TrustChain for Smartphones

Measuring reconnection latency when the network is interrupted

Alexandra Mihaela Nicola
Supervisors: Johan Pouwelse, Bulat Nasrulin
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Alexandra Mihaela Nicola
Final project course: CSE3000 Research Project
Thesis committee: Johan Pouwelse, Bulat Nasrulin, Koen Langendoen

An electronic version of this thesis is available at
https://github.com/mbakker520/smartphone-trustchain/tree/reconnectionLatency.

Abstract

TrustChain is a scalable, lightweight blockchain architecture that avoids global consensus by maintaining a personal chain of co-signed interactions for each peer, forming a directed acyclic graph. While its structure makes it promising for mobile and resource-constrained environments, its behavior under real-world conditions, such as the time it takes to restore connectivity, remains underexplored. This paper presents an implementation of TrustChain in Rust for Android mobile devices and evaluates its robustness under network disconnections across two protocols: UDP and Iroh over QUIC. Robustness is defined as the time elapsed between network connectivity restoration and the successful exchange of the first valid TrustChain message between peers. Experiments involved controlled Wi-Fi interruptions, measuring the reconnection time for each protocol. The findings reveal trade-offs between protocol simplicity and recovery performance. UDP demonstrated consistent low-latency reconnection times between 4 and 6 seconds, averaging ∼5 seconds, due to its stateless nature and lack of connection recovery overhead. In contrast, Iroh reconnection times ranged from 4.5 to 11.5 seconds, with most values between 6 and 8 seconds, because of QUIC's timeout strategy, DNS-based peer discovery, and relay reconnection overhead. The results provide insights for deploying decentralized systems in mobile contexts and highlight open challenges in peer reconnection for lightweight blockchain protocols. This study also proposes improvements and directions for future work, including multi-peer evaluation, measuring different Wi-Fi congestion levels, or experimentation with other network protocol stacks.

1 Introduction

Blockchain technology has emerged as a foundational infrastructure for decentralized applications, enabling trustless, tamper-proof transactions in systems ranging from cryptocurrencies to digital identity and smart contracts [1]. At its core, a blockchain is a distributed, append-only ledger that maintains a consistent and verifiable history of interactions across a network of peers. Early designs like Bitcoin introduced the concept of consensus through proof-of-work, allowing a permissionless group of actors to reach agreement on a shared ledger without central control. However, this approach imposes significant performance and scalability limitations. In real-time applications, global consensus systems restrict responsiveness and throughput due to their high energy costs, computing demands, and delays.

These limitations are problematic especially for mobile devices, which require efficiency, but remain poorly supported by traditional blockchains. The need for lightweight, scalable, and interaction-centric blockchain systems has led to the exploration of alternative architectures that deviate from monolithic global ledgers and consensus-based transaction validation.

One interesting substitute is TrustChain. Designed as a lightweight, user-centric blockchain protocol, TrustChain replaces the global ledger with a personal chain of interactions. Each participant maintains their own history in the form of a Directed Acyclic Graph (DAG), where each block represents a co-signed interaction with a peer. [2] Because of its scalability, avoidance of global synchronization, and reduced computing requirements, TrustChain is naturally suited for mobile deployments.

However, there are significant challenges when using TrustChain in the real world, particularly on smartphones or other mobile devices. One critical aspect is **robustness**: the

system's ability to recover quickly after network interruptions. These situations are common in mobile scenarios due to switching between Wi-Fi and mobile data, signal loss, or energy saving modes. A robust system must restore peer connectivity and resume operation with minimal delay after such disruptions.

This thesis addresses a critical but underexplored question:

How can a smartphone-optimized implementation of TrustChain be built from scratch to support real time peer-to-peer communication? How robust is TrustChain, in terms of time to re-establish connectivity, when running over different network protocols?

As part of a broader team project implementing a common TrustChain base over multiple networking protocols, UDP and Iroh [3], this thesis contributes to a focused investigation of reconnection performance. Specifically, it evaluates how long it takes TrustChain nodes to resume interaction after simulated network failures, across each protocol.

The core contributions of this thesis are:

- TrustChain's core features implementation
- Implementation of different network protocols: UDP and Iroh over QUIC.
- An experimental framework that measures reconnection time under identical failure patterns across all protocols
- A comparison of UDP and Iroh in terms of robustness, outlining the trade-offs between the protocols for constrained deployments.

