Sidecoin: a Snapshot Mechanism for Bootstrapping a Blockchain

Joseph Krug & Jack Peterson
www.sidecoin.net

Sidecoin is a mechanism that allows a snapshot to be taken of Bitcoin's blockchain. We compile a list of Bitcoin's unspent transaction outputs, then use these outputs and their corresponding balances to bootstrap a new blockchain. This allows the preservation of Bitcoin's economic state in the context of a new blockchain, which may provide new features and technical innovations.

I. SNAPSHOTS AND THEIR USES

A snapshot records all public addresses in the Bitcoin network with a substantial balance (over about 20 cents in USD) and sorts them from highest to lowest balance. This eliminates millions of addresses from places like Satoshi Dice that have small amounts of bitcoin in them, called 'dust'. We then save these addresses, along with their balances, and an alternative representation of the address known as a hash160.[1]

Snapshots allow us to do a few things. They can be used to examine addresses and balances to find high value holders, or to see the distribution of wealth among Bitcoiners: the economic state of the Bitcoin network. However, a much more fun use is to create altcoins – or, as coins created using this method are called, *spinoff* coins.[2] To create a spinoff coin, we first take a snapshot of Bitcoin's blockchain, then create an altcoin with an initial distribution based on Bitcoin's. *Sidecoin* is the reference implementation for bootstrapping a new blockchain using Bitcoin's present state. This technology allows bitcoin owners to trustlessly claim a proportional amount of a new cryptocurrency.

Imagine a coin development team like Gridcoin's[3]. Gridcoin uses a proof-of-work that, in principle, benefits society through computation, by directing it to to fold proteins for cancer research. Their team would probably be pleased if a tiny bit of the enormous mining power used to mine bitcoins could be redirected to their *useful* proof-of-work system. It provides a principled, transparent distribution method with which 'pre-mined' cryptocurrencies can disseminate their tokens. One example is Ripple, which does not use mining at all. Taking a snapshot, and allowing Bitcoiners to claim coins on the Gridcoin or Ripple networks, removes the profit gained by releasing new cryptocurrencies which do not contribute new technology. Coin developers can still profit by allowing mining after the initial claiming, but only by providing a true improvement to Bitcoin or other altcoins.

A snapshot-based distribution bootstraps an altcoin with a much larger user base than they otherwise would have had: any bitcoin owner is automatically 'bought-in' to the altcoin, provided they decide to claim their coins. Bitcoin holders who think an altcoin does not provide sufficient technological advancements over Bitcoin can simply claim their new coins, then sell them. Altcoin creators who believe that their coins hold value due to useful features are incentivized to buy more after the initial sharp selloff, because they will profit once the market realizes the value of their innovations.

II. TAKING A SNAPSHOT

There were several steps involved in creating our snapshot file. First, we downloaded the Bitcoin blockchain using `bitcoind`, then we parsed all the balances in the Bitcoin blockchain, along with their corresponding public keys.[4] Next, we converted these hash160s into the commonly used Bitcoin address: the base-58 strings seen everywhere, on `blockchain.info`, in your wallet, etc. We then used a shell script to manipulate the text to sort these addresses, hash160s, and their corresponding balances into a tab-delimited file. An altcoin developer could simply use this file, embed it in the first non-genesis block, then allow users to claim their coins through our coin verification process.

To create the snapshot file, we use Znort's blockchain parser[5], modified to use memory-lean hash tables. We then use a shell script[6] to call the appropriate functions on the block parser. We specify the maximum block number as well as the number of addresses to include in the snapshot; a value of about 2,000,000 includes all addresses with balances[7] above about $0.20. Next, the hash160s are converted into addresses[8], which is also written to the snapshot file. After parsing, a `snapshotToImport.txt` file is output containing the balance, hash160, and address. An example entry is:

