STRATUM V2
PROJECT DEVELOPMENT PROPOSAL

Definitions

- STRATUM V2 means the stratum v2 protocol as defined here.
- MINING PROTOCOL, JOB NEGOTIATION PROTOCOL, TEMPLATE DISTRIBUTION PROTOCOL, JOB DISTRIBUTION PROTOCOL means the (sub)protocols as defined by STRATUM V2.
- MINING DEVICE, POOL SERVICE, MINING PROXY, JOB NEGOTIATOR, TEMPLATE PROVIDER means the roles defined by STRATUM V2

Introduction

The advantages that STRATUM V2 can bring to the bitcoin ecosystem are well noted, particularly the ability to reduce the centralization of consensus groups drastically. So I will not elaborate on this further.

Even if a partial implementation of the STRATUM V2 protocol already exists, the remarkable work done by Braiins must be extended for several reasons.
Firstly, it is not complete. To date, only the MINING PROTOCOL has been partially implemented compared to the four protocols foreseen. Among the five possible roles, only the MINING DEVICE and the MINING PROXY have been partially implemented.
Secondly, as an integral part of the Bitcoin ecosystem, among the key requirements of the implementation of the STRATUM V2 there are (i) community-oriented, (ii) designed for two types of users, operators (i.e. miners and pool operators) and developers, and (iii) easily embeddable into the Bitcoin Core. The current implementation does not meet any of the above requirements: it is not community-oriented (a single company controls it), it is mainly designed for operators, it cannot be integrated into Bitcoin Core.
Finally, the project documentation must be improved taking into account the two different type of users.

Proposal

Implementing 3 (sub)protocols MINING PROTOCOL, JOB NEGOTIATION PROTOCOL, TEMPLATE DISTRIBUTION PROTOCOL and 4 roles POOL SERVICE, MINING PROXY, JOB NEGOTIATOR, TEMPLATE PROVIDER. JOB DISTRIBUTION PROTOCOL and MINING DEVICE role are outside the scope of work. The first is not yet specified and other developers will develop the second.

The implementation will be done according to a schema similar to the one defined in [Annex 1]. There will be a monorepository containing the products for developers [RepoDev] and there will be a monorepository containing the products dedicated to the operators [RepoOperators].

In RepoDev, there are:

- stratum-v1: a library crate. It implements the stratum v1 protocol (just a copy of the one implemented by Braiins).
- stratum-v2: workspace containing several library crates that implement the (sub)protocols as specification, 2 FFI library crates, maybe adaptor a library crate, several library crates the expose a higher-level API
- translator library crate translating v1 <-> v2

In RepoOperators, there is a binary crate that implements a MINING PROXY and a binary crate that implements a POOL SERVICE.

TEMPLATE PROVIDER will be implemented directly in the Bitcoin Core.

Since the project is community-driven, onboarding must be effortless.

The product dedicated to the developers must be designed with usability in mind. In particular, who will develop the implementation of the MINING DEVICE must rely on an easily integrable product.

STRATUM V2 has been designed to reduce the friction in adopting the protocol by miners and pool operators. To achieve this goal, it offers several incentives to both, but if the protocol implementation is not user friendly, all the efforts done by the designer of the protocol will have vanished. For this reason, it is necessary that deploying the implementation of the POOL SERVICEs and MINING PROXIEs be very easy.

Execution

A part of the RepoDev will be integrated into the Bitcoin Core. This is why security is fundamental. In fact, every crate in the RepoDev will have a minimal dependency set and will be buildable on Guix. The code will be well tested and CI will be added via Github actions.

The developers of the MINING DEVICE will use part of RepoDev. So it will be compilable for arm-openwrt-linux-muslgnueabi-gcc target without problems. If requested by the MINING DEVICE developers, it will be developed a library crate called adaptor that exposes an API identical to the one exposed by the already existing implementation developed by Braiins. In this way, the existing code will be reusable.