The rest of this paper is structured as follows: Section 2 reviews related work on blockchain robustness and protocol behavior in disrupted networks. Section 3 provides background on TrustChain and the selected protocols. Section 4 outlines the TrustChain and network protocols implementations, as well as the metrics for measuring robustness. Section 5 details the experimental setup, along with the results. Section 6 presents the responsible research, followed by a discussion in Section 7. Section 8 concludes with recommendations for future work.

2 Background and Related work

Blockchain technologies have evolved significantly since their inception, with increasing interest in adapting them to mobile and constrained environments. However, most of the literature still focuses on consensus mechanisms, scalability, or energy efficiency, with limited attention paid to system robustness, especially in terms of reconnection time after network interruptions. This chapter reviews existing literature relevant to blockchain deployment on constrained devices, TrustChain's architecture, and network protocol performance, identifying a critical research gap.

2.1 Blockchain on constrained devices

Several studies have explored the feasibility of deploying blockchain systems on smartphones and low-power devices. The Trinity protocol proposed by Zhidanov et al. [4] introduces

a hybrid model combining Proof-of-Work, Proof-of-Stake, and Proof-of-Activity to enable blockchain participation on mobile hardware. While the work includes a simulation and early-stage deployment across multiple countries, it does not quantitatively evaluate how the system behaves under disconnection or failure conditions. This raises concerns about the robustness in real world deployments, where a mobile system should support reliable peer-to-peer communication.

Other studies address blockchain fault tolerance more broadly. For example, wChain [5] is designed for multihop wireless networks and includes mechanisms for node recovery in dynamic topologies. Other research proposes hybrid consensus models incorporating machine learning to predict and mitigate faults [6]. However, these works are not focused on mobile device deployments or reconnection latency metrics.

Similarly, Kim et al. [7] propose a Proof-of-Phone consensus mechanism leveraging trusted smartphone hardware to reduce operational costs. Their architecture introduces high entry costs through dedicated hardware modules, to limit excessive competition. This enables a secure, low cost blockchain operation on mobile phones. Although the authors discuss resilience in economic and cryptographic terms, they do not measure robustness in the face of network churn or temporary loss of connectivity. This omission raises doubts about its practical usefulness in mobile contexts, where connectivity is frequently interrupted.

2.2 TrustChain

TrustChain is a decentralized ledger architecture designed for high scalability and Sybil resistance without relying on global consensus. Instead of a single shared blockchain, each participant maintains a personal chain of co-signed interactions, forming a DAG when all individual chains are considered collectively. This structure is illustrated in Figure 1. Every block records a transaction between two peers and is cryptographically signed by both, linking backward to each participant's previous block.

This structure enables parallel growth of chains and avoids the bottlenecks of consensus based validation. It also makes TrustChain well suited to mobile and resource constrained environments, where global synchronization is costly or infeasible. Blocks are disseminated using a gossip protocol, and nodes operate with partial knowledge of the overall system state. This decentralized, peer driven structure supports a degree of robustness under churn and intermittent connectivity, as each node independently manages its history.

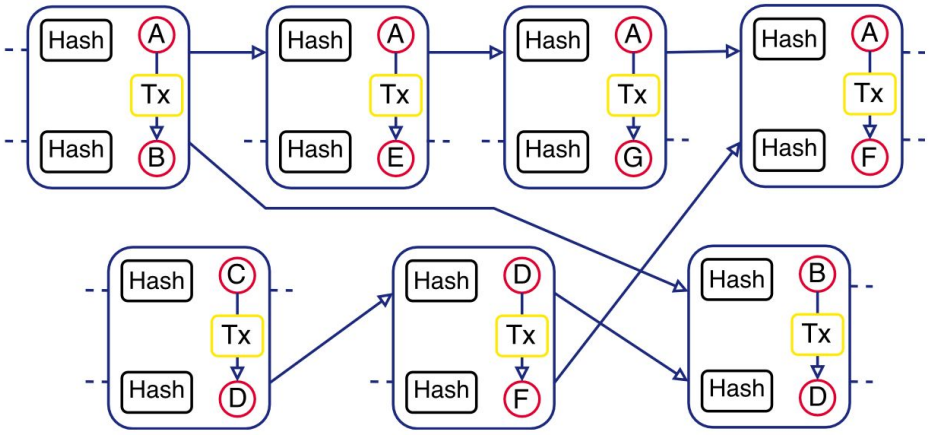

Figure 1: The tamper-proof TrustChain data structure to record transactions. Reproduced from Pouwelse et al. [2] with permission.

