Welcome Back,
*Chain*smokers in the House!

Ethereum WDTDS:

Abstraction & Mental Representation

Alex Xiong

BLOCKCHAIN
AT NTU

Review & Preview

- L1: Bitcoin Overview & Cryptography Basics
  - ↳ Bitcoin motivation, digital signatures, hash functions, data structure (hash pointer, UTXO model, Merkle Tree)
  - ↳ Identity on Blockchain (pseudonymity)
- L2: Nakamoto Consensus & Bitcoin Protocol
  - ↳ Distributed consensus → Proof of Work (PoW) → Bitcoin Protocol (mining + longest-chain win)
  - ↳ Security analysis, incentive of miners, mining pool and forks
- L3: Ethereum
  - ↳ Bitcoin script, limitations → Ethereum: account-based model, EVM, Gas
- L4: Smart Contract & Solidity
  - ↳ Smart contract: why, what and so what
  - ↳ Solidity basic syntax & live example

Review & Preview

"... too many theories!! Doesn't seem I'm learning anything applicable... "

→ more emphasis on hands-on practice during weekly sharing sessions

"... I don't have particular strong CS backgrounds, find many concepts hard to capture… "

→ 1. Education department prepares "questions board";

2. Slack channel : #quora, #stackoverflow, #research, please raise questions and requests;

3. Event calendar (soon available on website)

"... So many new concepts, protocols, I don't know what they are, or why should I care about them..."

→ 1. SoK sharing sessions, covering research frontiers (scalability, security, privacy, formal methods, p2p network, consensus algorithms, key managements etc.)

2. READ!! Publish what you've learned and your thought!! GET A BLOCKCHAIN JOB!

Review & **Preview**

Lecture:

- Ethereum: WDTDS
- Truffle Framework & Testing
- Blockchain Sharding by Zilliqa
- Smart Contract Security
- Web3 library
- ...

Weekly Sharing:

- Basic DevTool
- Smart Contract Lab Sessions (cryptozombies etc.)
- EthFoundation sharing
- Research SoK
- Hacking Smart Contract (ethernaut)
- ...

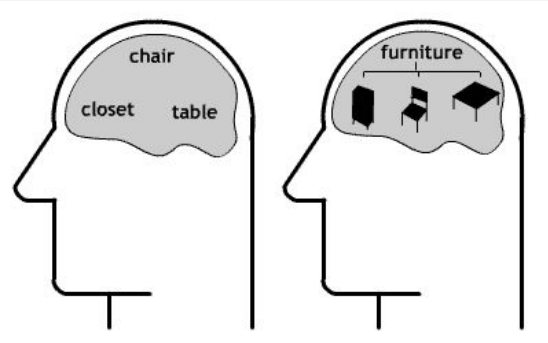
“

Any fool can know, the point is to understand

-- *Albert Einstein*

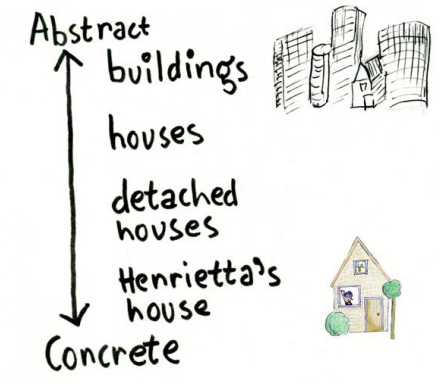
Goals for today

Mental Representation



Macro ↔ Micro View

Concrete Example



Source: Quizlet ; Giphy ; Enchanting Marketing

Satellite-level View: a blockchain?

A distributed, p2p software

Satellite-level View: a blockchain?

A distributed, p2p software

- Dispersedly located, rather than on a single machine.
- Coordinate through passing message across the network.
- Each instance is called a node/peer/client.

Satellite-level View: a blockchain?

