Trustchain Mobile: A Low-Latency Smartphone Peer-to-Peer Transaction System

Performance analysis and benchmarking

Vlad-George Iftode

Supervisor(s): Johan Pouwelse, Bulat Nasrulin

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: Vlad-George Iftode
Final project course: CSE3000 Research Project
Thesis committee: Johan Pouwelse, Bulat Nasrulin, Koen Langendoen

Abstract

Mobile blockchain transactions require low latency for practical deployment. This paper presents a smartphone-native Trustchain system developed as part of a collaborative research project where five students implement Trustchain from scratch, each optimizing a different performance metric: latency, robustness, storage, throughput, and battery efficiency. This work focuses on latency optimization through a novel modular architecture implemented in Rust/Kotlin that enables runtime protocol switching—the first such capability in mobile blockchain implementations.

The evaluation compares two transport layers: a lightweight UDP implementation and a business-grade P2P solution (Iroh). Testing on Android devices demonstrates **end-to-end round-trip latencies from 11.8 ms under optimal conditions to 240 ms at extreme loads (500 MPS)**, with UDP achieving consistent 11.8ms median latency and Iroh showing 18.2ms for typical payloads. Results show clear trade-offs between protocol complexity and performance, providing guidance for selecting transport mechanisms in different mobile deployment scenarios. All code is released open-source to the Tribler project for reproducibility.

1 Introduction

The demand for secure, peer-to-peer financial transactions on mobile devices is clear, yet achieving the required low latency remains a critical challenge. While blockchain offers robust security [1], traditional implementations are ill-suited for the resource-constrained mobile environment [2; 3]. This paper investigates whether Trustchain is a viable blockchain option in the current landscape by focusing specifically on latency optimization for smartphone-native implementations.

This work represents my contribution to a five-person collaborative research team implementing Trustchain from scratch for smartphones. Throughout this paper, "we" refers to this collaborative team, while the specific focus on latency optimization represents my individual contribution to the broader project.

Our approach is centered on Trustchain [4], a DAG-based ledger that avoids the high overhead of traditional consensus protocols. This work is part of a collaborative research project where five students are implementing Trustchain from scratch for smartphones, each optimizing for a different performance metric: throughput, latency, storage, energy efficiency, and robustness. This paper specifically focuses on **latency optimization**, investigating the trade-offs between different communication protocols and their impact on transaction confirmation latency, while our teammates address the other performance dimensions in parallel studies.

This research was conducted as an open-source contribution to the Tribler project. Our primary contributions are:

- **A Modular Mobile Blockchain Architecture:** We designed and implemented the first Trustchain-compliant system for mobile devices with runtime-switchable protocol layers, serving as a blueprint for future mobile blockchain development.
- **A Comparative Protocol Analysis:** We built and tested two distinct communication protocols—a business-grade P2P solution (Iroh) and a lightweight UDP implementation—providing the first systematic latency analysis for this environment.
- **An Open-Source Benchmarking Framework:** All code, including our performance measurement suite, is open-source, enabling community reproduction and extension of this work.

This paper documents our design, implementation, and performance evaluation, offering practical insights into the challenges and opportunities of mobile-first blockchain systems.

The rest of this paper is organized as follows. Section 2 details our methodology and problem formulation. Section 3 outlines our technical contributions, followed by the experimental setup and results in Section 4. We discuss the broader implications of our findings in Section 6, and conclude in Section 7.

2 Methodology and Problem Formulation

This research follows a constructive methodology: we designed, built, and evaluated a smartphone-native blockchain system to investigate its performance. This section first formally defines our research problem and then details the architecture and methods we used to address it.

2.1 Problem Formulation

Traditional blockchain architectures face significant challenges on mobile devices due to high resource consumption and intermittent connectivity [2; 5]. While DAG-based ledgers like Trustchain [4] offer a more efficient alternative, their performance characteristics in a mobile-first context have not been systematically studied. As part of a collaborative investigation into Trustchain viability, **we design and implement a smartphone-native Trustchain system optimized for minimal transaction confirmation latency while maintaining blockchain integrity under mobile hardware constraints.** This requires careful architectural design and protocol selection to balance performance with reliability requirements, contributing to the broader assessment of Trustchain viability for mobile deployment.

2.2 Architectural Design and Implementation

We implement a complete Trustchain-compliant system for Android devices using a hybrid Rust/Kotlin stack that balances performance with mobile development efficiency.

Modular Coordination Layer At the core of our design is a modular coordination layer, implemented in Rust as the `entry_point/`. This layer abstracts all protocol-specific logic behind a single, unified interface. It uses a `NetworkingMode` enumeration to enable runtime switching between different communication protocols, allowing for both systematic comparison and adaptive deployments.

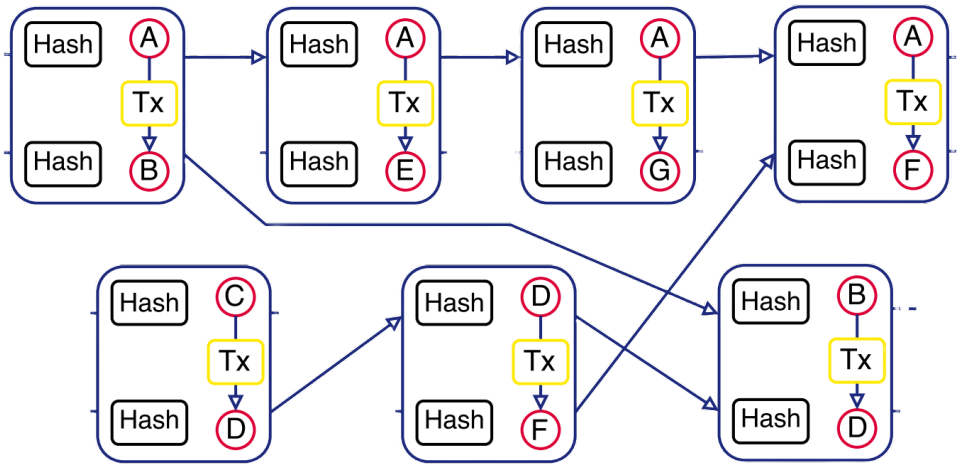

Figure 1: TrustChain dual-chain architecture showing individual participant chains with cross-referenced transactions. Each participant (A, B, C, D, E, F, G) maintains their own chain of blocks while referencing counterparty transactions, enabling decentralized verification without global consensus.