The TrustChain architecture was introduced as a scalable and Sybil-resistant blockchain system tailored for decentralized, permissionless environments without relying on global consensus [2]. Its design, further detailed in an IETF draft [8], relies on individual chains of co-signed blocks forming a Directed Acyclic Graph, which supports asynchronous operation and resilience to churn. While TrustChain's structure suggests robustness against temporary disconnections, the system has not been evaluated empirically for recovery time after network loss, particularly across different transport protocols.

2.3 Network protocol evaluations

Reliable communication is fundamental to the operation of blockchain protocols, particularly in decentralized and mobile environments where connectivity can be intermittent. Various network protocols offer different trade-offs between performance, reliability, and resource usage.

While many blockchain deployments rely on TCP-based stacks for guaranteed delivery, alternative protocols such as UDP, TFTP [9], and libp2p-based stacks like Iroh [3] offer lower overhead or improved adaptability in peer-to-peer contexts. These protocols differ significantly in how they handle connection state, packet loss, and recovery mechanisms, which can directly impact robustness in real-world deployments.

QUIC [10] is a modern transport protocol designed to provide fast, reliable, and secure communication, built on top of UDP. Unlike traditional TCP, it combines connection establishment, encryption, and stream multiplexing into a single protocol layer, which significantly reduces handshake delays. Its features such as early data transmission, active loss recovery and connection migration can be promising in the context of mobile TrustChain, where temporary disconnections are common.

Prior work in peer-to-peer networking, such as Kotlin-IPv8 and related overlays has observed limitations in data throughput and reliability, especially for UDP-based communication under high load [11]. These results highlight the need for alternative lightweight protocols that provide a certain level of reliability with minimal overhead. TFTP [9] introduces a lightweight acknowledgment system over UDP, while Iroh builds on modern transport technologies like QUIC and offers connection abstraction and peer discovery. However, existing studies tend to focus on throughput, latency, or energy efficiency, and do not measure reconnection behavior after network disruptions. To our knowledge, no prior work has systematically compared these protocols in the context of blockchain or DAG-based systems like TrustChain, where re-establishing peer communication quickly after disconnection is essential for usability.

Another protocol that was considered for implementation was the Micro Transport Protocol [12], a UDP based protocol primarily used in BitTorrent peer-to-peer file sharing. Its main characteristic is its ability to manage network congestion effectively, ensuring that large data transfers in the background do not significantly slow down other internet activities. However, despite its strengths in efficient background data transfer, the Micro Transport Protocol was ultimately not chosen for implementation. The primary reason lies in TrustChain's need for rapid and consistent reconnection after network interruptions, especially in mobile environments. On top of this, given the project's time constraints, it was decided to focus on protocols that offered more direct and simpler paths to achieving the core robustness requirements, such as those that allowed for more straightforward and simpler implementation.

These protocols range from minimalistic to highly abstracted communication layers, offering different connection managements, reliability, and reconnection logic. Understanding how each performs in a TrustChain based system under network churn provides insight into practical blockchain deployment trade-offs.

2.4 Research Gap

In summary, while prior work has investigated blockchain scalability, security, and energy efficiency for smartphones and IoT devices, the specific problem of reconnection delay after network interruptions remains largely unaddressed. Existing literature does not offer:

- Quantitative measurements of time to reconnect after disconnection;
- Comparisons between network protocols for TrustChain message recovery;
- Evaluations of how these delays impact TrustChain's functionality or usability in mobile contexts.

To bridge this gap, this thesis defines a performance metric for reconnection time, implementing a shared TrustChain base over two transport protocols, UDP and Iroh, and empirically comparing their behavior in controlled network disruption scenarios.

3 TrustChain and Network Protocols Implementation

This chapter outlines the implementation of the TrustChain core and its integration with two network protocols: UDP and Iroh. It also defines the robustness metric used to eval-

uate reconnection behavior and describes the system design enabling controlled protocol comparisons.

