Unittesting Smart Contracts in Python

POPULUS

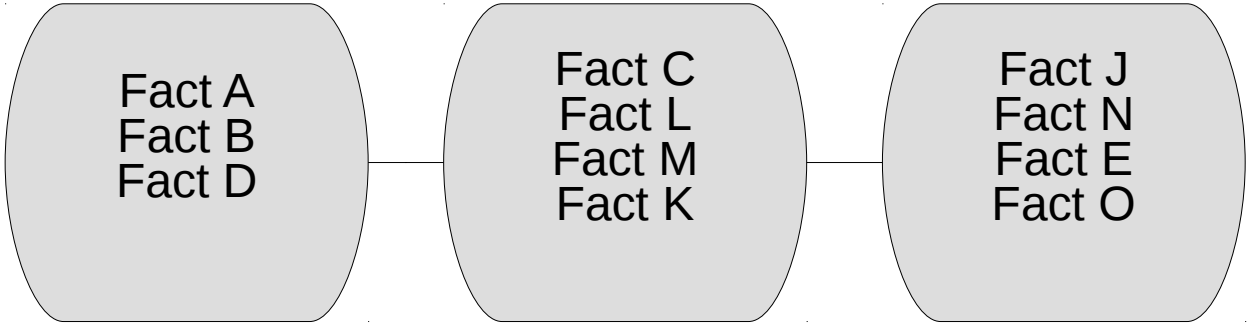
chat on gitter | build passing | docs passing

Development framework for Ethereum smart contracts

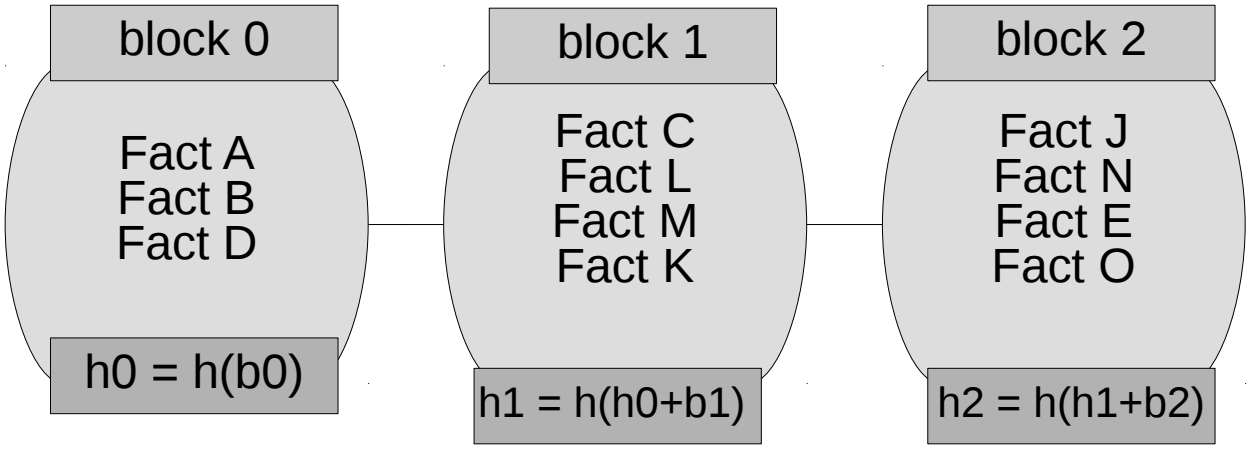
Robert Meyer – FlixBus – Data Science, Machine Learning, and Operations Research

