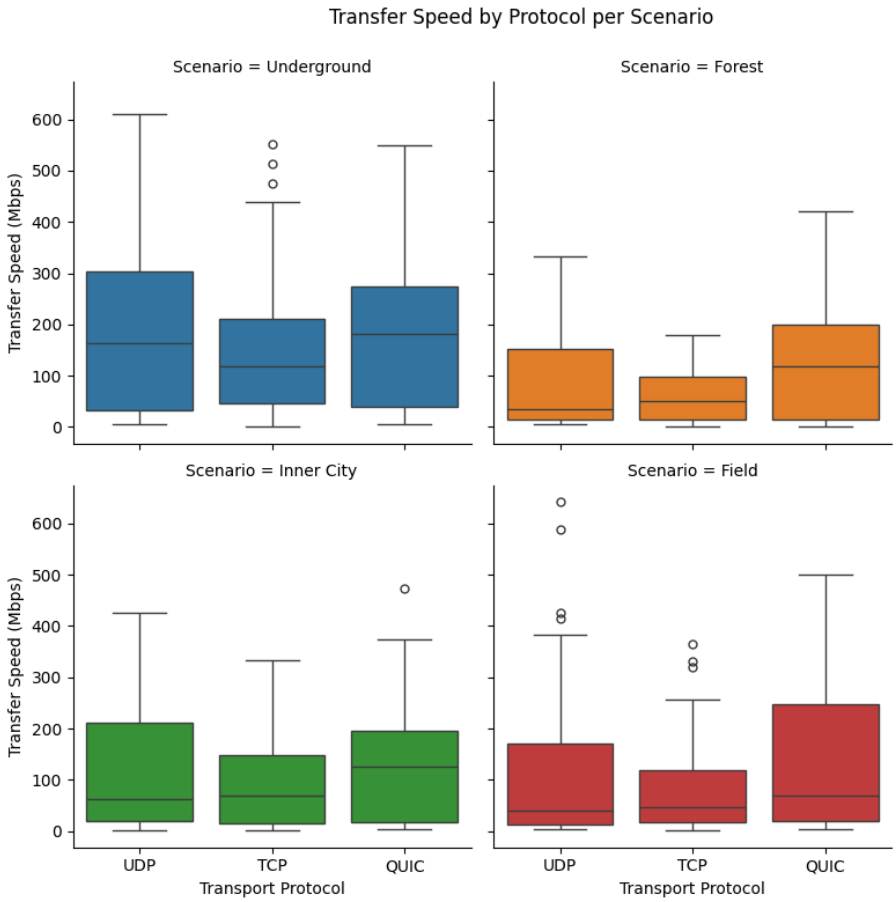

Figure 25: Transfer speed per protocol and scenario.

Figure 25 shows that using TCP leads to the lowest data transfer speed. UDP is particularly high because of unreliable delivery which makes the time the server received messages very low and leads to a higher transfer speed. QUIC has again performed very well considering reliability of data transfer, has outperformed TCP in some scenarios and even UDP on the Field and in the Forest.

6.5.2.1 Conclusion

Although TCP started with some seemingly major advantages due to Nagle's Algorithm QUIC outperformed it in terms of transfer speed in every scenario. Considering the high package loss of UDP especially when more packages were sent Figure 26 which inevitably lead to a higher transfer speed and the relatively close advantage in transfer speed compared to QUIC for most applications QUIC might be the best Transport Layer protocol for AWDL.

