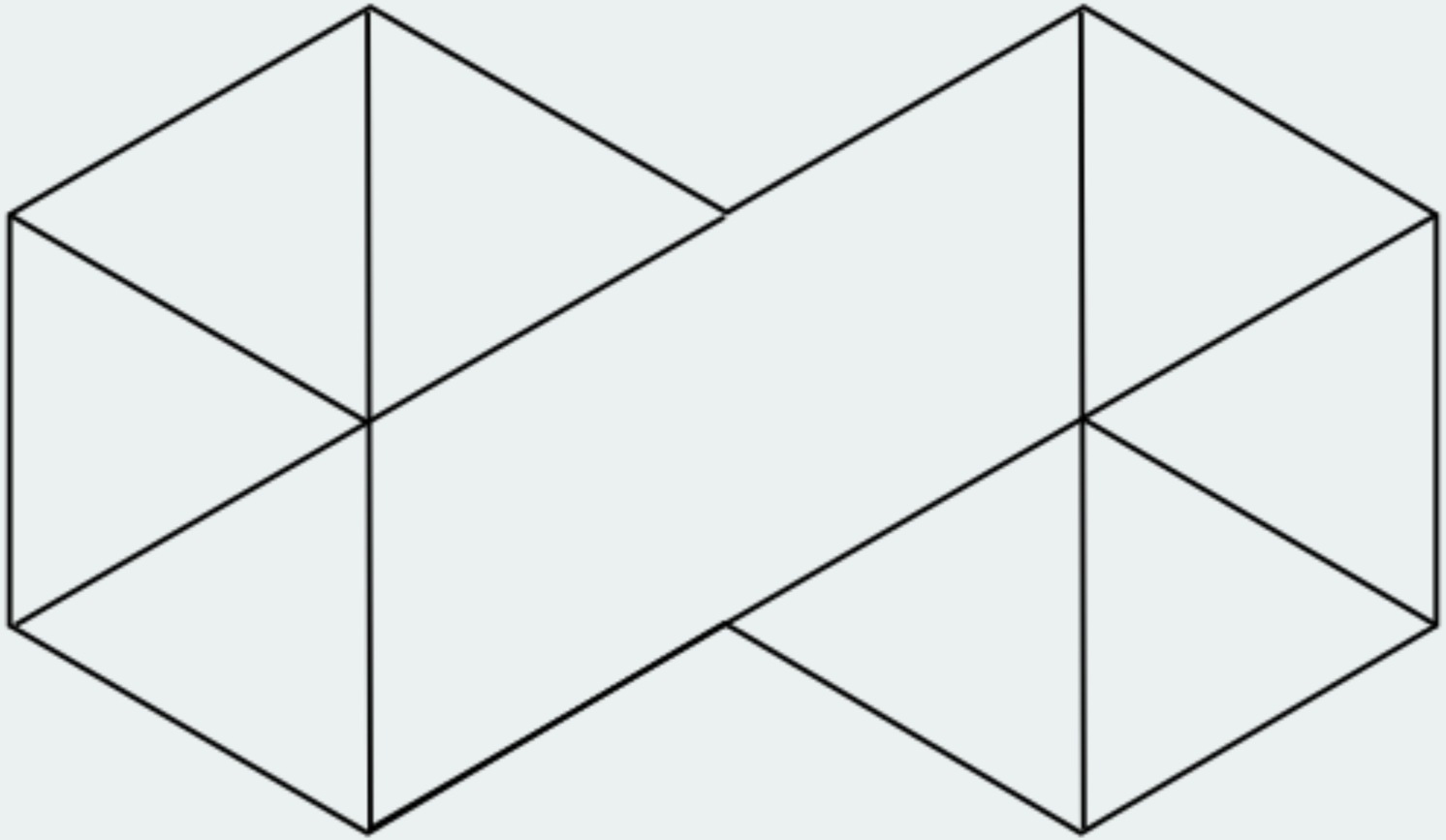
ethereum

vienna

Byzantium Release
(Metropolis Part 1)



