Throughput Analysis of a Trustchain Protocol Implementation

t.chirila@student.tudelft.nl
Author: Tudor Chirilă
Supervisors: Bulat Nasrulin, Johan Pouwelse

1. Overview

- **Context:** Bitcoin's[1] design trades scalability (≈7 TPS, 10 min blocks)[2] for security; DAG-based protocols[3] like Trustchain[4] offer higher throughput and Sybil resistance.
- **Why Mobile?** Smartphones dominate internet access and could enable fast, peer-to-peer payments - despite limited compute power and mobile testing tools.

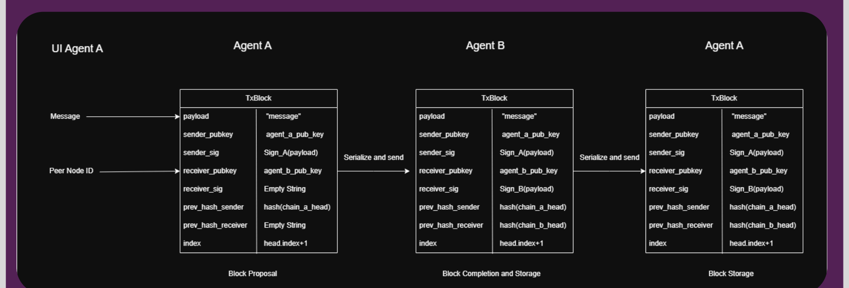
Research Questions

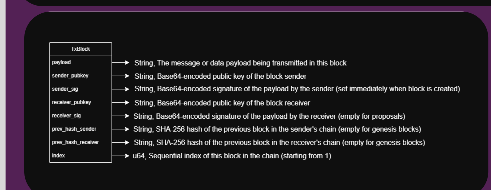
1. How can we build a smartphone application which implements the core features of the Trustchain protocol?
2. Given the base implementation of the application, what is its throughput performance (that is, the number of transactions per second that the app will be able to support)?

2. Methodology

- **Prototype**
  - Build Trustchain[5] client with P2P framework
  - Connect to peer, attach message, exchange & store blocks
  - Support UDP[6] transport
- **Throughput Tests**
  - Metric: completed blocks stored per ms, after full processing
  - Controlled: private Wi-Fi, identical phones/OS
  - Variables: test duration, message rate[7], payload size
  - Compare QUIC[8] (iroh) vs. UDP under same workloads

3. Overview of the Trustchain Protocol Implementation

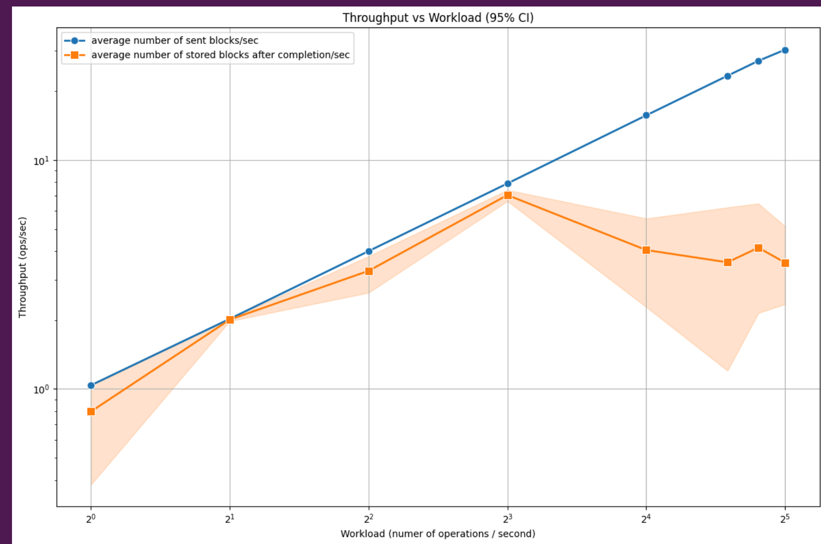




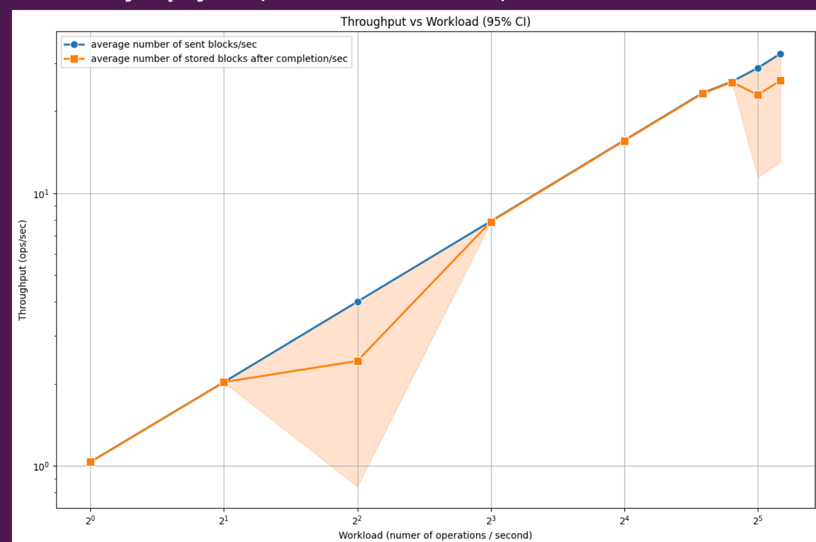
The implementation is done in the Rust and Kotlin languages (for Android agents), bridged by the Java Native Interface.