The protocol implementation is also exported as a library that exposes a C ABI to be used outside of rust.

To facilitate the developers' onboarding, each library crate will be full of examples and well documented. There will always be at least an untaken good-first-issue. Finally, a contribute file will be available for each repo, explaining how to do the first PR and the code structure.

To simplify the adoption by the operators. Both the POOL SERVICE and the MINING PROXY implementations will be very easy to deploy. Through GitHub's actions, each release will be made available as docker images and as a binary. A good quantity of examples will be provided. Every example will be composed of a detailed readme and a docker-compose file containing all the software needed (the pool service or the proxy or both and bitcoins). Between the examples, there will also be one for solo miners. Two tutorials will be written. All the use cases that will be supported by the POOL SERVICE and the MINING PROXY implementations are listed In the [Annex 2].

The TEMPLATE PROVIDER should be implemented directly into Bitcoin Core. Probably something like this PR will be done. It would be better to use the library from RepoDev, that are secure and buildable on Guix. If that is not possible, a part of STRATUM V2 should be implemented in Bitcoin Core using C++. If also that is not an option, a TEMPLATE PROVIDER can be implemented in rust. The implementation will talk with Bitcoin Core using the zmq interface so it will be able to receive push notification. The implementation of the TEMPLATE PROVIDER on BItcoin Core is fundamental for the project. Reaching consensus to implement it will take a long time, so a dialogue with the Bitcoin Core developers will be established from day one.

Annex 1

MOST LIKELY FINAL STATE OF THE PROJECT

RepoDev

stratum v1 *

stratum v2 (workspace)

--- implementation of the STRATUM V2 (sub)protocols---
utils *
mining-protocol * features (header-only)
job-negotiation-protocol *
template-distribution-protocol *
common *
types* features (header-only)

--- C ABI ---
FFI
startumv2-lib features (all, mining-protocol, job-negotiation-protocol, template-protocol, job-distribution-protocol, header-only)

---- this is intended to be used in bitcoin core ----
template-provider

--- maybe ---
adaptor

--- questi importano e riesportano stratum v2 top level per i vari use case ---
utils *
Mining-device * features (header-only, standard, ...)
pool-service * features (header-only, negotiator, ...)
proxy * features (header-only, translations, negotiator, ...)
template-provider * features (...)

--- v1 <-> v2 translations
translator *

--- utility for devs ---
utils **

--- this repo contains several binary crates, examples as docker-compose files that implements the use cases and likely several scripts ---

RepoOperators

hardware ***

--- the proxy can be a v1 v2 translator, redistribute the work at the downstream devices, negotiate with the pool, aggregate multiple channel in less
TCP connection ----

proxy

pool

* depends only on (getrandom, bytes, serde-derive, snow)
** to define during the development
*** I won't implement it

Annex 2

MOST LIKELY SESSION PERMITTED BY THE IMPLEMENTATION OF POOL SERVICE, MINING POOL
Everything between [] is something that can be 1 or more elements.
With session is intended a group with at maximum 1 proxy and 1 pool and at least 1 pool and 1 miner

pool <- group channel -> proxy <- [v1 channels] -> proxy <- [v1 channels] -> [mining devices]
pool <- group channel -> proxy <- [v1 channels] -> proxy <- [standard channels] -> [mining devices]
pool <- group channel -> proxy <- [v1 channels] -> [mining devices]
pool <- group channel -> proxy <- [standard channels] -> [mining devices]
pool <- group channel -> proxy <- [v1 channels] -> [mining devices]
pool <- [extended channels] -> proxy <- [standard channels] -> [mining devices]
pool <- [extended channels] -> proxy <- [v1 channels] -> [mining devices]
pool <- [standard channels] -> [mining devices]
pool <- [extended channels] -> negotiator (proxy) <- [standard channels] -> [mining devices]
pool <- [extended channels] -> negotiator (proxy) <- [v1 channels] -> [mining devices]