



CPSC 3640/5400: Decentralized Payments, Contracts, and Finance for Humans and AI

Lecture 7: Ethereum

Instructor: Prof. Fan Zhang

Yale University

Fall 2026

How is Lab I going?

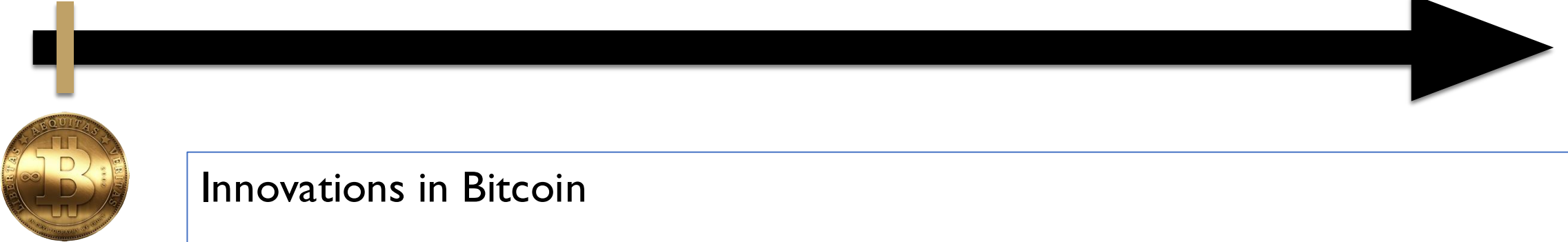
- If you need an extension, talk to me after the class.

Late submission policy:

- Definition: One late day equals a 24-hour extension.
- Each student gets one free late day.
- Each non-free late day incurs a 10% grade reduction.
- Extensions granted by deans or accommodation offices will be honored

Recap: Bitcoin

2009



Innovations in Bitcoin

- **Public append-only ledger** secured by decentralized network
- An **asset** (BTC) with code-enforced monetary policy
- Innovative **payment mechanisms**: UTXO model, Bitcoin Script

Bitcoin vs Ethereum

User Facing Tools

Bitcoin Core, wallets

Application Layer

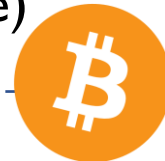
Bitcoin, payment channels, atomic swaps
...

Execution Layer

UTXO model, Scripts

Consensus Layer

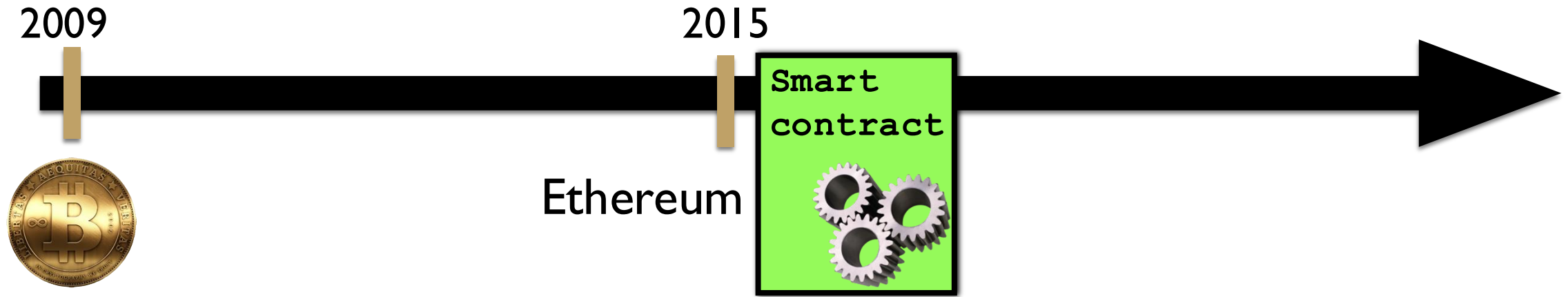
Nakamoto Consensus
(Proof of Work + longest chain rule)



Bitcoin Script is intentionally limited

- Bitcoin was geared primarily towards **payments**
 - Difficult to maintain state in multi-stage contracts
 - Difficult to enforce global rules on assets
- A simple example: rate limiting.
 - Desired policy: can only transfer 2BTC per day out of all my UTXOs
 - Can't be done with Bitcoin Scripts! UTXOs are independent and unrelated from each other
- All of these can be done easily with Ethereum/smart contracts

Ethereum



“Smart contracts”: programs run on a blockchain

Bitcoin vs Ethereum

User Facing Tools

Application Layer

Execution Layer

Consensus Layer

Bitcoin Core ...

