```
BIP: 42
Layer: Consensus (soft fork)
Title: A finite monetary supply for Bitcoin
Author: Pieter Wuille <pieter.wuille@gmail.com>
Comments-Summary: Unanimously Recommended for implementation
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0042
Status: Final
Type: Standards Track
Created: 2014-04-01
License: PD
```

Abstract

Although it is widely believed that Satoshi was an inflation-hating goldbug he never said this, and in fact programmed Bitcoin's money supply to grow indefinitely, forever. He modeled the monetary supply as 4 gold mines being discovered per mibillenium (1024 years), with equal intervals between them, each one being depleted over the course of 140 years.

This poses obvious problems, however. Prominent among them is the discussion on what to call 1 billion Bitcoin, which symbol color to use for it, and when wallet clients should switch to it by default.

To combat this, this document proposes a controversial change: making Bitcoin's monetary supply finite.

Details

As is well known, Satoshi was a master programmer whose knowledge of C++ was surpassed only by his knowledge of Japanese culture. The code below:

```
   int64_t nSubsidy = 50 * COIN;
// Subsidy is cut in half every 210,000 blocks
// which will occur approximately every 4 years.
nSubsidy >>= (nHeight / 210000);
```

is carefully written to rely on undefined behaviour in the C++ specification - perhaps so it can be hardware accelerated in future.

The block number is divided by 210000 (the "apparent" subsidy halving interval in blocks), and the result is used as input for a binary shift, applied to the original payout (50 BTC), expressed in base units. Thanks to the new-goldmine interval being exactly 64 times the halving interval, and 64 being the size in bits of the currency datatype, the cycle repeats itself every 64 halvings on all currently supported platforms.

Despite the nice showoff of underhanded programming skills - we want Bitcoin to be well-specified. Otherwise, we're clearly in for a bumpy ride:

Note that several other programming languages do not exhibit this behaviour, making new implementations likely to be slower and generally more bogus than Bitcoin Core. For example, Python unexpectedly returns 0 when shifting an integer beyond its size.

Other solutions

Floating-point approximation

An obvious solution would be to reimplement the shape of the subsidy curve using floating-point approximations, such as simulated annealing or quantitative easing, which have already proven their worth in consensus systems. Unfortunately, since the financial crisis everyone considers numbers with decimal points in them fishy, and integers are not well supported by Javascript.

Truncation

An alternative solution would be to represent the total number of bitcoins as a string:

```
"2100000000000000000000"
```

and then use string manipulation to remove the rightmost zero every 4 years, give or take a leap-year:

```
strSubsidy = strSubsidy.substr(0, strSubsidy.size() - 2);
```

This style relies less heavily on clever C++ and is more familiar to the Core Dev Team who are primarily PHP programmers.

Proposal

Instead, how about we stop thinking about long term issues when we'll all be dead (barring near lightspeed travel, cryogenic revival, or other technology— like cryptocurrency— which only exists in science fiction).

A softfork (see BIP16, BIP34, BIP62) will take place on april 1st 2214, permanently setting the subsidy to zero. The result of this will be that the total currency supply will be limited to 42 halfmillion (including the genesis coinbase output, which is not actually spendable).

Implementation

An implementation for the reference client can be found on https://github.com/bitcoin/bitcoin/pull/3842 .

Compatibility

Given the moderate time frame over which this change is to be implemented, we expect all miners to choose to screw themselves and deploy this change before

2214.

If they don't, and a minority remains on the old code base, a fork may occur. Essentially, they'll be mining fool's gold after that time.

Acknowledgements

Thanks to Gregory Maxwell for proposing this solution, and to Mike Hearn for insights into web development. Also thanks to "ditto-b" on github to implement a prototype ahead of time.

Copyright