3.1 TrustChain Implementation Scope

As part of the team's effort to implement a shared TrustChain base across multiple network protocols, the individual contribution centers on evaluating robustness under network disruptions. The implemented system is a partial realization of TrustChain's core functionality, written in Rust and integrated into an Android environment via JNI. The subset of the TrustChain's features and architecture that were implemented include:

- Block creation and proposal logic: A proposal is a one-sided block created by the sender, containing the payload and metadata such as index, previous hash, and signatures.
- Block completion and finalization: The receiver validates the proposal, signs it, and completes the block. Both sides store the block in their respective chains.
- Local storage: Each device maintains a per-peer chain of interactions in memory.
- Signature validation: All finalized blocks undergo dual signature verification using either ed25519-dalek or Iroh's signing key, depending on the networking mode.
- The public key and signing mechanisms adapt based on whether the node is running in UDP or Iroh mode.
- Duplicate message filtering: A deduplication cache ensures each finalized block is processed only once.

The implementation does not include full peer discovery, message broadcasting, or block gossiping. As such, all evaluation focuses on pairwise reconnection and message exchange.

3.2 Robustness

In the context of distributed systems, robustness refers to the system's ability to maintain correct operation under disruptions such as network partitioning, node failures, or reconnection. For blockchain systems, this includes aspects like data integrity, availability, and especially recovery from communication failures. Robustness is especially critical in mobile deployments, where network conditions may change frequently. For TrustChain, reconnection robustness means how quickly and reliably two nodes can resume valid block exchange after losing and regaining network contact.

To evaluate protocol robustness in the context of TrustChain, a specific metric was defined:

Reconnection Time: The time elapsed between the restoration of network connectivity and the successful exchange of the first valid TrustChain message between peers.

This metric reflects how well the underlying protocol can detect when connectivity is restored and resume the TrustChain workflow. The comparison between UDP and Iroh highlights the impact of the communication protocol on recovery speed, which is critical for mobile

environments.

Alternative robustness metrics were considered but ultimately not selected for this study. For instance, the percentage of messages dropped during disconnection could provide insights into protocol-level reliability. However, this was excluded because UDP is stateless by default and drops all messages sent after disconnection and the default implementation of Iroh over QUIC tears down the connection after exponential backoff and re-establishes a new one. Furthermore, another viable strategy would have been to measure energy consumption during failure and recovery events, which is also a form of robustness. However, this was outside the scope of the current work.

3.3 Network Protocol Implementations

UDP. The UDP implementation uses raw datagram sockets for direct peer-to-peer communication. The networking logic is handled in Kotlin using the standard *DatagramSocket API*. On the Rust side, blocks are created, signed, validated, and messages are serialized as JSON strings and transmitted over UDP, with no built-in reliability mechanisms. Each device locally maintains a per-peer chain and uses simple deduplication strategies and index tracking to manage incoming messages. The protocol remains fully stateless, and all session control or retransmission logic is absent, simplifying the stack, but limiting its resilience to disruptions. UDP was chosen as a baseline for minimal overhead communication in local or controlled environments.

Iroh. Iroh is a modern peer-to-peer protocol stack based on libp2p and QUIC. It supports decentralized peer discovery, stream multiplexing, NAT traversal, and secure communication via public-key-based NodeIds. In this implementation, each peer advertises itself using *Pkarr*, a DNS-based relay system provided by Iroh, and discovers others via relay lookups. Once a connection is established, data is exchanged over multiplexed QUIC streams. The Rust core handles TrustChain block logic independently of the transport layer, but dynamically adjusts its signing and identity logic depending on the networking mode, using Iroh's keypair and NodeId for identification.

A flow of actions happening inside Iroh when exchanging messages between peers and the network drops is:

- Initially messages are being sent normally, using QUIC connection over Iroh. The connection may use either a direct UDP path or a relay tunnel.
- The network drops for one peer, losing the possibility of sending or receiving any messages. The other peer continues to send messages, but QUIC does not see any acknowledgments. It starts exponential probe timeouts backoff (first probe sent after 1s, then 2s, then 4s).
- After multiple failed probes, QUIC declares the connection lost and tears down the old connection. Then, Iroh notices the disconnection at the Router layer.
- The network is back up and the Iroh's Endpoint auto-reconnects to home relay. The relay now knows the node is online again
- A QUIC handshake completes over relay and new logical connection established between peers. Iroh triggers hole punching in the background

- Iroh attempts NAT traversal using relay/STUN-style coordination. If it is successful, the direct path replaces the relay and congestion control is reset to avoid slow start. If not, the relay continues being the path
- The first message sent after the reconnection successfully arrives.