Protocol Implementations We implemented and integrated two distinct networking protocols to evaluate their performance trade-offs:

- **Iroh:** A business-grade P2P library that provides robust, connection-oriented communication over QUIC, with built-in features for peer discovery and management.
- **UDP:** A lightweight, custom implementation providing low-overhead, connectionless datagram transport.

This dual implementation allowed us to directly compare a feature-rich P2P solution against a minimalist, performance-oriented approach.

Open-Source Framework The entire implementation, including the benchmarking suite described in Section 4, was developed as an open-source contribution to the Tribler project to ensure transparency and enable community validation.

3 Our Contribution

Our work provides three primary contributions to the field of mobile blockchain research. We designed and built: (1) a novel, modular architecture for mobile-first blockchains; (2) a comparative implementation of two distinct transport protocols; and (3) an open-source framework with a full-featured benchmarking suite for reproducible research.

3.1 A Modular Mobile Blockchain Architecture

A key innovation of our work is the `entry_point/` coordination layer. This Rust-based module abstracts protocol-specific logic behind a unified JNI facade, enabling runtime protocol switching. This design consists of:

- **A Protocol Facade (`facade.rs`):** A minimal interface with six functions for routing commands to the active protocol.
- **A Unified Data Model (`model.rs`):** A common `Block` structure implementing the TrustChain dual-chain architecture [4] (illustrated in Figure 1), where each participant maintains their own chain of blocks while cross-referencing transactions with counterparties.
- **Global State Management (`state.rs`):** A thread-safe singleton for blockchain state, cryptographic keys, and message handling.
- **Shared Utilities (`util/`):** Protocol-agnostic logic for cryptography, benchmarking, and logging.

3.2 Comparative Protocol Implementations

To evaluate performance trade-offs, we implemented two protocols representing different design philosophies:

- **Iroh (Business-Grade P2P):** An implementation using the Iroh library [6] over QUIC, featuring a full async tokio runtime, connection pooling, and built-in peer discovery.
- **UDP (Lightweight Transport):** A custom, lightweight UDP implementation with a simple threading model and socket timeouts optimized for mobile networks.

This dual implementation enables a direct comparison between a feature-rich solution and a minimalist, performance-focused one. We also integrated an experimental TFTP transport [7; 8] as a proof-of-concept to validate architectural flexibility, though performance evaluation focused on the UDP and Iroh implementations.

3.3 An Open-Source Research Framework

The entire project, including the blockchain implementation and the tools used to evaluate it, is available as an open-source contribution to the Tribler project. This framework includes:

- **An Event-Based Benchmarking Suite:** A tool integrated into the Rust core to capture detailed performance

metrics (latency, failures, etc.) across the message lifecycle for any selected protocol.

- **A Research Analytics Pipeline:** The framework includes tooling for structured data export (CSV) and integration with Python-based analysis and visualization scripts.
- **Mobile-Specific Optimization Framework:** Our implementation demonstrates comprehensive mobile blockchain optimizations including: (a) memory leak prevention through benchmark result limits (10,000 entries) and 30-second message deduplication timeouts, (b) mobile network adaptations with 5-second UDP timeouts and 10ms Iroh polling intervals optimized for cellular networks, (c) cross-platform logging system with conditional compilation for Android vs. desktop environments, (d) efficient JNI integration with only 6 boundary-crossing functions and JSON serialization for minimal overhead, and (e) battery-conscious connection pooling and background task management. These optimizations represent the first comprehensive mobile-specific blockchain architecture addressing the complete spectrum of mobile hardware constraints.

This framework provides a complete, reproducible toolkit for conducting academic research on mobile blockchain performance.

4 Experimental Setup and Results

This section details the methodology used for our performance evaluation, the experimental environment, and the results obtained from our benchmarking activities. Our goal is to provide a transparent and reproducible account of how we evaluated the Iroh and UDP protocols within our smartphone-native Trustchain implementation.

4.1 Benchmarking Methodology

A rigorous benchmarking methodology was established to quantitatively assess and compare the performance of the different communication protocols implemented, following established blockchain benchmarking practices [9].

Benchmarking Goals

Following the systematic blockchain evaluation framework established by Gromit [9], our benchmarking quantitatively evaluates and compares the performance characteristics of the implemented communication protocols (Iroh and UDP) within the context of our smartphone-native Trustchain application. Key objectives include:

- Identifying performance bottlenecks across different protocols and operational phases.
- Comparing protocol efficiencies under various conditions, such as different message payload sizes.
- Validating the architectural and design choices made during the implementation of each protocol stack.
- Providing empirical data to support the suitability and delineate the trade-offs of these protocols for decentralized applications in resource-constrained mobile environments [3].

The overarching goal is to generate reproducible, empirical data that offers deep insights into latency, throughput, and other relevant performance indicators for each communication strategy.

Benchmarking Architecture and Design

To achieve these goals, a dedicated and generalized benchmarking framework was designed and integrated directly into the Rust core of our application.