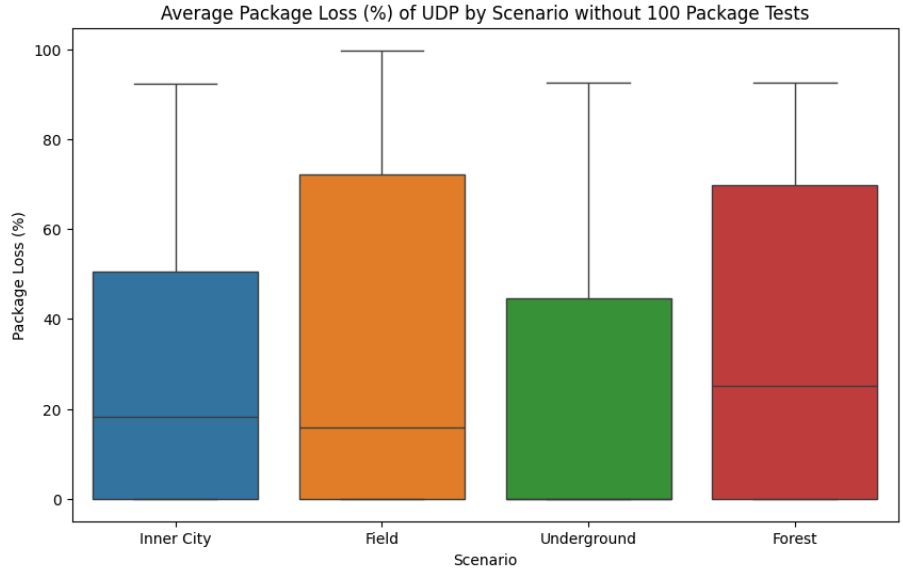

Figure 26: Package loss of UDP per scenario without tests using only 100 packages.

6.6 Other findings

During testing some other incidents happened which were not part of this research but should be shared anyways.

6.6.1 Establishing connection was difficult

Establishing a connection namely finding the Bonjour service in proximity got increasingly bad with more distance. It came to a point where connection establishment was done very close to only then create the distance that should be tested.

6.6.2 Distance effect on transfer speed

The distance between the iOS devices had a noticeable impact on the data transfer speed which is very nicely shown by Figure 27. Data however shows that the transfer speed decreases less from five to ten meters than from one to 5 meters.

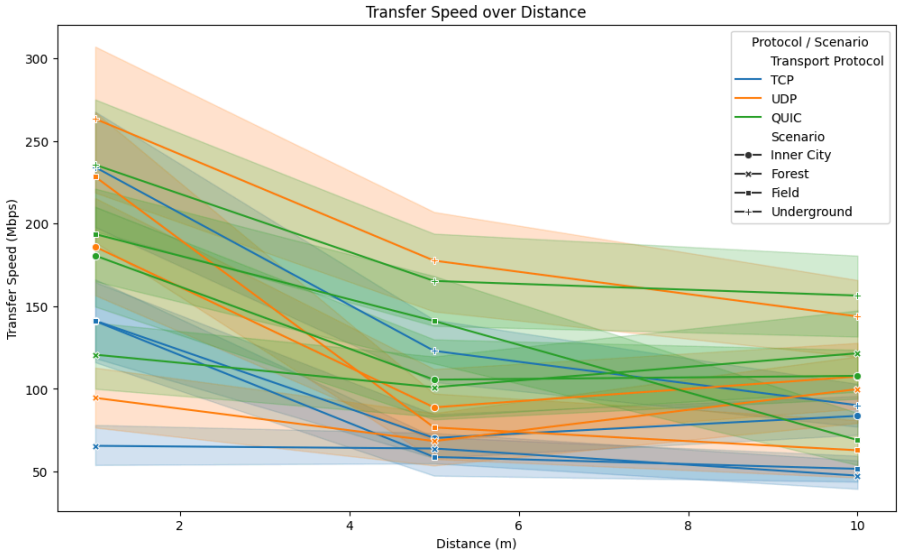

Figure 27: Transfer speed over distance.

While transfer speed decreased over distance no effect on the RTT can be seen.

