Summary

The Stratum protocol reduces client-server communications to levels that are usable with very low bandwidth, and reduces the strain on servers drastically.
The key concept behind Stratum based mining is "push" based work, where the server pushes work to the miner, and the miner is able to utilize that work until the next push, regardless of the hash rate.
This is done by allowing the miner to increase a counter in the coinbase transaction, and build a new merkleroot for the block header, which effectively means the miner generates new work continuously without contacting the server.
Additionally, this new protocol utilizes a single asynchronous socket connection for all communication, rather than opening/closing HTTP connections with each communication.

Building Local Work

The mining pool provides a miner with two pieces of information upon connecting. The first is ExtraNonce1. The second piece of information is ExtraNonce2_size. This is how many bytes should be used for the ExtraNonce2 counter. ExtraNonce2 is a hexadecimal counter, and should be padded to fill up the number of bytes identified as the ExtraNonce2_size.

When the pool pushes work, it provides the coinbase transaction in two pieces. When the miner is building the block header to hash, it builds a merkleroot by creating a coinbase transaction. This is done by combining: Coinbase1 + ExtraNonce1 + ExtraNonce2 (padded) + Coinbase2.

This transaction is then passed through a double SHA256 to get the coinbase tx hash. You then combine that with first merkle branch provided by the work (if any), and double SHA256 the combined string. You then take that result, and do the same with the next merkle branch, repeating this process until all merkle branch hashes have been combined with previous results.

This process creates a unique merkle root for the new block header, which can then be pushed through the miner using the standard 32-bit nonce counter.

Initial Connection Communication

```
Client: {"id": 1, "method": "mining.subscribe", "params": []}\n
Server:  {"id": 1, "result": [["mining.notify", "ae6812eb4cd7735a302a8a9dd95cf71f"], "08000002", 4], "error": null}\n
```

Connection uses 'mining.subscribe' to subscribe to work from the server.
Server result response contains:
result[0] = "mining.notify", "ae6812eb4cd7735a302a8a9dd95cf71f" - Unique string used for the subscription
result[1] = ExtraNonce1, used for building the coinbase.
result[2] = Extranonce2_size, the number of bytes that the miner users for its ExtraNonce2 counter

Server Work Communication

```
Server: {"params": ["bf", "4d16b6f85af6e2198f44ae2a6de67f78487ae5611b77c6c0440b921e00000000",
"01000000010000000000000000000000000000000000000000000000000000000000000000ffffffff20020862062f503253482f04b8864e5008",
"072f736c7573682f000000000100f2052a010000001976a914d23fcdf86f7e756a64a7a9688ef9903327048ed988ac00000000",
["c5bd77249e27c2d3a3602dd35c3364a7983900b64a34644d03b930bfdb19c0e5","049b4e78e2d0b24f7c6a2856aa7b41811ed961ee52ae75527df9e80043fd2f12"],
"00000002", "1c2ac4af", "504e86b9", false], "id": null, "method": "mining.notify"}\n
```

This is sent at regular intervals by the server. It should be sent almost immediately after an acknowledge subscription.
Work pushes contain:
params[0] = Job ID. This is included when miners submit a results so work can be matched with proper transactions.
params[1] = Hash of previous block. Used to build the header.
params[2] = Coinbase (part 1). The miner inserts ExtraNonce1 and ExtraNonce2 after this section of the coinbase.
params[3] = Coinbase (part 2). The miner appends this after the first part of the coinbase and the two ExtraNonce values.
params[4][] = List of merkle branches. The coinbase transaction is hashed against the merkle branches to build the final merkle root.
params[5] = Bitcoin block version, used in the block header.
params[6] = nBit, the encoded network difficulty. Used in the block header.
params[7] = nTime, the current time. nTime rolling should be supported, but should not increase faster than actual time.
params[8] = Clean Jobs. If true, miners should abort their current work and immediately use the new job. If false, they can still use the current job, but should move to the new one after exhausting the current nonce range.

Worker Authorization

```
Client: {"params": ["eleuthria_miner1", "password"], "id": 2, "method": "mining.authorize"}\n
Server: {"error": null, "id": 2, "result": true}\n
```

Client authorizes a worker using the method "mining.authorize". Client request contains:
params[0] = Worker Name
params[1] = Worker Password (can be "" or omitted if server does not require passwords)

Server response is "result": true for successful authorization, "result": false for unsuccessful authorization.

Work Submissions

```
Client: {"params": ["eleuthria_miner1", "bf", "00000001", "504e86ed", "b2957c02"], "id": 4, "method": "mining.submit"}\n
Server: {"error": null, "id": 4, "result": true}\n
```

Miners submit shares using the method "mining.submit".
Client submissions contain:
params[0] = Worker Name
params[1] = Job ID
params[2] = ExtraNonce 2
params[3] = nTime
params[4] = nonce
Server response is result: true for accepted, false for rejected.

Difficulty Adjustment

```
Server: { "id": null, "method": "mining.set_difficulty", "params": [2]}\n
```

The server can adjust the difficulty required for miner shares with the "mining.set_difficulty" method.
The miner should begin enforcing the new difficulty on the next job received. Some pools may force a new job out when set_difficulty is sent, using clean_jobs to force the miner to begin using the new difficulty immediately.

Cheat Sheet

```
Methods:
mining.subscribe     : Used to subscribe to work from a server, required before all other communication.
mining.authorize     : Used to authorize a worker, required before any shares can be submitted.
mining.notify        : Used to push new work to the miner.  Previous work should be aborted if Clean Jobs = true!
mining.submit        : Used to submit shares
mining.set_difficulty: Used to signal the miner to start submitting shares under a new difficulty.
```