---

[1] 'hash160' is shorthand for the 160-bit hash created by SHA256 followed by RIPEMD160.
[2] `http://bitcointalk.org/index.php?topic=563972.0`
[3] `http://www.gridcoin.us`
[4] More precisely, the hash160, which is derived from the public key.
[5] `https://github.com/znort987/blockparser`
[6] `https://github.com/AugurProject/SideCoin/blob/master/snapshot`
[7] The balances are given in Satoshis.
[8] `http://bitcoin.stackexchange.com/questions/5021/how-do-you-get-the-op-hash160-value-from-a-bitcoin-address`

```
180893019187.00000000 8c1d15231afa4868330f8af694ba637b69fdc2d7 14CKu2rJN2f8
    fdPrtmRLWChhXESgN5qaA7
```

The snapshot file has as many rows as the number of addresses we specified to the blockparser.

III. INCORPORATING A SNAPSHOT INTO A BLOCKCHAIN'S INITIAL STATE

Once the snapshot file has been created, it must to be recorded on the new blockchain. We incorporate the snapshot data as a series of unspent transaction outputs into block one of the new blockchain. Then, we add a claim transaction function and make it available via RPC call. This allows bitcoin owners to trustlessly claim their coins in the spinoff blockchain. Finally, we discuss how to create your own snapshot and incorporate it in a spinoff.

First, we mine the genesis block. Next, we incorporate the snapshot data into the first block.[9] We begin by creating coinbase transactions[10] for each of the addresses in the snapshot:

```
Input:
   <pubKeyHash from snapshot>
Output:
   Value: <Bitcoin address's balance>
   scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

where the Bitcoin address's balance is the balance recorded in the snapshot. After these transactions are loaded into block one, we find the block's hash, process the block (i.e., the same thing a miner does upon finding a new block), then submit it to the network.

Finally, we provide a trustless method of claiming these transactions. Claim transactions give our users who own bitcoins a way of claiming their portion of sidecoins. To do this, the user must submit a *claim transaction*:

```
Input:
   <previous transaction ID>
   <index of the output being claimed> (should be 0)
   scriptSig: <signature> <pubKey>
```

where `<pubKey>` is the public key of the user's Bitcoin address. This input Script has successfully unlocked the new coins on the Sidecoin network! These coins are locked up in an address belonging to the user's Sidecoin wallet:

```
Output:
   Value: <Bitcoin address's balance>
   scriptPubKey: OP_DUP OP_HASH160 <Sidecoin pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

To facilitate submitting these claim transactions, we created a new RPC command, `claimtx`. Users call this by inputting `claimtx` followed by the user's Bitcoin address:

```
claimtx "1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX"
```

The *claimtx* function (in `rpcserver.cpp`) searches through the snapshot file to find the address's corresponding hash160 and unspent output in block one. Then, it automatically builds the claim transaction (described above), locking the coins in an output belonging to a new address in the user's Sidecoin wallet:

```
signrawtransaction
'0100000001ec70650bf05a75d62f3bcf83d183064e118d39070c2a7237d33ee9a4d4930da20000000000
    ffffffff01606b042a010000001976a914345cd34789f945f0cbd952ce254bf5246e63be0c88ac00000000'
     '[{"txid":"a20d93d4a4e93ed337722a0c07398d114e0683d183cf3b2fd6755af00b6570ec","vout":0,
    "scriptPubKey":"76a914f4dfe70369cc98dc8113bba69bda324e4f1386a088ac"}]'
```

Next, the user is given instructions to open the RPC console of their Bitcoin client containing the address which held the bitcoins the user is claiming at the time the snapshot was taken. The user pastes in the output supplied by the `claimtx`, which is the unsigned transaction appended to the RPC command `signrawtransaction`. The user's bitcoin client will then sign this transaction.

Finally, the user takes the results, appends them to the command `sendrawtransaction`, and submits it using Sidecoin's RPC console:

---

[9] The genesis block is loaded from memory every time `bitcoind` is started. However, aside from the genesis block, only the last one-hundred blocks are loaded into memory. Since the snapshot block is so large, we do not want it loaded into memory every time the coin daemon is called!