Bitcoin, payment
channels, atomic swaps
...

UTXO model, Scripts

Nakamoto Consensus



Web3 apps, wallets,
RPC...

Smart contracts

Account Model, EVM

ETH2: PoS + Gasper

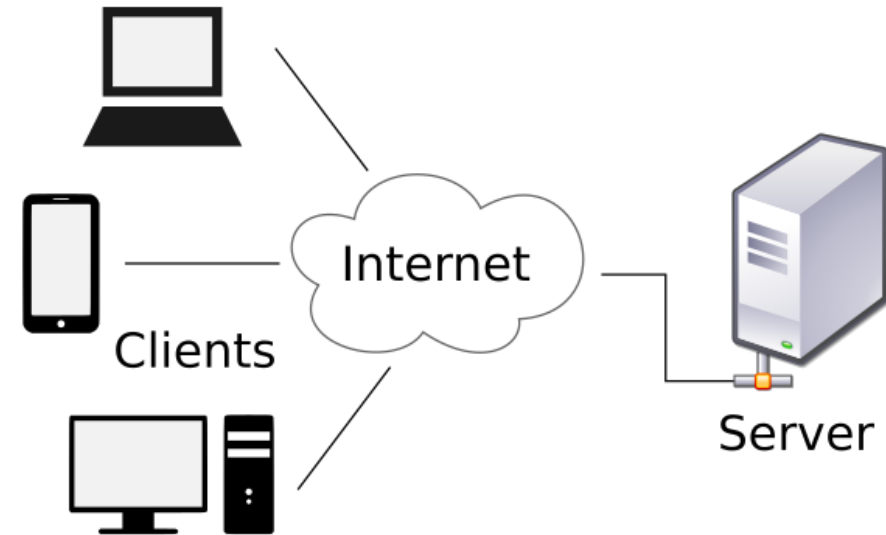


From the lens of Internet evolution



https://en.wikipedia.org/wiki/IBM_mainframe

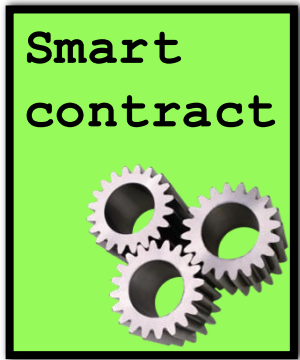
Mainframe w/o Internet can only run local computation



Internet enables **digital services** (accounts for 70 percent of global GDP)

From the lens of Internet evolution

- Smart contracts are digital services with built-in security properties: integrity, verifiability, censorship-resistance, etc.



Decentralized Apps (DAPPs) are a new kind of digital services (web3?)

Today: Transaction model & mechanism

Bitcoin

- Blockchain tracks UTXOs
- Transactions consume UTXOs and create UTXOs.



Ethereum

- Blockchain tracks per-account state.
- Transactions run **user programs**
- ... that could change state
- ... that could create more programs



Ethereum: the account model

- Blockchain stores the following for **each account**

address	code	storage	balance	nonce
---------	------	---------	---------	-------

- There are two types of accounts
 - 1) Externally Owned Account (EOA)
 - 2) Smart contract account
-
- EoA is externally managed by the holder of its secret key; smart contract accounts are deterministically managed by a smart contract.

EoA account

address	code	storage	balance	nonce
H(pub_key)	∅	∅	ETH balance	#txns sent

- One of vitalik's EoA accounts:
[0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045](https://etherscan.io/address/0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045)
- (This link takes you to Etherscan, a popular blockchain explorer)

Sidebar: Ethereum units

- **Ether**: the primary unit
- **Wei**: The smallest denomination of ether ($1 \text{ ETH} = 10^{18} \text{ wei}$). It's named after Wei Dai, a cryptography activist.
- **Gwei**: Also known as gigawei, or 10^9 wei . $1 \text{ ETH} = 10^9 \text{ gwei}$.
 - Transactions fees are often denominated in gwei.
 - E.g., lately the average fee is around $\sim 10 \text{ gwei}$ per “gas unit”
- There are also Finney, Szabo, Shannon named after other scientists