PyData Berlin 2018

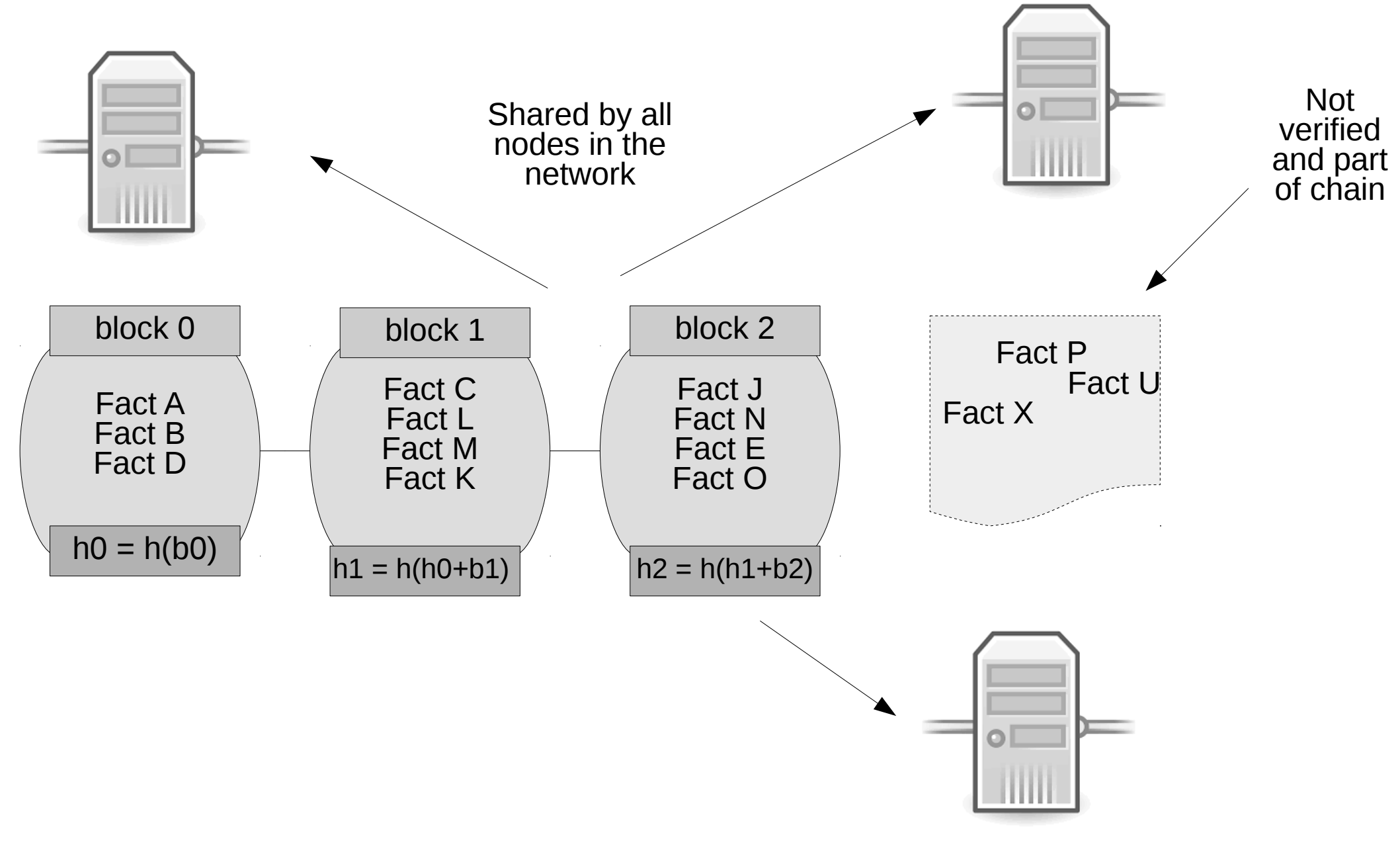
Blockchain


Fact A
Fact B
Fact D
Fact C
Fact L
Fact M
Fact K
Fact J
Fact N
Fact E
Fact O

Blockchain

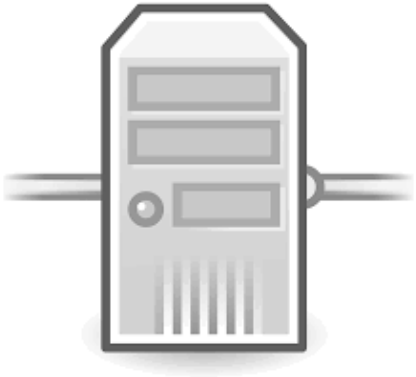

block 0
Fact A
Fact B
Fact D
h0 = h(b0)
block 1
Fact C
Fact L
Fact M
Fact K
h1 = h(h0+b1)
block 2
Fact J
Fact N
Fact E
Fact O
h2 = h(h1+b2)

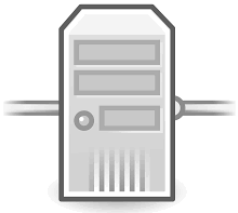
Blockchain

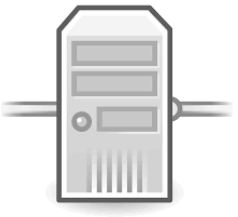

Shared by all nodes in the network
Not verified and part of chain
block 0
Fact A
Fact B
Fact D
h0 = h(b0)
block 1
Fact C
Fact L
Fact M
Fact K
h1 = h(h0+b1)
block 2
Fact J
Fact N
Fact E
Fact O
h2 = h(h1+b2)
Fact P
Fact U
Fact X

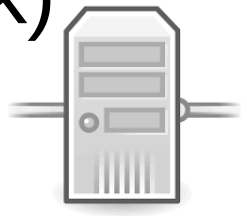
Miners

- Miner
  - Collect, verify, and, combine new facts
  - Create new blocks
  - Come up with a **proof of work**
  - Get a reward for the work
  - Publish the new block
- Other Nodes
  - Verify new block (facts + proof of work)



