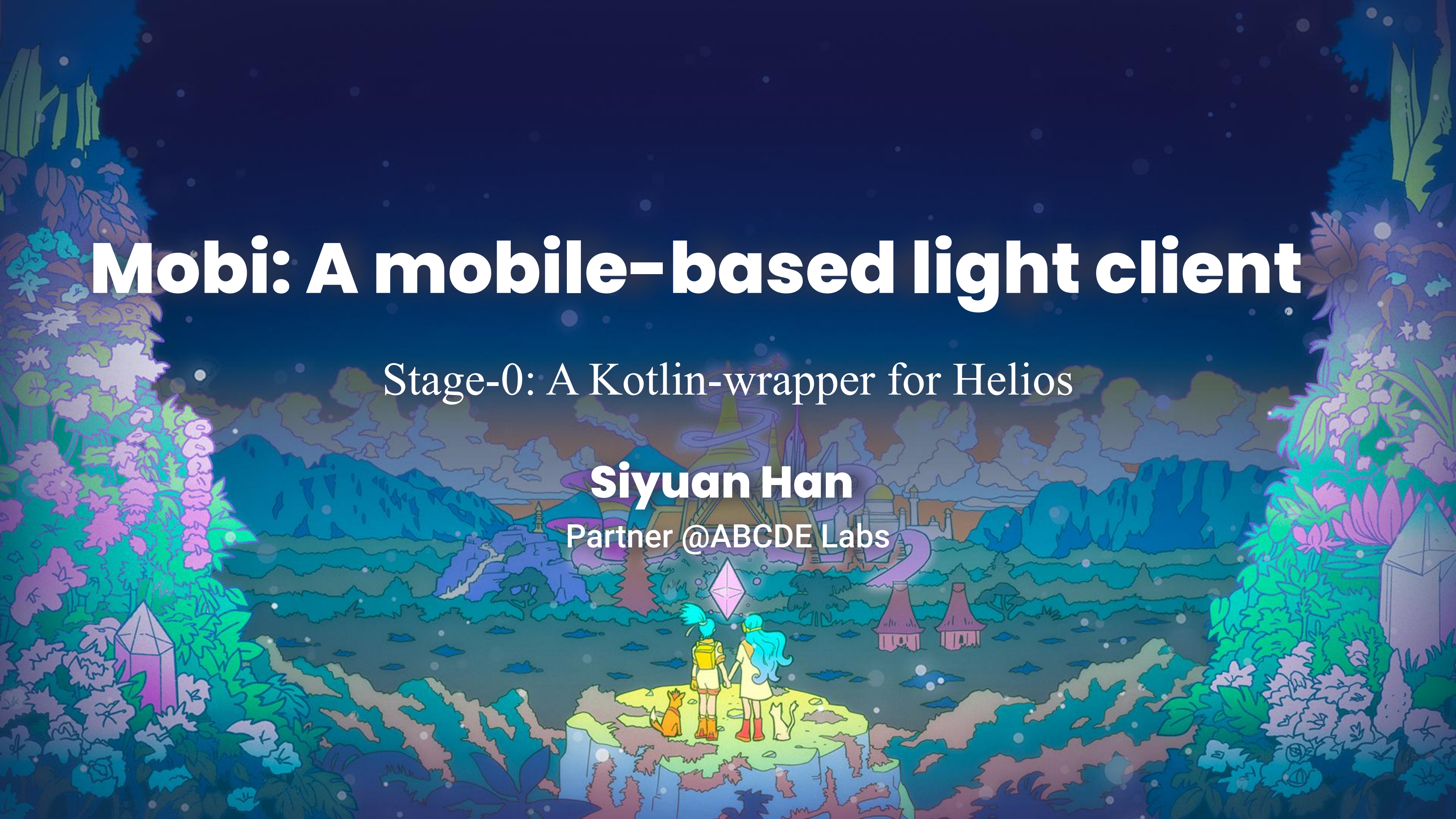


Mobi: A mobile-based light client

Stage-0: A Kotlin-wrapper for Helios

Siyuan Han

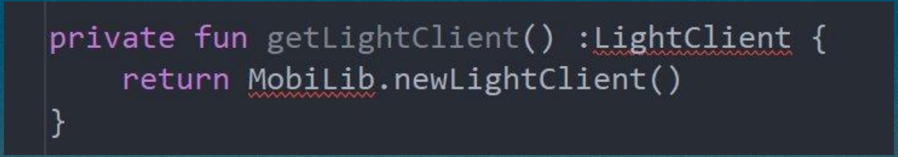
Partner @ABCDE Labs

Mobi

Powered by ABCDE CYSIC

Don’t trust, verify

- The entire Blockchain community is founded on a core principle: **Don’t trust, verify.** This principle serves as the foundational pillar for the architecture of the Blockchain network.
- Several third-party teams have initiated their own implementations, such as **Nimbus (Nim)**, **Lodestar (TypeScript)**, and **Helios (Rust)**. However, none of these light clients can natively run on the most ubiquitous devices: mobile phones.



```
private fun getLightClient() :LightClient {
    return MobiLib.newLightClient()
}
```