Smart contract account

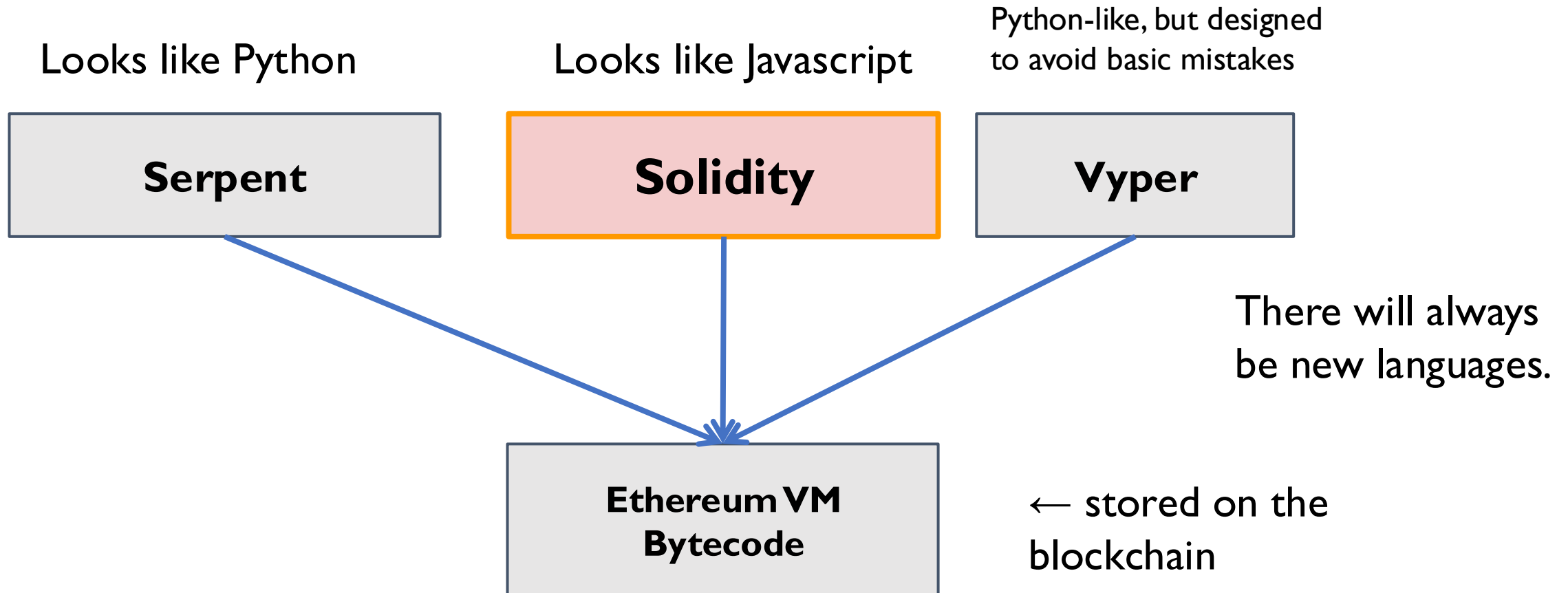
- In addition to EoA account state, smart contract has
 - **Code**
 - **Storage**

address	code	storage	balance	nonce
H(creator, nonce)	EVM bytecode	Account storage root	ETH balance	#contracts created

Example: USDC (the stablecoin):

<https://etherscan.io/token/0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48>

Code: Ethereum Languages



Example: Solidity

```
pragma solidity ^0.5.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

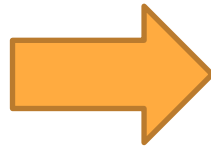
Example: Solidity

```
pragma solidity ^0.5.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```



```
PUSH1 0x80 PUSH1 0x40 MSTORE
CALLVALUE DUP1 ISZERO PUSH2 0x10
JUMPI PUSH1 0x0 DUP1 REVERT
JUMPDEST POP PUSH1 0xE6
0x1000000000000000000000000000000000
00000000000000000000000000000000 SWAP1
DIV DUP1 PUSH4 0x60FE47B1 EQ
PUSH1 0x48 JUMPI DUP1 PUSH4
0x6D4CE63C EQ PUSH1 0x7F JUMPI
JUMPDEST PUSH1 0x0 DUP1 REVERT
JUMPDEST CALLVALUE DUP1 ISZERO
JUMPDEST DUP2 ADD SWAP1 DUP1
DUP1 CALLDATALOAD SWAP1 PUSH1
0x20 ADD
0x85DE0029000000000000000000000000
00000000000000000000000000000000
```