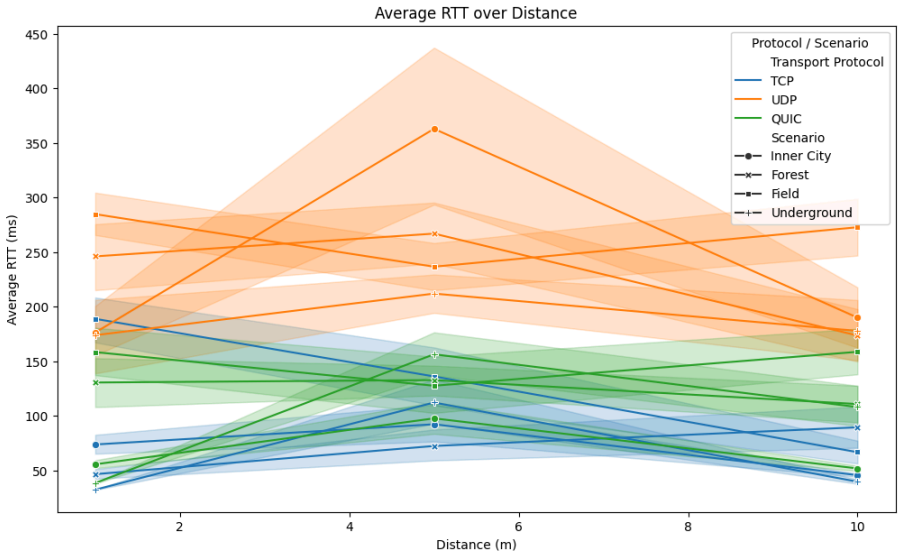

Figure 28: RTT over distance.

6.6.3 Differences of Field and Underground

The first testing scenario was the field. The initial planned distances were 1/10/30 meters but during testing at the distance of 10 meters on the field the connection was so unstable that it was nearly impossible to get data so we decided to change the distance of thirty to five meters. The last testing scenario was the underground where testing overall went very well and we gave some bigger distances another try. There distances over thirty five meters were no problem at all and transferred data super fast. As already mentioned the good connection in the underground could be due to the metal plates mounted on the roof and the walls that perfectly reflect electromagnetic signals and might act as a big antenna. On the other extreme on the field the bad performance could be attributed to the extreme

heat since it also was the only place with direct sun light exposure which is generally known to heat up smartphones very much.

7 | Conclusion and Outlook

Overall, considering the big research question, forming big mesh networks over wide areas throughout a city might not be possible yet due to technical and especially regulatory reasons. What we found in this research however is that connection quality highly depends on the surroundings, although not in the sense we might anticipated in the first place. Forest and inner city did not differ in metrics the way we thought it would do so we conclude that RF-EMF pollution might not be as negatively impacting as we thought. What we found however in the inner city, while generally performing well considering Jitter and data transfer speed, is that groups of people that were blocking the view of sight caused outliers in RTT and Jitter. Also we found that using different transport protocols also effects metrics like RTT, Jitter and data transfer speed, but showed that for most cases QUIC might be the best fit, especially because there might soon be a standard to also support an unreliable datagram extension for QUIC which has not been tested in this research (Pauly, Kinnear and Schinazi, 2022).

7.1 Outlook

What was unfortunately not covered by this research but would be still interesting to explore considering the issues we had on the field is how device temperature effects connection quality metrics and also connection establishment using the same protocol stack (AWDL and Bonjour). Another known unknown is why service discovery seemed more difficult, considering the distance than transferring data to a known host or how different devices, newer or older, iPad, iPhone or Mac change these metrics.

Another topic we did not focus on in this research per se but was very present in Section 3 was device security. It is needless to say that smartphones nowadays own enough personal data to make a huge part of ones identity. If this is a good or a bad thing is a philosophical question we do not want cover here in more detail but we urge everybody, especially big vendors like Apple, Samsung and co to be much more sensitive and responsive to these topics and also support research in this field since the trend for these devices to gather more and more personal data is only to increase.