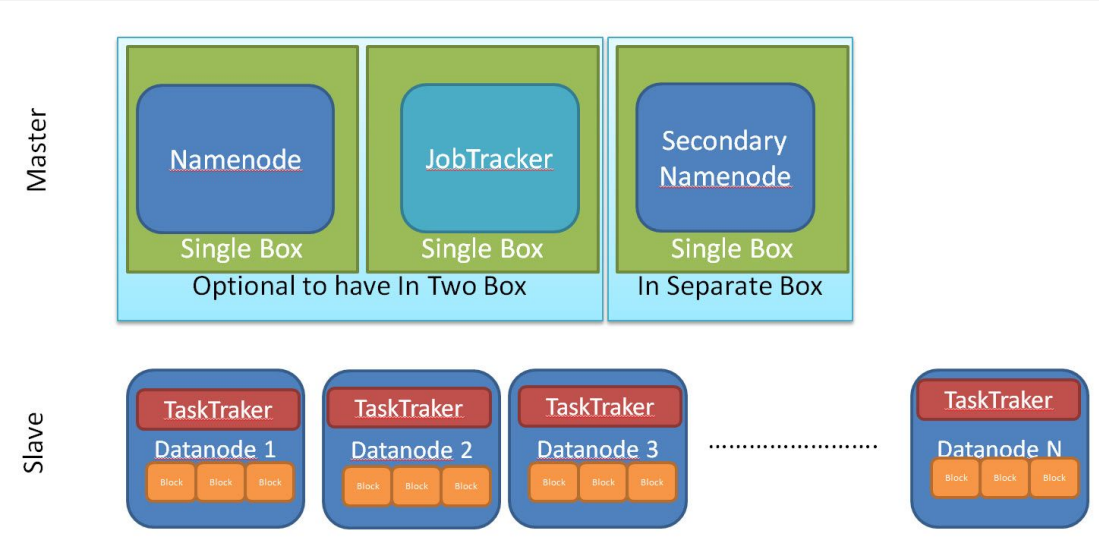
A distributed, p2p software

Same piece of code on each node for all distributed softwares?

Not necessarily.
E.g. Hadoop or MongoDB master-slave model.

Hadoop: everyone doing different jobs

Master & Slave/Worker model:



Source: https://blogs.msdn.microsoft.com/avkashchauhan/2012/02/24/master-slave-architecture-in-hadoop/

Satellite-level View: a blockchain?

A distributed, p2p software

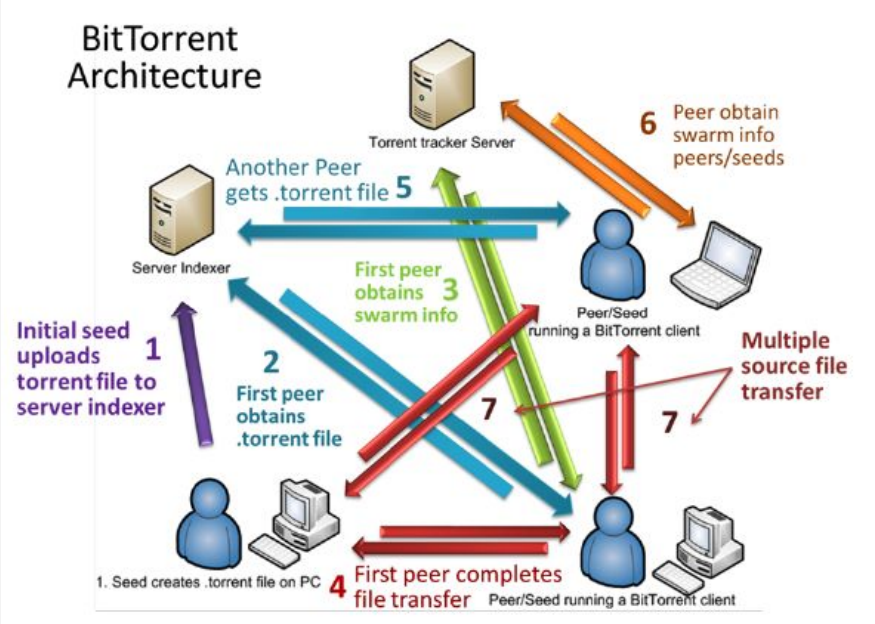
All nodes have the same "view"?