Example: Solidity

```
pragma solidity ^0.5.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

```
PUSH1 0x80 PUSH1 0x40 MSTORE  
CALLVALUE DUP1 ISZERO PUSH2 0x10  
JUMPI PUSH1 0x0 DUP1 REVERT  
JUMPDEST POP PUSH1 0xE6  
0x1000000000000000000000000000000000  
00000000000000000000000000 SWAP1  
DIV DUP1 PUSH4 0x60FE47B1 EQ  
PUSH1 0x48 JUMPI DUP1 PUSH4  
0x6D4CE63C EQ PUSH1 0x7F JUMPI  
JUMPDEST PUSH1 0x0 DUP1 REVERT  
JUMPDEST CALLVALUE DUP1 ISZERO  
JUMPDEST DUP2 ADD SWAP1 DUP1  
DUP1 CALLDATALOAD SWAP1 PUSH1  
0x20 ADD  
0x85DE002900000000000000000000000000  
00000000000000000000000000
```

```
6080604052348015610010
57600080fd5b5060e68061
001f6000396000f3fe60806
0405260043610604357600
0357c010000000000000000
000000000000000000000000
00000000000000000000000900
4806360fe47b1146048578
0636d4ce63c14607f575b6
000....
```

Solidity

Assembly Code

Hexadecimal format

Code is immutable

address	code	storageRoot	balance	nonce
---------	------	-------------	---------	-------

- Code is set at contraction creation time and **cannot be changed**.
- What if code has bugs???
- Solutions: “updatable pattern”

Smart contract storage data structure

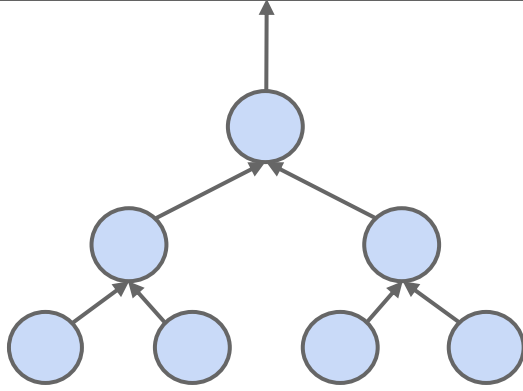
address	code	storageRoot	balance	nonce
---------	------	-------------	---------	-------

Key (32 bytes)	Value (32 bytes)
000...000	000.000
000...001	
...	
FFF...FFF	

- storageRoot: “hash” of a dictionary (32-byte key/value pairs)
- Infeasible to hash 2^{256} entries
 - going to take 10^{59} years
- Observation: dictionary is sparse (most cells are zeroes)
- Introducing *Merkle Patricia Trie*

Each contract has a dictionary of 2^{256} entries (initialized to zero)

state root



The global **World State Trie**

Key: Address

Value: (code, storageRoot, balance, nonce)

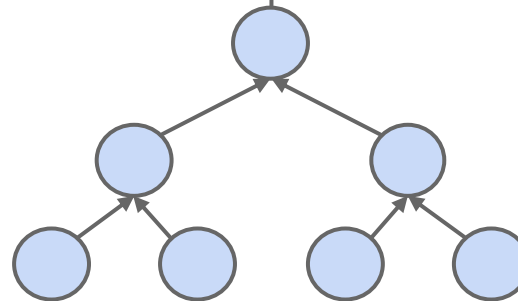
address

code

storageRoot

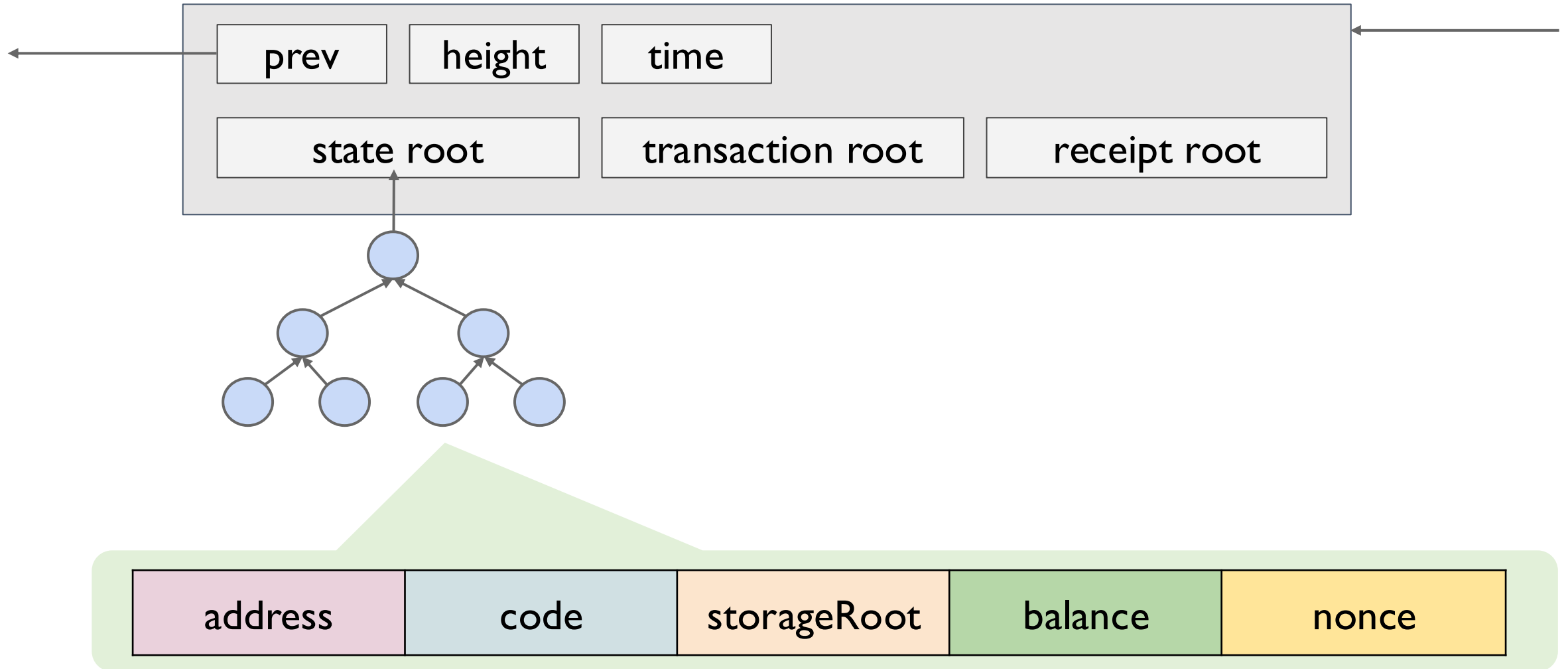
balance

nonce