7.2 Software

To help enthusiasts get started in this field we share the iOS application as well as the Jupyter Notebook to create the graphics and statistical metrics on GitHub. If any help is needed or questions arise related to these applications we suggest to create an issue on GitHub describing your problem.

iOS Application: https://github.com/MatthiasBart/PtPPrototype.git

Jupyter Notebook: https://github.com/MatthiasBart/evaluation_notebook.git

Glossary

5G i, ii, 1, 7

API – Application Programming Interface 15, 20, 21

AWDL – Apple Wireless Direct Link i, ii, 2, 3, 9, 10, 12, 13, 20, 21, 33, 36, 37, 41

BLE – Bluetooth Low Energy 2, 6, 12, 13

BSD – Berkley Software Distribution 9

CAA – Congestive-Avoidance Algorithms 9

CDMA – Code Division Multiple Access 10

CSV – Comma Separated Values 29

D2D – Device-to-Device iii, 10, 12

DoS – Denial of Service 12, 13

gNB – Next Generation Node B 7, 12

GSM – Global System for Mobile Communications 10

GUI – Graphical User Interface 17, 18

HO – Handoff 1, 13

IAB – Internet Architecture Board 4

ICANN – Internet Corporation for Assigned Names and Numbers 4

IDE – Integrated Development Environment 17

IEEE – Institute of Electrical and Electronics Engineers v, 5, 6

IETF – Internet Engineering Task Force 4, 8

iOS i, ii, iii, 1, 2, 3, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 20, 27, 33, 38, 41

IoT – Internet of Things 6

IP – Internet Protocol 8, 9, 15, 26, 36

iPadOS 7

IPS – Internet Protocol Suite vi, 2, 8, 9

ISP – Internet Service Provider 1

LTE – Long Term Evolution 1, 7, 10, 12

MAC – Media Access Control 12

macOS 20

MANET – **Mobile Ad Hoc Networks** iii, 10, 11

mDNS – **multicast Domain Name System** 8, 15, 21, 23

MitM – **Man in the Middle** 12, 13

MP3 10

MVVM – **Model View View Model** 17, 18

NFC – **Near Field Communication** 7, 10

NIST – **National Institute of Standards and Technology** 4

NR – **New Radio** 10, 12

OF – **Offline Finding** 13

OWL – **Open Wireless Link** 10, 12, 13, 14

PC – **Personal Computer** 13

PoC – **Proof of Concept** 13

ProSe – **Proximity Services** 12

PSI – **Private Set Intersection** 13

P2P – **Peer to Peer** i, ii, 1, 2, 3, 6, 10, 11, 15, 16, 20, 33, 36

PWS – **Password Sharing** 1, 13

QUIC i, ii, iv, 2, 9, 21, 22, 24, 26, 28, 29, 36, 37, 41

RFC – **Request For Comment** 4

RF-EMF – **Radio Frequency Electro Magnetic Field** 2, 27, 35, 41

RTT – **Round Trip Time** i, ii, iv, 2, 26, 28, 29, 33, 34, 35, 36, 39, 41

SDK – **Software Development Kit** 2

SEEMOO – **Secure Mobile Networking Lab** 12, 14

SL – **Sidelink** iv, 1, 7, 10, 12

SMS – **Short Message Service** 10

Swift 17

TCP – **Transport Control Protocol** 2, 8, 9, 20, 21, 26, 28, 29, 36, 37

TLS – **Transport Layer Security** 9, 13, 20, 22, 23

tvOS 20

UC – **Universal Clipboard** 1, 13

UDP – **User Datagram Protocol** i, ii, iv, 2, 9, 20, 21, 24, 26, 28, 29, 36, 37, 38

UE – **User Equipment** 7, 12

URL – **Universal Resource Locator** 17

UUID – **Universal Unique Identifier** 13

UVA – **Unmanned Aerial Vehicles** 7, 12

***UWB* – Ultra Wide Band** 7, 8, 20

***V2X* – Vehicle To Everything** 12

Wi-Fi i, ii, v, 1, 2, 5, 6, 9, 10, 11, 12, 13, 20

***WWDC* – World Wide Developer Conference** 20

Xcode 17

Bibliography

Android Developers (2025) *Ultra-wideband (UWB) communication.* [Online]. 10 February 2025. https://developer.android.com/develop/connectivity/uwb [Accessed: 29 April 2025].