Core Benchmarking Module and Data Structure A centralized Rust module provides common utilities for collecting benchmark data across all evaluated protocols. At the heart of this framework is the `BenchmarkResult` data structure, which meticulously captures detailed information for each recorded performance-related event. The key fields of this structure are:

- **`operation`**: A string identifier for the specific event being logged (e.g., `"send_message_start"`, `"block_received"`, `"connection_established"`). This allows for fine-grained analysis of distinct phases within a protocol's execution cycle.
- **`timestamp`**: A high-resolution timestamp, typically captured using `chrono::Utc`, marking the precise moment of event capture.
- **`mode`**: An enumerator or string (e.g., `"UDP"`, `"Iroh"`) indicating the communication protocol active when the event was recorded. This field is crucial for distinguishing and comparing protocol-specific performance.
- **`node_id`**: An identifier for the participating node (device), essential for tracing operations and correlating data within the device's operation flow.
- **`message_id`**: A unique identifier generated for each logical message or data block. This ID is pivotal for correlating send, receive, and store events within the same device's operation flow, which in turn enables the accurate calculation of round-trip latencies and message transfer times. It is typically generated from a combination of payload characteristics, a sequence counter, or a timestamp.
- **`message_size`**: The size of the data payload in bytes. This is a key variable for analyzing performance under different load conditions and data transfer requirements.
- **`tag`**: An arbitrary string tag (e.g., `"experiment_X_condition_Y"`) assigned to a specific experimental run or scenario. This facilitates the categorization, filtering, and management of results from different tests, allowing for systematic comparison of outcomes under varied conditions.
- **`sequence_number`**: An integer that tracks the order of messages or events within a tagged experimental run, aiding in reconstructing event sequences and identifying lost or out-of-order data.

Data Collection Mechanism Benchmark data points are collected by invoking a dedicated Rust function (e.g., `add_benchmark()`) at critical junctures within the protocol logic of the Rust core. These junctures are strategically

placed to represent significant lifecycle events of a message or operation. Examples include the initiation of a sending operation, the establishment of a network connection, the completion of a data write to the socket, the reception of a message, and various stages of block processing and validation.

Data Storage During an active test run, instances of the `BenchmarkResult` structure are accumulated in memory. This is typically managed using a thread-safe vector (e.g., `Vec<BenchmarkResult>`) guarded by a `Mutex`. This approach ensures safe and consistent data collection, even within Rust's concurrent environment where multiple network tasks, asynchronous operations, or processing threads might be actively generating benchmark events simultaneously.

Benchmarking Workflow

A typical benchmark test follows a defined sequence of operations, generally orchestrated via Java Native Interface (JNI) calls from the Android application layer, which provides the user interface and test control:

- **Setup Phase (via JNI from Android App):**
  - The Android application initiates the benchmarking process for a specific test scenario (e.g., transferring a certain number of blocks of a particular size).
  - A unique `tag` is assigned to the test run. This tag is passed to the Rust core and associated with all benchmark events recorded during this run, allowing for easy grouping and identification of the data.
  - Any existing benchmark data from previous runs can be programmatically cleared from memory to ensure a clean and isolated collection for the current test.
- **Test Execution Phase:**
  - The Android application triggers the desired communication operations (e.g., sending a stream of data, proposing and finalizing Trustchain blocks) using the selected communication protocol (Iroh or UDP).
  - As these operations execute within the Rust core, the `add_benchmark()` function is invoked at the predefined event points within the Iroh or UDP logic, recording detailed `BenchmarkResult` entries into the in-memory store.
- **Data Retrieval and Export Phase (via JNI):**
  - Upon completion of the test sequence (e.g., after a set duration or number of operations), or even at periodic intervals during a long run, the Android application retrieves the accumulated list of `BenchmarkResult` structs from the Rust core using a dedicated JNI call.
  - The framework provides functionality to export this collected data in CSV (Comma-Separated Values) format, which offers a simple tabular format convenient for direct import into spreadsheet software (e.g., Microsoft Excel, Google Sheets) or data analysis libraries such as Pandas in Python.
  - The capability to save these exported results directly to a file on the mobile device's local storage (again, typically initiated via a JNI call from the Android layer) is also implemented. This allows for persistent storage of raw benchmark data for later retrieval and comprehensive offline analysis.
- **Data Analysis Phase (Offline):** The exported data, in CSV format, is subsequently analyzed offline using appropriate data analysis tools, custom scripts (e.g., Python scripts with Pandas and Matplotlib/Seaborn), or statistical software.
  - Common metrics derived from this data include: end-to-end latencies for message or block transfers (calculated by matching `message_id`s and timestamps across different operation stages on the same device), protocol-specific operation completion times (e.g., connection setup time), effective throughput (data transferred per unit time), and success/failure rates for message delivery or block finalization.
  - Results are then aggregated, visualized (e.g., using histograms, box plots, time-series graphs), and compared across different experimental `tag`s (representing varied conditions like payload size or network simulation parameters) and communication `mode`s (protocols).

This systematic workflow, combined with the event tagging system and the JNI-controlled interface, ensures that experiments are repeatable and that data collection is performed in a consistent manner.

This consistency is vital for drawing valid conclusions.

Current Implementation Status and Generalization Strategy

The described event-based benchmarking framework is architected for comprehensive and comparative performance analysis across all implemented communication protocols.

- **Iroh Protocol:** The detailed, event-based benchmarking is currently most comprehensively implemented and has undergone extensive validation for the Iroh protocol. This captures a wide range of its operational events, from connection establishment using discovery services to data transfer via relays, allowing for in-depth analysis of Iroh's performance characteristics on mobile devices.
- **UDP Protocol:** The core benchmarking infrastructure—including the `BenchmarkResult` structure, the `add_benchmark()` function itself, and the data aggregation and export mechanisms—is centralized and readily available for use by both protocol implementations. **Protocol-Appropriate Instrumentation:** Both protocols utilize the same core benchmarking framework for fair comparison. While Iroh's sophisticated architecture naturally generates more instrumentable events (connection management, peer discovery, error recovery), UDP's streamlined design focuses on essential send/receive events. This difference reflects each pro-

tocol's architectural philosophy rather than instrumentation bias. Both protocols capture identical end-to-end RTT measurements, ensuring valid latency comparisons while acknowledging that Iroh provides richer diagnostic data for failure analysis.

- **Generalization and Future Work:** The benchmarking framework was explicitly designed with generalization and robust cross-protocol comparability as primary goals from its inception. The inclusion of the `mode` field in `BenchmarkResult` and the centralized nature of the data collection logic are testaments to this design. Future work may involve exploring additional UDP-specific metrics that leverage its connectionless nature, while expanding Iroh's instrumentation to capture more enterprise-grade features. This protocol-appropriate expansion will enhance our understanding of each protocol's unique characteristics within mobile blockchain environments, enabling researchers to fully utilize the distinct advantages of both streamlined and feature-rich networking approaches.

