





Learning blockchain basics with PHP

FrOSCon 2018 PHP Track, 25.08.2018

\$whoami

- Matthias Gutjahr, freelance software developer
- Co-orga of PHP User Group Rheinhausen (Mainz)
-  @mattsches
-  mattsches/phplockchain

\$whoami

- Matthias Gutjahr, freelance software developer
→ You can hire me ;-)
- Co-orga of PHP User Group Rheinhessen (Mainz)
-  @mattsches
-  mattsches/phplockchain

Agenda

- Short introduction to Blockchain in general
- Main technical concepts
 - Transactions
 - Blocks
 - Mining
 - Consensus
- Code examples in PHP
- Live demo
- Conclusion

Disclaimers

- For Educational Purposes
 - ➔ Many examples are simplified
 - ➔ Work In Progress
- If you have any questions, please ask!
- If you think I made a mistake, please tell me!
- Please leave feedback on joind.in! 🙏

Bitcoin / Blockchain

- Will be 10 years old in Jan 2019
- Won't go away quickly
- The amount of energy is insane
- But the technology has great potential
- Lack of killer application
- Alternatives addressing the problems
- We have to know what we're talking about!

Bitcoin

„Bitcoin (₿) is a **cryptocurrency**, a form of electronic cash. It is a **decentralized** digital currency without a central bank or single administrator.“ (Wikipedia)

„Bitcoin has been **criticized** for its use in **illegal transactions**, its **high electricity consumption**, price volatility, **thefts** from exchanges, and the possibility that bitcoin is an **economic bubble**. Bitcoin has also been used as an investment, although several regulatory agencies have issued investor alerts about bitcoin.“

Energy consumption

„Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa“

(<https://powercompare.co.uk/bitcoin/>)

„New study quantifies bitcoin's ludicrous energy consumption. Bitcoin could consume 7.7 gigawatts by the end of 2018.“

(<https://arstechnica.com/tech-policy/2018/05/new-study-quantifies-bitcoins-ludicrous-energy-consumption/>)

1.21 Gigawatts! Great Scott!!!



Why PHP?



?





libsodium
(paragonie/halite)



The Fantastic Four

Client

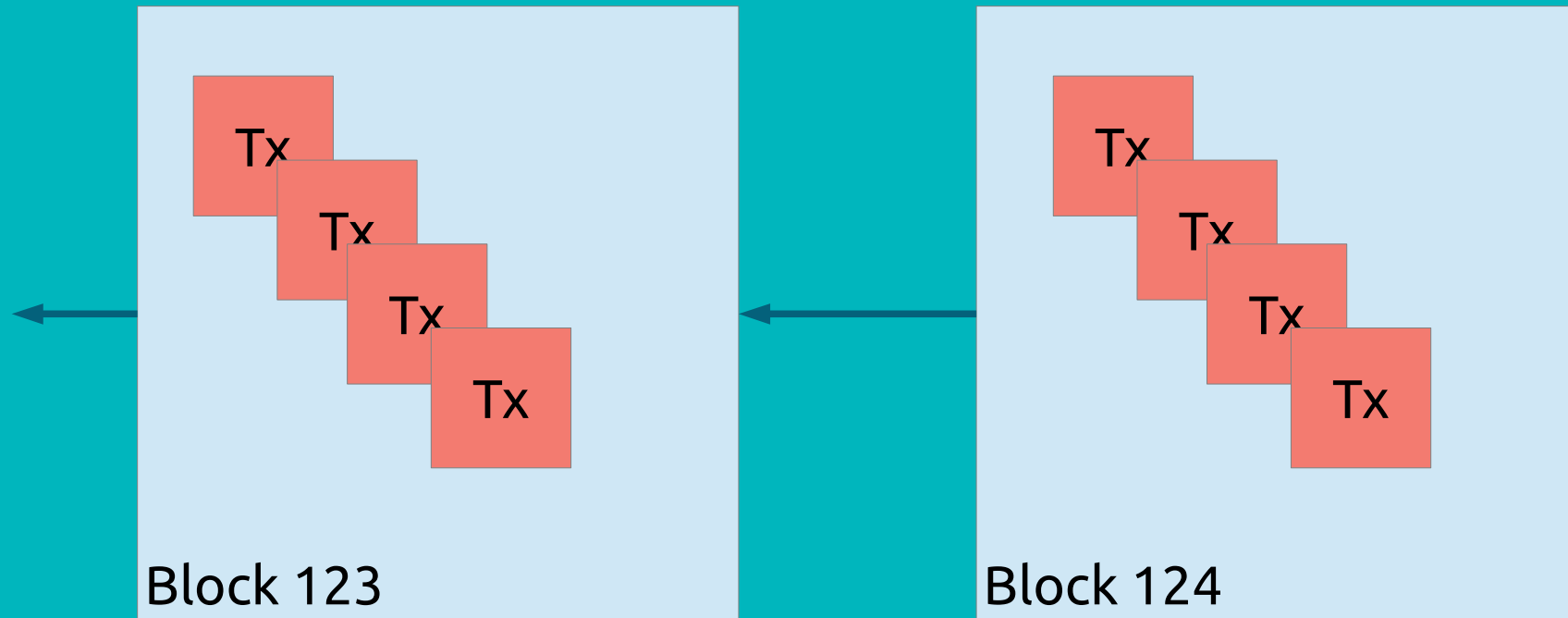
Block

Blockchain

Transaction

Client (Node, Wallet)

Blockchain



Let's see some code!



```
1 use GuzzleHttp\Client;
2 use Mattsches\Util;
3 use Psr\Http\Message\ServerRequestInterface;
4 use React\Http\Response;
5 use React\Http\Server;
6 use React\Socket\Server as SocketServer;
7
8 $loop = React\EventLoop\Factory::create();
9 $client = new Mattsches\InitialClient(Util::createSignatureKeypair(), new Client(), 4);
10 $server = new Server(
11     function (ServerRequestInterface $request) use ($client) {
12         switch ($request->getRequestTarget()) {
13             case '/mine': // mine a new block
14                 $block = $client->mine();
15                 $response = ['message' => 'New block forged', 'block' => $block];
16                 break;
17             // more routes ...
18             default:
19                 $response = ['message' => 'We\'re up and running!'];
20         }
21
22         return new Response(
23             200,
24             ['Content-Type' => 'application/json'],
25             json_encode($response, JSON_PRETTY_PRINT, 1000)
26         );
27     }
28 );
29 $socket = new SocketServer(5001, $loop);
30 $server->listen($socket);
31 $loop->run();
```

API

- /mine
 - ➔ Mine new block
- /transaction
 - ➔ Create new transaction
- /transaction/validate
 - ➔ Validate transaction by id
- /chain
 - ➔ Get blockchain
 - ➔ Validate blockchain
- /getblockchaininfo
 - ➔ Get length & difficulty
- /getblock
 - ➔ Get block by id
- /keys
 - ➔ Get public/private keys