TFTP. The Trivial File Transport Protocol was initially selected for implementation because of its simplicity and lightweight overhead, making it a potential middle ground between raw UDP and Iroh over QUIC. The implementation followed a file-based approach, where TrustChain messages were transferred as JSON files, with accompanying markers to indicate transfer completion. Although this setup aimed to provide basic reliability in a local environment, the integration proved to be unstable and difficult to manage. Due to frequent issues with file handling, transfer timing, and message loss, the TFTP based system could not be reliably evaluated and was excluded from the final robustness experiments.

It also needs to be considered what happens to the blocks transmitted when the connection is interrupted and analyze the failure detection and recovery strategies embedded in each protocol. UDP, as a stateless protocol, offers no mechanism to detect peer loss or automatically resume communication. QUIC, by default does not send keep-alive packets unless configured, to avoid unnecessary traffic on healthy connections. [13] In Quinn's *TransportConfig*, the *keep-alive interval* is $None$ by default.

4 Experimental Setup and Results

To evaluate the robustness of TrustChain under network disruptions, a series of controlled experiments were conducted, which compared the two network protocols: UDP and Iroh over QUIC. Each experiment measures the system's ability to recover from temporary disconnections by recording the reconnection time. This chapter presents the software and hardware setup for the experiments along with the results.

4.1 Setup

The system is deployed on Android devices with protocol mode selected at runtime. The same TrustChain core handles block generation and validation across all modes. The evaluation setup includes: one peer initiates a proposal, the network connection is dropped and later restored. The peer resumes communication and finalizes the block. Timestamps are logged at: disconnection, network restoration, first successful proposal or finalized block received. These are used to compute the reconnection delay.

The experiments were conducted using two Android devices to represent both physical and virtual environments:

- Device 1: Samsung Galaxy S20+, physical, running Android 13
- Device 2: Google Pixel 7 emulator, running Android API level 36 via Android Studio

The use of both a real device and an emulator allows testing in different runtime conditions and network interface behaviors. Because of hardware limitations, the available computer could not support two emulators at the same time and this setup was the only option for running the experiments. For consistency, the battery level of the physical phone was kept

above 80%. Both devices were connected to the same local Wi-Fi network, with mobile data disabled on the physical device to avoid fallback routing. The TrustChain core was compiled with identical settings for all test cases, and the networking mode was selected at runtime.

To evaluate reconnection behavior, the connection drop was simulated by disabling and re-enabling the device's Wi-Fi using ADB shell commands (*svc wifi disable/enable*). This was automated through a benchmarking script that ran for 150 iterations, each disabling Wi-Fi for 30 seconds, then re-enabling it and waiting additional 10 seconds to allow for message delivery. During each run, a timestamp was recorded at the moment of reconnection (T_{up}), and another when the first message was received (T_{recv}). These were used to compute reconnection time as $T_{\text{recv}} - T_{\text{up}}$.

To generate traffic, the app sent messages automatically at fixed 0.5-second intervals using an auto-send feature. This interval was chosen to ensure prompt message delivery right after reconnection, while avoiding excessive load of the application. A higher interval might have resulted in a slightly higher reconnection time, due to how it is computed. The chosen 0.5 interval provided a balance between responsiveness and system stability during reconnection testing. Logs were collected from Logcat and parsed using a custom Python script, which extracted timestamps with millisecond precision and plotted results for analysis.

4.2 Results and Analysis

The results of the experiment using **UDP** as the network communication protocol can be found in the histogram from Figure 2. The reconnection time can take anywhere between **4 and 6 seconds**, the average being approximately 5 seconds.

