Syndica

Sig Take-Home Project:
Understanding Solana's Proof-of-History (PoH)

The Task:

The task is to understand the PoH component in the Solana Repository (https://github.com/solana-labs/solana at commit d0b1f2c).

The task consists of three sub-tasks:

1. **Read and understand** how Solana's PoH chain is created
2. **Write code** that emulates the same logic to create a continuous PoH chain over time while receiving hashes to mix into the chain. You can write this in any language you choose.
3. **Write a short report**/blog post on the methods/structures listed on page 2. You can assume your audience is the rest of the Sig team who understands Rust but not how PoH works or how it is implemented. You can also use https://excalidraw.com/ to create diagrams.

Note: the full document is 3 pages - please read all of them before starting.

The Goal of the Task:

It's important to note that the goal for this task is to showcase your skills for …

- Diving in to read and trace out the Solana codebase (a complex Rust codebase with little documentation)
- Writing code that is easy to understand while also being highly performant
- Communicating complex technical concepts into material that another developer can read and understand

Take charge of the task and try to showcase these skills as best as you can.

Note: reading the solana whitepaper may help but you should focus on understanding the rust code for this assignment.

Background:

PoH, at a high level, is an infinite hash loop which hashes itself over and over. Since a hash is a one-way function, the chain of hashes is proof that some time has passed. This passage of time acts as a consistent clock, which in turn runs the leader schedule which decides who is chosen to produce the next block.

Including the hash of transactions inside this hash chain is a way of ordering transactions. For example:
```
```

```
tx_hash = hash(vec![transactions]) // hash using a merkle tree
self.hash_state = hash(tx_hash, self.hash_state)
```

This is how Solana represents a block of ordered transactions.

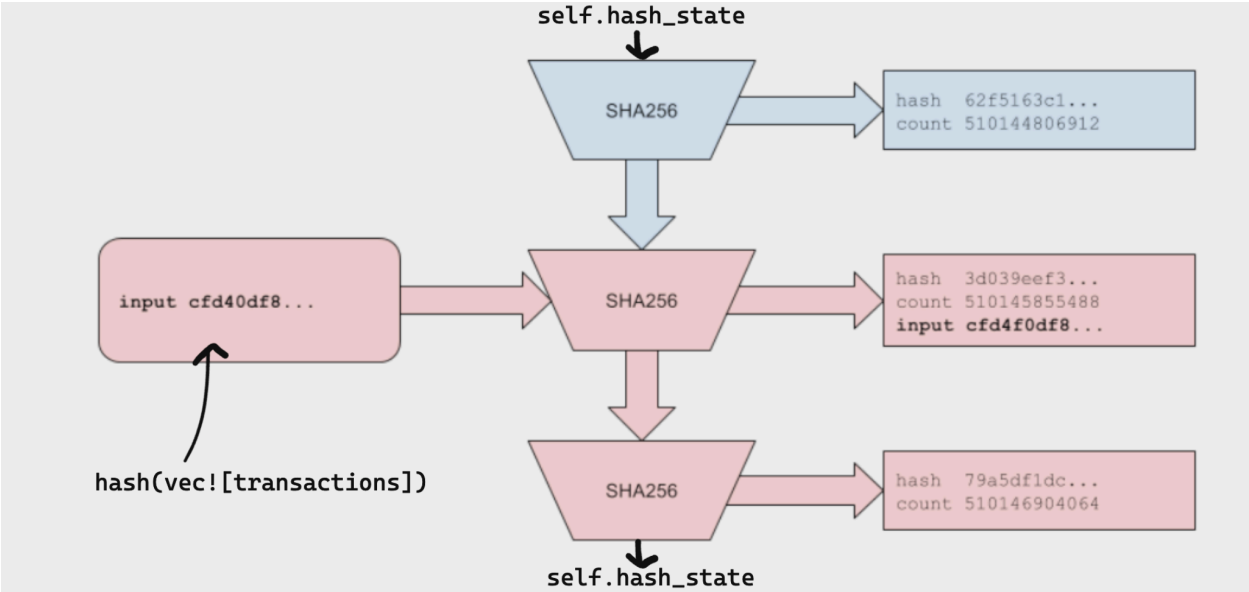

Figure 3: Inserting data into Proof of History

Structures and Methods to Understand:

There are a few main structures which we want to understand more in-depth and how they connect:

- Poh : entry/src/poh.rs
- Entry : entry/src/entry.rs
- PohService : poh/src/poh_service.rs
- PohRecorder: poh/src/poh_recorder.rs

The main functions we want to understand include:

- Poh :: tick(...)
- Poh :: record(...)
- Poh :: hash(...)
- PohRecorder :: record(...)
- PohService :: tick_producer(...)

Include documentation on these structures and functions in the final report. Feel free to include/explain additional structures/methods which you think are important to understand.

Syndica

Why PoH:

It may also help to understand why we decided on PoH for the task:

- It's one of the simpler and better (relatively speaking) documented components of the Solana protocol
- While its logic is simple (which can help guide you through the codebase), the actual Rust implementation of building can be fairly complex/difficult to navigate
- We keep the task open-ended so use your time-management skills as best as you can, it will likely be worthwhile to research existing documentation before diving into the code
- This is a typical day-to-day task that a developer on the Sig team would be doing