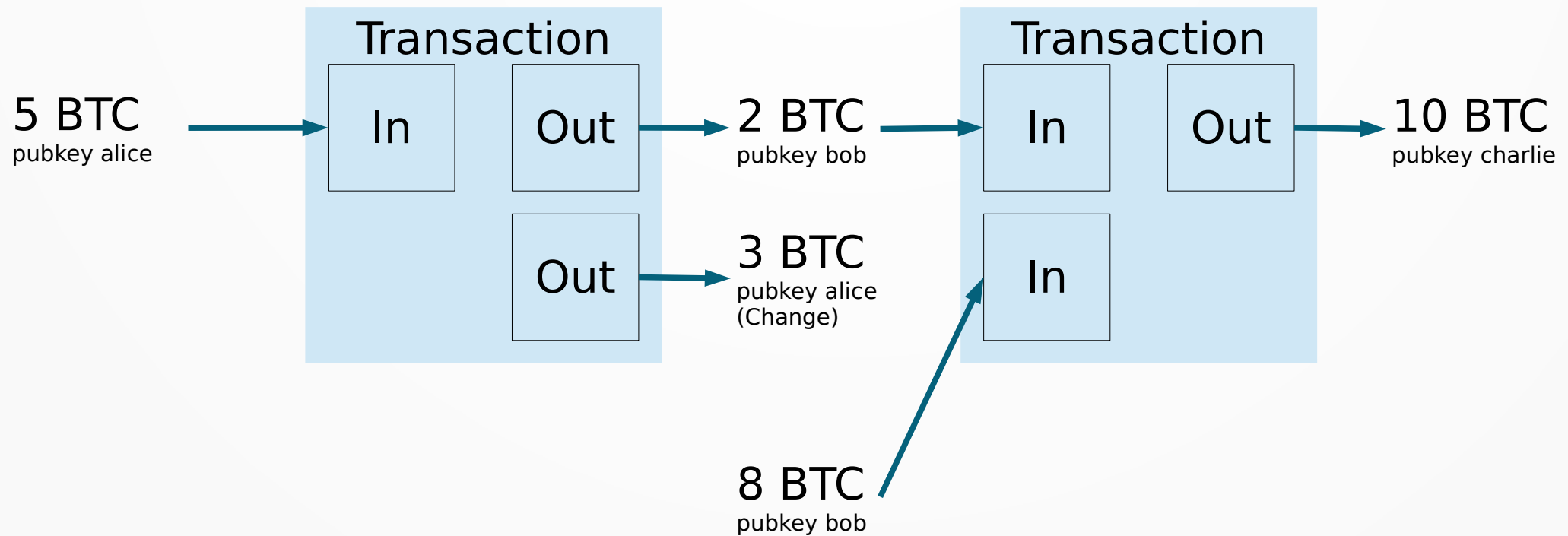
```
1 class Client
2 {
3     /** @var Blockchain */
4     protected $blockChain;
5
6     /** @var Transaction[] */
7     protected $currentTransactions = [];
8
9     /** @var SignatureKeyPair */
10    protected $keyPair;
11
12    /** @var HttpClient */
13    protected $httpClient;
14
15    public function __construct(SignatureKeyPair $keyPair, HttpClient $httpClient)
16    {
17        $this->keyPair = $keyPair;
18        $this->httpClient = $httpClient;
19        $this->blockChain = $this->downloadBlockChain();
20    }
21
22    public function addTransaction(Transaction $transaction): int
23    {...}
24
25    public function mine(): Block
26    {...}
27
28    private function downloadBlockChain(): Blockchain
29    {...}
30 }
```


```
1 use ParagonIE\Halite\Asymmetric\SignaturePublicKey as SPK;
2
3 class Transaction
4 {
5     /** @var SignaturePublicKey */
6     private $sender;
7
8     /** @var SignaturePublicKey */
9     private $recipient;
10
11     /** @var int */
12     private $amount;
13
14     /** @var string */
15     private $signature;
16
17     /** Constructor */
18     public function __construct(SPK $sender, SPK $recipient, int $amount, string $signature)
19     {
20         $this->txid = Uuid::uuid4();
21         $this->sender = $sender;
22         $this->recipient = $recipient;
23         $this->amount = $amount;
24         $this->signature = $signature;
25     }
```




```
1 /* @var string $sender SignaturePubkey */
2 $sender = 'c8eb997fa5734ba3041795aa84b842dcd3a730db682ae4c3df238dcbc8347553';
3 /* @var string $recipient SignaturePubkey */
4 $recipient = '98eb40f9f0177ef42f6bd9519288955be97faec9eefb1336d671cad45ab7560c';
5 /* @var int $amount */
6 $amount = 10;
7
8 /* @var string $signature */
9 $signature = Crypto::sign(
10     $sender.$recipient.$amount,
11     new SignatureSecretKey(new HiddenString(sodium_hex2bin($privateKey)))
12 );
13
14 new Transaction($sender, $recipient, $amount, $signature);
```

Bitcoin Transactions





```
1 class Block
2 {
3     /**
4      * @var int
5      */
6     private $index;
7
8     /**
9      * @var Transaction[]
10     */
11     private $transactions;
12
13     /**
14      * @var int
15      */
16     private $timestamp;
17
18     /**
19      * @var int
20      */
21     private $proofOfWork;
22
23     /**
24      * @var string
25      */
26     private $previousHash;
```



```
1 class Blockchain
2 {
3     /** @var Block[] */
4     private $blocks = [];
5
6     /** @var int */
7     private $difficulty;
8
9     /** @param int $difficulty */
10    public function __construct(int $difficulty)
11    {
12        $this->difficulty = $difficulty;
13    }
14
15    public function addBlock(): Block
16    {...}
17
18    public function getLatestBlock(): Block
19    {...}
20
21    // ...more methods...
22 }
```



```
1 class Block
2 {
3     public function calculateHash(): string
4     {
5         $treeNodes = array_map(
6             function (Transaction $transaction) {
7                 return new Node($transaction→getHashableString());
8             },
9             $this→transactions
10        );
11        $merkleTree = new MerkleTree( ... $treeNodes);
12        $rootHash = $merkleTree→getRoot();
13
14        return hash('sha256', $this→index.$rootHash.$this→proofOfWork.$this→
15        >previousHash.$this→timestamp);
16    }
17
18 // In the client:
19 $latestBlock = $blockChain→getLatestBlock();
20 $previousHash = $latestBlock→calculateHash();
21 $proof = $blockChain→getProofOfWork($latestBlock→getProofOfWork(), $previousHash);
22 // Add the reward and a new block to the blockchain:
23 $blockChain→addTransaction(new Transaction(...));
24 $blockChain→addBlock($proof, $previousHash);
```