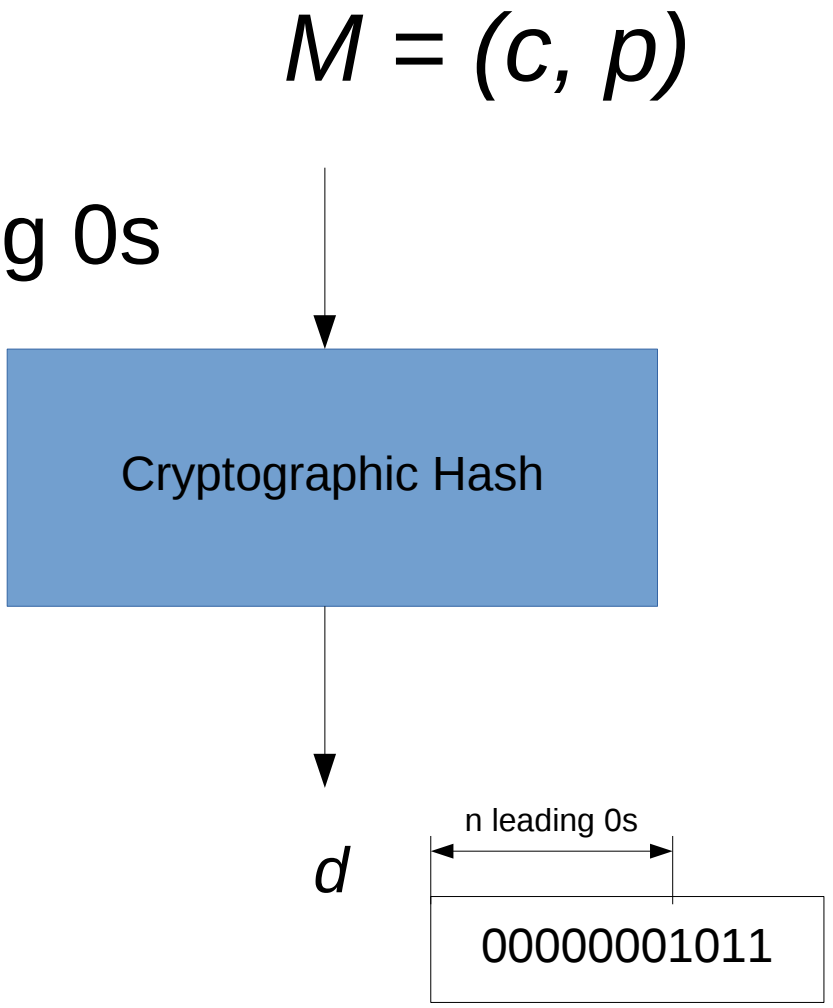




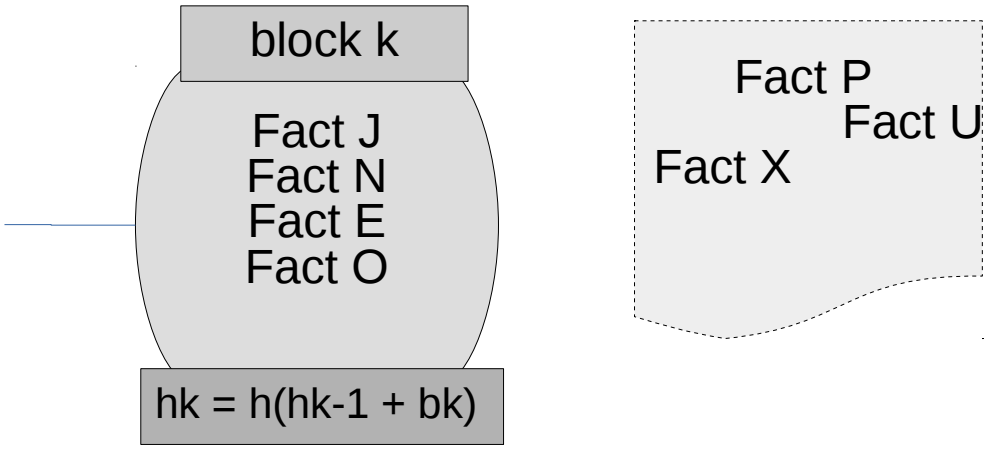


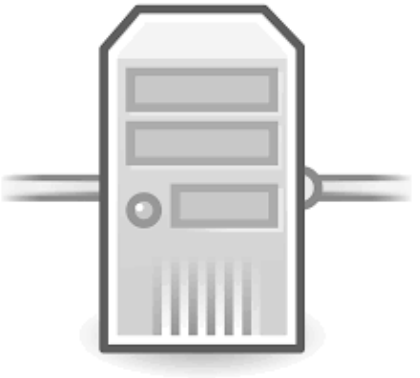
Proof of Work

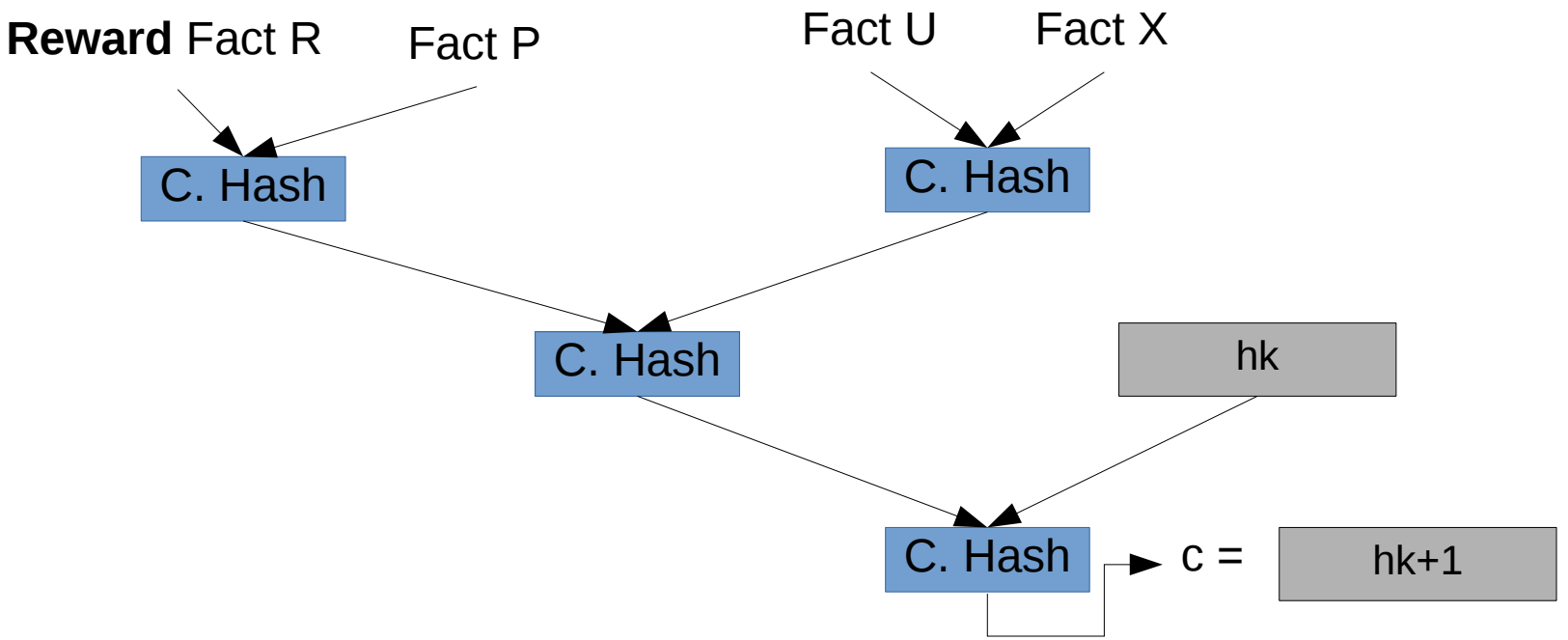
- Solve a puzzle
  - Compute hash with *n* leading 0s
  - Msg = challenge + proof
  - Iteratively pick proof strings



