COMP4137 Blockchain Technology and Applications
COMP7200 Blockchain Technology

Lecturer: Dr. Hong-Ning Dai (Henry)

Lecture 7

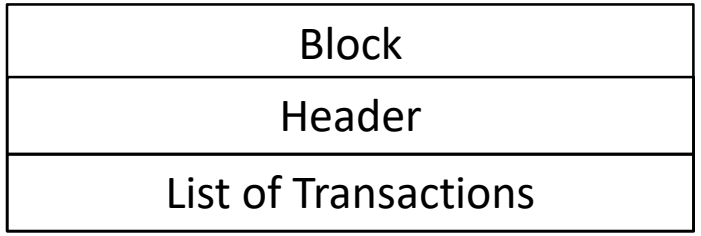
Permissionless blockchain 2

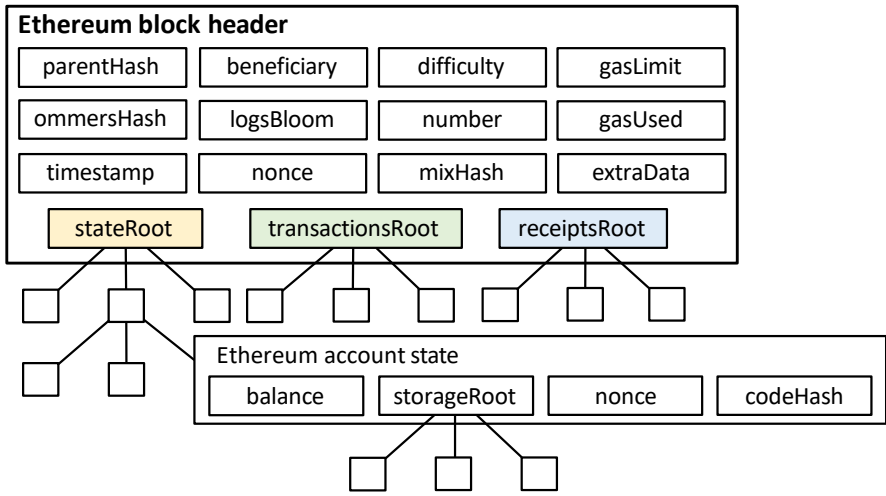
Outline

- Ethereum Block
  - Patricia Trie
- Ethereum Consensus
- Ethereum DApps

Ethereum Block

- **Block header**
  - Consensus data: parent hash, difficulty, PoW solution, etc
  - Beneficiary: where TX fees will go (address)
  - **World state root**: updated world state
    - Merkle Patricia Tree hash of all accounts in the system
  - **TX root**: Merkle hash of all TXs included in block
  - **TX receipt root**: Merkle hash of log messages generated in block
  - Gas used: Tells verifier how much work to verify block





Ethereum Block

- Block header contains three Merkle trees for **Transactions**, **Receipts** and **States**

- Enable light clients to conduct various types of queries
  - Has this transaction been included in a particular block? (Transaction tree)
  - Tell me all instances of an event of type X (e.g., a crowdfunding contract reaching its goal) emitted by this address in the past Y days (Receipt tree)
  - What is the current balance of my account? (State

    tree)
  - Does this account exist? (State tree)

Radix Trie

- How does Ethereum manage the storage with 256-bit address?
  - Key-value pair
- Radix Trie: used for key-value pair storage management

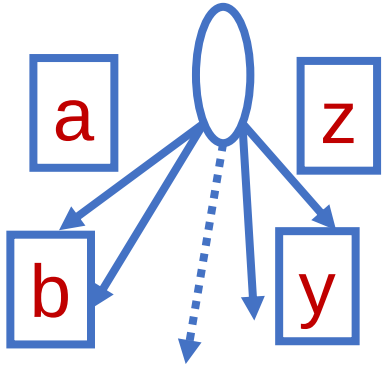
do:0
dog:1
data:2
down:3
cat:4
cats:5

One possible child for each letter

Each path spells a key word

Value at end of path

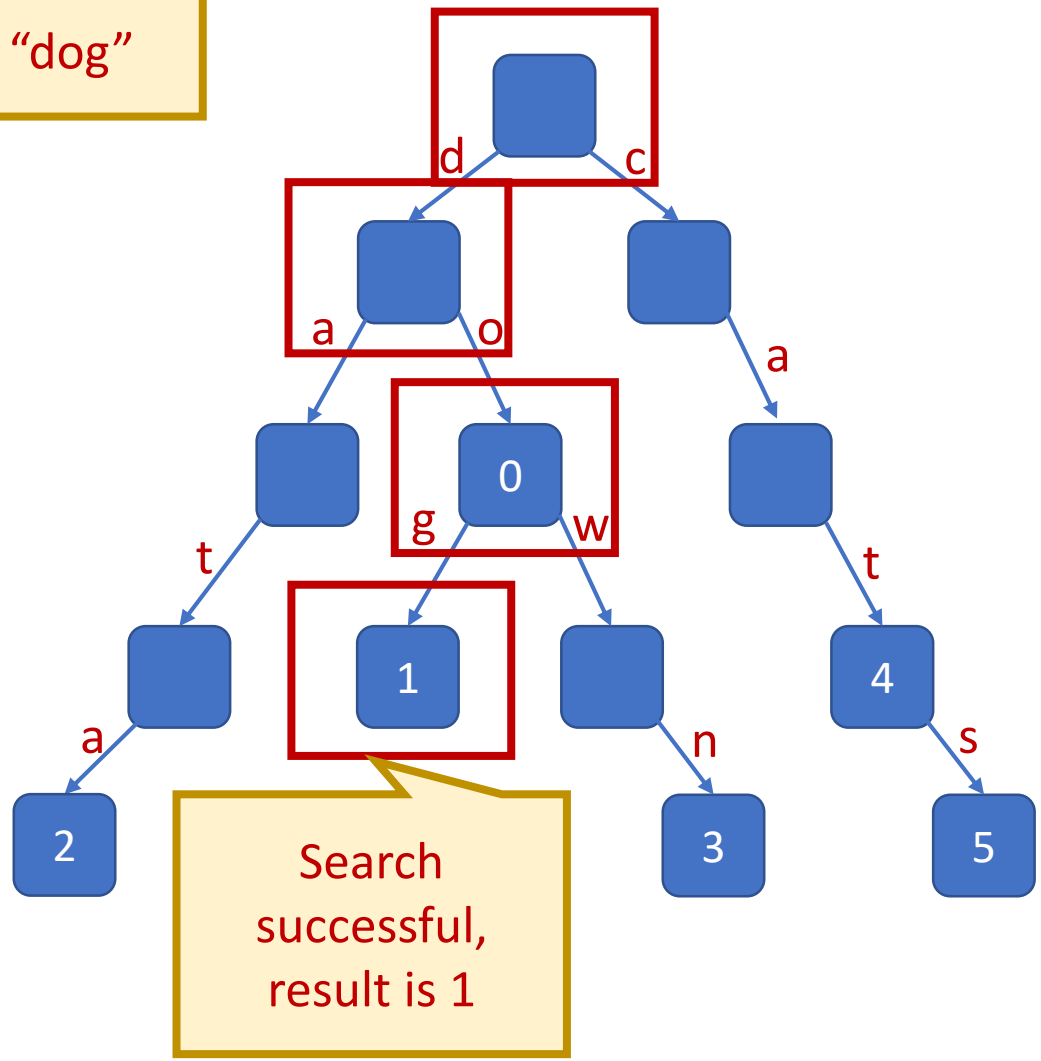
Search time = key length



Radix Trie

Search for “dog”

