Mining Overview

James Hilliard

Stratum Mining Protocol

- Originally Created by Slush to replace getwork protocol
- Based on Electrum wallet sync protocol
- Extensible “\n” terminated line based JSON-RPC over TCP
- Allows miners to locally generate work by rolling the extranonce
- Virtually unlimited hashpower can mine on a single stratum connection
- Spec is largely defined by implementations
- Stratum servers have to handle clients that don’t always comply with the same spec
- Cgminer is the de facto client reference implementation

Stratum Overview

- Stratum Documentation
  - https://bitcointalk.org/index.php?topic=557866.0
  - https://slushpool.com/help/manual/stratum-protocol
  -
- Stratum examples taken from ck’s documentation

Stratum Subscription

On the beginning of the session, client subscribes current connection for receiving mining jobs:
{"id": 1, "method": "mining.subscribe", "params": []}\n
{"id": 1, "result": [[["mining.set_difficulty", "b4b6693b72a50c7116db18d6497cac52"], ["mining.notify", "ae6812eb4cd7735a302a8a9dd95cf71f"]], "08000002", 4], "error": null}\n
The result contains three items:

1. Subscriptions details - 2-tuple with name of subscribed notification and subscription ID.
2. Extranonce1 - Hex-encoded, per-connection unique string which will be used for coinbase serialization later.
3. Extranonce2_size - Represents expected length of extranonce2 which will be generated by the miner.

Stratum Authorize

After the miner is subscribed it must authorize itself:
{"params": ["slush.miner1", "password"], "id": 2, "method": "mining.authorize"}\n
{"error": null, "id": 2, "result": true}\n

- The result indicates if the miner successfully authorized
- Passwords are sent plain text
- Password is optional(most pools will accept any password)

Stratum Notify

Server start sending notifications with mining jobs:

{"params": ["bf", "4d16b6f85af6e2198f44ae2a6de67f78487ae5611b77c6c0440b921e00000000",
"01000000010000000000000000000000000000000000000000000000000000000000000000000000ffffffff20020862062f503253482f04b8864e5008",
"072f736c7573682f000000000100f2052a010000001976a914d23fcdf86f7e756a64a7a9688ef9903327048ed988ac00000000", [],
"00000002", "1c2ac4af", "504e86b9", false], "id": null, "method": "mining.notify"}

This contains:

1. job_id - ID of the job. Use this ID while submitting share generated from this job.
2. prevhash - Hash of previous block.
3. coinb1 - Initial part of coinbase transaction.
4. coinb2 - Final part of coinbase transaction.
5. merkle_branch - List of hashes, will be used for calculation of merkle root. version - Bitcoin block version.
6. nbits - Encoded current network difficulty
7. ntime - Current ntime/
8. clean_jobs - When true, server indicates that submitting shares from previous jobs will be stale and such shares will be rejected.

Stratum Submit

When miner find the job which meets requested difficulty, it can submit share to the server:

{"params": ["slush.miner1", "bf", "00000001", "504e86ed", "b2957c02"], "id": 4, "method": "mining.submit"}
{"error": null, "id": 4, "result": true}

Coinbase Transaction Reconstruction

To produce the coinbase, we just concatenate Coinb1 + Extranonce1 + Extranonce2 + Coinb2 together.
We can use the following to produce the double-sha256 hash of given coinbase

```
import hashlib
import binascii
coinbase_hash_bin = hashlib.sha256(hashlib.sha256(binascii.unhexlify(coinbase)).digest()).digest()
```

We can then generate the merkle root. We use merkle_branch from the notify and coinbase_hash_bin from previous snippet as an input:

```
import binascii

def build_merkle_root(self, merkle_branch, coinbase_hash_bin):
    merkle_root = coinbase_hash_bin
    for h in self.merkle_branch:
        merkle_root = doublesha(merkle_root + binascii.unhexlify(h))
    return binascii.hexlify(merkle_root)
```

Block Header Reconstruction

We have to put all together to produce block header for hashing:
version + prevhash + merkle_root + ntime + nbits + '00000000' + '000000800000000000000000000000000000000000000000000000000000000000000000000000000000000080020000'

Stratum Servers

- Generates stratum templates for miners using getblocktemplate from bitcoind
- Responsible for reconstructing blocks from shares above network diff and submitting to bitcoind
- Validates and records share records
- Responsible for serializing generation transaction
- Serializes witness commitment for generation transaction

Stratum Clients

- Cgminer - forks of cgminer power the majority of the Bitcoin mining network, requires an ASIC miner https://github.com/ckolivas/cgminer
- BFGMiner - fork of cgminer maintained by luke-jr which still supports GPU mining, has basic stratum proxy functionality https://github.com/luke-jr/bfgminer
- Slush stratum proxy - getwork bridge for miners that don't support stratum protocol https://github.com/slush0/stratum-mining-proxy
- Ckpool proxy - connection combining stratum proxy, functions as both a stratum client and server https://bitbucket.org/ckolivas/ckpool

Common Pool Software

- Original stratum server reference implementation written by Slush in Python
  https://github.com/slush0/stratum-mining
- Eloipool stratum server written in Python by luke-jr
  https://github.com/luke-jr/eloipool
- Ckpool stratum server written in c by Con Kolivas
  https://bitbucket.org/ckolivas/ckpool
- NOMP stratum server written in javascript by Matthew Little
  https://github.com/zone117x/node-open-mining-portal
- BTCPool stratum server written in c++ by Kevin Pan
  https://github.com/btccom/btcpool

Payout Methods

- PPLNS - Pay Per Last N Share
- PPS - Pay Per Share
- PROP - Proportional Payout
- Score Based(Slush)
- SPLNS - Score Per Last N Shares(ckpool)
- CPPSRB - Capped PPS with Recent Backpay(eligius)
- P2pool - PPLNS Sharechain with 0.5% block finder bonus

Pool Attack Vectors

- Block Withholding Attack
  - Very difficult to detect intentional attacks
  - Costly to PPS pool operators
  - Costly to PPLNS miners
  - Can not be fully mitigated without a hard fork
  - Can happen accidentally due to software/hardware bugs
  - Accidental block withholding can often be detected faster than an intentional attack
  - Multiple pools have detected block withholding miners
- DDOS Attack
  - Common in early days of mining
  - Multiple mitigation methods available
  - Not all that common recently

Stratum Attack Vectors

- BGP Hijacking
  - stratum reconnect(“client.reconnect”) used to persist hijacked connections
  - Persistence mitigated by enforcing same domain rule for stratum reconnect
  - Stratum clients do not authenticate servers(TLS support partially exists but is unused)
  - Attack requires access to BGP and is detectable(typically only ISP employees have this access)
- DNS cache poisoning
  - Typically requires vulnerable DNS server
- Cloud host interception
  - Significant percentages of network hashpower mine on pools that use the same host