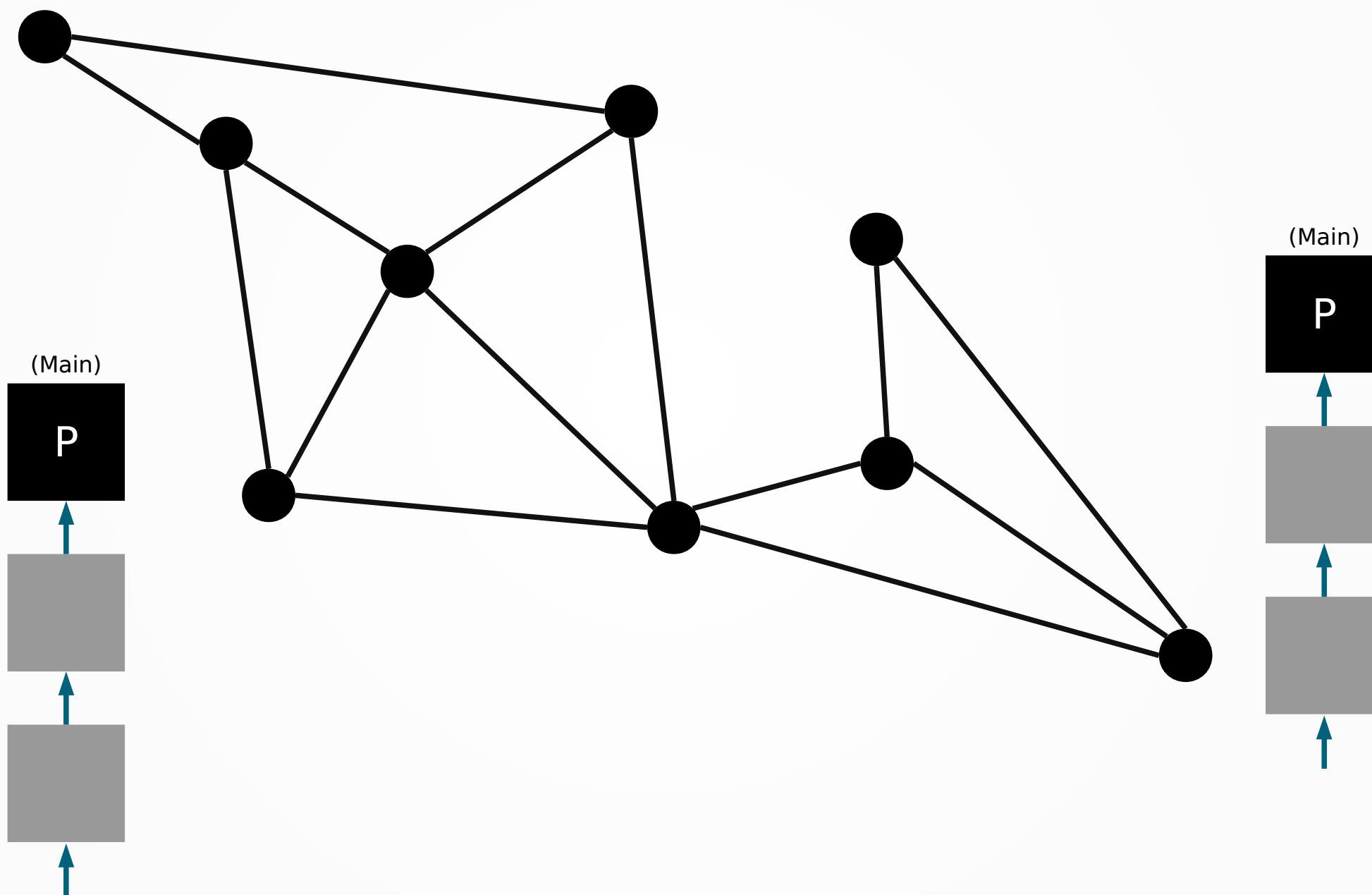
```
1 /**
2  * @param string $previousProofOfWork
3  * @param string $previousHash The hash of the previous block
4  * @return int
5  */
6 public function getProofOfWork(string $previousProofOfWork, string $previousHash): int
7 {
8     $proof = 0;
9     while ($this->isValidProof($proof, $previousProofOfWork, $previousHash) === false) {
10         $proof++;
11     }
12
13     return $proof;
14 }
15
16 /**
17 * @param int $proof
18 * @param string $previousProofOfWork
19 * @param string $previousHash
20 * @return bool
21 */
22 private function isValidProof(int $proof, string $previousProofOfWork, string $previousHash):
    bool
23 {
24     $guessHash = hash('sha256', $previousProofOfWork.$proof.$previousHash);
25
26     return 0 === strpos($guessHash, str_repeat('0', $this->difficulty));
27 }
```