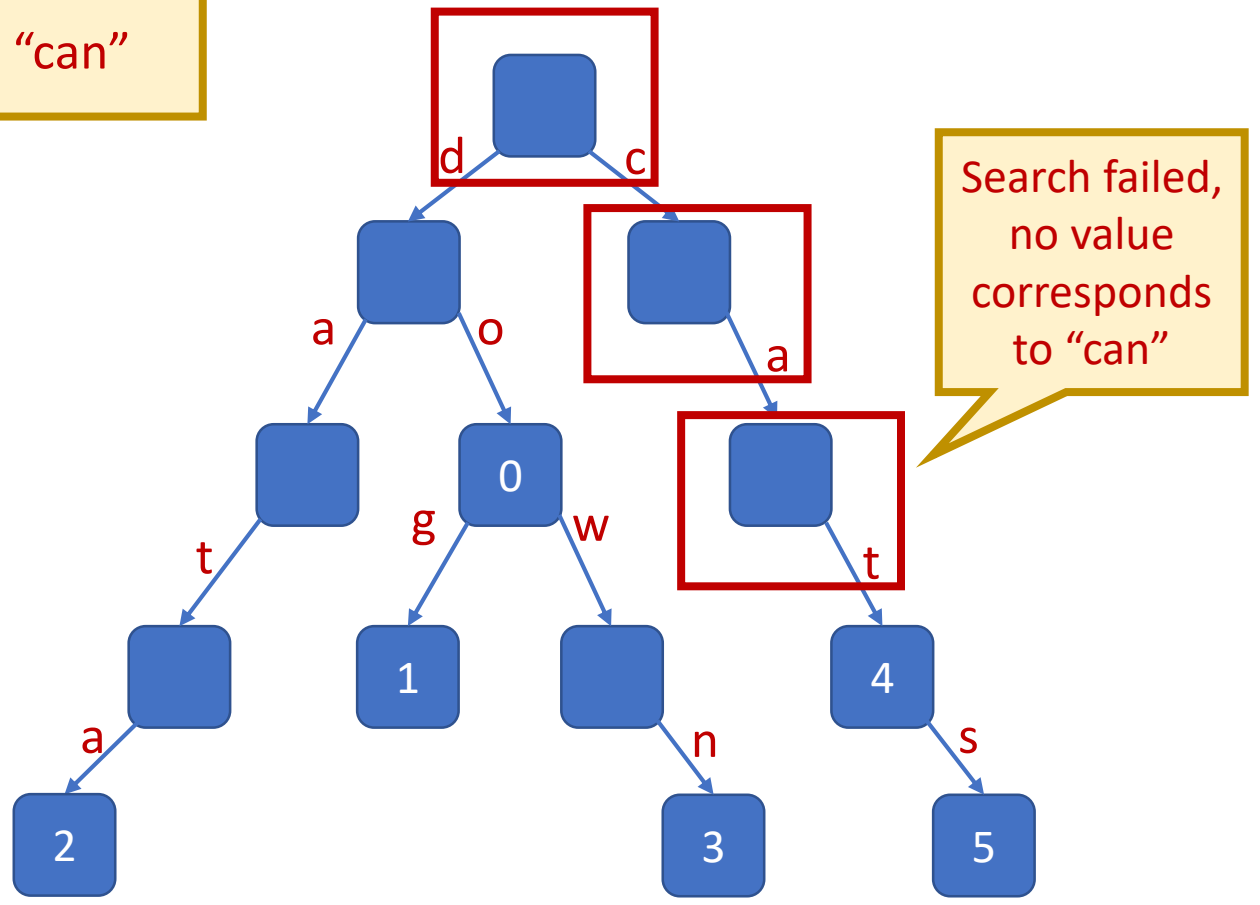
do:0
dog:1
data:2
down:3
cat:4
cats:5



Radix Trie

Search for “can”

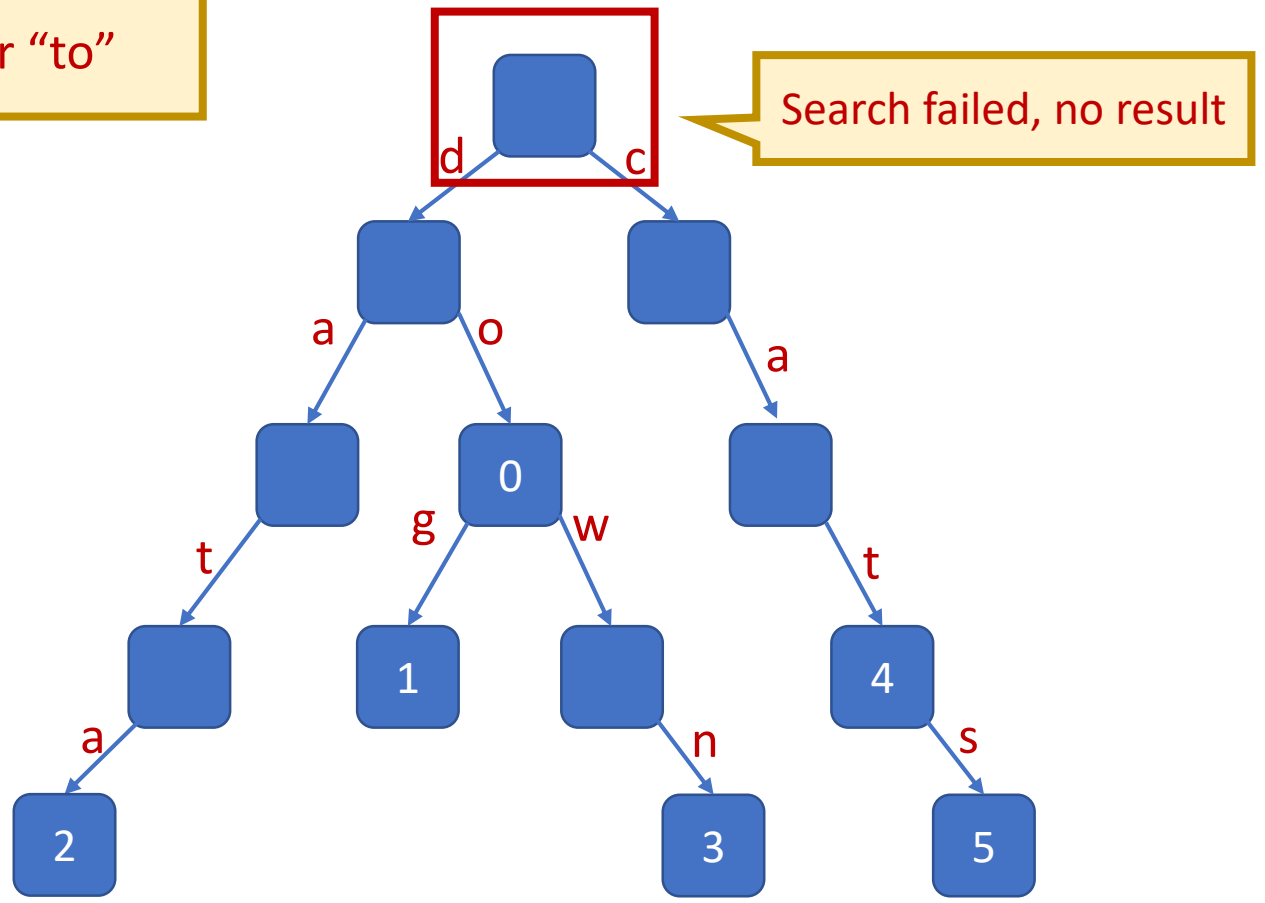
do:0
dog:1
data:2
down:3
cat:4
cats:5



Radix Trie

Search for “to”