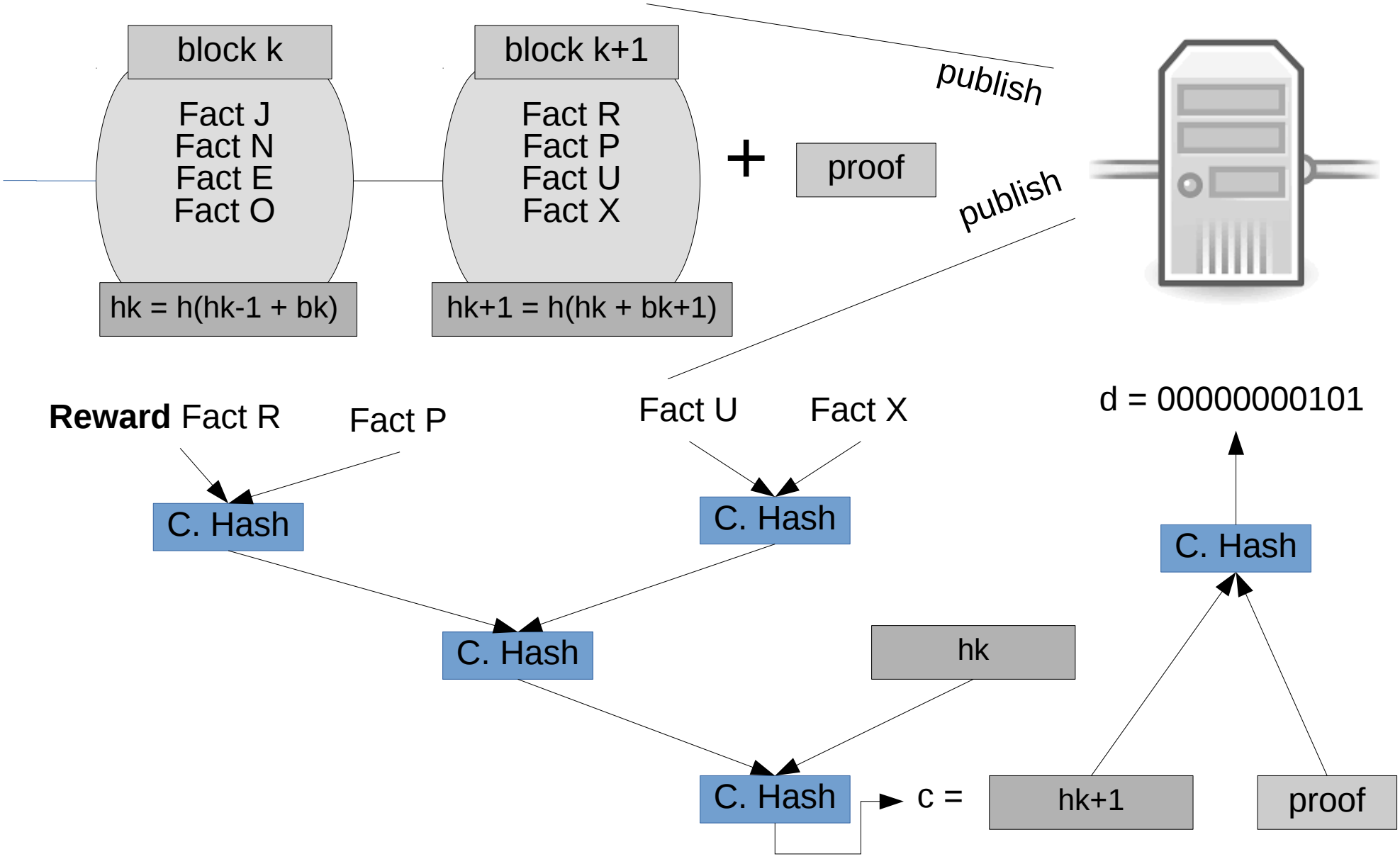
Blockchain Proof of Work

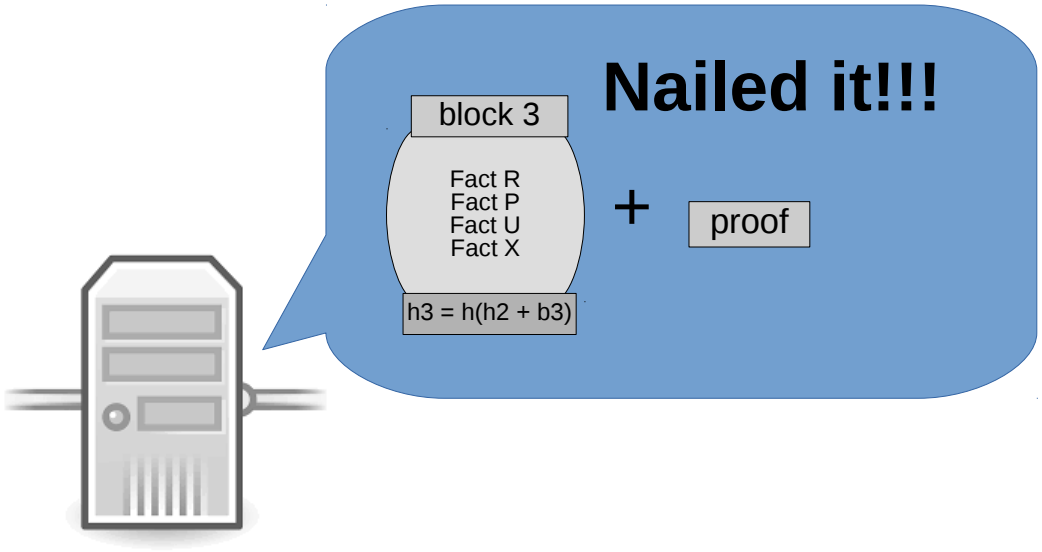


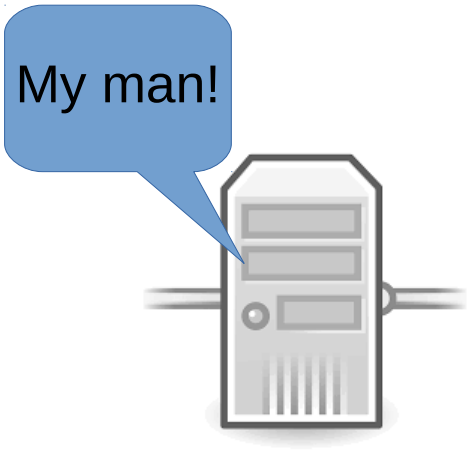


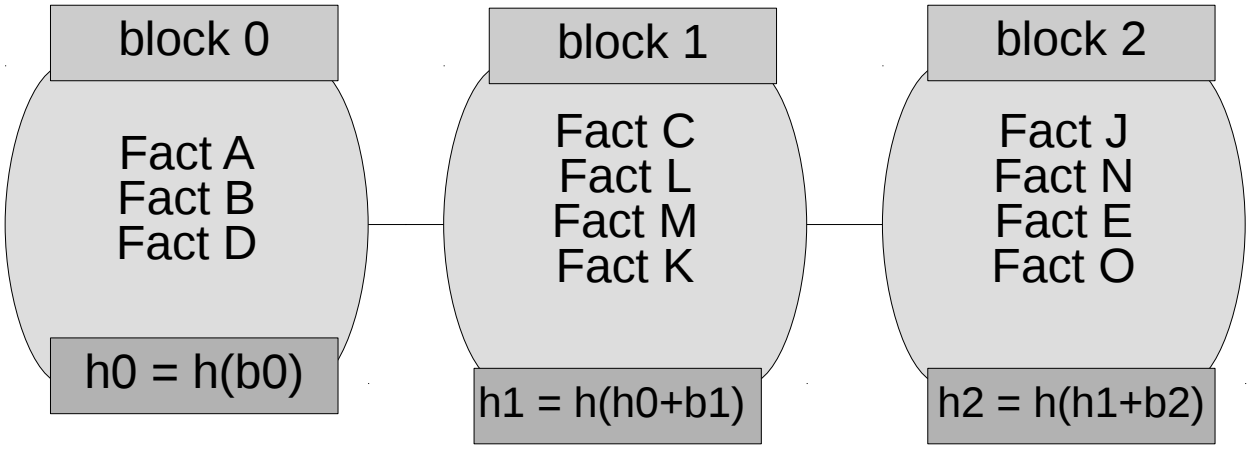


Blockchain Proof of Work



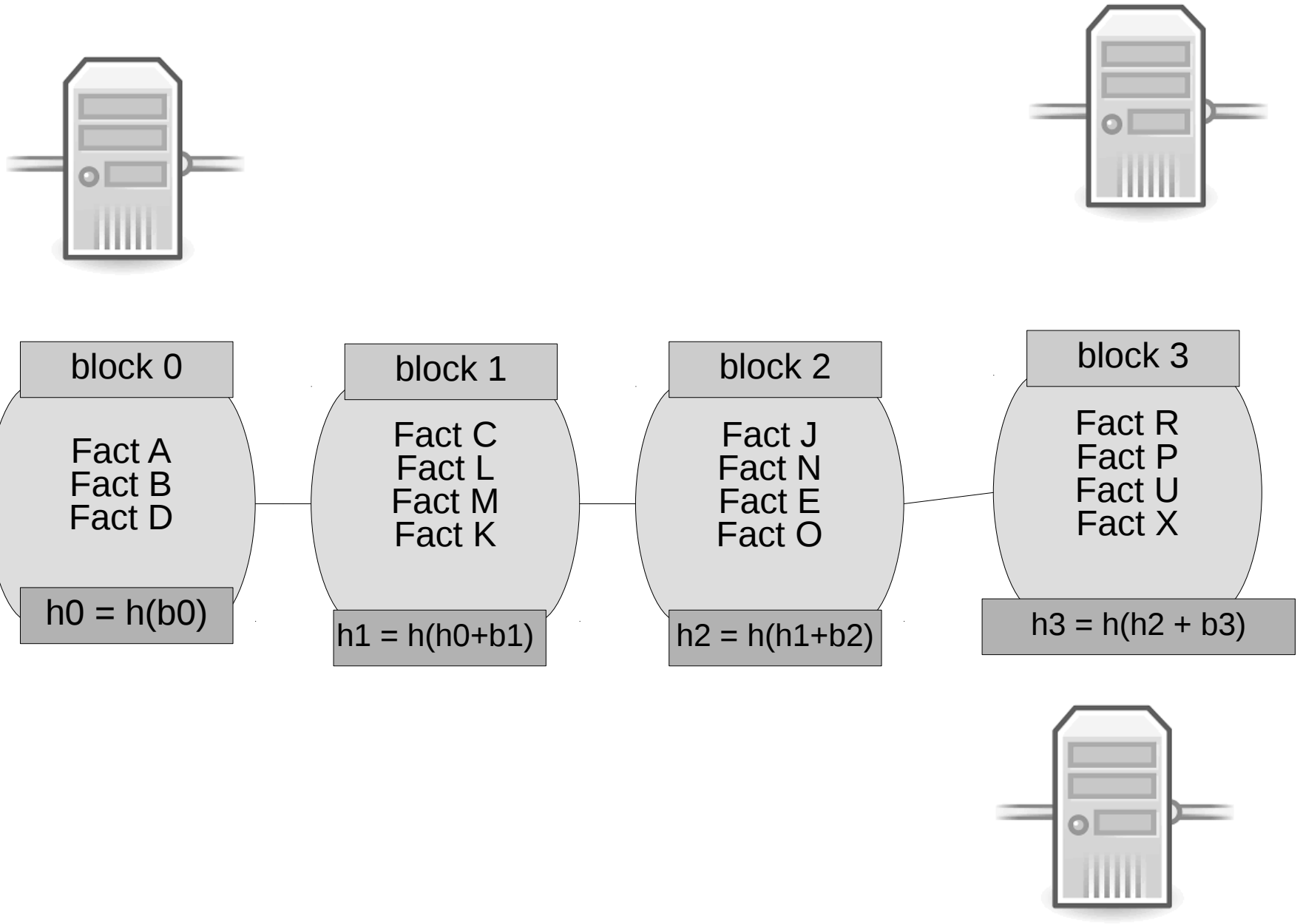

Nailed it!!!
block 3
Fact R
Fact P
Fact U
Fact X
h3 = h(h2 + b3)
+
proof


My man!

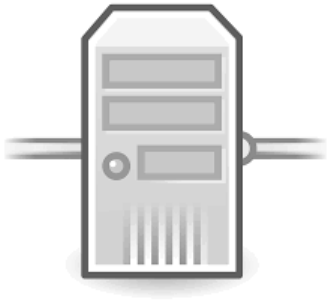

block 0
Fact A
Fact B
Fact D
h0 = h(b0)
block 1
Fact C
Fact L
Fact M
Fact K
h1 = h(h0+b1)
block 2
Fact J
Fact N
Fact E
Fact O
h2 = h(h1+b2)

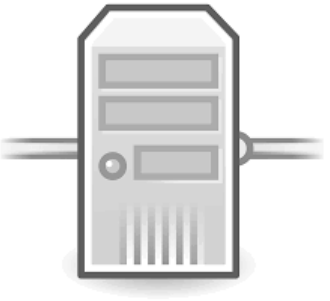

Dig you!