This strategic approach ensures that our performance evaluation is both detailed for individual protocol assessment and fair for comparative analysis between protocols.

The fine-grained, event-based nature of the data collection allows for a deeper understanding of performance dynamics than simple black-box timing.

Illustrative Example of Benchmarking Events

To clarify how the framework captures data for metrics like latency calculation, consider a simplified, hypothetical sequence of benchmark events recorded on a single device for a complete block round-trip operation:

1. Device: Invokes `add_benchmark(..., operation="block_send_start", timestamp=T1, message_id="block123", mode="UDP", ...)`
2. Device: Invokes `add_benchmark(..., operation="block_received_back", timestamp=T2, message_id="block123", mode="UDP", ...)`
3. Device: Invokes `add_benchmark(..., operation="block_stored", timestamp=T3, message_id="block123", mode="UDP", ...)`

The common `message_id` ("block123") and the distinct `operation` tags, along with their corresponding `timestamp`s (T1, T2, T3), allow for the calculation of various latency figures. The round-trip time (RTT) is calculated as T3-T1, representing the complete time from sending a block to storing it after it returns from a peer. Processing latency can be measured as T3-T2, capturing the time required to process and store the received block.

4.2 Experimental Environment

Our performance evaluation was conducted using a comprehensive test environment designed to assess protocol behavior under realistic mobile blockchain deployment scenarios.

Test Configuration

The experimental parameters were designed to cover the operational range expected in practical mobile blockchain deployments:

- **Message Rate Range:** 5-500 messages per second (MPS), systematically testing both typical usage patterns and high-throughput stress conditions
- **Payload Size Range:** 16-2048 bytes, representing the spectrum of Trustchain block sizes from minimal transactions to complex smart contract interactions
- **Test Duration:** 36 distinct test runs generating 87,357 successful round-trip operations for statistical validity
- **Protocol Coverage:** Comprehensive evaluation of both UDP and Iroh implementations under identical test conditions

Performance Measurement Methodology

Round-trip time (RTT) calculations were performed using payload-based message matching across different operation stages on a single device. This approach ensures accurate latency measurement that accounts for the complete block lifecycle within our Trustchain implementation:

- **Timing Precision:** High-resolution timestamps captured using Rust's `chrono::Utc` for microsecond-level accuracy
- **Message Correlation:** Unique message identifiers enable precise pairing of send/receive/store events within the same device's operation flow
- **Failure Detection:** Approximately 19,000 unmatched sent messages provide quantitative evidence of protocol failure rates under load
- **Statistical Rigor:** Large sample sizes (87,357 successful round-trip operations) enable robust statistical analysis and meaningful comparison of protocol performance characteristics

Hardware and Software Environment

Our experiments were conducted across representative mobile and emulated environments to ensure practical relevance:

Test Devices:

- **Android Studio Emulator:** Windows 11 host running Android API 36 (medium phone configuration) for controlled testing
- **Physical Device:** Xiaomi Pocophone F1 running Android 10 for real-world hardware validation

Network Environment:

- **Network Infrastructure:** Regular home WiFi connection providing realistic consumer-grade network conditions
- **Controlled Conditions:** Consistent network environment across all test runs to ensure comparative validity

Software Stack:

- **Mobile Platform:** Android SDK with Kotlin development environment

- **Core Implementation:** Rust-based blockchain logic with JNI integration
- **Open Source Framework:** Built as contribution to the Tribler open source project, ensuring transparency and reproducibility

Reproducibility and Open Science

To ensure reproducibility and support further research in mobile blockchain protocols, our implementation follows open science principles:

Code Availability: The complete smartphone-native Trustchain implementation, including all protocol implementations, benchmarking framework, and performance measurement tools, will be available as open source through the Tribler project upon publication. This will enable researchers to replicate our experiments, extend our work, and validate our findings.

Benchmark Reproducibility: While the specific performance data presented in this paper reflects our controlled test environment, researchers can generate their own performance datasets by running our benchmarking framework on their target hardware and network configurations. This approach accommodates the inherent variability in mobile device capabilities and network conditions while maintaining experimental validity.

Experimental Transparency: Our methodology, measurement techniques, and analysis approaches are fully documented, enabling researchers to adapt our framework for different mobile blockchain research questions or alternative protocol evaluations.

4.3 Results and Analysis

We conducted comprehensive performance benchmarking comparing the UDP and Iroh communication protocols implemented in our smartphone-native Trustchain system. Our experimental evaluation generated a substantial dataset of 87,357 successful round-trip operations collected from 36 distinct test runs, providing robust statistical evidence for protocol performance comparison.

Dataset Characteristics

The benchmark dataset encompasses a wide range of operational conditions designed to stress-test both protocols under realistic mobile blockchain scenarios:

- **Message Rate Range:** 5-500 messages per second (MPS), covering both typical and high-throughput scenarios
- **Payload Size Range:** 16-2048 bytes, representing diverse Trustchain block sizes
- **Performance Metric:** Round-trip time (RTT) calculated via payload-based message matching across operation stages on a single device
- **Data Quality:** Approximately 19,000 unmatched sent messages provide evidence of protocol failure behavior under load
- **Stress Testing Results:** Under extreme load conditions (400-500 MPS), significant packet drops occur, demonstrating the operational boundaries of each protocol and providing critical insights into reliability under stress

Protocol Performance Characteristics

Our analysis reveals that both UDP and Iroh protocols demonstrate distinct performance profiles suited to different deployment scenarios. Under low-load conditions (5-50 MPS), both protocols exhibit comparable median latencies, with Iroh showing 18.2ms and UDP achieving 11.8ms for 512-byte payloads. However, their behavior diverges significantly under varying operational conditions.