Each smart contract account has a **storage trie** (over dictionary)

Ethereum blockchain structure



So far so good?

- Each smart contract has
 - **Code**: the logic set at creation time
 - **State**: a dictionary with up to 2^{256} slots
- Code can modify *its own* state
 - Can't modify other smart contracts' state directly
- Next: Transactions (invoking smart contracts)

Ethereum Transactions

Transactions

- Three types of transactions
 - **creates** smart contracts
 - **invokes** an existing one
 - simple **transfer** of Ether (doesn't invoke any smart contract)
- Sharing the same format:

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------

sender

sender's sig and nonce

Transactions

- Three types of transactions
 - **creates** smart contracts
 - **invokes** an existing one
 - simple **transfer** of Ether (doesn't invoke any smart contract)
- Sharing the same format:

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------



- Which EoA to send ether to
- Which contract to invoke

Transactions

- Three types of transactions
 - **creates** smart contracts
 - **invokes** an existing one
 - simple **transfer** of Ether (doesn't invoke any smart contract)
- Sharing the same format:

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------



if “to” is a smart contract,
provide input data

Transactions

- Three types of transactions
 - **creates** smart contracts
 - **invokes** an existing one
 - simple **transfer** of Ether (doesn't invoke any smart contract)
- Sharing the same format:

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------

/

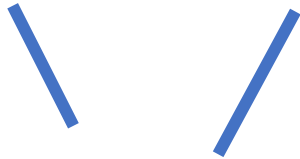
Amount of ether to
transfer

Transactions

- Three types of transactions
 - **creates** smart contracts
 - **invokes** an existing one
 - simple **transfer** of Ether (doesn't invoke any smart contract)
- Sharing the same format:

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------

transaction fee