Nope, almost always no! Take a snapshot!

1. *Each node could run different code, thus in charge of different subset of data.*
2. *Even all are running the same code, some could be out of sync (e.g. by going offline).*
3. *Any update takes time to transmit across the network, thus induce a delay.*

BitTorrent: everyone storing different data

One or few resource holders for each file, no one stores everything:



Source: https://www.researchgate.net/figure/Classical-BitTorrent-P2P-Protocol-Sequence_fig1_241972547

Satellite-level View: a blockchain?

A distributed, p2p software

Good, what about Ethereum?

Everyone running the "same"[1] software, responsible for storing the same set of data, but still possibly share different views.

1. Same specs, could vary in actual code implementation. (Geth v.s. Trinity)

Satellite-level View: a blockchain?

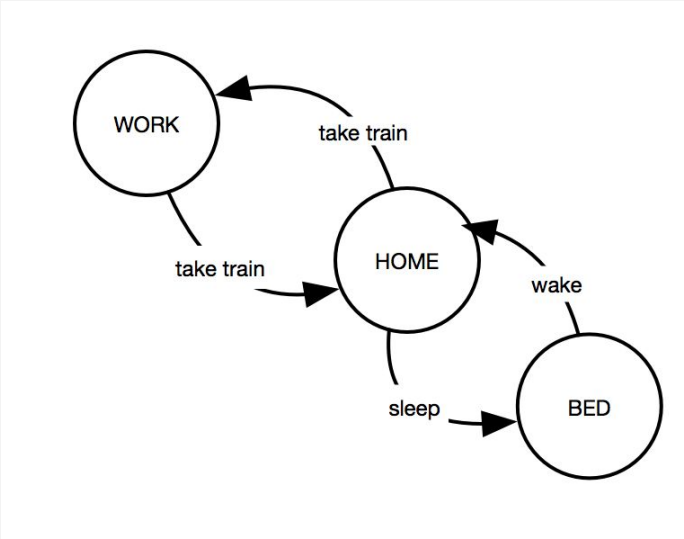
A distributed, p2p software

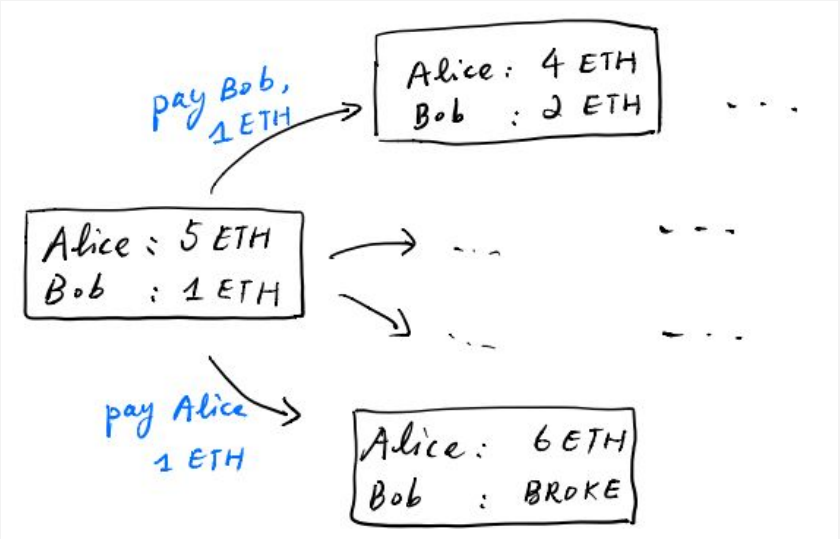
- Not Client-Server paradigm
- Everyone doing the same job:
  - listening to TXs,
  - verifying them,
  - mining new blocks
  - broadcast to the rest
  - verify others' block
- Everyone keeps track of the same data
  - All transaction history (blockchain/ledger)

Zooming in to *Continent-level View* now…

What is a node exactly?