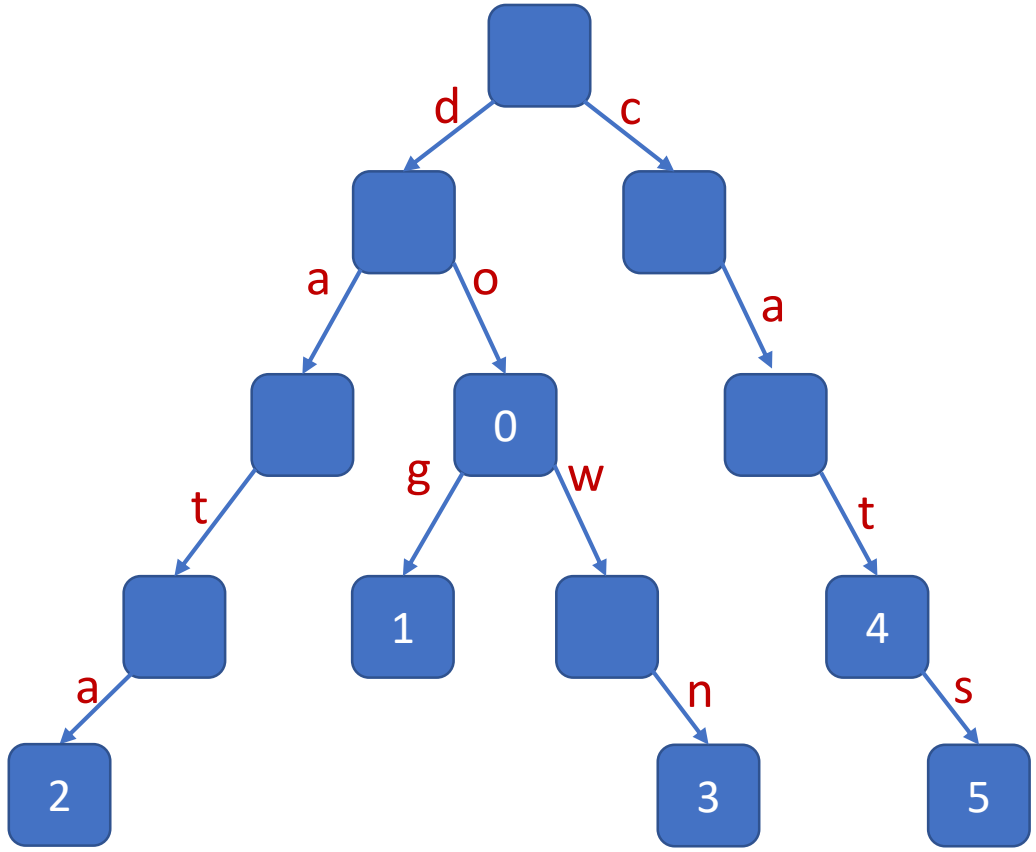
do:0
dog:1
data:2
down:3
cat:4
cats:5



Radix Trie

- Can we optimize the tree?

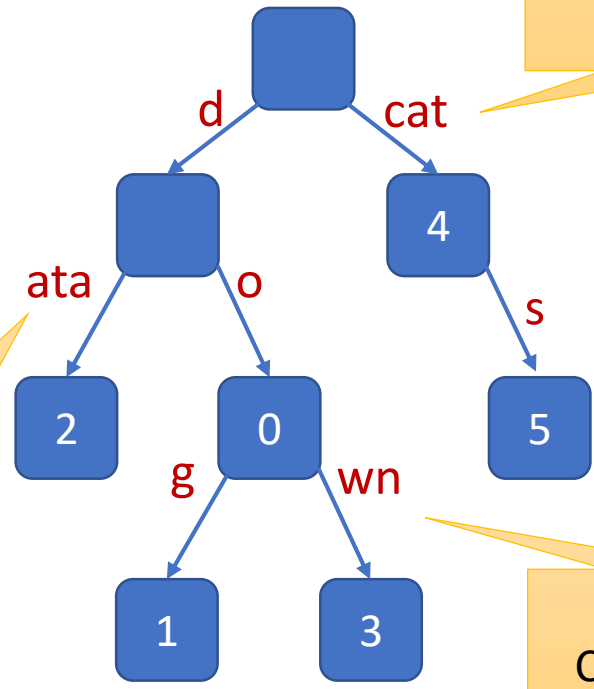
do:0
dog:1
data:2
down:3
cat:4
cats:5



Radix Trie

- Can we optimize the tree?
  - Combine some edges

do:0
dog:1
data:2
down:3
cat:4
cats:5



Combine the edges

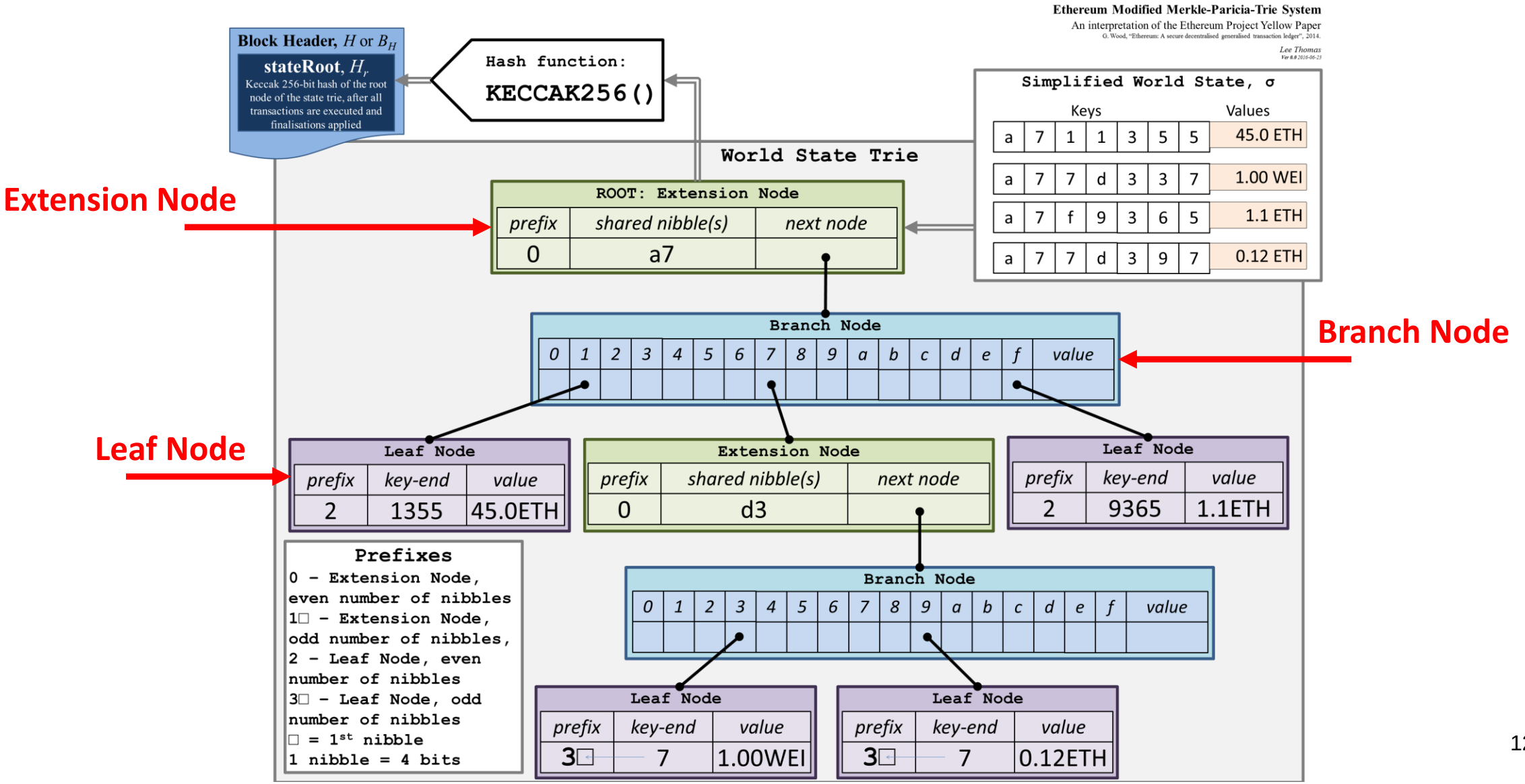
Combine the edges

Combine the edges

Patricia Trie

- PATRICIA=“Practical Algorithm To Retrieve Information Coded In Alphanumeric”
- It is an optimized Trie with lower tree height
- Features
  - Efficient look up
  - Short membership proof and non-membership proof