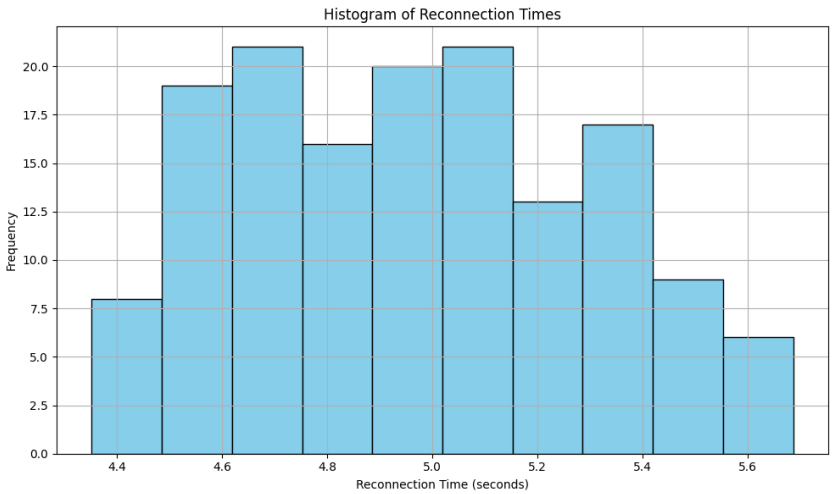

Figure 2: Histogram showing the frequency of reconnection times (in seconds) observed after Wi-Fi recovery using the **UDP** protocol across multiple test runs.

Given that the raw UDP protocol implemented is a stateless protocol with no built-in con-

nection or session management, the reconnection times are low. Once the Wi-Fi connection is restored and the OS has completed IP configuration and routing updates, any messages sent by the other peer are immediately deliverable. Since messages are transmitted every 0.5 seconds, the first packet after recovery is received shortly after the network is available. Therefore, the variable but small delays observed in the histogram is attributed to the underlying OS-level latency, such as Wi-Fi initialization, IP address assignment, rather than any action in the UDP layer. No reconnection or detection mechanism is needed, which can explain the consistent and low reconnection times in the experiments.

The results of the experiment using **Iroh** as the network communication protocol can be found in the histogram from Figure 3. The reconnection time can take anywhere between 4.5 and 11.5 seconds, the most frequent being between 6 and 8 seconds. Compared to the results from UDP, Iroh has a higher reconnection time, with more fluctuations.

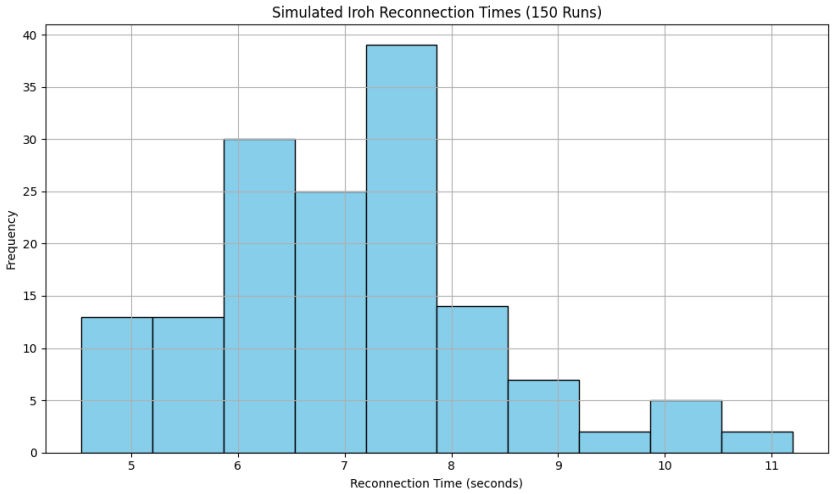

Figure 3: Histogram showing the frequency of reconnection times (in seconds) observed after Wi-Fi recovery using the **Iroh over QUIC** protocol across multiple test runs.

Several components in Iroh's implementation can introduce latency and fluctuations when recovering a connection. Studying the Crate Iroh documentation [13], the following possible causes of the reconnection delay were identified, along with possible mitigations:

- **QUIC Timeout Detection**. When the network is lost, QUIC does not immediately assume the connection is dead. Instead, it relies on a series of exponentially delayed probe timeouts (PTOs) to detect the absence of acknowledgments. These delays can result in connection loss detection taking a few seconds. To reduce this latency, the QUIC idle timeout can be shortened and periodic keep-alive packets (e.g., every 2-3 seconds) can be enabled to force earlier detection. Moreover, tuning the initial round-trip time (RTT) estimation to a lower value can help reduce the duration of each PTO interval, which will accelerate the disconnection recognition.