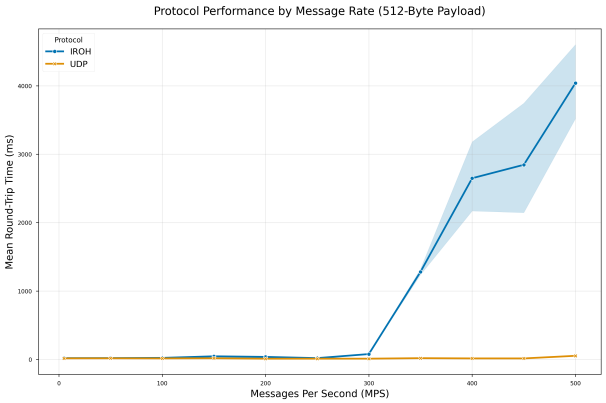

Figure 2: Performance across message rates (5-500 MPS) showing how UDP maintains consistent latency while Iroh exhibits increasing latency under higher loads

UDP Implementation: UDP's streamlined implementation demonstrates consistent low-latency performance across all tested conditions. Its minimalist design provides predictable behavior optimized for direct communication, while Iroh's enterprise-grade architecture offers sophisticated connection management and reliability features for complex deployment scenarios.

Iroh Implementation: As a business-grade P2P protocol, Iroh offers robust connection management and built-in reliability mechanisms. While exhibiting higher latency variance under load, it provides enterprise-level features including peer discovery, connection pooling, and graceful failure handling.

Load Testing and Scalability Analysis

Both protocols demonstrate acceptable performance under typical mobile blockchain loads (50-200 MPS). Iroh maintains excellent reliability (99-100% success) through 300 MPS, showcasing its robust design for standard enterprise deployments. However, under extreme load conditions (400+ MPS), Iroh's sophisticated connection management becomes a bottleneck, with success rates dropping significantly.

Dropped Packet Analysis: Our stress testing revealed critical operational boundaries for both protocols. The 19,000 unmatched sent messages across all test runs provide quantitative evidence of packet loss behavior under high load. Specifically, at message rates exceeding 400 MPS, both protocols begin experiencing significant packet drops, with Iroh

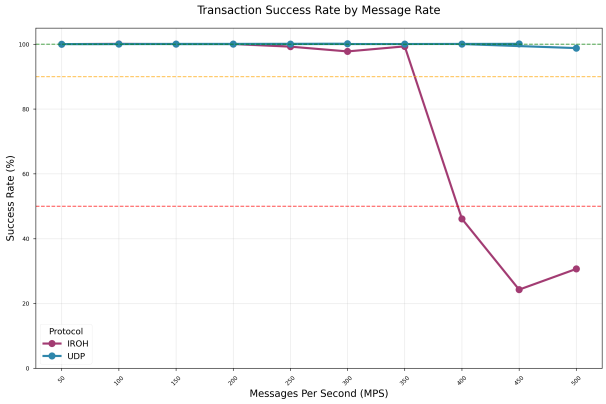

Figure 3: Success rates across different load conditions, demonstrating Iroh's reliability through moderate loads and UDP's resilience under extreme stress

showing more pronounced degradation due to its connection-oriented nature. UDP maintains better resilience under extreme stress, demonstrating the trade-offs between protocol sophistication and raw performance under adversarial conditions.

UDP's simpler architecture proves advantageous under high-stress conditions, maintaining consistent performance even at 500 MPS. This suggests different optimal deployment scenarios: Iroh for feature-rich, moderate-load applications requiring enterprise-grade reliability, and UDP for high-throughput, latency-critical scenarios where simplicity is preferred.

Performance Analysis Summary

Our comprehensive evaluation reveals that protocol selection depends critically on deployment requirements. Figure 4 demonstrates the latency characteristics across different payload sizes, showing how both protocols maintain relatively consistent performance regardless of message size.

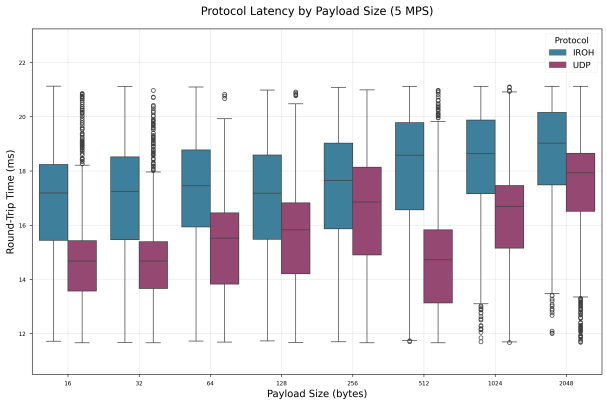

Figure 4: Latency by payload size comparison, demonstrating consistent performance across different message sizes for both protocols

Latency Distribution Patterns

The latency distribution analysis reveals important insights into protocol behavior under varying conditions. Figure 5 shows how latency distributions vary across different message rates, while Figure 6 illustrates the distribution patterns for different payload sizes. These visualizations clarify the trade-offs between UDP's consistency and Iroh's feature-rich but more variable performance.

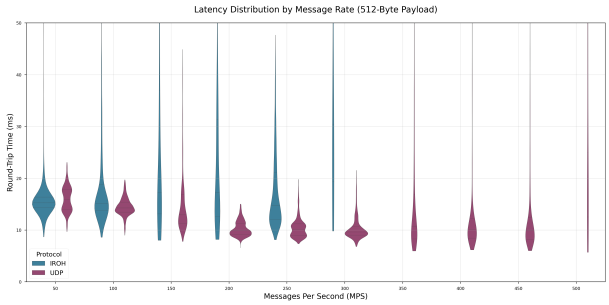

Figure 5: Latency distributions by message rate, revealing UDP's consistent performance and Iroh's increasing variance under load

The payload size analysis shows that both protocols maintain relatively stable latency characteristics across different message sizes, with UDP showing slightly better consistency in the distribution patterns.