Patricia Trie

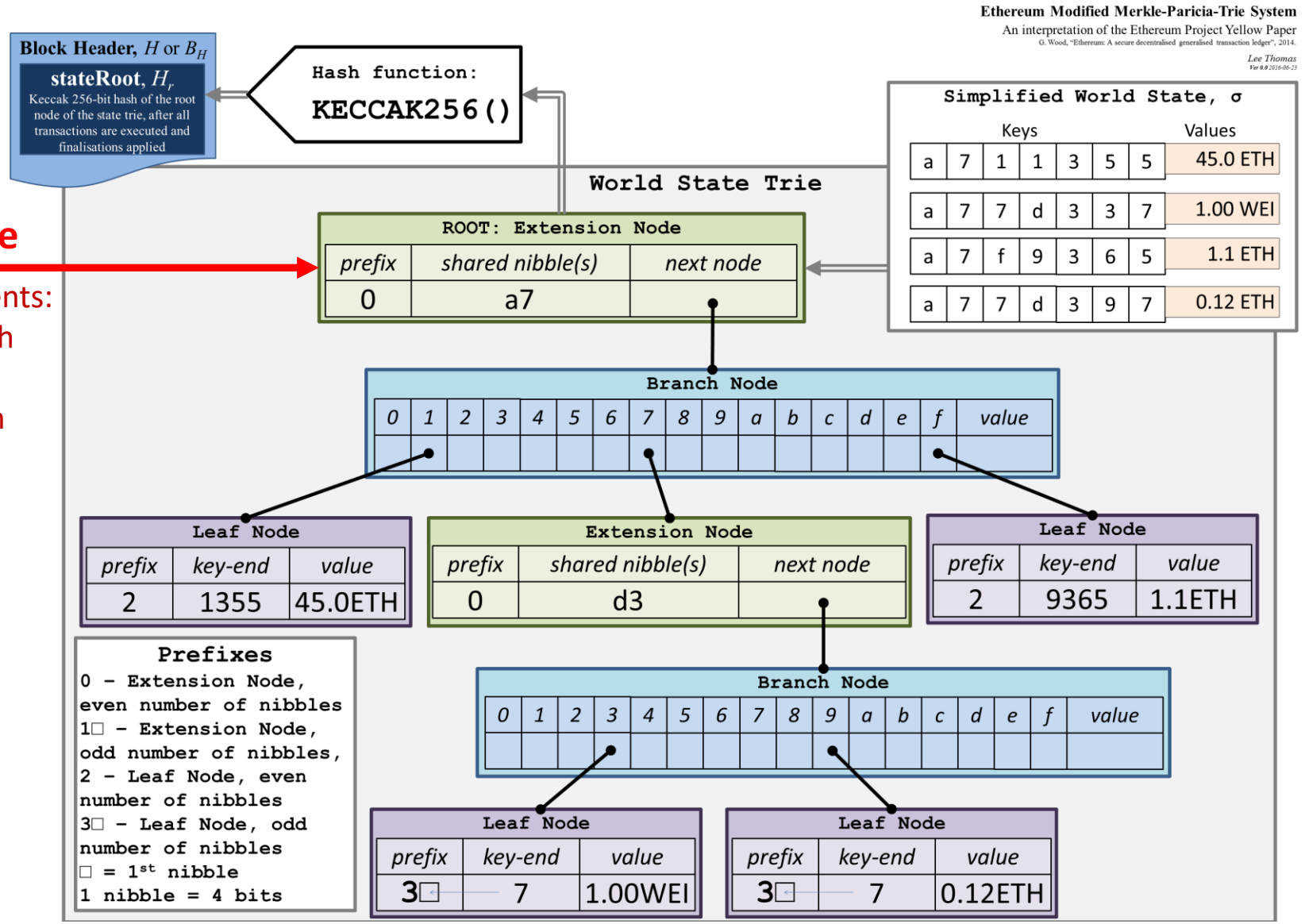


Patricia Trie

Extension Node

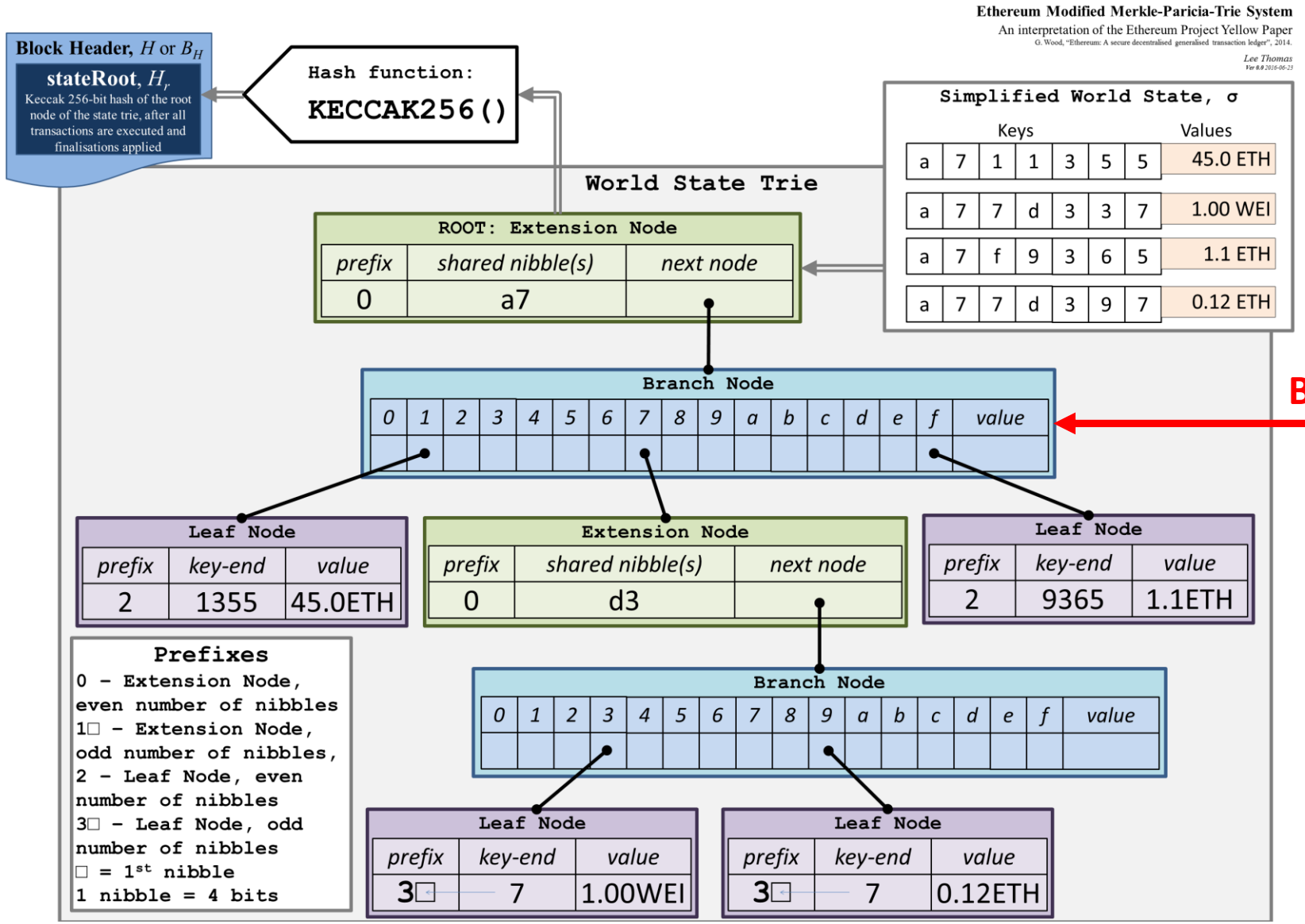
Contains two elements:

- Key: encode path
- Value: path of following branch node



0: 0000
1: 0001
2: 0010
3: 0011

Patricia Trie

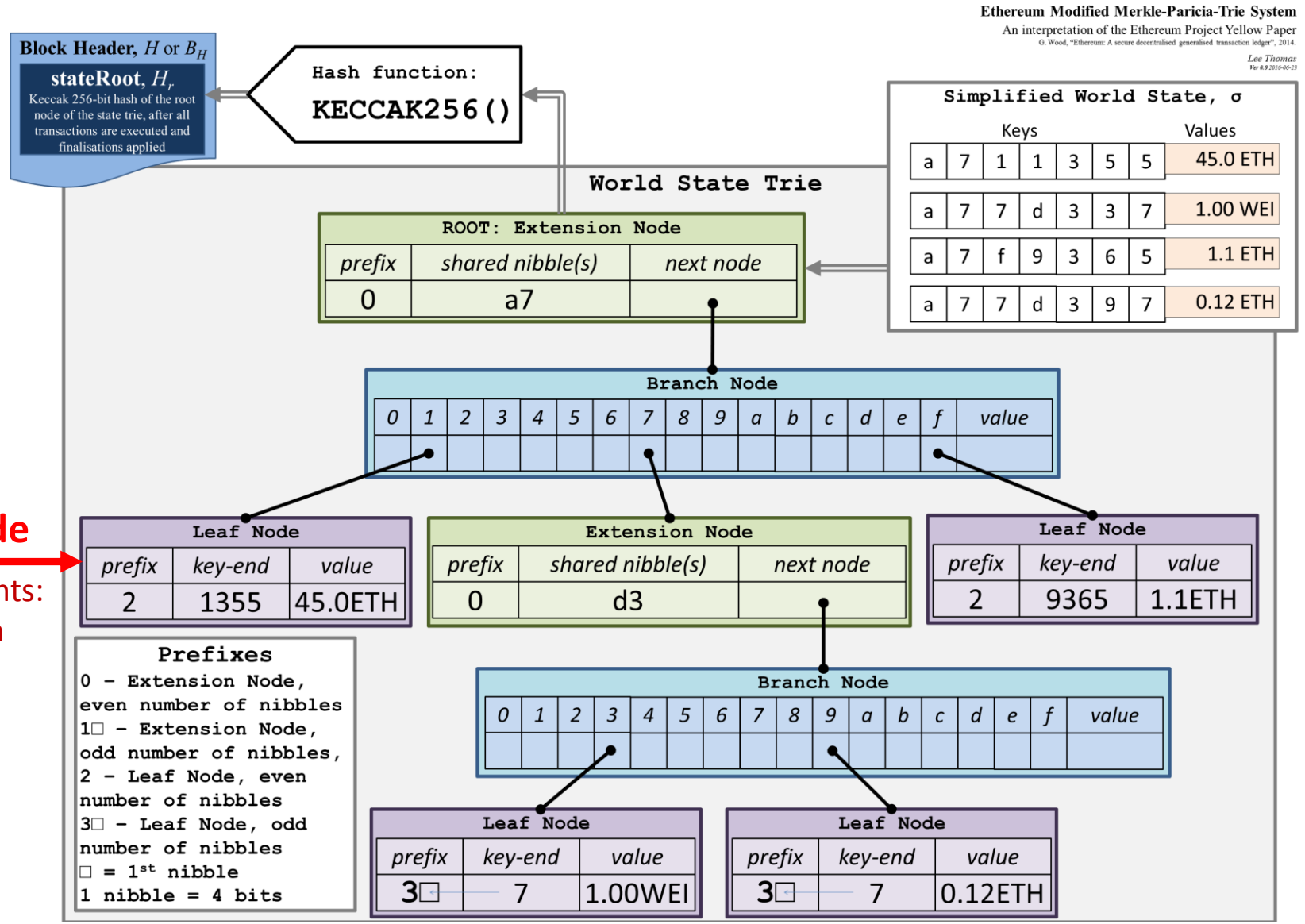


Branch Node

Contain 17-elements:

- Point to the next level

Patricia Trie



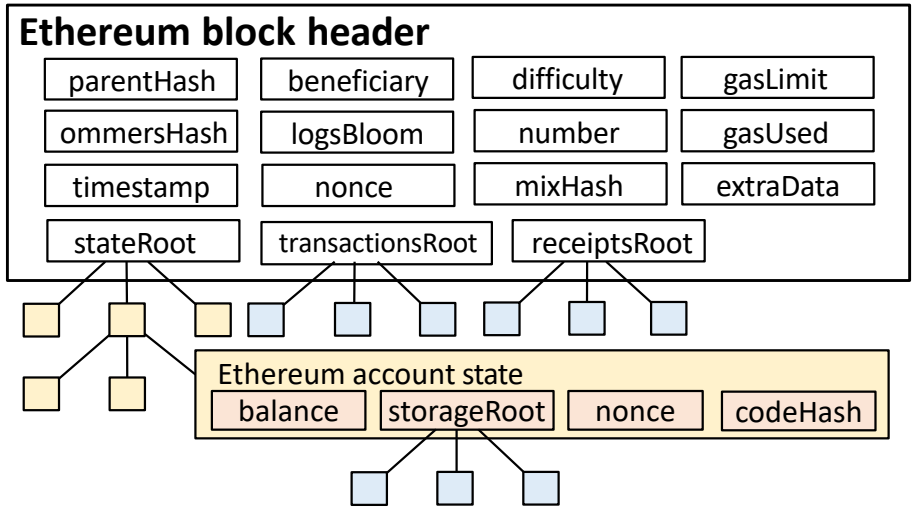
Leaf Node

Contains two elements:

- Key: encode path
- Value: data of account

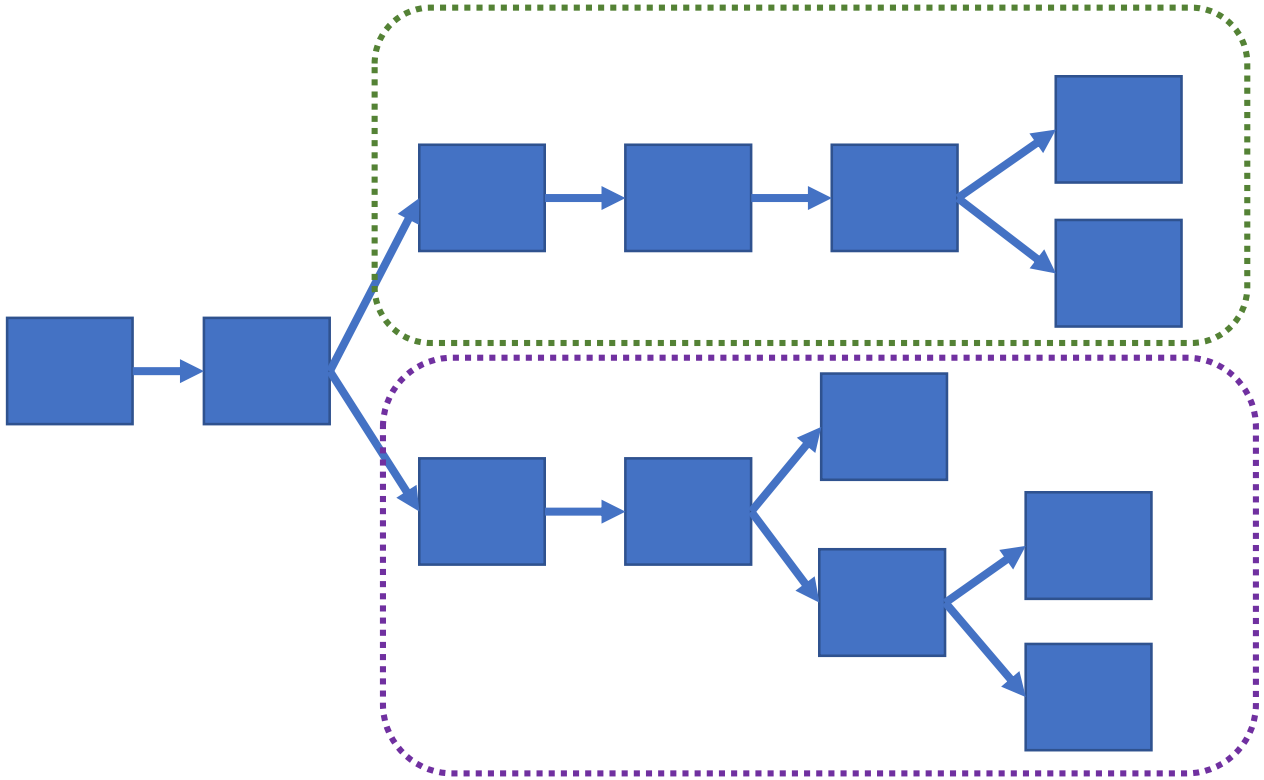
Patricia Trie in Ethereum

- All Merkle Hash Trees in Ethereum use Merkle Patricia Trie
- Three types of tries
  - State Trie
    - include all accounts in ETH
    - balance and storage
  - Transaction Trie
    - include Txs in this block
  - Receipts Trie
    - include transaction's receipts
    - such as logs and gas used