Our goal is to offer the Ethereum community an alternative: **a light client that runs natively on mobile devices.**

Motivation

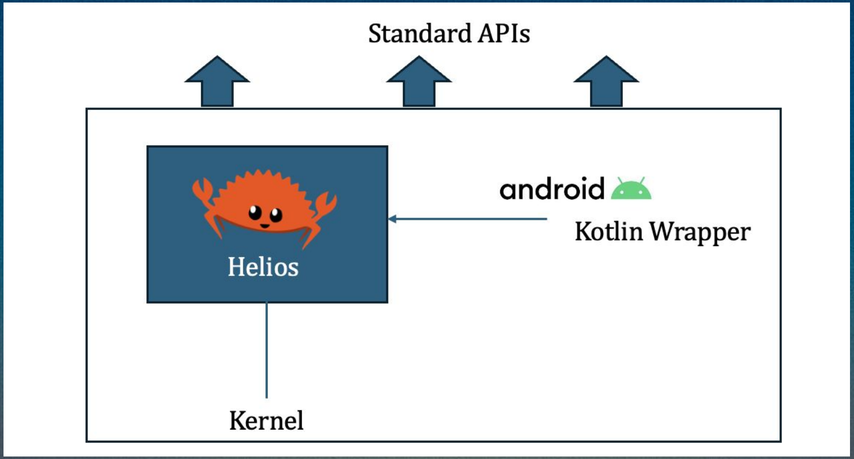
Why this is technical feasible?

- As of 2024, the performance of edge devices has advanced significantly. We are already seeing some developers running LLM inference on edge devices.

- Running a light node on mobile devices primarily requires periodic network communication and some storage space. Therefore, running an Ethereum light client on mobile devices is entirely feasible.

Arch and Roadmap

1. Our first goal is to build a **SDK** for a natively running light client on mobile platforms (Android).
2. Our secondary goal, after achieving compatibility with the Ethereum mainnet, is to provide a hub function, offering light client services for Ethereum L2s.

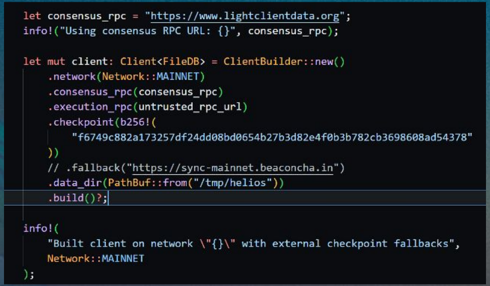


Challenges

- Helios is a codebase written in Rust, containing a significant number of asynchronous threads. Android applications, on the other hand, are currently written in Kotlin and run on the JVM.

- Rust does not have the garbage collection (GC) mechanism, whereas the JVM has GC. This can result in asynchronous threads being incorrectly garbage collected by the JVM.

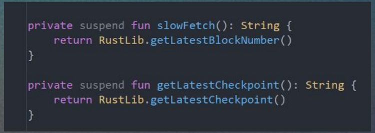
- Moreover, Android's unique permission protection mechanism introduces various permission access errors when directly invoking Helios' code.



```
let consensus_rpc = "https://www.lightclientdata.org";
info!("Using consensus RPC URL: {}", consensus_rpc);

let mut client: Client<FileDB> = ClientBuilder::new()
    .network(Network::MAINNET)
    .consensus_rpc(consensus_rpc)
    .execution_rpc(untrusted_rpc_url)
    .checkpoint(b256!(
        "f6749c882a173257df24dd08bd0654b27b3d82e4f0b3b782cb3698608ad54378"
    ))
    // .fallback("https://sync-mainnet.beaconcha.in")
    .data_dir(PathBuf::from("/tmp/helios"))
    .build()?;

info!(
    "Built client on network \"{}\" with external checkpoint fallbacks",
    Network::MAINNET
);
```



```
private suspend fun slowFetch(): String {
    return RustLib.getLatestBlockNumber()
}

private suspend fun getLatestCheckpoint(): String {
    return RustLib.getLatestCheckpoint()
}
```