A State Machine, that's all … → State + State transition





Source: https://medium.com/@brianray_7981/tutorial-write-a-finite-state-machine-to-parse-a-custom-language-in-pure-python-1c11ade9bd4

What does a node do?

A State Machine, that's all … → State + State transition

- READ: look up some states.
- WRITE: apply state transition and update the state.

What does a node do?

A State Machine, that's all … → State + State transition

- READ: look up some states.
- WRITE: apply state transition and update the state.

What can be read?

1. *Everything related to every account (e.g. balance, nonce, address etc.).*
2. *Every transactions included.*
3. *Every transaction receipts.*
4. *Some general global blockchain data in block headers (e.g. previousHash, blockHeight, difficulty level etc.)*

In what format should I read them?

Mostly through a JSON wrapper, defined by the JSON RPC spec.

i.e. `web3.eth.blockNumber`
`web3.eth.getBalance('0x12..a')`

What's actually been stored on blockchain are strings of bytes, but most developer tool will parse them into Json object.

Demo time!

Demo :: reading data from blockchain

- Block Explore: https://etherscan.io
- Geth: command line tool
- Ganache-cli: your local blockchain for testing

Make sure node and npm is installed before the lectures.

Reading data from blockchain using ganache-cli

1. Install ganache command line tool: `npm install -g ganache-cli truffle`
2. Install web3 locally in a npm folder: `npm install web3`
3. Make sure your port 8545 is cleaned: `lsof -ti :8545`
    a. If not kill it: `lsof -ti :8545 | xargs kill`
4. Open another terminal and let ganache to run in the background
5. Type in the following:
    a. Start a truffle interactive console: `truffle console`
    b. Initialize web3 to connect to your local host and choosing ganache as your http provider: `web3.setProvider("http://localhost:8545")`
    c. Now follow me!

What does a node do?

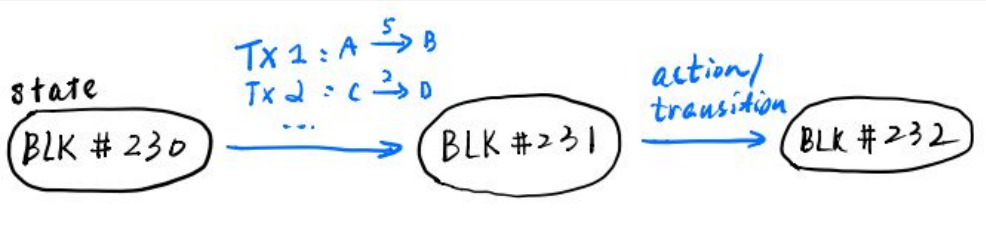
A State Machine, that's all … → State + State transition

- READ: look up some states.
- WRITE: apply state transition and update the state.

How to modify/write to blockchain?

Command for State Transition is done via "transactions" which contains all necessary parameters.

Transaction is no longer just for balance update, but arbitrary state update.