RIAT

RESEARCH INSTITUTE FOR ARTS AND TECHNOLOGY

Agenda

General Introduction to Ethereum

Metropolis: Byzantium and Constantinople

Socialising

Organisational Updates

Events

To accommodate both the very and not-so technical crowd

future meetups will have either a **TECH** or a **USER** tag

- **USER** meetups will be more high level while
- **TECH** meetups will be significantly more deep

This separation will probably begin in January

ethereum

Byzantium Release

(Metropolis Part 1)

Upgrade Process

EIPs

Changes to Ethereum happen through **Ethereum Improvement Proposals**

Those are proposals in a standardised format

They go through several phases: from **draft** to **final**

EIPs are usually publicly discussed for many months before getting accepted

Everything happens on **https://github.com/ethereum/EIPs**

Difficulty bomb

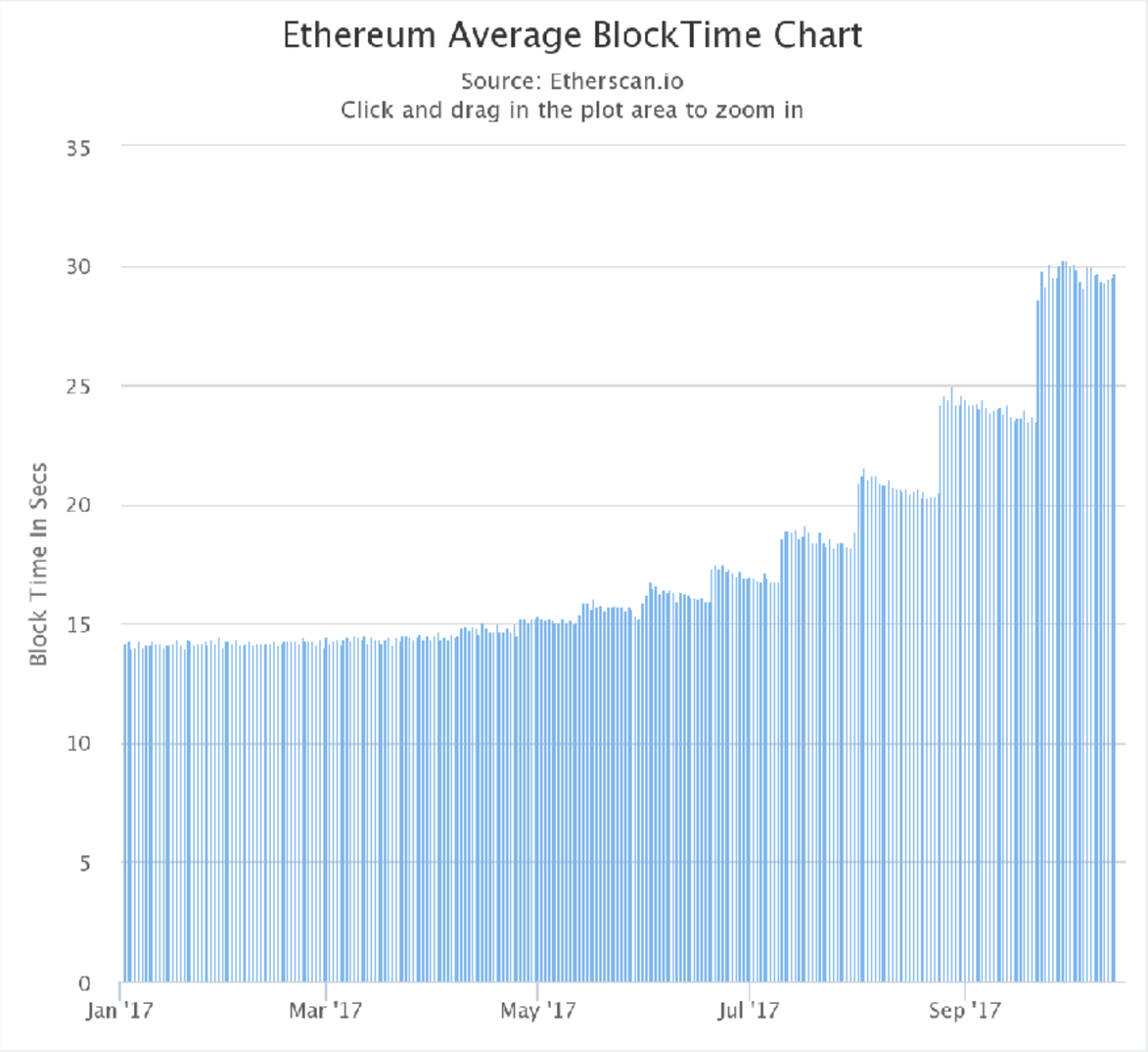
Built in mechanism to force a hardfork

Originally meant for the PoS transition

Blocktime reached ~30s in September

Would have increased exponentially

Now postponed by 1.4 years



Hardfork

Eventually a set of EIPs is collected into a **Hardfork**

Once testing has been a success for a while

1. A block number is set for the fork
2. Updated client versions with the block number are released
3. People need to update their clients
4. New rules go in effect at the set block number

Metropolis

Metropolis

Metropolis was meant to be the third release phase of Ethereum

Many unexpected things happened since the last regular release (**Homestead**)