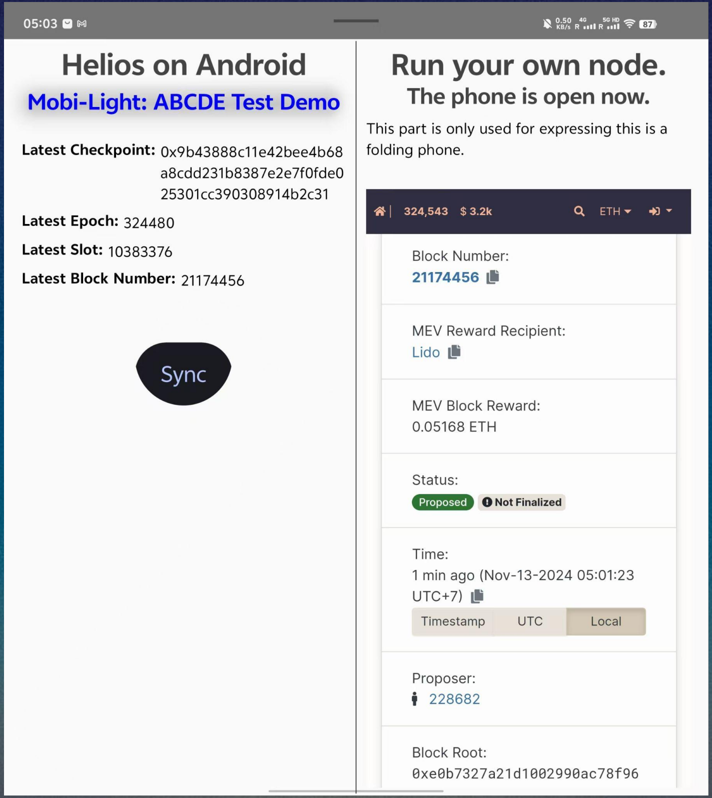
Long-term Goal: Smart Client

Apart from these basic functionalities, we have further expectations for the mobile light client, including but not limited to the following two directions:

1.(Functionality) The core function of a light client is verification. However, given the strong performance of modern mobile devices, we aim to build a 'Light-client plus'. Beyond simple verification, we intend to add storage capabilities. For example, users could selectively store all state and historical data relevant to themselves. Additionally, we can develop RPC functionality for use by other light nodes.

2.(Entertainment) In addition to the local light node, mobile devices can also run local LLM inference services. For instance, Android provides a local Gemini inference API. We also want to explore whether we can combine the Node and a local LLM at the client layer to conduct some interesting experiments.

Demo



05:03

Helios on Android

Mobi-Light: ABCDE Test Demo

Latest Checkpoint: 0x9b43888c11e42bee4b68a8cdd231b8387e2e7f0fde025301cc390308914b2c31

Latest Epoch: 324480

Latest Slot: 10383376

Latest Block Number: 21174456

Sync

Run your own node.

The phone is open now.

This part is only used for expressing this is a folding phone.

324,543 $ 3.2k ETH

Block Number:
21174456

MEV Reward Recipient:
Lido

MEV Block Reward:
0.05168 ETH

Status:
Proposed Not Finalized

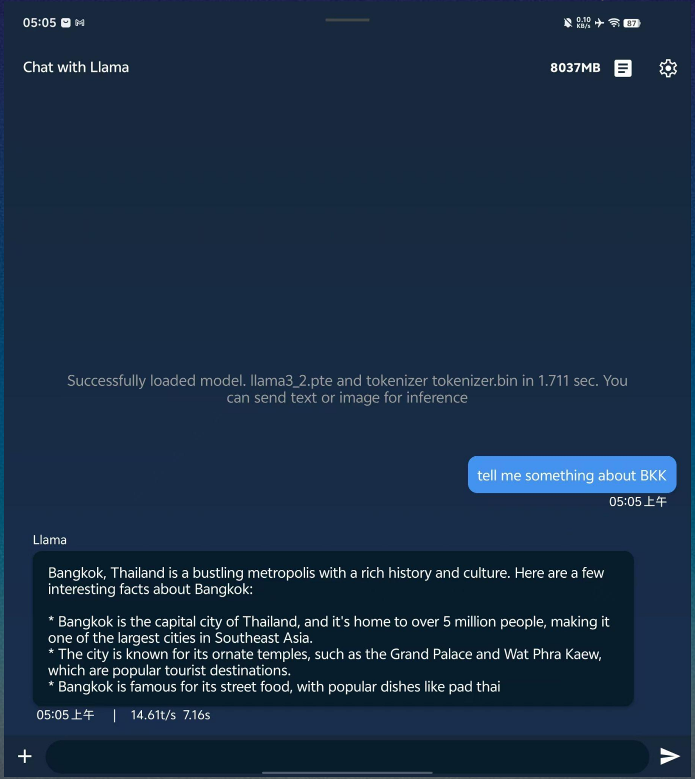
Time:
1 min ago (Nov-13-2024 05:01:23 UTC+7)