AO Kaspersky Lab (2025) *Jailbreaking – Definition und Erläuterung.* [Online]. 18 March 2025. https://www.kaspersky.de/resource-center/definitions/what-is-jailbreaking [Accessed: 29 April 2025].

Apple Inc. (2024b) *Apple device support for private 5G and LTE networks.* [Online]. 25 September 2024. https://support.apple.com/en-gb/guide/deployment/depac6747317/web [Accessed: 29 April 2025].

Apple Inc. (2013) *Bonjour Concepts.* [Online]. 23 April 2013. https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Articles/about.html#//apple_ref/doc/uid/TP40002458-TPXREF108 [Accessed: 29 April 2025].

Apple Inc. (2025a) *Connect to a satellite with your iPhone.* [Online]. 7 April 2025. https://support.apple.com/en-us/105097 [Accessed: 29 April 2025].

Apple Inc. (2024a) *Continuity features and requirements for Apple devices.* [Online]. 6 November 2024. https://support.apple.com/en-us/108046 [Accessed: 29 April 2025].

Apple Inc. (2021) *Explore Nearby Interaction with third-party accessories.* [Online]. 2021. https://developer.apple.com/videos/play/wwdc2021/10165/?time=1249 [Accessed: 30 April 2025].

Apple Inc. (2025c) *Extending advanced direction finding and ranging.* [Online]. 2025. https://developer.apple.com/documentation/nearbyinteraction/extending-advanced-direction-finding-and-ranging [Accessed: 29 April 2025].

Apple Inc. (2023a) *Peer-to-peer networking.* [Online]. 19 September 2023. https://developer.apple.com/documentation/technotes/tn3151-choosing-the-right-networking-api#Peer-to-peer-networking [Accessed: 30 April 2025].

Apple Inc. (2023b) *TN3151: Choosing the right networking API.* [Online]. 19 September 2023. https://developer.apple.com/documentation/technotes/tn3151-choosing-the-right-networking-api [Accessed: 29 April 2025].

Apple Inc. (2025b) *Unauthorized modification of iOS.* [Online]. 2025. https://support.apple.com/en-gb/guide/iphone/iph9385bb26a/ios [Accessed: 29 April 2025].

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]. 22 June 2009 Kraków Poland, ACM. pp. 291–304. DOI:10.1145/1555816.1555846 [Accessed: 7 December 2024].

Barnes, L., Maheu, W. & Kuzin, J. (2023) *How 5G sidelink benefits public safety and critical communications*. [Online]. 4 April 2023. https://www.qualcomm.com/news/onq/2023/04/how-5g-sidelink-benefits-public-safety-and-critical-communications [Accessed: 29 April 2025].

Beer, I. (2020) *An iOS zero-click radio proximity exploit odyssey*. [Online]. 12 January 2020. https://googleprojectzero.blogspot.com/2020/12/an-ios-zero-click-radio-proximity.html [Accessed: 26 March 2025].

Camps-Mur, D., Garcia-Saavedra, A. & Serrano, P. (2013) Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE Wireless Communications*. [Online] 20 (3), 96–104. DOI:10.1109/MWC.2013.6549288 [Accessed: 29 April 2025].

Cheshire, S. D. (2018) *Proximity Wi-Fi*. [Online]. https://patents.google.com/patent/US20180083858A1/en [Accessed: 29 April 2025].

Cisco Systems, Inc. (2025) *What is WiFi?*. [Online]. 2 April 2025. https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html [Accessed: 29 April 2025].

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 [Accessed: 7 December 2024].