Ethereum Consensus

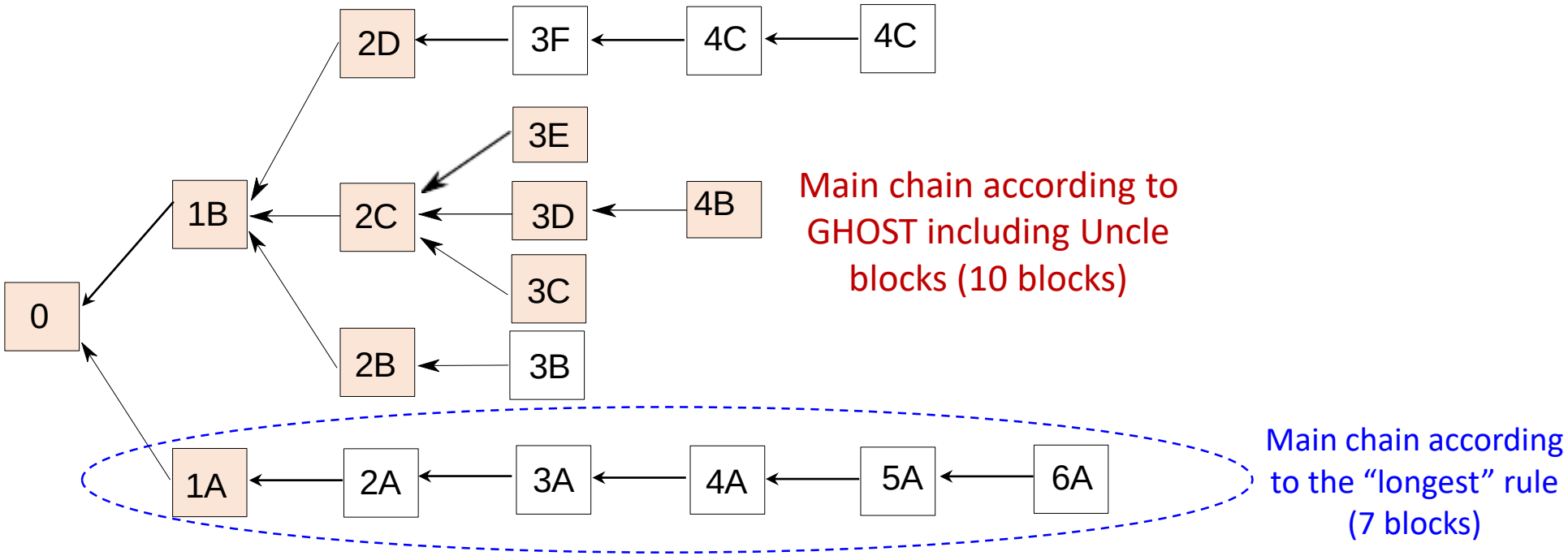
- Initially, use PoW with faster block generation rate (~14s)
- In 2022, switch to PoS
- Faster block generation rate incurs more forks!



Which chain should we choose?

Ethereum's Main Chain Selection

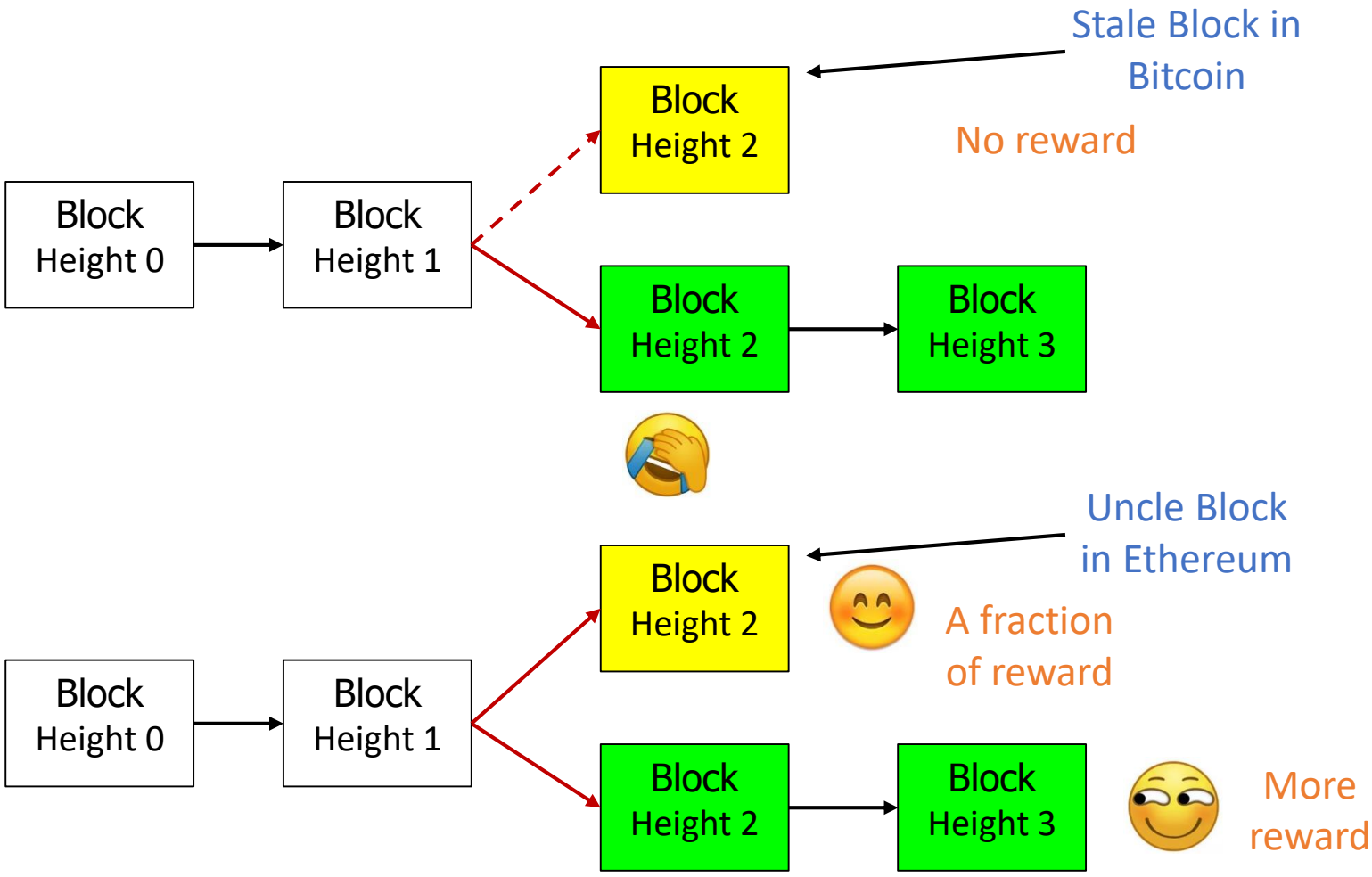
- Greedy Heaviest Observed Sub-Tree (GHOST) protocol proposed by Sompolinsky and Zohar in December 2013
- Ethereum uses a simpler version of GHOST



Ethereum Block Reward

- Recall: Bitcoin only rewards the new block mined in the main chain
  - Orphan blocks in Bitcoin don't contribute to longest chain rule-based consensus
- In GHOST, orphan blocks ("uncles") are counted when determining the heaviest sub-tree
  - Incentivize the honest-but-not-luck works
- Reward stale block miners and also miners who include stale block headers
  - Rewarded stale blocks are called ***uncles*** or ommers
  - Transactions in uncle blocks are invalid
  - Only a fraction of block reward goes to uncle creator; no transaction fees
- Block = (Block Header, Transaction List, Uncle Header List)
  - *ommersHash* in block header is hash of uncle header list

Example: Uncle Incentive in Ethereum



Ethereum Block Reward

- Normal Block Reward
  - Intrinsic reward: 5 ETH (now reduced to 2 ETH)
  - All transaction fees in the block
  - If include uncle blocks, (5 ETH / 32 = 0.15625 ETH) for each uncle block

- Uncle Block Reward
  - (Uncle height + 8 – height of block including this uncle) * intrinsic reward (5 ETH) / 8

Block Reward Examples

Height:	< Prev 4222300 Next >
TimeStamp:	4 mins ago (Aug-31-2017 05:05:31 AM +UTC)
Transactions:	107 transactions and 56 contract internal transactions in this block
Hash:	0xfe8c1080bfa54fc8396f739b73e7d47f9f1ad947497ab191a836d0107edfa75e
Parent Hash:	0x479ea5613fdc054e5a98b8da682b4e880c6d73a7a1277645812dd0493f3fd621
Sha3Uncles:	0xc31bbd9e8088f3c7c596d15d3ffd431609875970318af96b6dac3647d2fc65b6
Mined By:	0x829bd824b016326a401d083b33d092293333a830 in 75 secs
Difficulty:	2,255,032,776,672,791
Total Difficulty:	813,558,265,078,432,542,096
Size:	26592 bytes
Gas Limit:	6,712,390
Gas Used:	6,697,815
Nonce:	0x883206036c1fabd23b
Block Reward:	5.594337168043699381 Ether (5 + 0.281837168043699381 + 0.3125)
Uncles Reward:	8.75 Ether (2 Uncles at Position 0, Position 1)
Extra Data:	ä ƒå½©ç¥žä»™é±¼ (Hex:0xe4b883e5bda9e7a59ee4bb99e9b1bc)