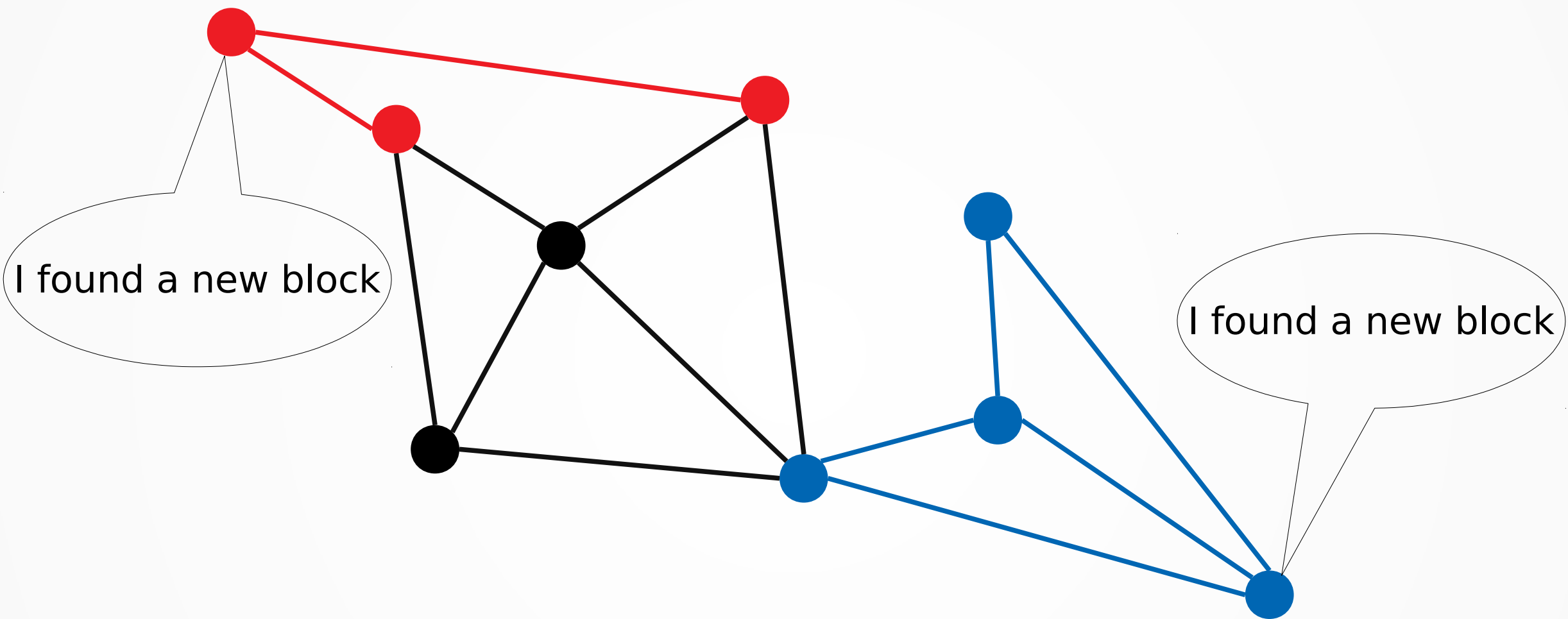


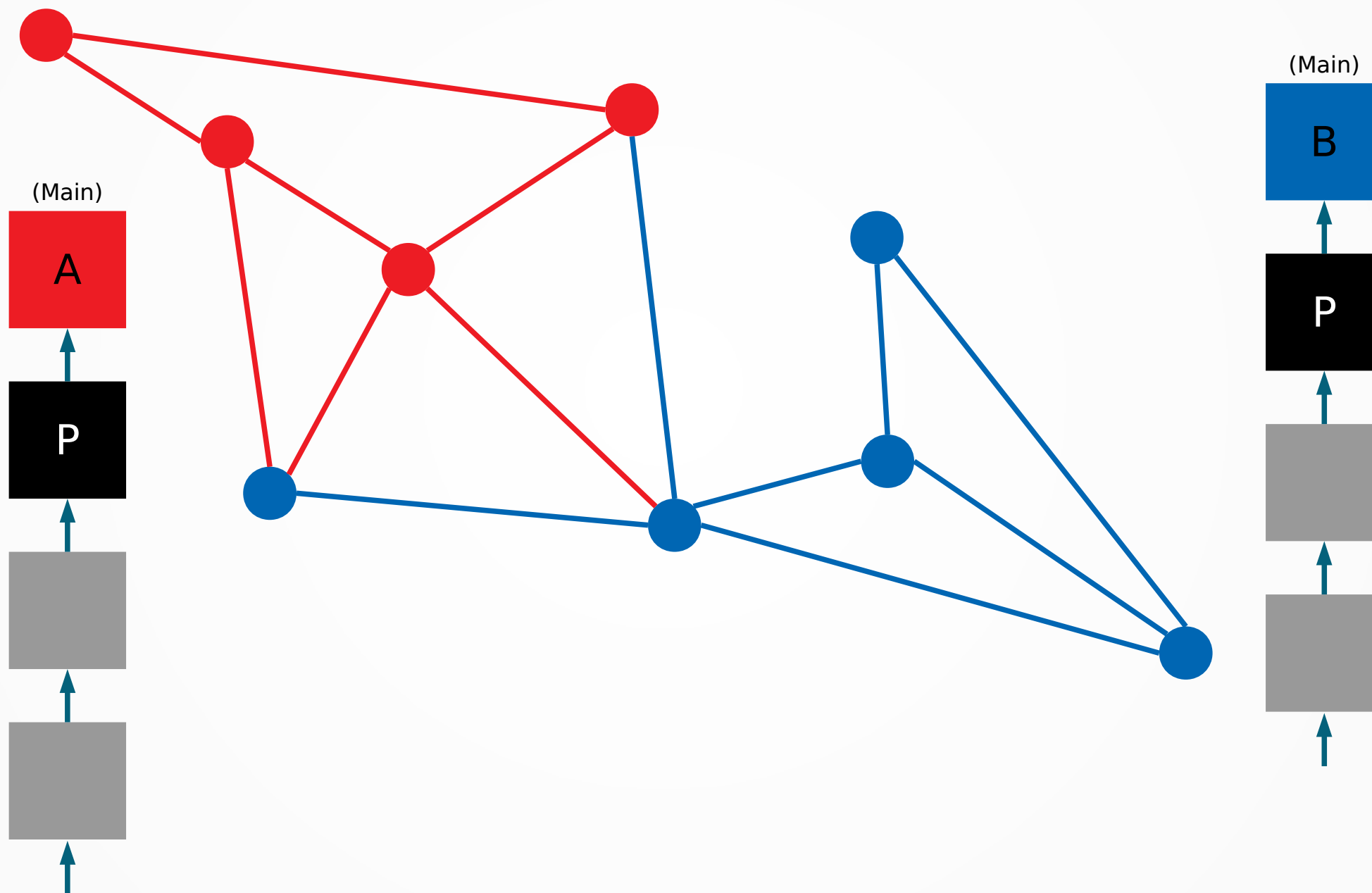

Demo Time

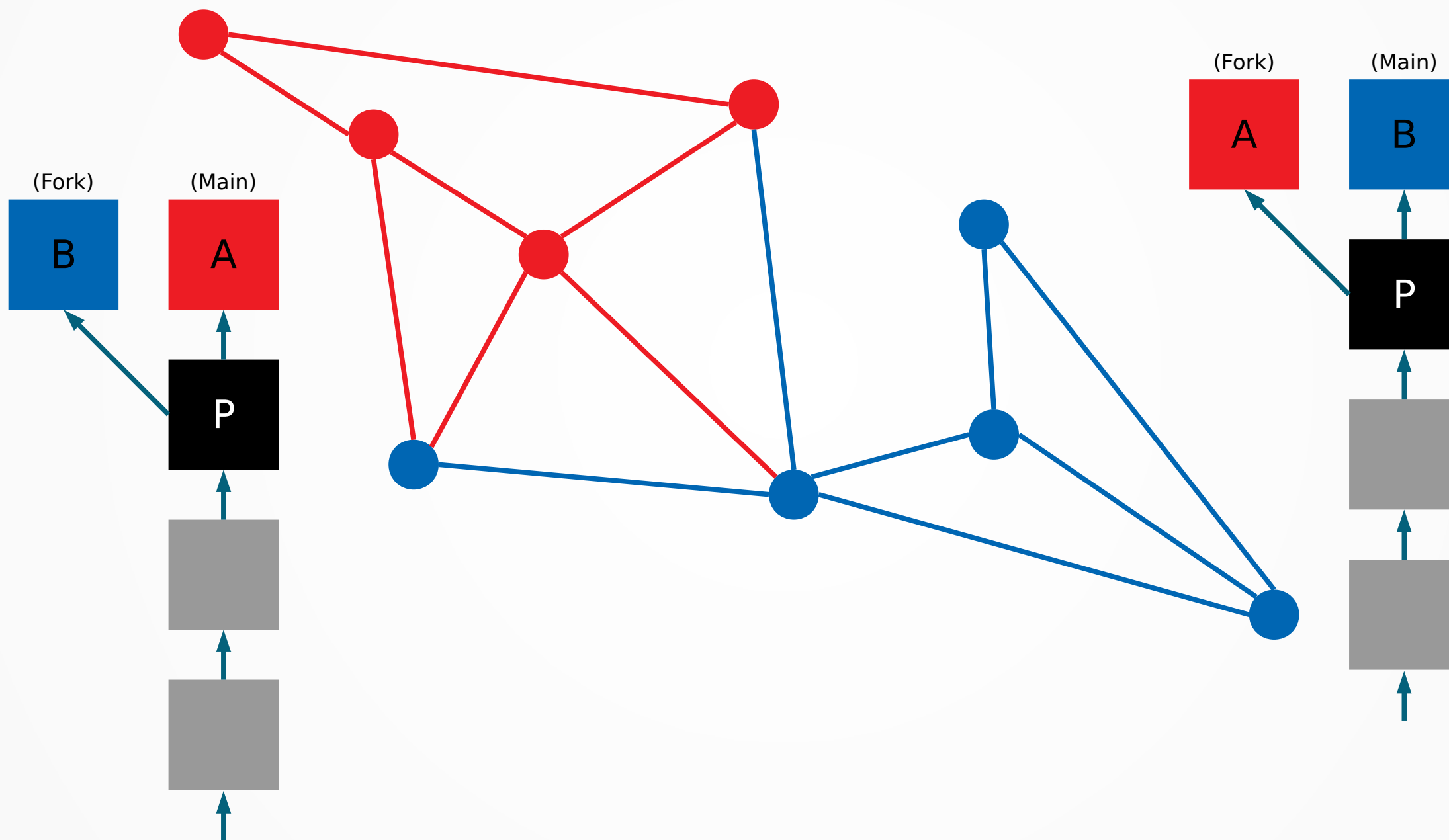
1. Start 2 nodes/clients
2. Add a transaction from node 2 → node 1
3. Mine a new block
4. Verify the transaction