[10] `http://bitcoin.org/en/developer-guide#transaction-data`

```
sendrawtransaction "01000000017b1eabe0209b1fe794124575ef807057c77ada213\
                    8ae4fa8d6c4de0398a14f3f0000000049483045022100894 9f0\
                    cb400094ad2b5eb399d59d01c14d73d8fe6e96df1a7150deb38\
                    8ab8935022079656090d7f6bac4c9a94e0aad311a4268e082a7\
                    25f8aeae0573fb12ff866a5f01ffffffff01f0ca052a0100000\
                    01976a914cbc20a7664f2f69e5355aa427045bc15e7c6c77288\
                    ac00000000"
```

The user has now unlocked their sidecoins! This transaction is like any other transaction: it just spends unspent outputs. The key difference is that it involves data from two different blockchains! We do the claim transaction in this manner to trustlessly verify that the user owns the Bitcoin private key associated with the address to which the sidecoins have been assigned.

In order to create a snapshot for use on one's own computer, one simply runs the snapshot script which asks the user a few questions about which block they'd like to take the snapshot in and how many addresses they'd like in the snapshot. After this, take the `snapshotToImport.txt` file and paste it in the *Appdata/Sidecoin/balances* folder on your respective OS. Place a `sidecoin.conf` file there, as well. Open the `snapshot.h` file in Sidecoin's source folder and set `GENESIS_SWITCH` to true. Then, launch Sidecoin, and it will mine the mainnet, testnet, and regtest genesis blocks. (Upon completion, the block hashes and Merkle root fields in `chainparams.cpp` need to be updated.)

After the genesis block is mined, launch Sidecoin again and it will load the transactions into block one from the snapshot file, then it will add block one to the blockchain. If it outputs `accepted` everything has been done correctly! Next, change `GENESIS_SWITCH` to false. Finally, relaunch Sidecoin to see a new successful spinoff implementation.

For users who want to use an already created spinoff coin, one can simply wait to download block one from other peers. However, when the network is first starting off this is very slow and impractical due to the large size of block one. Instead, one can try to load block one in on their own by enabling the `GENESIS_SWITCH` and editing the `blockOne.nNonce` and `nTime` to the values that they were mined with if not already set. Then place the `snapshotToImport.txt` file in the *balances* directory, start Sidecoin, wait for it to load the block, then disable the `GENESIS_SWITCH`.[11]

IV. PROOF-OF-CONCEPT

Sidecoin[12] is our proof-of-concept implementation of a snapshot/spinoff mechanism. It is a fork of Bitcoin Core 0.9.1. It does not support pay-to-script-hash addresses in the snapshot, nor does it support simplified claim verification schemes, as proposed on Bitcointalk.[13]

We made a few changes to our fork of Bitcoin Core to make Sidecoin. We had to change a couple things in the `CheckBlock` code segments: allowing multiple coinbase transactions, as well as modifying block size limits, signature operations limits, and transaction limits. We only made these modifications for block one, which is determined as the block with a previous block hash equivalent to the genesis block. We also added functions allowing Bitcoin to read in snapshot data from a flat file into block one, the `claimtx` function, and manually loading block one into the blockchain, similar to how the genesis block is built.

Finally, we recognize that most of the utility of snapshots and spinoffs will ultimately be replaced by the interoperability provided by Sidechains.[14] However, for those looking to make an altcoin and dramatically boost community adoption, spinoffs could be a solution. Another possible application of the snapshot is using it to bootstrap a replacement for Bitcoin in the unlikely event something catastrophic occurs.

ACKNOWLEDGMENTS

Financial support for this project was provided by Joe Costello.

---

[11] The text file is needed for anyone looking to claim a transaction anyway, so this a nice method. An alternative method is to paste the blocks and chainstate folders from a peer who has a blockchain with block one in it into one's Sidecoin app data folder. Block one is stored as a checkpoint, so there is not much concern of someone falsifying the snapshot and giving a faulty block one.

[12] `https://github.com/AugurProject/SideCoin`

[13] `http://bitcointalk.org/index.php?topic=563972.0`

[14] `http://www.blockstream.com/sidechains.pdf`