All transaction fees

Intrinsic reward
5 ETH

Include 2 uncles
(2*0.15625)

Block Reward Examples

Uncle Information

Uncle Height:	4222271
Uncle Position:	0
Block Height:	4222272
Hash:	0x1c2cbba0403f1079dcdb70e5971a87ce0fbc03d4572be30e2d17e4e4a0f136d5
Parent Hash:	0x0dfe11b91ccb68294a2b60ed574398b979673fb888c3fe2bc0cbbff1175d3e82
Sha3Uncles:	0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
Mined By:	0x829bd824b016326a401d083b33d092293333a830 in 19 secs
Difficulty:	2,258,524,587,473,917
Gas Limit:	6,735,996 Wei
Gas Used:	3,846,939 Wei
TimeStamp:	32 mins ago (8/31/2017 4:53:07 AM)
Uncle Reward:	4.375 Ether

Uncle height

Height of block including this uncle

(4222271 + 8 - 4222272) * 5 / 8 = 4.375 ETH

Ethereum Mining

- Ethash Proof of Work
  - Keccak-256 (SHA3 variant)
  - Memory-hard computation
  - Memory-easy validation
  - Cannot exploit ASIC
- Mining similar to Bitcoin

```
nonce = rand()
while (SHA3(block,nonce) * difficult > threshold
      nonce++
return nonce
```

Ethereum Mining

- **Difficulty adjustment**
  - After every block (vs. after 2016 blocks in bitcoin)

```
Block_diff = parent_diff  +    parent_diff / 2048 *
             max(1 - (block_timestamp - parent_timestamp) / 10,-99) +
              int(2**((block.number / 100000) - 2))
```

  - If the difference (`block_timestamp - parent_timestamp`) is
    - < 10 secs, adjust upwards by `parent_diff / 2048 * 1`
    - 10 - 19 secs, unchanged
    - >= 20 seconds, adjust downwards from `parent_diff/ 2048 *-1` to

`parent_diff / 2048 *  -99`

Ethereum PoS Transition

- Ethereum is moving to Proof of Stake (PoS) consensus (ETH 2.0 phase 1)

  PoS does not incur huge computation resource and energy consumption

  - Also reduce 51% attack and fast TX validation
    Disadvantage: may be more centralized

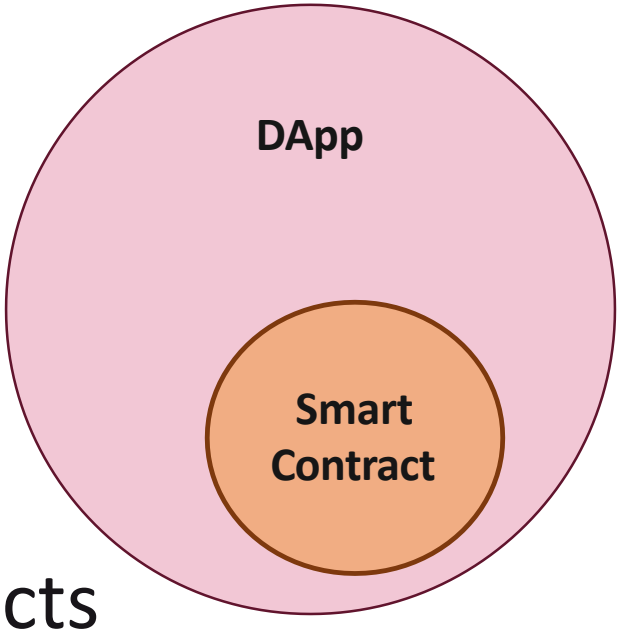
- Miners become “validators” and deposit to an escrow account

- The more escrow a miner deposit, the higher chance it will be chosen to mint next block

- Lose deposit if minting a block with invalid transactions

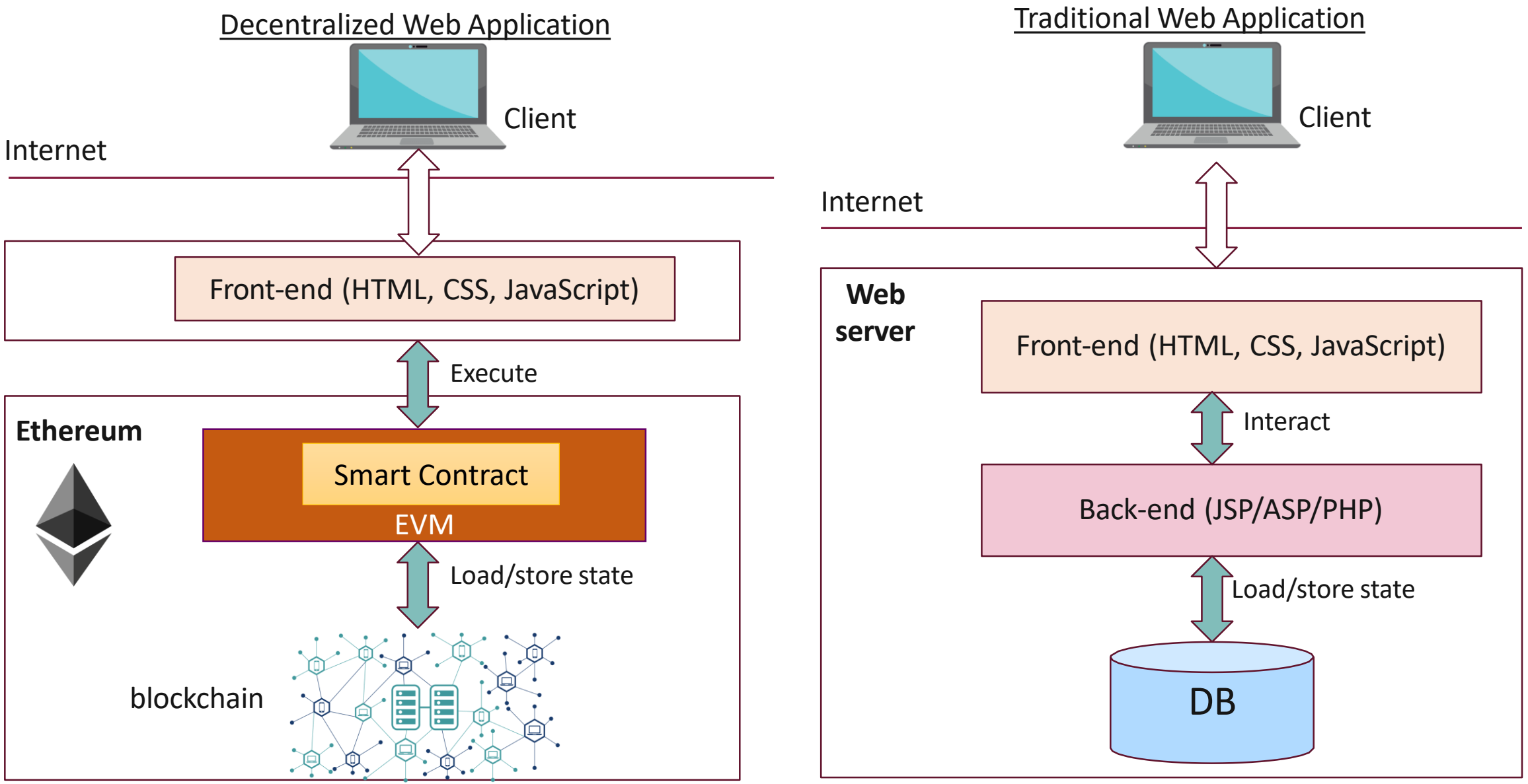
DApp vs. Smart Contract

- DApp is a complete application containing
  - Front-end (e.g., GUI)
  - Back-end (e.g., blockchain)



- Smart contract is only a part of DApp that interacts with the blockchain

Dapp vs. Centralized App



Building DApp

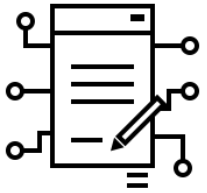
- Main principles to develop a DApp
  - **Develop Front-end:** create app's user interface