- TheDAO
- DoS attacks and two rescue Hardforks

Plans for Metropolis have changed significantly

Nothing in common with the original milestone other than name

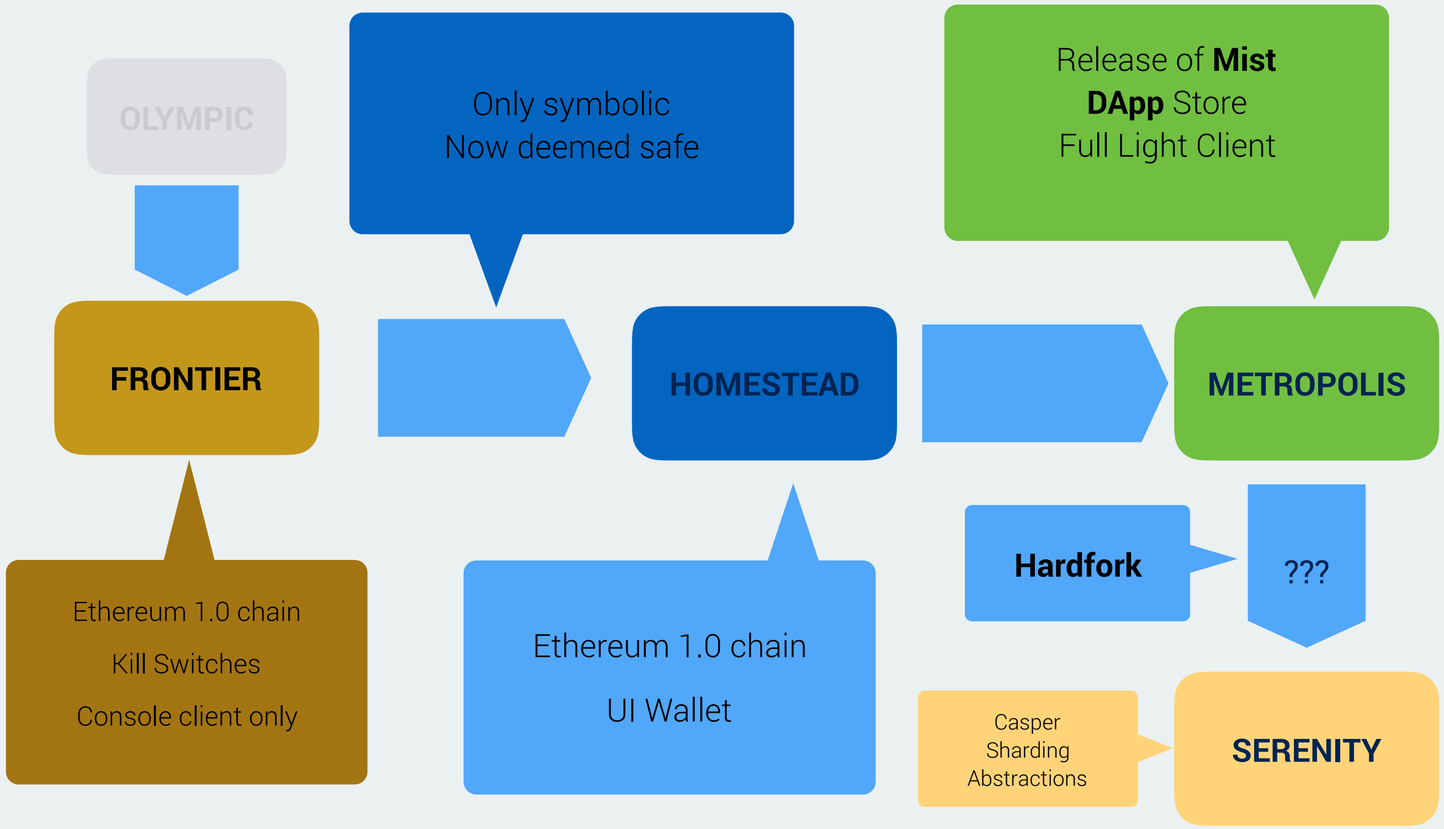
The original Metropolis

The original roadmap looked very different

Metropolis was supposed to be

- DApp Store
- Release of Mist
- Light client

Scheduled for Summer 2015



The original Metropolis

DApp Store

- no ongoing effort (effectively abandoned)

Release of Mist

- not pretty but can do most of what was promised
- can fetch pages via swarm, resolve through ens, provides web3 api

Light client

- also exists, but has connectivity issues (to be fixed soon)

A tale of two forks

To not push the release back any further **Metropolis** was split into two forks

Byzantium (already happened)

- delays the difficulty bomb
- implements most EIPs

Constantinople (expected in early 2018)

- implements the most complex EIPs
- is the beginning of the "abstraction" updates

Byzantium

Byzantium

Happened earlier today at 7:22 am at block #4370000

```
2017-10-16 05:22:44 UTC Imported #4370000 b1fc…785e (97 txs, 6.61 Mgas, 1281.98 ms, 16.47 KiB)
```

So far everything is fine

geth and parity in sync

old chain at #4369999

Byzantium

Byzantium implements 9 EIPs in total:

EIP 649 delays the difficulty bomb by 1.5 years and changes block rewards

EIP 100 fixes some other difficulty related issues

EIP 196, 197 and 198 add various precompiles for advanced elliptic curve crypto

EIP 214, 211 and **140** add various opcodes

EIP 658 changes the transaction receipt

EIP 649

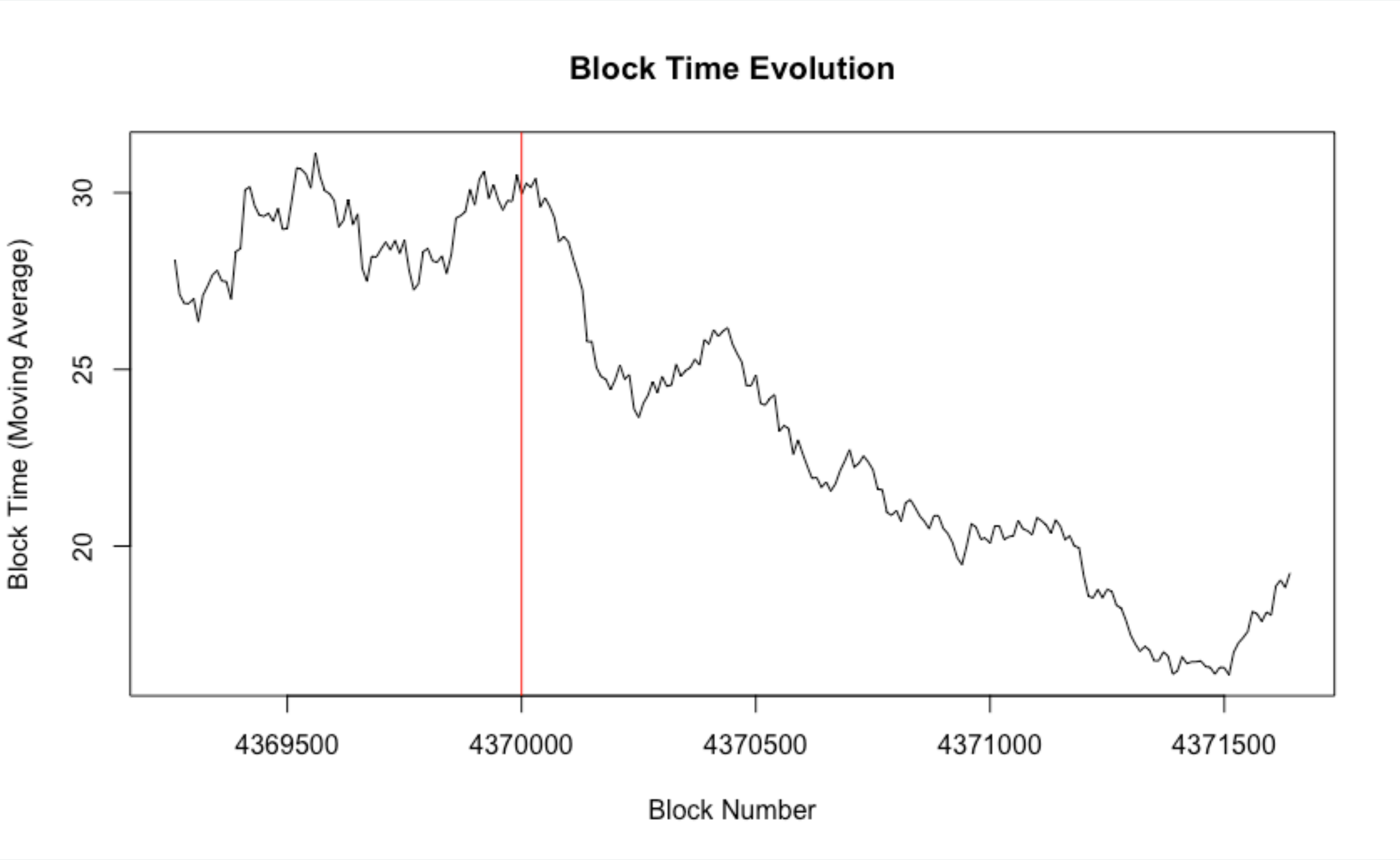
The effect of the difficulty bomb depends on the **block number**

This EIP replaces this with a **fake block number**

- which is 3,000,000 blocks behind
- this equals about 1.42 years

This means block time will now return to 15s

At the end of 2018 it will be back where it was now


Block Time Evolution
Block Time (Moving Average)
30
25
20
4369500
4370000
4370500
4371000
4371500
Block Number

EIP 649

It also reduces the block reward from **5 ETH** to **3 ETH** per block

Uncle rewards are also adjusted

This is actually a higher reward than immediately before the fork

Before: 5 ETH / 30s ~ 0,16 ETH/s

After: 3 ETH / 15s ~ 0,2 ETH/s

EIP 100

This EIP changes the other part **difficulty formula**

The new algorithm targets a constant rate of blocks **including** uncles

This is to avoid possible manipulations of the issuance rate by producing uncles

max(1 - ((timestamp - parent.timestamp) // 10), -99)

changed to

max((2 if len(parent.uncles) else 1) - ((timestamp - parent.timestamp) // 9), -99)

EIP 100

Before

Block Time	Adjustment
<10s	+1/2048
<20s	0
<30s	-1/2048
<40s	-2/2048
…	,,,

After

Block Time	Adjustment if no uncles	Adjustment if uncles
<9s	+1/2048	+2/2048
<18s	0	+1/2048
<27s	-1/2048	0
<36s	-2/2048	-1/2048
…	,,,	,,,

EIP 196-198

EIP 196 adds precompiles for

- addition of points of the **alt_bn128** curve
- multiplication of points of the **alt_bn128** curve with a scalar

This is not the curve that Ethereum itself uses (**secp256k1**)!