Cybersecurity and Infrastructure Security Agency (2021) *Understanding Bluetooth Technology*. [Online]. 1 February 2021. https://www.cisa.gov/news-events/news/understanding-bluetooth-technology [Accessed: 29 April 2025].

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]. 18 August 2013 Saint Petersburg Russia, ACM. pp. 687–690. DOI:10.1145/2491411.2494573 [Accessed: 7 December 2024].

European Space Agency (2017) *About space debris*. [Online]. January 2017. https://www.esa.int/Space_Safety/Space_Debris/About_space_debris [Accessed: 29 April 2025].

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]. 14 May 2024 Herndon, VA, USA, IEEE. pp. 1–6. DOI:10.1109/WFPST58552.2024.00043 [Accessed: 26 March 2025].

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]. 30 October 2023 Boston, MA, USA, IEEE. pp. 807–814. DOI:10.1109/MILCOM58377.2023.10356291 [Accessed: 26 March 2025].

Heinrich, A., Hollick, M., Schneider, T., Stute, M., et al. (2021) PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop. In: *30th USENIX Security Symposium (USENIX Security 21).* [Online]. August 2021 USENIX Association. pp. 3577–3594. https://www.usenix.org/conference/usenixsecurity21/presentation/heinrich.

Heinrich, A., Stute, M., Kornhuber, T. & Hollick, M. (2021) Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System. *Proceedings on Privacy Enhancing Technologies.* [Online] 2021 (3), 227–245. DOI:10.2478/popets-2021-0045 [Accessed: 29 April 2025].

IEEE Communication Society (2023) *5G Sidelink Graphic.* [Online]. February 2023. https://www.comsoc.org/sites/default/files/styles/768wide/public/images/2023-2023-02/ctn-feb-2023-figure5.png?itok=GVi73s6A [Accessed: 30 April 2025].

Intel Corporation (2022) *What Is Bluetooth® Technology?.* [Online]. 2022. https://www.intel.com/content/www/us/en/products/docs/wireless/what-is-bluetooth.html [Accessed: 29 April 2025].

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 [Accessed: 7 December 2024].

Loizeau, N., Zahner, M., Schindler, J., Stephan, C., et al. (2023) Comparison of ambient radiofrequency electromagnetic field (RF-EMF) levels in outdoor areas and public transport in Switzerland in 2014 and 2021. *Environmental Research.* [Online] 237116921. DOI:10.1016/j.envres.2023.116921 [Accessed: 26 March 2025].

LoRa Alliance (2021) *LoRaWAN Graphic.* [Online]. 2021. https://lora-alliance.org/wp-content/uploads/2021/10/LA-TC-Stack-01-1536x708.png [Accessed: 30 April 2025].

LoRa Alliance (2024) *What is LoRaWAN?.* [Online]. 17 June 2024. https://lora-alliance.org/about-lorawan/ [Accessed: 29 April 2025].

Nagle, J. (1984) *Congestion Control in IP/TCP Internetworks.* [Online]. (RFC896) p.RFC896. DOI:10.17487/rfc0896 [Accessed: 29 April 2025].

Network Working Group (1989) *Requirements for Internet Hosts – Communication Layers.* [Online]. October 1989. https://datatracker.ietf.org/doc/html/rfc1122 [Accessed: 29 April 2025].

Newport, C. (2017) Gossip in a Smartphone Peer-to-Peer Network. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing.* [Online]. 25 July 2017 Washington DC USA, ACM. pp. 43–52. DOI:10.1145/3087801.3087813 [Accessed: 7 December 2024].

NIST (2022) *Adhoc Network.* [Online]. 19 January 2022. https://csrc.nist.gov/glossary/term/ad_hoc_network [Accessed: 23 April 2025].

NIST (2018) *Infrastructure Network.* [Online]. 19 October 2018. https://csrc.nist.rip/glossary/term/infrastructure_network [Accessed: 23 April 2025].

Pauly, T., Kinnear, E. & Schinazi, D. (2022) *An Unreliable Datagram Extension to QUIC*. [Online]. March 2022. DOI:10.17487/RFC9221.