- **Discovery delays**. When the connection is lost, the peers need to find each other again. By default, Iroh uses a DNS-based discovery mechanism (*Discovery n0*) that publishes and resolves NodeId records via a dynamic DNS service. While generally efficient, this process can introduce latency due to DNS query times, caching behavior, or delays in peer re-registration. To reduce this dependency, applications can provide known addresses directly, which allows Iroh to skip the discovery step entirely.
- **Relay Reconnection and Handshake**. When the Wi-Fi drops, the connection to the relay server is also lost and when the network returns, Iroh must reconnect to the relay, introducing unavoidable latency during reconnection. Once a peer regains network access, it must re-establish its connection to a home relay, and the other side may attempt to reconnect via that relay before any direct path is available. The relay based handshake adds at least one full round-trip time (RTT), and often slightly more due to TLS negotiation overhead, adding more delay. To minimize this, applications could maintain a persistent Endpoint instance across network transitions to avoid rebuilding the relay connection from scratch.

5 Responsible Research

In line with TU Delft's standards for scientific integrity and transparency, this chapter outlines the ethical considerations and reproducibility of the work presented.

5.1 Ethical considerations

Evaluating communication protocols for mobile blockchain systems involves trade-offs that can impact users indirectly, particularly in terms of energy consumption and resource use. While no sensitive or personal data was used during testing, energy efficiency remains relevant due to the deployment of TrustChain on resource-constrained devices.

The implementation avoids heavy consensus algorithms and relies on lightweight block signing and message validation. However, different protocols may induce varying levels of retry overhead or idle connection maintenance, which could affect battery life in practical settings. Although energy use was not measured directly in this work, it is an important factor for future research, especially when scaling to multipeer or real time deployments.

Privacy risks are minimal in the current setup. All communication occurred on a private, local Wi-Fi network between two test devices, with synthetic payloads and no user identifiable information. Key material was generated locally and used only for message validation within the controlled test environment.

5.2 Reproducibility

The implementation was designed with reproducibility in mind. Understanding the implementation and design choices is crucial for potentially replicating the experiments' results. Thus, Chapter 3 describes the implementations of TrustChain and network protocols in detail, as well as a comprehensive definition of robustness. The details of setting up the experimental setup can be found in Chapter 4.

Moreover, all experiments were conducted using open-source tools and publicly available libraries. The TrustChain core was implemented in Rust and integrated with an Android application using JNI. All protocol-specific logic was implemented in a modular way, and the networking mode could be selected dynamically at runtime. We also aim to make the implementation of the Trustchain logic, the communication protocols (Iroh, UDP), and the benchmarking framework available as open-source software. This will be useful for other researchers who want to validate or extend our work.

While aiming for reproducibility, there are some challenges that need to be addressed. The devices used for measuring the reconnection latency can influence the results, because of OS background tasks or emulator behavior. Moreover, the command for simulating the network drop (svc wifi disable/enable) may have different behavior depending on the Android version of the smartphone.

By considering these ethical factors and ensuring reproducibility, this work aims to support the responsible development of decentralized mobile communication systems.

6 Discussion, Future Work and Conclusions

This thesis set out to answer the following central research question:

How can a smartphone-optimized implementation of the TrustChain protocol be built from scratch to support real time peer-to-peer communication? How robust is TrustChain, in terms of time to restore connectivity, when running over different network protocols?

To address this, a minimal TrustChain core was implemented in Rust and integrated into an Android environment, supporting two transport protocols: UDP and Iroh. The system was evaluated for reconnection robustness, defined as the time between Wi-Fi recovery and successful message exchange after simulated disconnection. The experiments were conducted using a real device and an emulator, with 150 runs for each protocol.

6.1 Summary of the findings

The experiments conducted show the difference between using UDP and Iroh over QUIC as the network communication protocol in a smartphone-based implementation of TrustChain. There is a measurable difference regarding the reconnection time, UDP achieving a consistent average of 5 seconds, due to its stateless nature and lack of overhead. It is a simple protocol, but comes with the cost of reliability, as UDP does not provide any delivery guarantees or built-in reconnection handling.

The implementation of Iroh over QUIC presents more variable reconnection times, ranging between 4.5 seconds to 11.5 seconds, with the most frequent being between 6 and 8 seconds. Some possible factors for the reconnection latency were identified: **QUIC timeout detection**, which uses delayed probes to detect disconnections, **discovery delays** from DNS-based peer resolution, and **relay reconnection and handshake overhead**, which includes relay server reregistration and TLS negotiation. Playing with these parameters could decrease the reconnection time, this remains a future work.

These results highlight the tradeoffs between simplicity, resilience and complexity. While UDP is well-suited to fast, local deployments, its stateless design makes it unreliable under churn. Iroh, though designed for decentralized robustness, can introduce unexpected delays due to its internal complexity.