Two blue lines originate from the 'gas limit' and 'gas price' fields of the table above and converge towards the 'transaction fee' text below.

Gas in Ethereum is a necessary evil

- What if someone wrote a smart contract with dead loops `while (true) {...}`
- Can't be detected: no program can decide if a program will run forever [proven by Turing in 1936]
- Tx fee solved the problem!
 - Each step of computation costs some fee
 - Computation halts when fee is depleted
- Gas is necessary: gas ensures that all transactions terminate, and miners/validators work are compensated.
- Another reason: to separate high value transactions from low value ones (improving market efficiency)

Every instruction costs gas, examples:

	opcodes	gas cost
Basic operations	ADD, MUL, PUSH, JUMP	2-10
Contract call	CALL, CODECALL, etc.	700
Transaction overhead	n/a	21000
Contract creation	CREATE	$32000 + 200 \times (\text{code size})$

Every byte costs gas, examples:

SSTORE **addr** (32 bytes), **value** (32 bytes)

- zero → non-zero: 20,000 gas
- non-zero → non-zero: 5,000 gas
- non-zero → zero: 15,000 gas **refund**

Refund is given
for reducing
size of
blockchain state

Gas limit, gas price, transaction fees

- How do miners pick what transactions to include?
- User specifies “gas price”, how much to pay for each gas used
- Transaction fee = gas used x gas price

Example: How much would it cost to store 1 MB of new data? (assuming 100 gwei gas price and ETH price at \$3,000)



TX fee = gas usage x price



Gas calculation workflow

- **If** `GAS_LIMIT * GAS_PRICE > caller.balance`, **halt**
- **Deduct** `GAS_LIMIT * GAS_PRICE` from `caller.balance`
- **Set** `GAS = GAS_LIMIT`
- Run code, deducting gas from `GAS` for each op code
- For refunds, add to `GAS`
- After termination, add `GAS` back to `caller.balance`

-
- Prior EIP-1559, user sets `GAS_PRICE`
 - Post EIP-1599 (live on 8/2021), `GAS_PRICE` is set by user + congestion