Blockchain

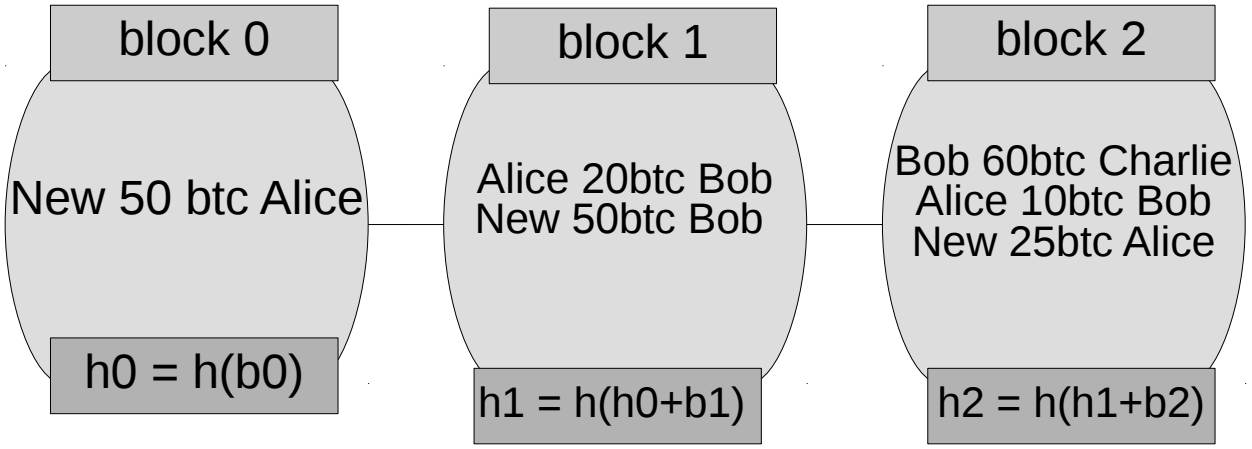

block 0
Fact A
Fact B
Fact D
h0 = h(b0)
block 1
Fact C
Fact L
Fact M
Fact K
h1 = h(h0+b1)
block 2
Fact J
Fact N
Fact E
Fact O
h2 = h(h1+b2)
block 3
Fact R
Fact P
Fact U
Fact X
h3 = h(h2 + b3)

Bitcoin





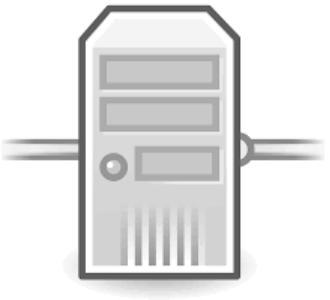
Not verified and part of chain



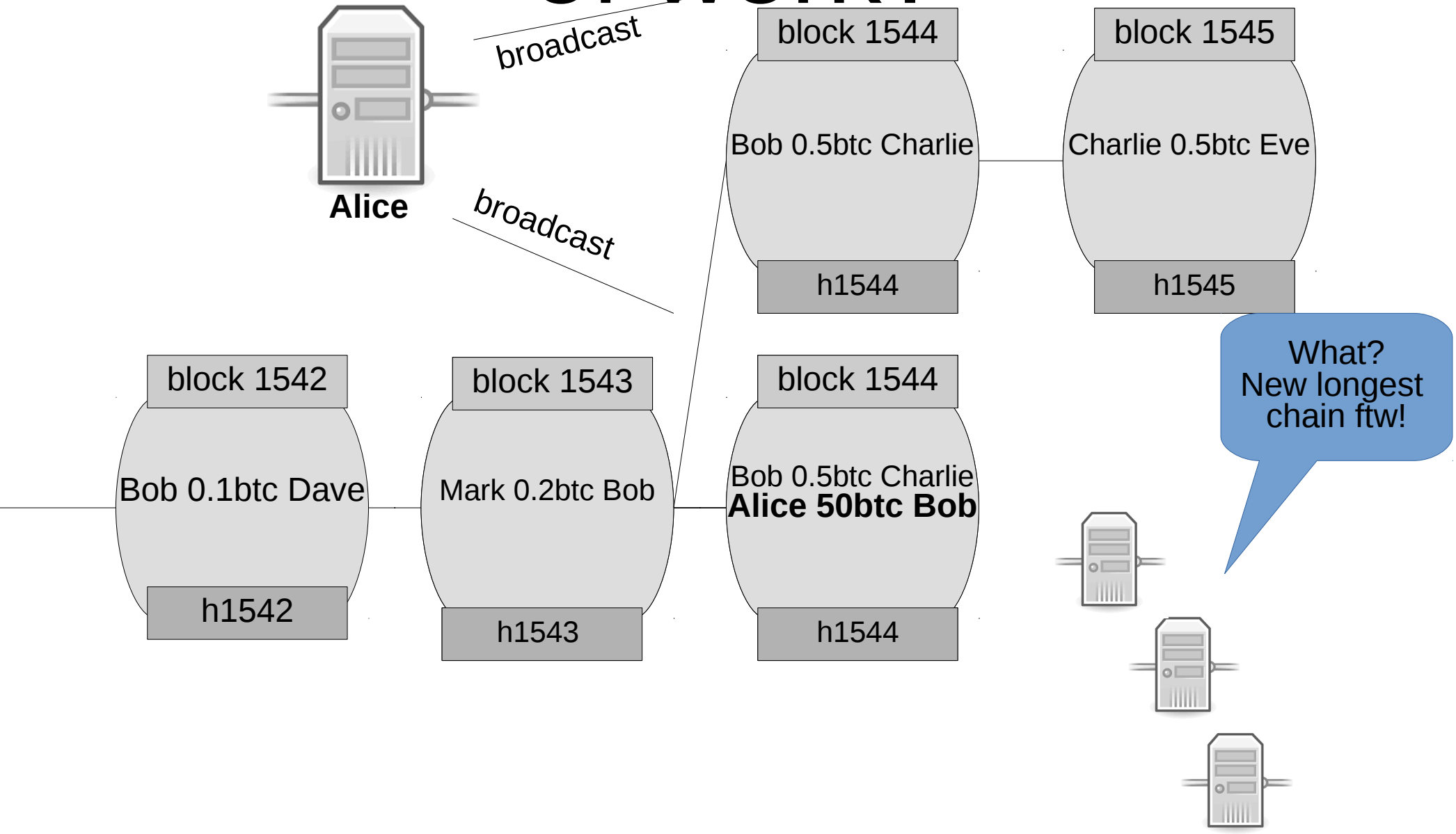
Charlie 20btc Dave

Bob 0.5btc Eve

Alice 0.1btc Dave



Why this expensive proof of work?