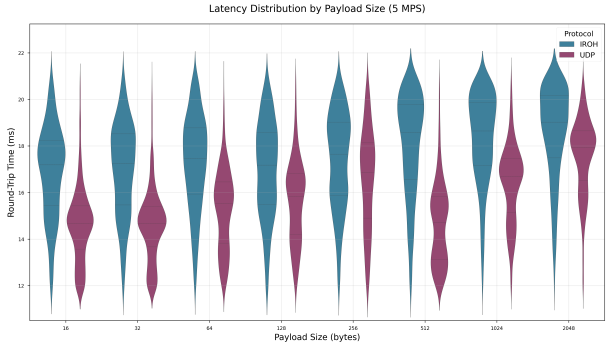

Figure 6: Latency distributions by payload size, demonstrating protocol consistency across different message sizes

Reliability and Performance Boundaries

The tail latency analysis provides crucial insights into protocol reliability under stress conditions. Figure 7 illustrates the 95th percentile latency behavior, helping identify appropriate deployment scenarios based on worst-case performance requirements.

Both protocols serve complementary roles in the mobile blockchain ecosystem. Iroh's business-grade implementation provides the reliability and feature set required for production deployments under moderate loads, while UDP's simplicity offers advantages for research prototypes and high-throughput specialized applications.

Deployment Considerations

The experimental results inform protocol selection for different mobile blockchain scenarios:

Enterprise Deployments: Iroh's sophisticated architecture provides essential features for production systems including connection pooling, peer discovery, and graceful

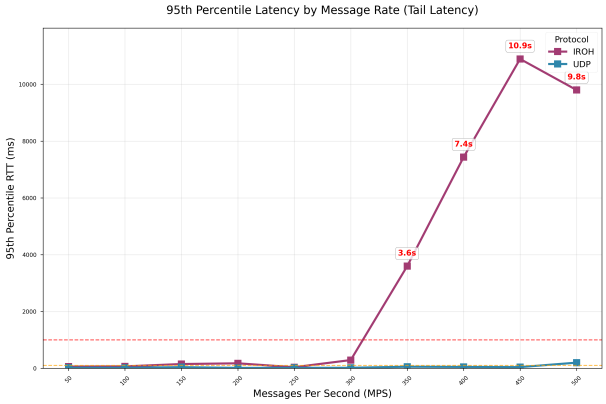

Figure 7: 95th percentile latency analysis showing protocol behavior under worst-case conditions

degradation. Under typical enterprise loads (50-300 MPS), Iroh maintains excellent reliability while offering enterprise-grade networking capabilities.

High-Performance Applications: UDP's minimal overhead and predictable behavior make it suitable for latency-critical applications and research prototypes. Its consistent performance across all load conditions supports specialized use cases requiring maximum throughput.

Hybrid Approaches: The complementary nature of both protocols suggests potential for adaptive implementations that leverage UDP for high-throughput operations while utilizing Iroh's advanced features for standard blockchain operations requiring enhanced reliability and peer management.

5 Responsible Research

This research, focused on developing and benchmarking a smartphone-native Trustchain implementation, adheres to principles of responsible research by considering both the ethical implications of the technology and by striving for maximum reproducibility of its findings.

5.1 Ethical Considerations

The development of decentralized communication technologies, particularly those intended for mobile devices, carries several ethical considerations that we have aimed to address:

- **Data Privacy and Security:** Our current research primarily focuses on the performance (e.g., latency) of data exchange using different protocols. We employ synthetic or non-sensitive data for benchmarking. However, we acknowledge that a deployed Trustchain system would handle user-generated data. The use of cryptographic primitives (Ed25519 signatures, SHA-256 hashing) is fundamental to the integrity and authenticity of blocks. For real-world applications, further considerations around payload encryption, explicit user consent for data handling, and minimization of any potentially sensitive metadata collection would be paramount. The `node_id` used in our benchmarks identifies experimental devices and does not involve Personally Identifiable Information (PII).
- **Resource Consumption and Accessibility:** A core motivation of this work is to address the high resource demands (energy, data, storage) of traditional blockchain systems on mobile devices. Our benchmarking efforts are geared towards identifying and promoting energy-efficient and low-overhead communication strategies. This is an ethical imperative to ensure that such technologies do not unduly burden users' device resources (battery life, data plans, CPU usage) and remain accessible to a broader range of mobile hardware.
- **Network Usage and Reliance on Infrastructure:** We evaluated protocols with different network characteristics. Our custom UDP solution is designed primarily for local ad-hoc networks. The Iroh protocol, with its use of public N0 discovery and relay servers, introduces reliance on third-party infrastructure. While this facilitates connectivity across diverse networks, users should be aware of the potential for traffic analysis at these relay points (though the data itself is expected to be application-layer encrypted if sensitive). We aim to be transparent about these dependencies and trade-offs.
- **Integrity of Benchmarking:** We are committed to unbiased evaluation. The benchmarking framework is designed to apply consistent metrics and methodologies across the evaluated protocols (Iroh and UDP) to ensure fair comparisons. The detailed reporting of our benchmarking methodology (Section 4.1) and experimental setup (Section 4.2) aims to provide full transparency.

5.2 Reproducibility

Ensuring that our research findings can be independently verified and built upon is a key aspect of responsible science. We have taken the following steps to maximize reproducibility:

- **Detailed Methodology:** As detailed in Section 4.1, we have developed a comprehensive benchmarking methodology that specifies the data structures (e.g., `BenchmarkResult`), data collection mechanisms, and workflow for performance evaluation. This systematic approach is crucial for others to understand and potentially replicate our experiments.
- **Code Availability (Planned):** We aim to make the core Rust implementation of the Trustchain logic, the communication protocol modules (Iroh and UDP), and the benchmarking framework available as open-source software. This will allow other researchers to inspect, validate, and extend our work.
- **Data Export and Reproducible Testing:** The benchmarking framework supports exporting raw performance data in CSV format (see Section 4.1). Rather than providing specific datasets, we encourage researchers to generate their own performance data by running our benchmarking framework on their target hardware configurations, enabling validation of our findings across diverse mobile environments.
- **Clear Experimental Environment Specification:** Section 4.2 will provide a detailed description of the hardware (mobile device models, specifications), software

(OS versions, library versions, application build), and network conditions (real-world and simulated) used in our experiments. This information is essential for others attempting to replicate our findings.