Qualcomm Technologies, Inc. (2014) *LTE Direct Always-on Device-to- Device Proximal Discovery*. [Online]. 2014. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/lte_direct_always-on_device-to-device_proximal_discovery.pdf [Accessed: 26 March 2025].

Quinn "The Eskimo!" (2015) *iOS and Wi-Fi Direct*. [Online]. July 2015. https://developer.apple.com/forums/thread/12885 [Accessed: 29 April 2025].

Quinn "The Eskimo!" (2017) *iOS disabled peer-to-peer over Bluetooth*. [Online]. September 2017. https://developer.apple.com/forums/thread/38476?answerId=263873022#263873022 [Accessed: 29 April 2025].

Quinn "The Eskimo!" (2024) *Network API Lowlevel*. [Online]. June 2024. https://developer.apple.com/forums/thread/757385?answerId=791547022#791547022 [Accessed: 29 April 2025].

Quinn "The Eskimo!" (2023) *Use iphone antenna without sim card - Answer*. [Online]. August 2023. https://developer.apple.com/forums/thread/97688?answerId=762404022#762404022 [Accessed: 26 March 2025].

Sherif, A. (2025b) *Apple worldwide shipments of smartphones from 2010 to 2024, by quarter*. [Online]. 14 January 2025. https://www.statista.com/statistics/299153/apple-smartphone-shipments-worldwide/ [Accessed: 30 April 2025].

Sherif, A. (2025a) *Market share of mobile operating systems worldwide from 2009 to 2025, by quarter*. [Online]. 11 March 2025. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [Accessed: 12 March 2025].

Shirey (2007) *Internet Security Glossary, Version 2*. [Online]. August 2007. https://www.rfc-editor.org/rfc/rfc4949.txt [Accessed: 26 April 2025].

Starlink (2025) *Satellite Technology*. [Online]. 2025. https://www.starlink.com/gb/technology [Accessed: 29 April 2025].

Statista Research Department (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 [Accessed: 8 December 2024].

Stute, M., Heinrich, A., Lorenz, J. & Hollick, M. (2021b) Disrupting Continuity of Apple's Wireless Ecosystem Security: New Tracking, DoS, and MitM Attacks on iOS and macOS Through Bluetooth Low Energy, AWDL, and Wi-Fi. In: *30th USENIX Security Symposium (USENIX Security 21)*. [Online]. August 2021 USENIX Association. pp. 3917–3934. https://www.usenix.org/conference/usenixsecurity21/presentation/stute.

Stute, M., Heinrich, A., Lorenz, J. & Hollick, M. (2021a) *USENIX Security '21 - Disrupting Continuity of Apple's Wireless Ecosystem Security: New Tracking*. [Online]. 4 September 2021. https://youtu.be/6dUqEA5MVBQ?si=T_iZ7PDCC92u14Qw&t=179 [Accessed: 2 May 2025].

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]. 15 October 2018 New Delhi India, ACM. pp. 529–543. DOI:10.1145/3241539.3241566 [Accessed: 17 February 2025].

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

Stute, M., Narain, S., Mariotto, A., Heinrich, A., et al. (2019) A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. In: *28th USENIX Security Symposium (USENIX Security 19)*. [Online]. August 2019 Santa Clara, CA, USENIX Association. pp. 37–54. https://www.usenix.org/conference/usenixsecurity19/presentation/stute.

Vijitha, Weerackody, Kent, Benson & Sumit, Roy (2023) *Who Needs Basestations When We Have Sidelinks?*. [Online]. 24 February 2023. https://www.comsoc.org/publications/ctn/who-needs-basestations-when-we-have-sidelinks [Accessed: 26 March 2025].

Wi-Fi Alliance (2023) *Discover WiFi*. [Online]. 2023. https://www.wi-fi.org/discover-wi-fi [Accessed: 29 April 2025].