EIP 197 adds a precompile for elliptic curve pairing operations

EIP 198 adds a precompile for big integer exponentiation

EIP 196 -198

Together they are supposed to enable advanced cryptographic primitives

Developed test contracts thus far include

- zcash-style **zk-snarks**
- monero-style **ring signatures**

Most of this is probably not going to be useful until **EIP 86** from **Constantinople**

EIP 214

EIP 214 introduces the new opcode **STATICCALL**

Allows the calling of other contracts while prohibiting state changes

Primarily a security feature

Could also be used for optimisations

EIP 211

EIP 211 introduces the opcodes **RETURNDATASIZE** and **RETURNDATACOPY**

Enables contracts to get the return data of a call afterwards

Solves the issues of

- contracts not being able to call dynamic length return value functions
- not being able to make general proxy contracts (e.g. for updatability)

EIP 140

EIP 140 introduces the new opcode **REVERT**

This reverts all state changes - but without going out of gas!

This means you will no longer overpay in case of errors!

e.g. sending ETH into an ICO after the cap has been reached

EIP 140

REVERT also works in concert with the **RETURNDATA*** opcodes

Together they will allow contracts to

- have proper error handling
- return an error message (no more useless **invalid opcode** message)

The tooling (solidity etc.) does not support this yet

EIP 658

EIP 658 changes a field in the **transaction receipts**

The intermediate state root ("**medstate**") has been removed

Instead there is a new **status** field

- 0 indicates success
- 1 indicates failure (revert or out-of-gas)

EIP 140 & 658 on etherscan

TxHash:	0x7147cd1a322954d9c0f3997b1f7c99c072f32608f4a32e6a8fd189d6adaecbd4
Block Height:	1880141 (9 block confirmations)
TimeStamp:	2 mins ago (Oct-16-2017 06:24:51 AM +UTC)
From:	0xaf586d5df634e014e083faec7b08de101594004b
To:	Contract 0x82576fa6b63b30197bf11eecc40d4aac44ff9da3
Value:	0 Ether ($0.00)
Gas Limit:	21625
Gas Used By Txn:	21624
Gas Price:	0.00000002 Ether (20 Gwei)
Actual Tx Cost/Fee:	0.00043248 Ether ($0.000000)
Cumulative Gas Used:	1021624
TxReceipt Status:	Fail
Nonce:	15

Looks like success

Not all gas used

But it failed

Constantinople

Constantinople

Constantinople implements 3 EIPs in total

EIP 86 implements account abstraction

EIP 96 moves block hashes into a contract

EIP 145 introduces new bit shifting opcodes

EIP 145

EIP 145 introduces new opcodes for bitwise shifting

- shift left
- logical shift right
- arithmetic shift right

Can make certain computations more efficient

EIP 96

EIP 96 introduces a special system contract keeping track of past block hashes

Simplifies the protocol by

- putting some of the block processing logic into a contract
- putting more of the state into the state trie (rather than as something extra)

Can enable efficient proofs about past properties of the chain

Can enable more efficient light clients

EIP 86

Implements the beginning of "account abstraction"

Anyone can send transactions from the (**null**) **address** 2^160 - 1

These transactions

- don't have a signature
- have a gasprice of 0 wei
- don't have a nonce
- have a value of 0 wei

EIP 86

These transactions should always go to a contract

This contract then takes care of

- checking the signature
- paying for the gas
- preventing replays

Miners have to start running transactions and check themselves wether they get paid

EIP 86

This EIP also changes the process of how contracts are assigned addresses

Now it depends on **sender** and **nonce**

This changes it to **sender**, **salt** and **init code**

Makes addresses more predictable

Also gives guarantees about the code of a contract, even after a reorganisation

EIP 86

This can enable many things like

- contracts being able to pay the fee
  - important for end users (don't need to hold ether)
  - important for **privacy** technologies like **zksnarks** or **ring signatures**
- better and cheaper **multisig** contracts
  - only one transaction instead of many
  - fee can be paid directly be the wallet

EIP 86

This can enable many things like

- **custom cryptography** instead of **secp256k1**
  - everybody can upgrade to quantum safe crypto at their own pace
  - or any other cryptographic primitive
- other non cryptographic things
  - transaction only becomes valid if some other property holds
  - parallelizable nonces

Schedule

planned for the first quarter of 2018

since everything in Ethereum is usually delayed

=> likely not before the next summer

at the very latest at the end of 2018 (=> difficulty bomb resurfaces)

very possible that more EIPs will be added by then

All informations about our events at

https://www.meetup.com/Ethereum-Vienna

All available slides and materials at

https://github.com/ethereum-vienna-meetup