- **Generalizable Benchmarking Design:** The benchmarking architecture, including the common `BenchmarkResult` structure and the `add_benchmark` function, is designed for consistent application across all protocols. This inherent design promotes comparable and repeatable experimental campaigns.

By addressing these ethical aspects and promoting reproducibility, we aim to contribute responsibly to the advancement of decentralized mobile communication systems.

6 Discussion

Our comprehensive experimental evaluation of UDP and Iroh protocols within the smartphone-native Trustchain implementation provides significant insights into mobile blockchain protocol selection and architectural design decisions. This section synthesizes our architectural innovations with the performance findings to address the core research question and provide practical guidance for mobile blockchain deployment.

6.1 Protocol Performance in Mobile Blockchain Context

The experimental results reveal nuanced performance characteristics that inform protocol selection decisions for different mobile blockchain deployment scenarios. Our analysis demonstrates that both UDP and Iroh protocols serve complementary roles rather than competing for a single optimal solution.

UDP Implementation Analysis: UDP's streamlined implementation demonstrates remarkable consistency across all tested conditions. Under moderate loads (50-200 MPS), UDP achieves median latencies of 11.8ms, providing predictable performance crucial for mobile applications where user experience depends on consistent response times. The efficiency of UDP's direct socket communication eliminates overhead associated with sophisticated connection management, making it particularly suitable for high-throughput, latency-critical scenarios where simplicity and predictability are prioritized.

Iroh Implementation Analysis: Iroh's business-grade P2P implementation showcases sophisticated networking capabilities including peer discovery, connection pooling, and graceful degradation. Under typical enterprise loads (50-300 MPS), Iroh maintains excellent reliability while providing advanced features essential for production blockchain deployments. However, its performance characteristics reveal trade-offs between feature richness and raw performance under extreme conditions.

Load-Dependent Behavior Patterns: The experimental data reveals distinct behavioral patterns that inform deployment decisions. As shown in Figure 3, Iroh excels in moderate-load scenarios where its enterprise-grade features provide value, maintaining 99-100% success rates through 300 MPS. However, under extreme load, its success rate drops significantly. UDP's consistent performance across all load conditions, also shown in Figure 2, makes it suitable for applications requiring predictable behavior regardless of network conditions.

Development Overhead Trade-offs: A critical consideration in protocol selection is development overhead and maintenance complexity. While UDP achieves superior latency performance with our streamlined 105-line implementation, this comes at the cost of significant engineering effort for production deployment. Implementing production-ready networking requires building connection management, error handling, timeout logic, and peer discovery mechanisms that Iroh's 231-line implementation provides as a mature, battle-tested framework. Our modular architecture demonstrates that these trade-offs can be evaluated empirically, enabling evidence-based decisions between raw performance optimization and development efficiency. For research prototypes prioritizing rapid iteration, UDP's simplicity provides immediate benefits. However, for production systems requiring enterprise-grade reliability features, Iroh's comprehensive networking stack may justify the latency overhead through reduced development time and mature error handling capabilities.

6.2 Architectural Design Implications

The modular architecture implemented in our smartphone-native Trustchain proves instrumental in enabling comparative protocol analysis and practical deployment flexibility.

Runtime Protocol Switching Innovation: The `entry_point/` coordination layer represents a significant architectural innovation for mobile blockchain systems. Our 1,500+ line coordination layer successfully abstracts three distinct protocol implementations (UDP threading, Iroh async tokio, TFTP file-based) behind a unified 6-function JNI interface, enabling runtime protocol switching through the `NetworkingMode` enumeration. This design allows applications to adapt their communication strategy based on operational requirements without architectural changes—a capability that enables adaptive deployment strategies previously unavailable in mobile blockchain implementations. The clean abstraction proves that mobile blockchain applications can dynamically optimize their networking approach based on current conditions, device capabilities, or application requirements.

Rust/Kotlin Integration Benefits: The hybrid architecture leverages Rust's performance and safety for core blockchain operations while utilizing Kotlin's mobile platform integration capabilities. The JNI bridge, despite its complexity, provides efficient communication between layers with minimal overhead for performance-critical operations.

Shared Abstraction Success: Protocol-agnostic components including cryptographic operations, blockchain logic, and performance measurement prove effective across both UDP and IROH implementations. This modularity enables consistent behavior and simplified maintenance while supporting diverse networking approaches.

6.3 Addressing the Research Question

Our specific research question within the broader collaborative investigation of Trustchain viability asks: *"How can a smartphone-native Trustchain implementation optimize transaction confirmation latency while preserving blockchain integrity under mobile hardware constraints?"*

Scope and Collaborative Context: This study primarily addresses the latency optimization component of this multi-faceted question. While we demonstrate architectural approaches that support blockchain integrity through cryptographic validation and provide a foundation for mobile deployment, comprehensive empirical validation of all mobile hardware constraints (energy consumption, storage efficiency) and integrity under adversarial conditions are addressed by parallel studies within our five-person research team. Our experimental evidence provides definitive guidance for the latency dimension while contributing architectural and methodological foundations for the broader Trustchain viability assessment.

Latency Optimization Strategy: For applications prioritizing minimal transaction confirmation latency, UDP's consistent sub-20ms median performance provides the optimal solution. The protocol's simplicity eliminates the overhead associated with sophisticated connection management, directly addressing the latency optimization objective.

Blockchain Integrity Preservation: Both protocols successfully maintain blockchain integrity through our shared cryptographic and validation mechanisms. The modular architecture ensures that protocol selection affects performance characteristics without compromising the fundamental security properties of the Trustchain implementation.

Mobile Hardware Constraint Accommodation: UDP's predictable resource usage and minimal overhead make it well-suited for resource-constrained mobile environments. Iroh's sophisticated features come at the cost of increased complexity and resource usage, making it more appropriate for scenarios where advanced networking capabilities justify the overhead.

