Demystifying Blockchain Series

Slack Channel

(Slides, materials)

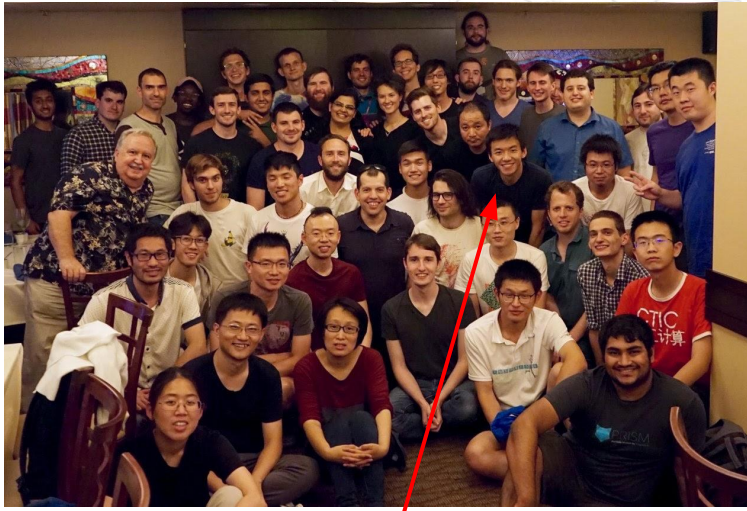
BLOCKCHAIN
@ NTU SINGAPORE

Demystifying Blockchain

Building ÐApp , yo!

> $ whoami

- **Alex -- NTU** electrical engineering student
- Started in late 2016, interested in **applied cryptography**, distributed system.
- Worked at **ConsenSys Diligence** team in New York (Summer 2017)
  - Smart contract development
  - Whitehat + Code auditing (0x Project)
  - Found bug in memory management in Solidity Compiler (v0.4.12)
- **Built**
  - Bytecode verifier (command line tool)
  - TEE chain prototype (with Ittay, Odded from Cornell and Aparna from Berkeley)
  - Authview decentralized review system (with Zhiyao)
- **Research:**
  - Practical **cross-chain exchanges using trusted hardware and state channel** (with Loi Luu from Kyber Network)



Me and my teeth

BLOCKCHAIN @ NTU SINGAPORE

Previously…

Week 1:

history →cryptography basics →peer-to-peer network + proof of work →Bitcoin design →mining, storing, using Bitcoin →attacking Bitcoin →applications →from Bitcoin to Altcoin to Ethereum

Previously…

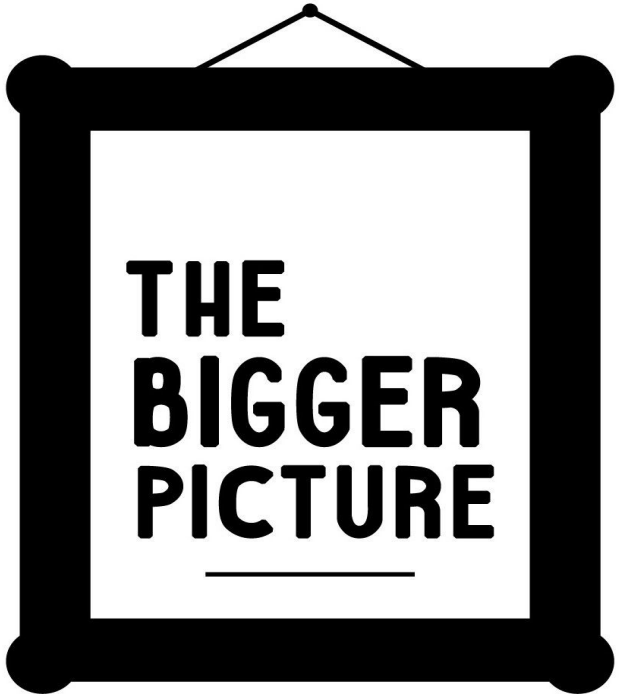
Week 2 :

Ethereum crash course →smart contract maniac → decentralized application → token fever →private chain and case study →Blockchain@NTU →existing bottleneck, solution, research frontier →reading list