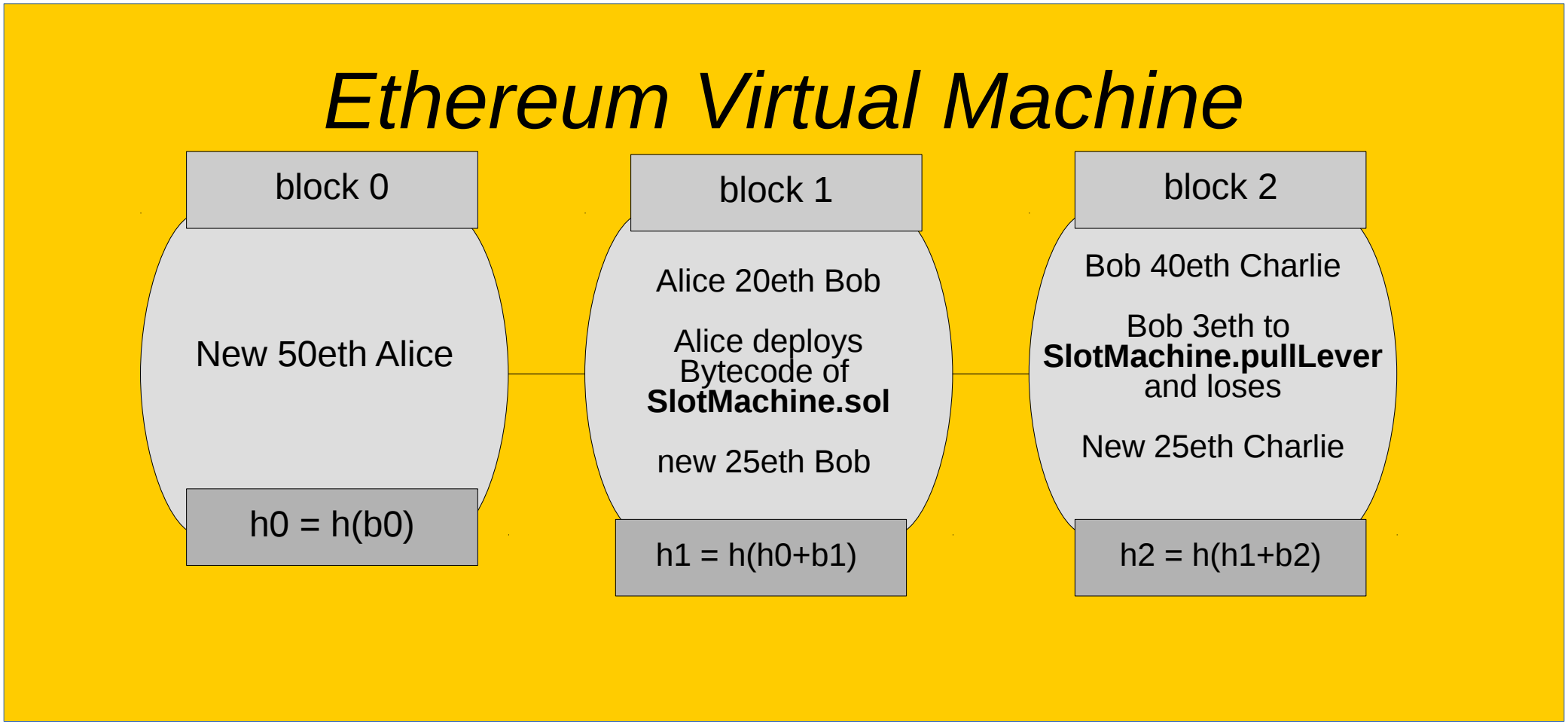

ethereum
HOMESTEAD RELEASE
BLOCKCHAIN APP PLATFORM

Ethereum In a Nutshell

- Cryptocurrency
  - Units are called Ether
- Distributed Database
  - Not only transaction but arbitrary data
- Executable code
  - Smart Contracts

Ethereum Blockchain

Transactions + Bytecode + Persistent Data



Miners

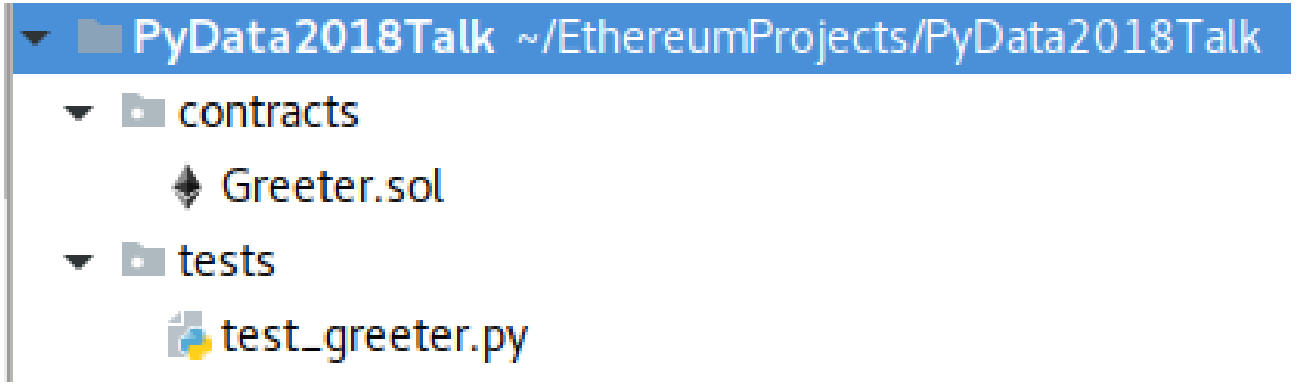
- 1. Data collection and verification
  - Check ETH transactions
  - Execute contract code in Ethereum Virtual Machine
    - Create new contract instances
    - Execute contract function calls
    - Change state of contracts
- 2. Do proof of work
  - Collect ETH for block generation
- 3. Publish new block
  - Nodes need to check if code is valid (re-execute)

Contract Programming is HARD

- The DAO Hack (2016)
  - Recursive Function Execution Exploit
  - ~50 million $ lost → 15% of ETH
- Parity Funds Freeze (2017)
  - „devops1999“ took ownership of a wallet library
    - Killed the library
  - ~150 million $ are stuck (forever)

Testing is of utmost importance!