Timestamp UTC Local

Proposer:
228682

Block Root:
0xe0b7327a21d1002990ac78f96

Demo


05:05

Chat with Llama

8037MB

Successfully loaded model. llama3_2.pte and tokenizer tokenizer.bin in 1.711 sec. You can send text or image for inference

tell me something about BKK

05:05上午

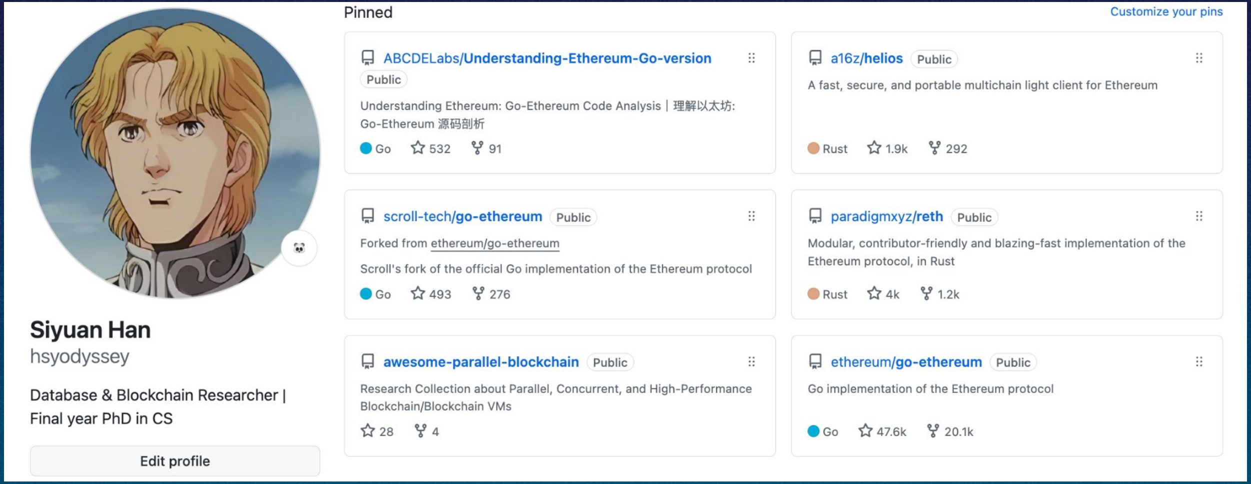
Llama

Bangkok, Thailand is a bustling metropolis with a rich history and culture. Here are a few interesting facts about Bangkok:

* Bangkok is the capital city of Thailand, and it's home to over 5 million people, making it one of the largest cities in Southeast Asia.
* The city is known for its ornate temples, such as the Grand Palace and Wat Phra Kaew, which are popular tourist destinations.
* Bangkok is famous for its street food, with popular dishes like pad thai

05:05上午 | 14.61t/s 7.16s

Thanks & About me



Pinned

Customize your pins

Siyuan Han
hsyodyssey

Database & Blockchain Researcher | Final year PhD in CS

Edit profile

ABCDELabs/Understanding-Ethereum-Go-version Public
Understanding Ethereum: Go-Ethereum Code Analysis | 理解以太坊: Go-Ethereum 源码剖析
Go 532 91

a16z/helios Public
A fast, secure, and portable multichain light client for Ethereum
Rust 1.9k 292

scroll-tech/go-ethereum Public
Forked from ethereum/go-ethereum
Scroll's fork of the official Go implementation of the Ethereum protocol
Go 493 276

paradigmxyz/reth Public
Modular, contributor-friendly and blazing-fast implementation of the Ethereum protocol, in Rust
Rust 4k 1.2k

awesome-parallel-blockchain Public
Research Collection about Parallel, Concurrent, and High-Performance Blockchain/Blockchain VMs
28 4

ethereum/go-ethereum Public
Go implementation of the Ethereum protocol
Go 47.6k 20.1k

I'm partner of ABCDE Labs and a final year Ph.D. student at HKUST, CSE. My research interests include database engines, blockchain storage optimization, distributed transaction management, ZK-based systems. Some of my research has been published in ICDE and SIGMOD.

https://github.com/hsyodyssey

s@abcde.com