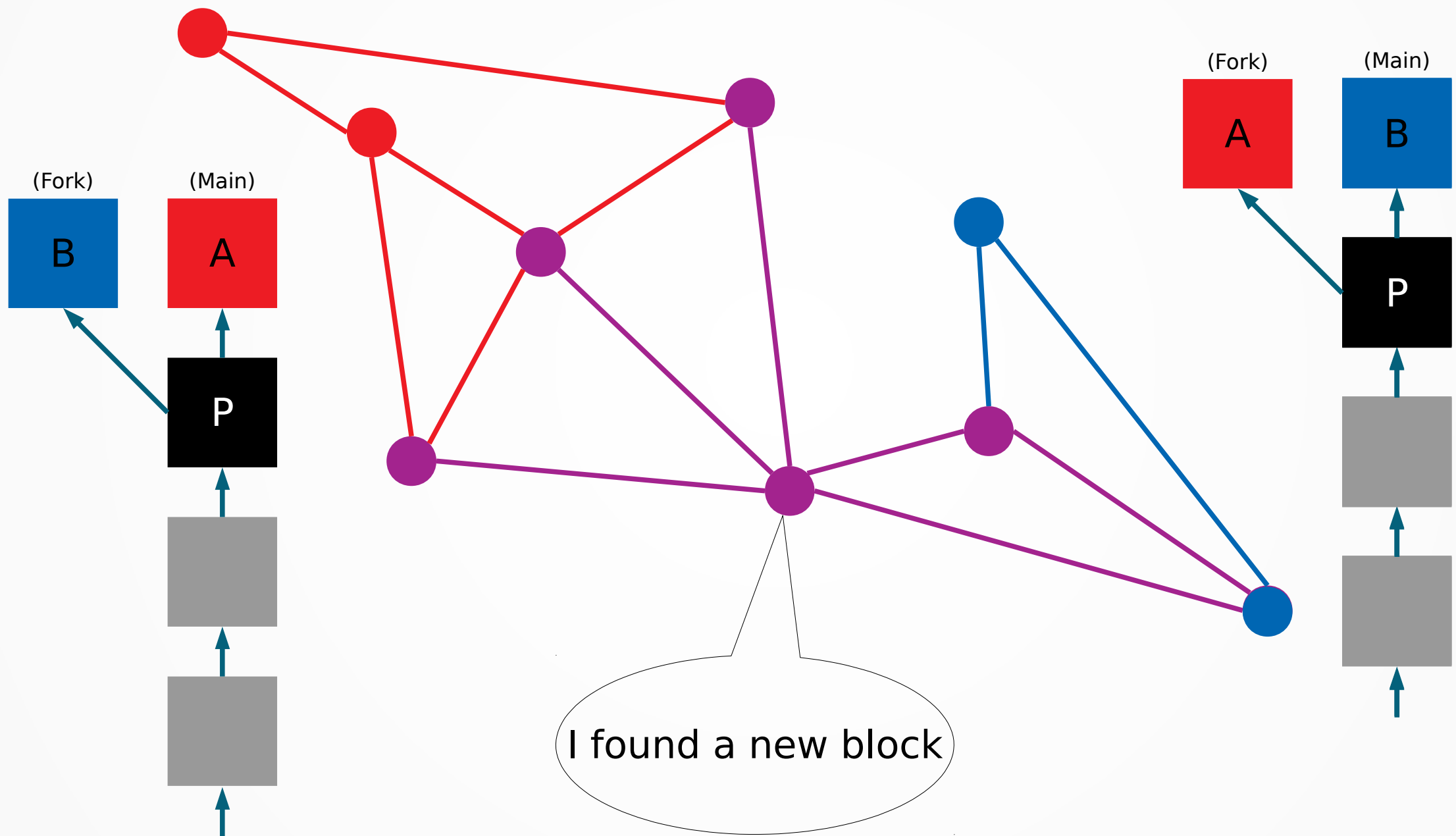
Consensus

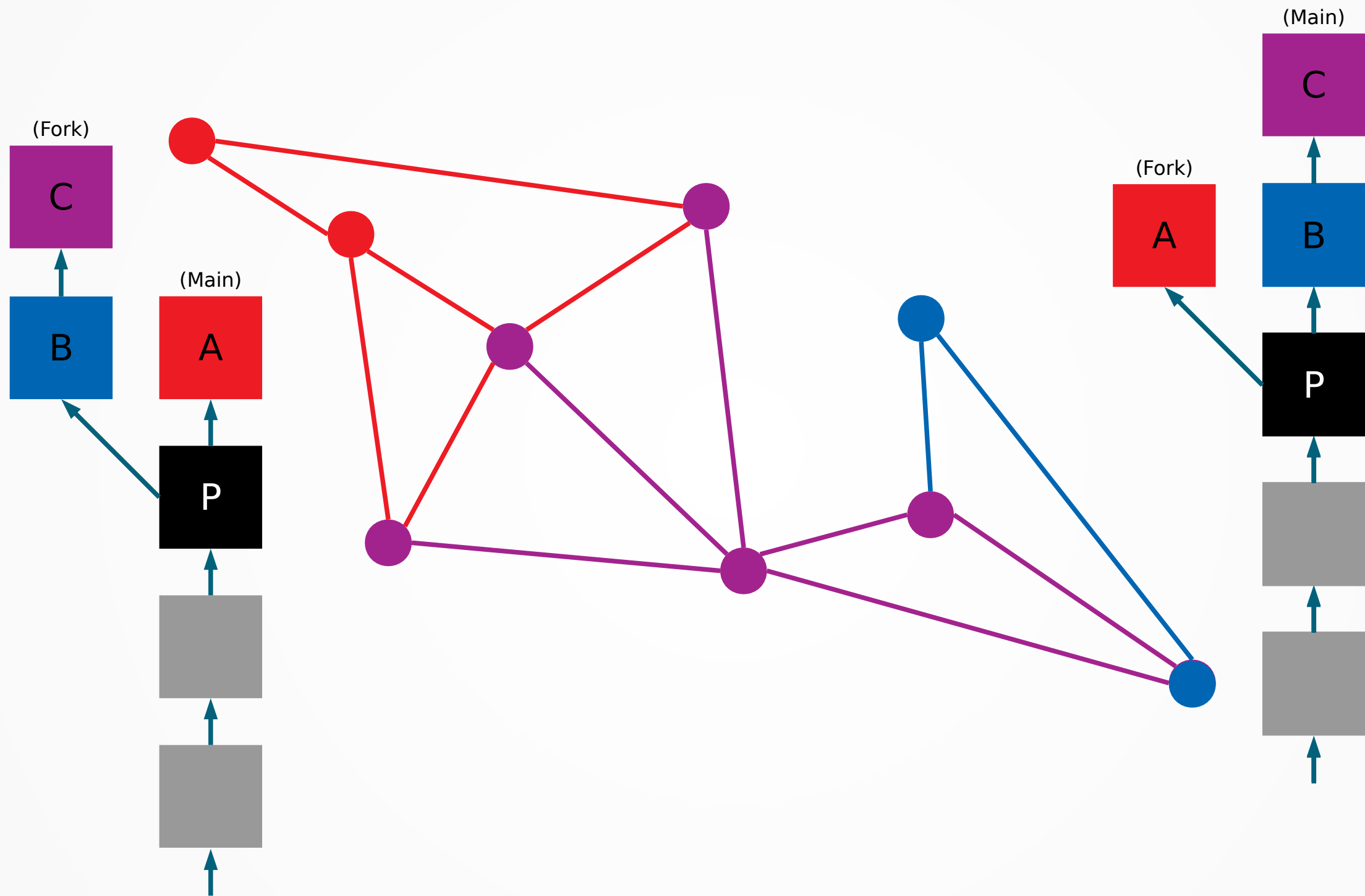


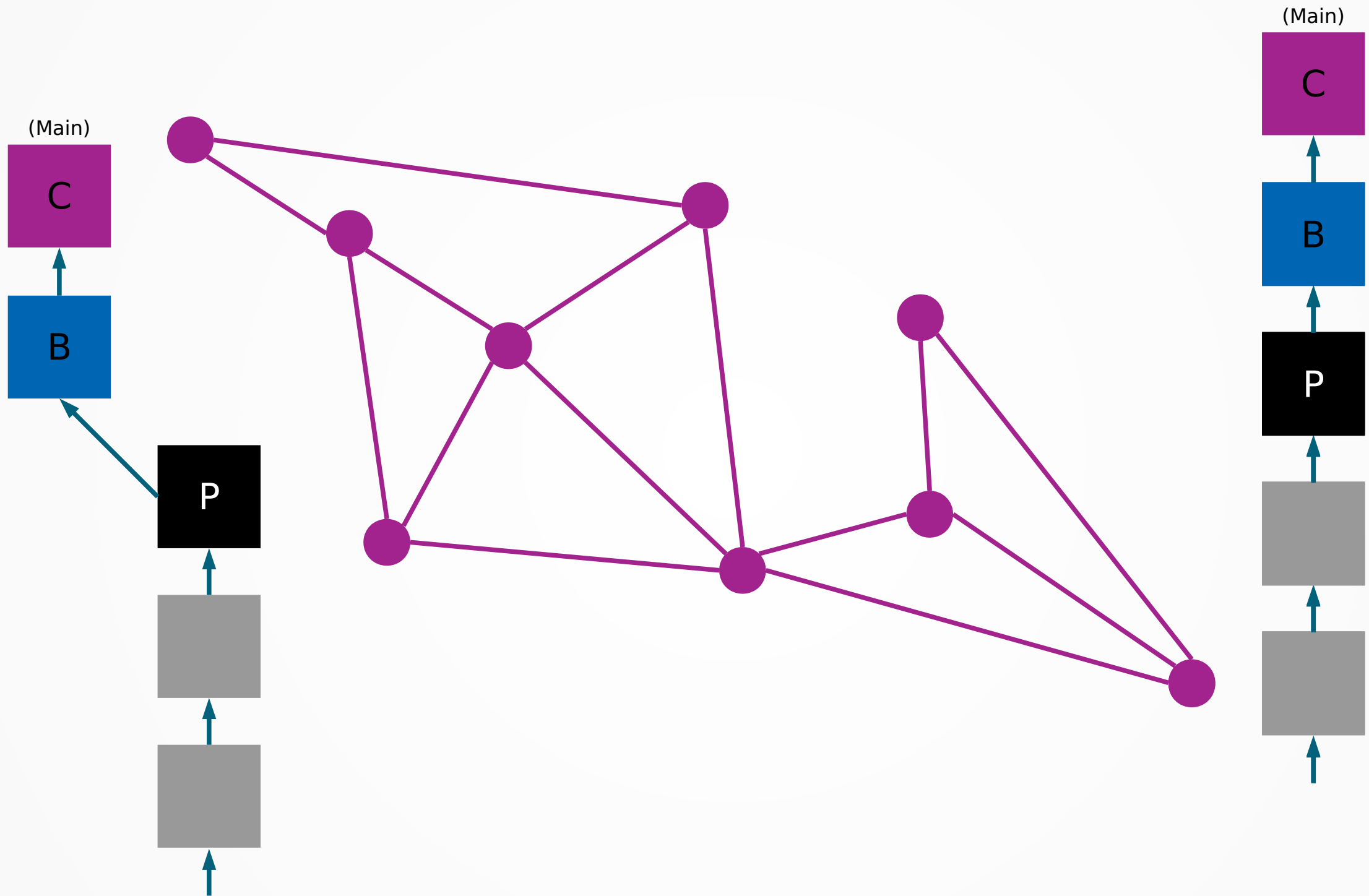


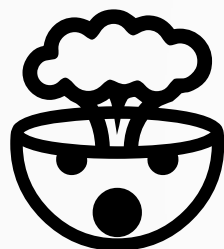








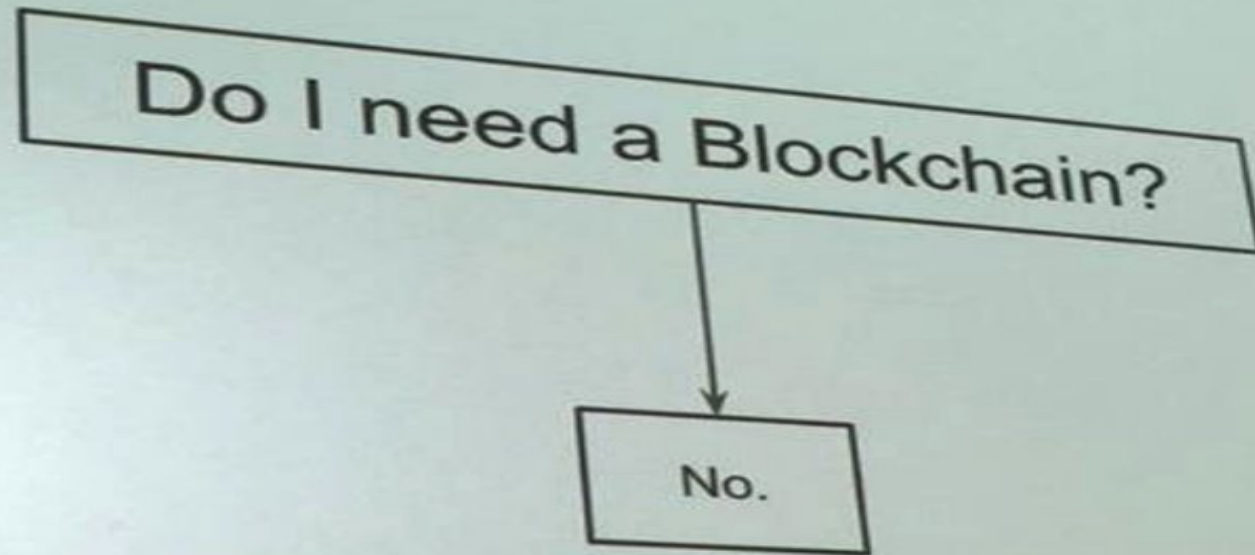




Learnings

- The tech behind blockchain is cool
- and here to stay
- Putting the right technologies together pays off in the end
- ReactPHP rocks 🤘

<http://doyouneedablockchain.com/>