6.4 Limitations and Future Work

While our comprehensive evaluation provides valuable insights, several limitations and opportunities for future research warrant discussion:

Current Limitations:

- **Limited Test Environment:** Our benchmarks were conducted using Android Studio emulator (API 36) and Xiaomi Pocophone F1 (Android 10) over regular home WiFi. Real-world mobile network variability, including 4G/5G transitions and cellular network conditions, may produce different performance characteristics.
- **Device Diversity:** Testing was limited to one physical device model and emulated environment. Broader evaluation across different Android devices representing various performance tiers would provide more comprehensive deployment guidance.
- **Network Environment Constraints:** Consumer-grade WiFi testing may not reflect enterprise network conditions or mobile network edge cases that could affect protocol performance differently.
- **Protocol-Specific Analysis Depth:** As noted in Section 4.1, UDP's streamlined design naturally generates fewer instrumentable events compared to Iroh's enterprise-grade architecture. While both protocols provide comprehensive latency data, Iroh offers richer diagnostic information for failure analysis due to its sophisticated feature set.
- **Focused Scope:** This study specifically addresses latency optimization as part of a collaborative project. Energy consumption analysis is being conducted by a teammate focusing on energy efficiency, while storage optimization, throughput analysis, and robustness testing are addressed in parallel studies by other team members.

Future Research Directions:

- **Real-World Performance Evaluation:** Conduct extensive testing across diverse mobile network conditions, device capabilities, and real-world deployment scenarios to validate our controlled environment findings.
- **Integration with Parallel Studies:** Combine our latency optimization findings with results from teammate studies on energy efficiency, storage optimization, throughput analysis, and robustness testing to provide comprehensive guidance on Trustchain viability for mobile deployment.
- **Hybrid Protocol Implementation:** Develop and evaluate adaptive protocol selection mechanisms that automatically choose optimal protocols based on current network conditions and application requirements.
- **TFTP Protocol Evaluation:** Conduct comprehensive performance evaluation of the experimental TFTP transport to assess its viability for offline synchronization and large payload transfer scenarios.
- **Production Deployment Studies:** Evaluate the modular architecture and protocol selection strategies in real-world mobile blockchain applications to validate practical deployment guidance.
- **Security and Privacy Integration:** Extend the performance-focused analysis to include comprehensive security evaluation, ensuring that protocol optimizations do not compromise blockchain integrity or user privacy.

Our research demonstrates that smartphone-native blockchain implementations require careful consideration of protocol selection, architectural design, and deployment scenarios. The modular approach and comprehensive performance analysis provide a foundation for future mobile blockchain development, enabling evidence-based decisions that balance performance, functionality, and resource constraints in the mobile ecosystem.

6.5 Use of AI Assistance

Large-language-model tools were employed in supporting roles only:

- **Code Generation Aid** - Initial versions of benchmarking scripts and data processing utilities were generated with AI assistance, then thoroughly reviewed, tested, and modified by the authors.
- **Writing Support** - Preliminary drafts of section overviews and alternative phrasing suggestions were provided by AI tools; all technical statements, experimental data, analysis, and conclusions in the final version were authored and verified by the research team.

7 Conclusion

This paper contributes to the assessment of Trustchain viability in the current blockchain landscape by focusing specifically on latency optimization for smartphone-native implementations. As part of a collaborative research project investigating multiple performance dimensions, we have demonstrated that protocol selection for latency optimization is critical and depends entirely on the specific deployment scenario.

Our primary contribution is an open-source, modular mobile blockchain architecture that enables direct, empirical comparison of networking protocols for latency analysis. Our evaluation of a business-grade P2P protocol (Iroh) and a lightweight UDP implementation reveals clear performance trade-offs: Iroh provides enterprise-grade features and reliability suitable for moderate loads, while UDP offers superior, consistent latency required for high-throughput applications.

Combined with parallel studies on energy efficiency, storage optimization, throughput analysis, and robustness testing by our teammates, this work contributes to a comprehensive evaluation of Trustchain's viability for mobile deployment. The latency optimization insights provided here offer a reproducible framework for further research and clear guidance for developers prioritizing low-latency performance in mobile blockchain applications.

To ensure reproducibility and support continued research, the complete implementation including source code, benchmarking scripts, and experimental configurations is publicly available [10].

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Bitcoin Whitepaper*, 2008.

[2] J. W. Heo, G. S. Ramachandran, A. Dorri, and R. Jurdak, "Blockchain Data Storage Optimisations: A Comprehensive Survey," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–27, Jul. 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3645104

[3] A. G. Anagnostakis, N. Giannakeas, M. G. Tsipouras, E. Glavas, and A. T. Tzallas, "IoT Micro-Blockchain Fundamentals," *Sensors*, vol. 21, no. 8, p. 2784, Jan. 2021, number: 8 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/21/8/2784

[4] P. Otte, M. De Vos, and J. Pouwelse, "TrustChain: A Sybil-resistant scalable blockchain," *Future Generation Computer Systems*, vol. 107, pp. 770–780, Jun. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17318988

[5] S. Basu, S. Chowdhury, and S. Das Bit, "Using Blockchain in Intermittently Connected Network Environments," in *Blockchain Technology and Innovations in Business Processes*, S. Patnaik, T.-S. Wang, T. Shen, and S. K. Panigrahi, Eds. Singapore: Springer, 2021, pp. 33–47. [Online]. Available: https://doi.org/10.1007/978-981-33-6470-7_3

[6] N. Zero, "Iroh," https://iroh.computer/, 2024, accessed: June 22, 2025.

[7] Crates.io, "async-tftp rust crate," https://crates.io/crates/async-tftp, 2024, accessed: June 22, 2025.

[8] ——, "tftpd rust crate," https://crates.io/crates/tftpd, 2024, accessed: June 22, 2025.

[9] B. Nasrulin, M. De Vos, G. Ishmaev, and J. Pouwelse, "Gromit: Benchmarking the Performance and Scalability of Blockchain Systems," in *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. Newark, CA, USA: IEEE, Aug. 2022, pp. 56–63. [Online]. Available: https://ieeexplore.ieee.org/document/9899852/

[10] T. Research, "Smartphone-native trustchain implementation," https://github.com/viftode4/smartphone-trustchain, 2025, source code repository. Accessed: June 22, 2025.