Front-end 

  - **Add library:** to connect front-end with wallet and blockchain User's wallet connect to the network and send TXs

library 

  - **Write smart contract:** contains your app's core functions, including anything that modifies user's wallet "contents"

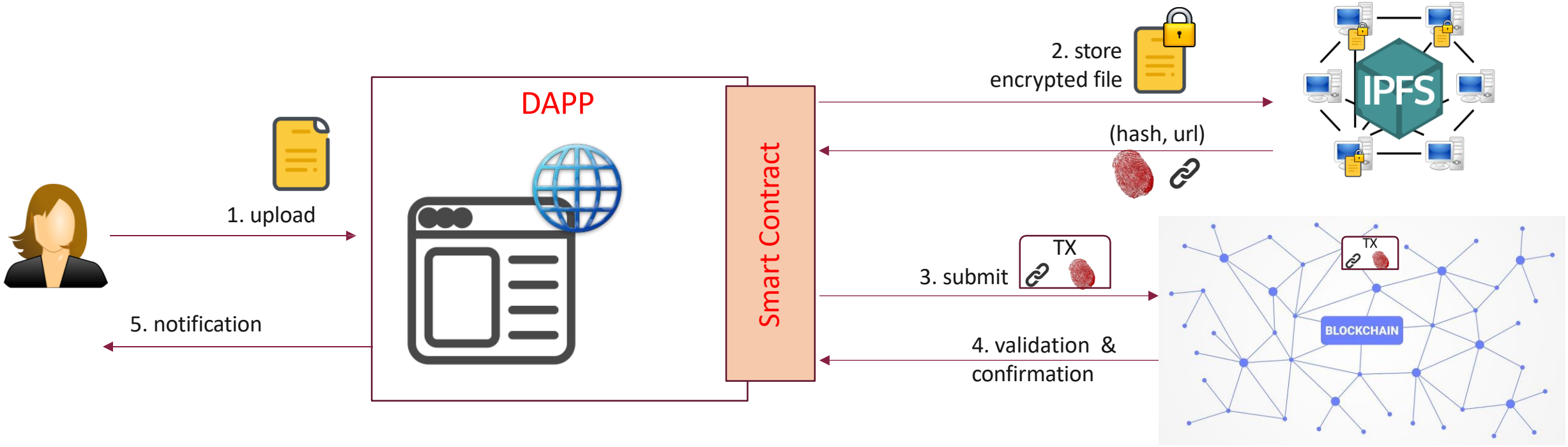
Smart contract 

  - **Deploy:** deploy smart contract to the blockchain
    - Submit TX containing compiled smart contract without specifying any recipients

Blockchain 

Off-chain Storage

- Sometime data is too large to store directly on blockchain
  - Increase block size, computation (validation) and storage overhead on blockchain nodes
- **Solution**: store data content off chain, and its hash and address on chain
  - Example: IPFS, Swarm, Filecoin



Ethereum Smart Contract

- Programming language: Solidity
  - Contract-oriented
  - Syntax similar to Javascript
- A contract is similar to a class in Object Oriented Programming
  - State variables
  - Functions (methods)
  - Events
- Types
  - Integer
  - String
  - Array
  - Mapping
  - …

Example

- Let’s write a simplest form of a cryptocurrency using Solidity
- Requirement
  - Anyone can transfer money to each other
  - The minter can mint some money and transfer to others
  - A sender cannot transfer money that exceeds the owned one (double spending)
  - There should be an event mechanism to notify the state changes

Example

- Components
  - Public storage
    - minter with address type
    - balances with mapping (address => uint) type
  - Function
    - constructor: initialize
    - mint(receiver, amount
      and transfers it to the receiver
    - send(receiver, amount): the sender sends some money to the receiver
  - Event
    - Sent(from, to, amount): log elements of from, to, and amount

The require function call defines conditions that reverts all changes if not met

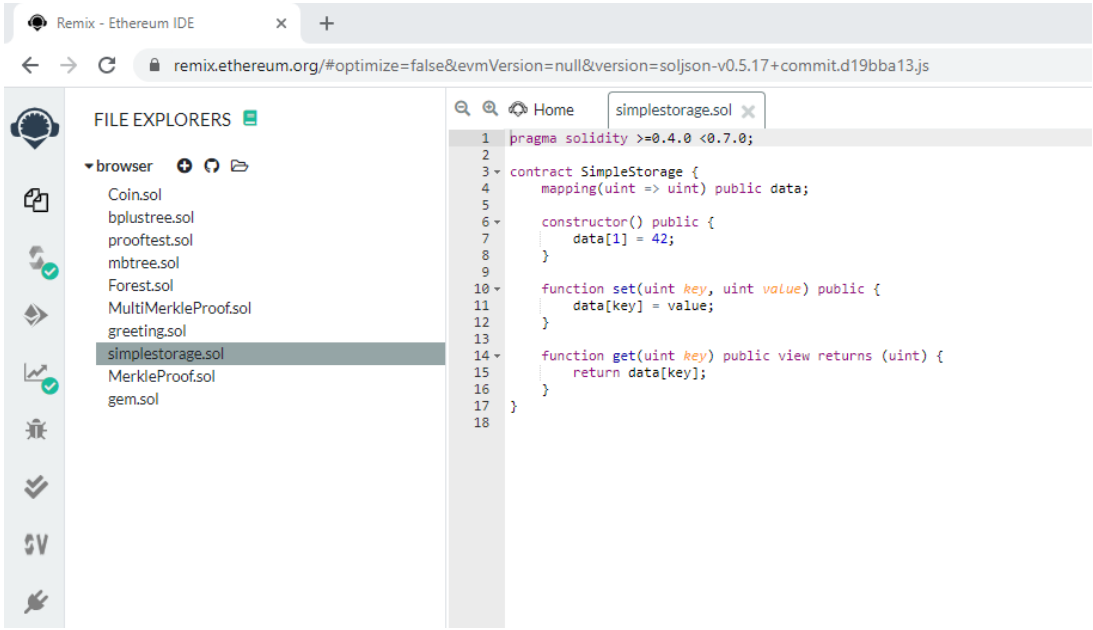
This ensures that the sender has enough money to transfer to the receiver

Emit an event after the successful money transfer

```
pragma solidity >=0.5.0 <0.7.0;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping (address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent(address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() public {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Ethereum Smart Contract

- Solidity Documentation
  - https://solidity.readthedocs.io/en/develop/index.html
- IDE
  - A web-based IDE: https://remix.ethereum.org/



Applications Built on Ethereum

https://www.augur.net/

Golem allows users to rent-out their computing power or develop and sell their softwares.

https://golem.network/

4G CAPITAL

4G Captial provides access to credit for small business growth in Africa.

question.

http://www.4g-capital.com/

Ethlance

Ethlance is the first job market platform built on Ethereum blockchain.

https://ethlance.com/

AA

Ampliative Art allows artists to create their own galleries and exhibit their work for free.

http://www.ampliativeart.org/

Companies are starting to accept Ethers

Summary

- In Permissionless Blockchains, participation is open to the public with a fully decentralized network.
- Ethereum is a decentralized platform that runs smart contracts or dApps, without downtime, censorship, fraud or third party interference.
- Smart contracts are executed on Ethereum Virtual Machine (EVM).
- Uncle Incentive: reward stale block miners and also miners who include stale block headers
  - Rewards for normal block and uncle block are different
- Each operation in a transaction execution costs some gas
- Main chain selection via GHOST protocol

Summary

- Ethereum accounts: Externally owned accounts and Contract accounts
- Ethereum transactions: Contract creation transaction and Message call transaction
- Altcoins are digital money created using encryption techniques with public blockchain.
- Tokens are blockchain-based abstractions that can be owned and represent assets, currency, or access rights.
- A company can raise funds to create a new coin, app, or service through ICO.

References

- Mastering Ethereum, by Gavin Wood, Andreas M. Antonopoulos
- Ethereum White paper: *https://github.com/ethereum/wiki/wiki/White-Paper*
- Ethereum Yellow paper: *https://ethereum.github.io/yellowpaper/paper.pdf*
- Ethereum Wikipedia Article: *https://en.wikipedia.org/wiki/Ethereum*
- A Prehistory of the Ethereum Protocol: *https://vitalik.ca/general/2017/09/14/prehistory.html*
- Ethereum announcement on Bitcointalk: *https://bitcointalk.org/index.php?topic=428589.0*