- populus init



Testing is of utmost importance!

- populus init

```
pragma solidity ^0.4.0;

contract Greeter {
    string public greeting;

    // TODO: Populus seems to get no bytecode if `internal`
    function Greeter() public {
        greeting = 'Hello';
    }

    function setGreeting(string _greeting) public {
        greeting = _greeting;
    }

    function greet() public constant returns (string) {
        return greeting;
    }
}
```

Testing is of utmost importance!

- populus init

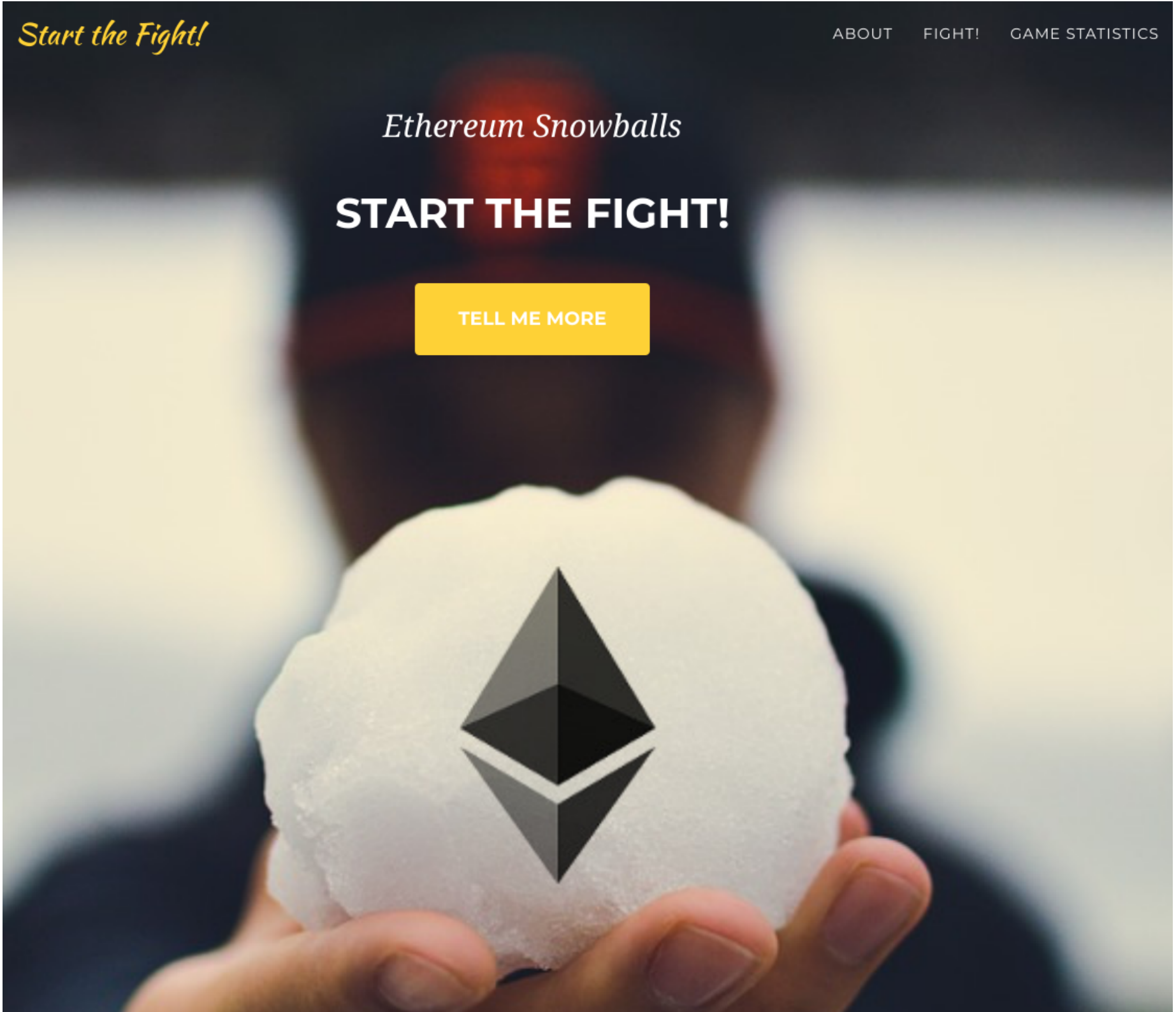
```
def test_greeter(chain):
    greeter, _ = chain.provider.get_or_deploy_contract('Greeter')

    greeting = greeter.call().greet()
    assert greeting == 'Hello'


def test_custom_greeting(chain):
    greeter, _ = chain.provider.get_or_deploy_contract('Greeter')

    set_txn_hash = greeter.transact().setGreeting('Guten Tag')
    chain.wait.for_receipt(set_txn_hash)

    greeting = greeter.call().greet()
    assert greeting == 'Guten Tag'
```


Start the Fight!
ABOUT
FIGHT!
GAME STATISTICS
Ethereum Snowballs
START THE FIGHT!
TELL ME MORE

Take Home Messages

- Ethereum allows you to run **Smart Contracts** on a Blockchain!
- **Populus** allows you to quickly test these Programs!
- Writing good **Solidity** code is RIDICULOUSLY difficult!

Thank You!

- Checkout/References
  - Populus: https://github.com/ethereum/populus
  - Solidity Docs: http://solidity.readthedocs.io
  - Remix: https://remix.ethereum.org
  - Metamask: https://metamask.io/
  - Ethereum Hacks: https://blog.sigmaprime.io/solidity-security.html
- Contact:
  - Mail: robert.meyer@flixbus.com
  - Github: https://github.com/SmokinCaterpillar
  - Steemit: https://steemit.com/@smcaterpillar