 - We'll study transaction fee mechanism several lectures later.

Three* types of transaction in Ethereum

type	from	sig, n	to	data	value	startgas	gasprice
create	creator	sig, n	∅	code	start_bal	53,000	?

Three* types of transaction in Ethereum

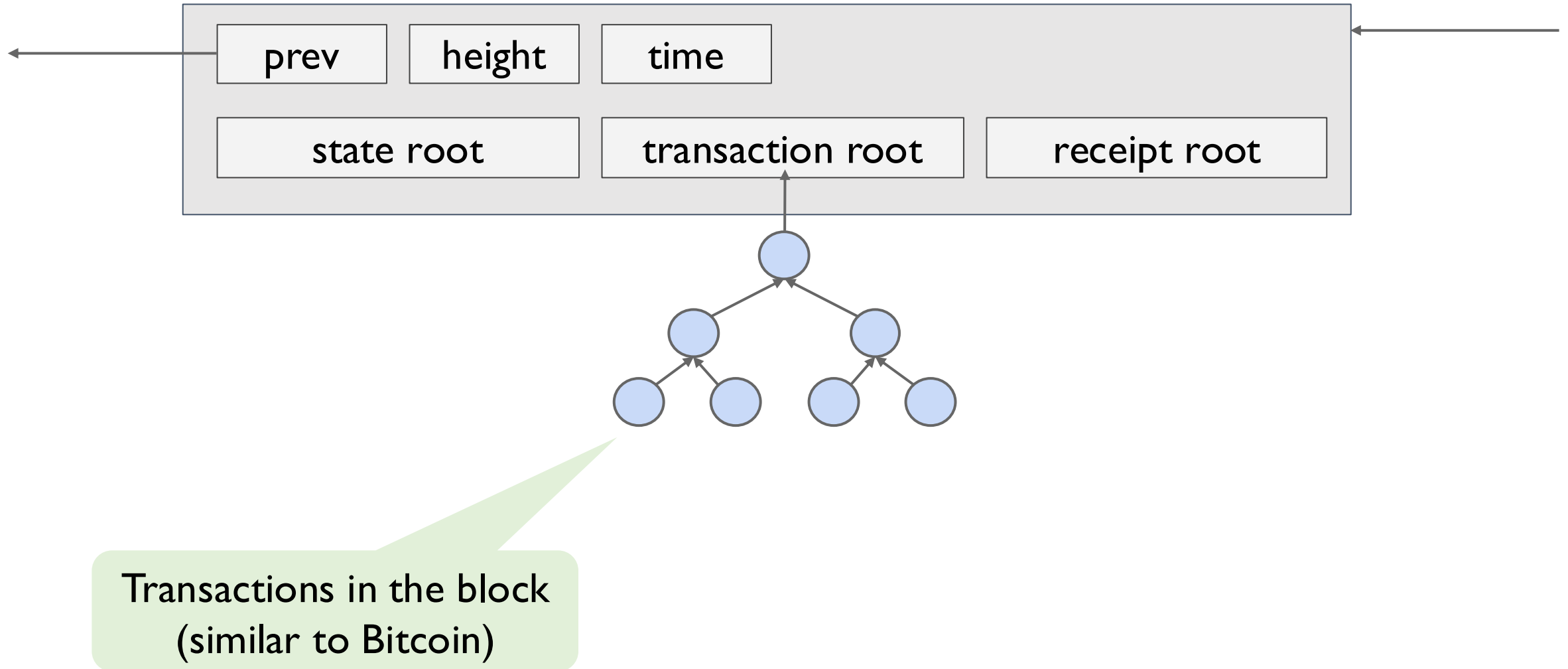
type	from	sig, n	to	data	value	startgas	gasprice
create	creator	sig, n	∅	code	start_bal	53,000	?
send	sender	sig, n	receiver	∅	transfer_val	21,000	?

Three* types of transaction in Ethereum

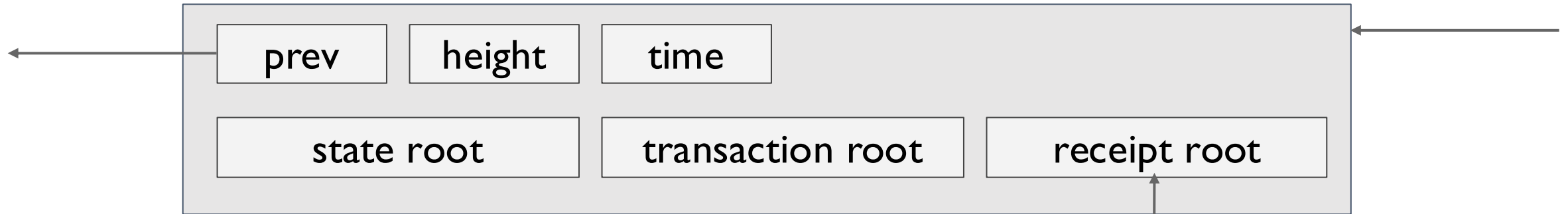
type	from	sig, n	to	data	value	startgas	gasprice
create	creator	sig, n	∅	code	start_bal	53,000	?
send	sender	sig, n	receiver	∅	transfer_val	21,000	?
call	caller	sig, n	contract	func, args	transfer_val	?	?

func selector (method id): first 4 bytes of the hash of function name and type (e.g., Hash("foo(address,uint256)")

Ethereum blockchain structure

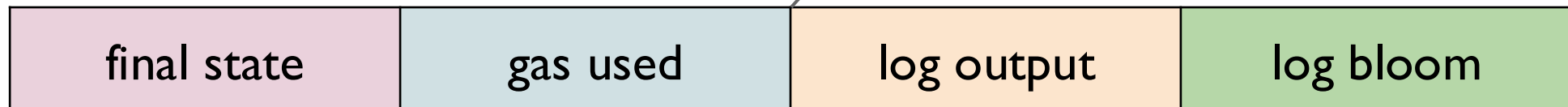


Ethereum blockchain structure



Receipt is the **execution trace** of a transaction:

- The final state after execution
- gas used
- events / logs invoked by the transaction
- a Bloom filter that makes it easy to search events.
- Storing all receipts takes many TBs!



Ethereum in one slide

- States: per-account state stored in Merkle Patricia Trie

address	code	storage	balance	nonce
---------	------	---------	---------	-------

- Transactions

from	sig, n	to	data	value	gas limit	gas price
------	--------	----	------	-------	-----------	-----------

- State Transition
 - Validate signature against “from”
 - Check nonce n against # txns sent so far
 - **If to.code $\neq \emptyset$** : run to.code(from, data, value, gaslimit, gasprice)
 - **else**: transfer “value” to “to”

Hands-on component

- Install Metamask wallet
- Get some testnet (Sepolia) ETH to pay txn fees
- Use a web3 application via Metamask
- Go through steps in <https://cpssc3640.netlify.app/hands-on/course-nft/>