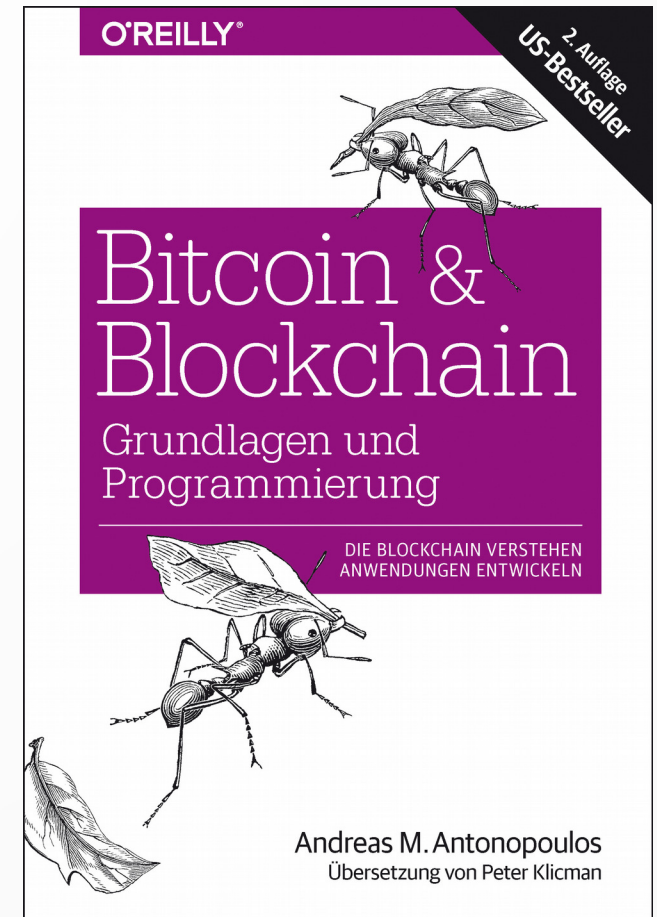


@vgcerf: „Simple Flowchart“

<https://mobile.twitter.com/vgcerf/status/1019987651301081089>

For further reading

- Antonopoulos: Bitcoin & Blockchain. O'Reilly, 2018.
- github.com/Bit-Wasp/bitcoin-php
- hackernoon.com/learn-blockchains-by-building-one-117428612f46
- anders.com/blockchain
- blockchaindemo.io & coindemo.io
- github.com/mattsches/phplockchain