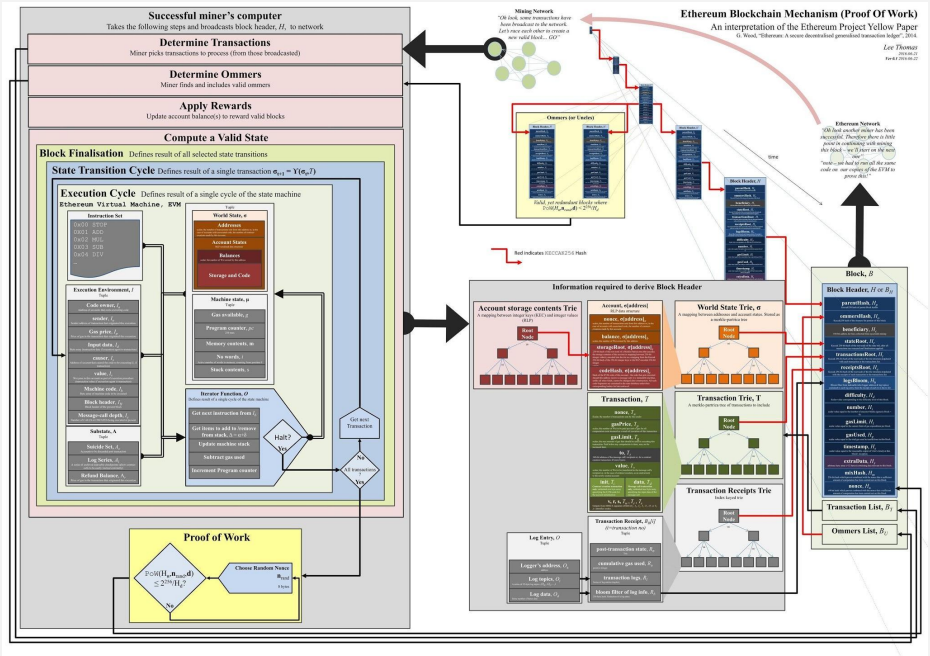
Requirement for a legitimate WRITE?

- *Clearly specify "which state to modify" (e.g. Alice and Bob's balances), "changes or new values", gas payment, valid signature(s)*

Zooming in to *City-level View* now…

All kinds of states, and where to find them.

“ To find 5th Ave, you first need to understand the grid system of Manhattan. ”



Source: https://www.6sqft.com/204-years-ago-today-the-manhattan-street-grid-became-official/; https://i.stack.imgur.com/afWDt.jpg

BLOCKCHAIN AT NTU

Now, let's break it down

https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture

Or

http://bit.ly/lazyasme

Demo: two types of account

Always refer to the web3.js@1.0 API document

Ex 1. Create a new Externally-owned Account (EOA)

→ read balance, read nonce

Ex 2. Create a new contract account

→ contract address (how to compute),

Demo: WRITE to blockchain

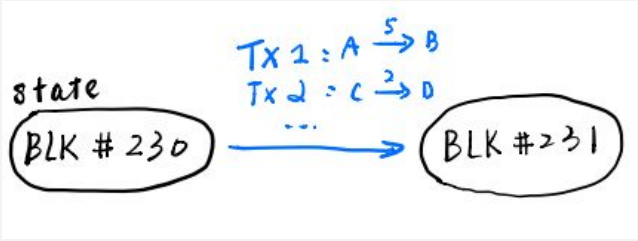
Ex 1: Simple Payment

Ex 2: Contract Creation (both Truffle & Remix)

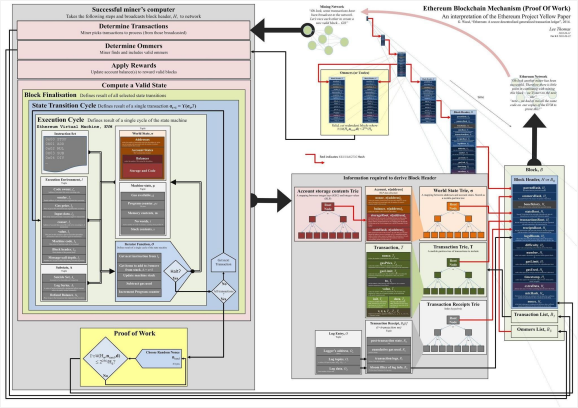
Ex 3: Contract Function Call

→ Remix

→ (optional) abi encoding, function selectors, parameters encoding.



holistic understanding, Yo!



BLOCKCHAIN AT NTU

Thank You!

🔨 with ❤ by

Alex Xiong

@ALuoyuan

Or https://t.me/ntublockchain

Attendance
Passphrase: deeep