- **The iroh implementation** builds on Rust's quinn library to offer a peer-to-peer messaging API over QUIC, complete with built-in asymmetric encryption for agent key pairs and peer-discovery support.
- **The UDP implementation** is deliberately kept simple: using native Rust networking crates, the code opens a pre-defined UDP socket, sends block-sized messages to it, and runs a dedicated receiver thread.

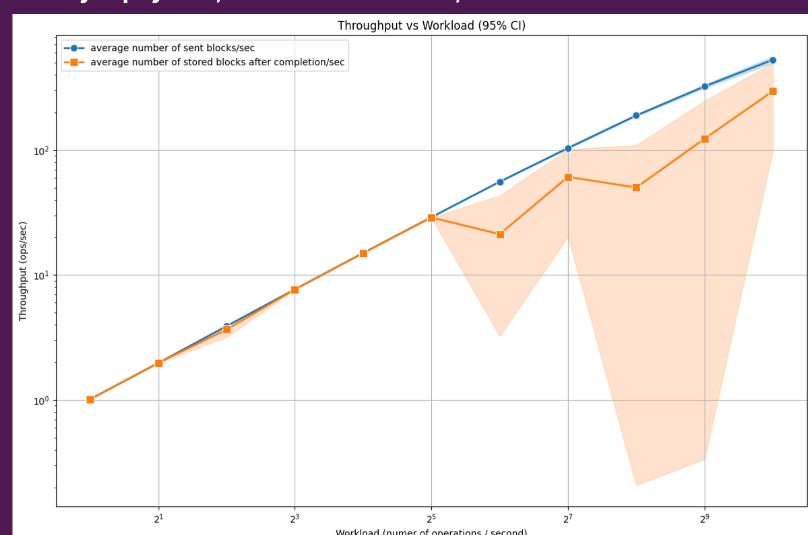
4. Experiments



100 Byte payload, 25 second duration, iroh with reconnection



100 Byte payload, 25 second duration, iroh with connection retainment



128 Byte payload, 25 second duration, iroh with connection retainment, high workloads

5. Conclusions

- **Research Question 1**
  - Built a smartphone app using the Rust-iroh framework, yielding a functional Trustchain prototype.
  - Implemented block creation, payload attachment, peer exchange of signed blocks, and local storage.
- **Research Question 2**
  - Maintaining a persistent QUIC connection delivered over three times the throughput compared to reconnecting on each message.
  - Achieved approximately 28 blocks/s with a 128 B payload at 29 msg/s. When considering a full block size of minimum 536B recorded, this corresponds to ~15 KB/s storage throughput.
  - The simple UDP implementation outperformed the iroh version, storing at least 500 blocks/s of 668 B each.

6. Limitations

- **Protocol fidelity**
  - Our implementation supports throughput tests but skips the exact block format and consensus layer; a full re-implementation is needed for deeper protocol insights.
- **Limited testbed**
  - Experiments run on two devices in a fixed network; multiple devices and varied network topologies are required to generalize findings.
- **Benchmarking overhead**
  - Capturing full timestamps, block structure, and message content yields rich data but adds overhead; a leaner metric (e.g., counting stored blocks) is faster but sacrifices structural validation.
- **Low repeatability**
  - Manually executing the QUIC-based (iroh) experiments restricts the number of runs; an automated test bench is needed to scale repetitions.
- **Code robustness**
  - The Rust-based iroh optimization sometimes disconnects under heavy load, and the original iroh version fails around 40 msg/s; improving stability and diagnosing these faults is critical.

7. References

[1]Nakamoto, S. – Bitcoin: A peer-to-peer electronic cash system (2008)
[2]Croman et al. – On scaling decentralized blockchains, Financial Cryptography (2016)
[3]Wang et al. – SoK: DAG-based blockchain systems (2022)
[4]Otte et al. – Trustchain: A sybil-resistant scalable blockchain, FGCS (2020)
[5]Pouwelse – Trustchain protocol, IETF draft-01 (2018)
[6]RFC 768 – User Datagram Protocol (1980)
[7]Nasrulin et al. – Gromit: Benchmarking blockchain performance, DAPPs (2022)
[8]RFC 9000 – QUIC: A UDP-Based Multiplexed and Secure Transport (2021)