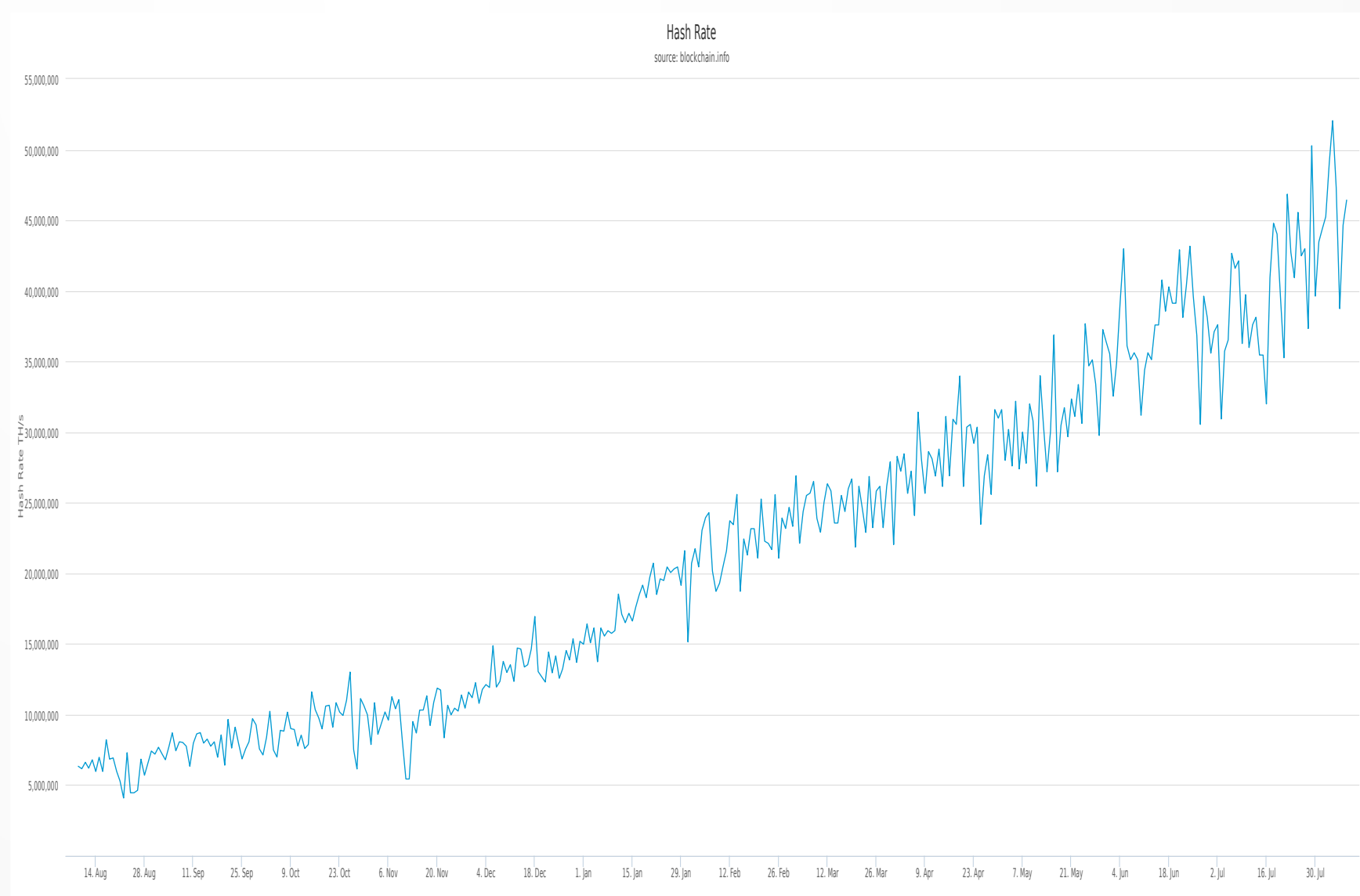
Thank you, FrOSCon!

(don't forget to rate this talk on joind.in)

And now enjoy the social event!

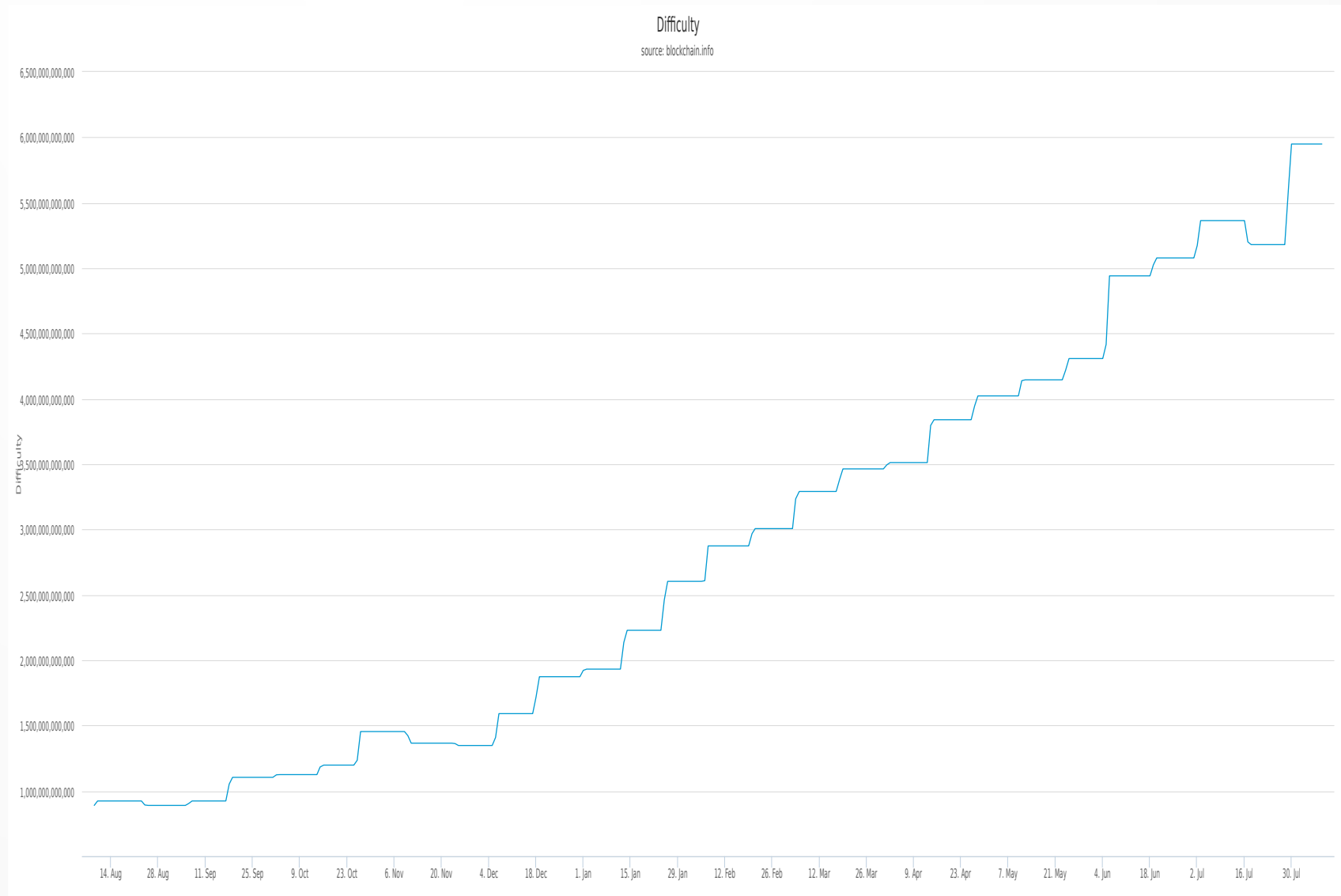
Bonus: Statistics

- **Hashrate**
- Difficulty
- Market Price



Bonus: Statistics

- Hashrate
- **Difficulty**
- Market Price



Bonus: Statistics

- Hashrate
- Difficulty
- **Market Price**