6.2 Limitations and Future Work

One limitation of this study is that it focuses on pairwise communication only. Multi-peer interactions and their impact on robustness were not explored, because of hardware limitation: the available computer could not support more than one emulator running and the availability to physical devices was limited. Moreover, another limitation is that the implementation supports only Android devices, with no support for IOS.

Furthermore, the TFTP protocol was a strong candidate for implementation, because it could potentially provide a middle ground between UDP and QUIC. However, the implementation was not successful. This limits the study's comparison to two protocols, even though a third protocol was partly implemented.

This study opens the path to future investigations. Listed based on the impact, future researchers can explore:

- Change QUIC's parameters and evaluate the impact on robustness: shorten idle timeout, introduce keep-alive packets, maintain a persistent Endpoint instance, or skip the discovery step for known addresses
- Support and evaluate multipeer support for measuring reconnection latency
- Evaluate robustness under different Wi-Fi congestion levels and mobile data
- Explore protocol switching strategies, where nodes dynamically choose between UDP or Iroh, depending on the context
- Complete the TrustChain feature set implementation, such as peer discovery, block gossiping, and conflict resolution and re-evaluate reconnection latency

6.3 Conclusions

This study investigated the reconnection latency under two different network protocols, UDP and Iroh over QUIC, in the context of a lightweight blockchain protocol, TrustChain, deployed on Android devices. Specifically, it analyzed how quick peers resume communication after temporary network disruptions, such as Wi-Fi interruption. The goal was to define robustness in the context of mobile devices and evaluate the performance of a built-from-scratch TrustChain mobile application.

The implementation includes core TrustChain features, implemented from scratch, such as block proposal, completion and signature validation, along with different network protocols. UDP offered fast recovery but no reliability or state awareness, while Iroh showed higher and more variable reconnection times. This was attributed to factors such as QUIC's delayed connection timeout, DNS-based peer discovery, and relay re-establishment.

The key findings are: UDP achieves a reconnection time between 4 and 6 seconds, due to its simplicity, while Iroh over QUIC has longer and more variable results, ranging between 4.5 and 11.5 seconds, with an average between 6 and 8 seconds. Although Iroh provides more complex transport features, these introduce reconnection overhead.

In conclusion, UDP offered fast recovery but no reliability, while Iroh, despite its advanced features like QUIC, stream multiplexing, and NAT traversal, showed higher and more variable reconnection times. This was attributed to factors such as QUIC's delayed connection timeout, DNS-based peer discovery, and relay re-establishment.

References

[1] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE 6th International Congress on Big Data (BigData Congress)*, pages 557–564. IEEE, 2017.

[2] Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. 2020. Internet-Draft, Internet Engineering Task Force (IETF).

[3] n0-computer. Iroh: A peer-to-peer networking stack based on libp2p and quic. https://www.iroh.computer/docs, 2025. Accessed: 2025-05-21.

[4] Konstantin Zhidanov, Sergey Bezzateev, Alexandra Afanasyeva, Mikhail Sayfullin, Sergei Vanurin, Yulia Bardinova, and Aleksandr Ometov. Blockchain technology for smartphones and constrained iot devices: A future perspective and implementation. 07 2019.

[5] Minghui Xu, Chunchi Liu, Yifei Zou, Feng Zhao, Jiguo Yu, and Xiuzhen Cheng. wchain: A fast fault-tolerant blockchain protocol for multihop wireless networks. 02 2021.

[6] Venkatesan .K and Syarifah Rahayu. Blockchain security enhancement: an approach towards hybrid consensus algorithms and machine learning techniques. volume 14, 01 2024.

[7] Jae Min Kim, Jae Won Lee, Kyungsoo Lee, and Junho Huh. Proof of phone: A low-cost blockchain platform. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2019.

[8] Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. Internet-Draft, IETF, 2020. Available at https://datatracker.ietf.org/doc/html/draft-pouwelse-trustchain-01.

[9] Karen R. Sollins. The TFTP Protocol (Revision 2). RFC 1350, 1992. Accessed: 2025-05-21.

[10] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

[11] R. Koning. Performance analysis of an offline digital euro prototype. 2023.

[12] Wikipedia contributors. Micro transport protocol — Wikipedia, the free encyclopedia, 2025. [Online; accessed 10-June-2025].

[13] Crate iroh. https://docs.rs/iroh/latest/iroh/.