Menu of the day

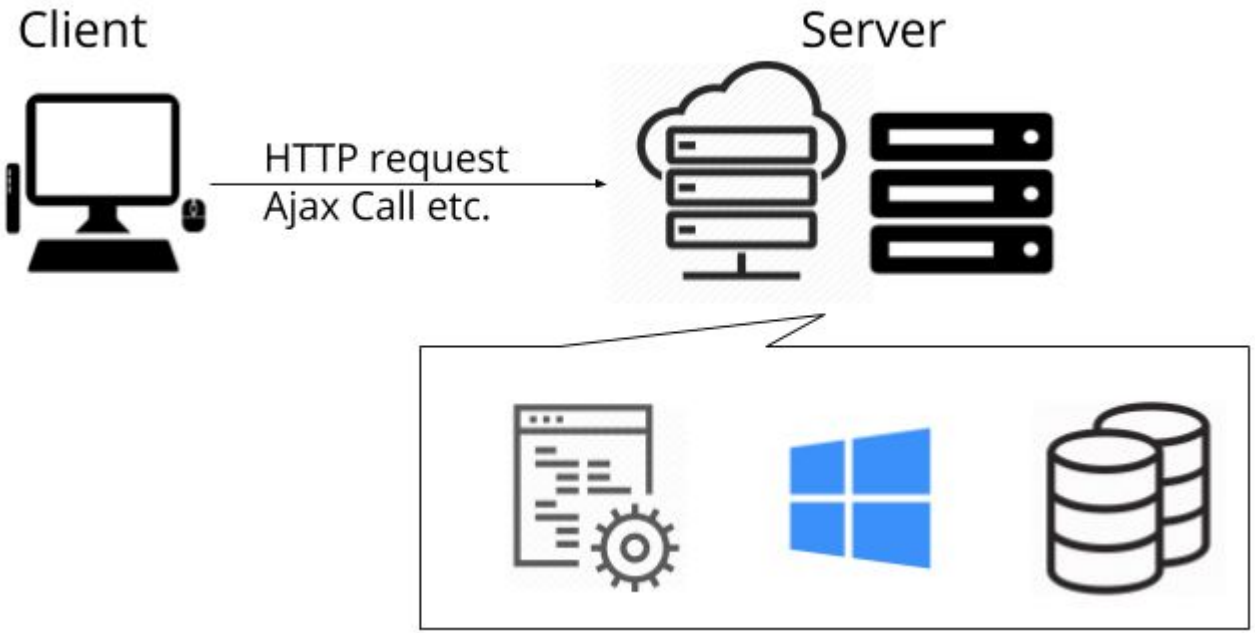
- Ethereum from developers' perspective
- Solidity Language by examples
- *Remix* IDE + *Truffle* Framework + *Ganache* Testrpc + Testnet Contract Deployment via *Metamask*

Wait… What am I doing exactly?

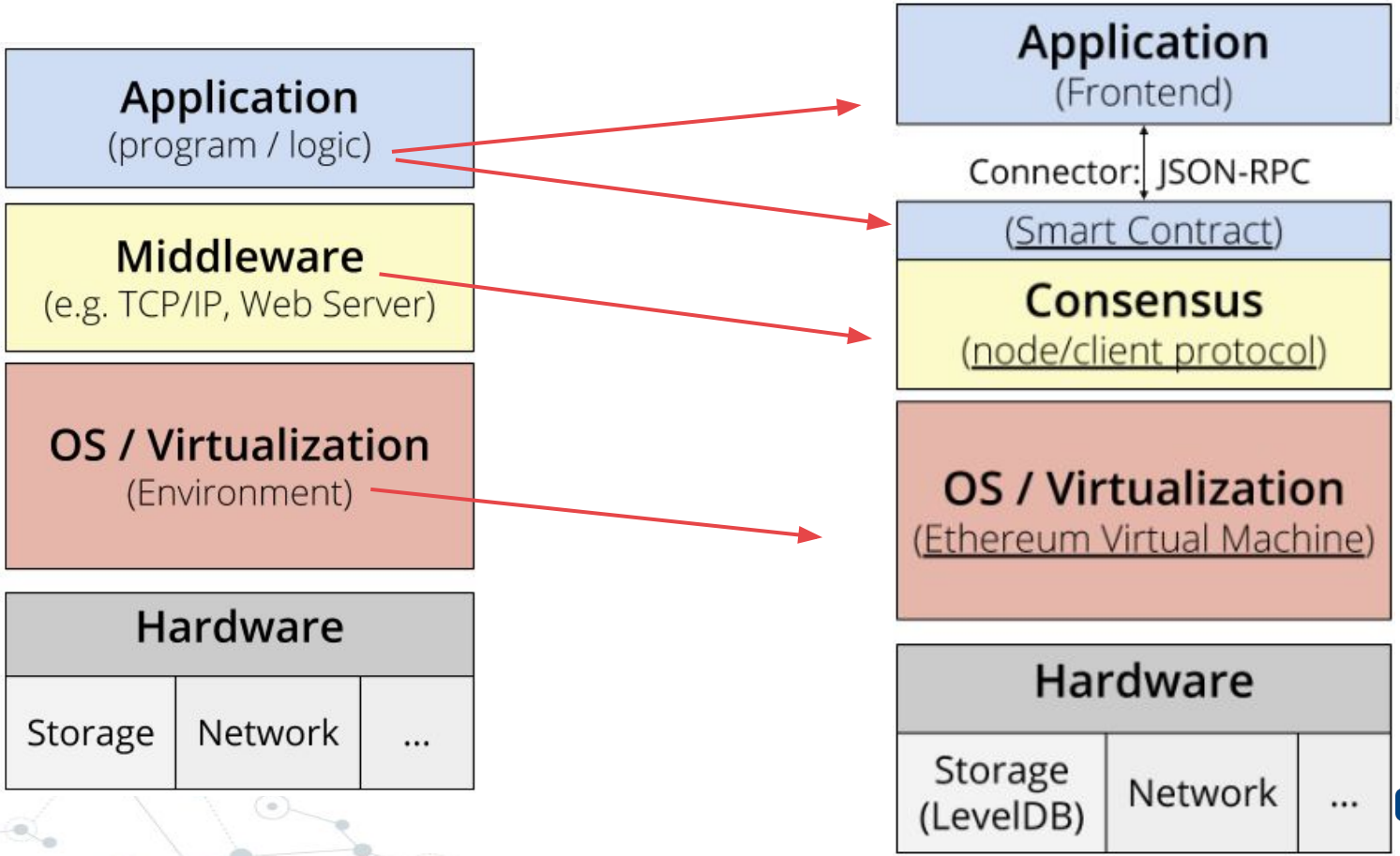


Traditional Software Development

- Client-Server

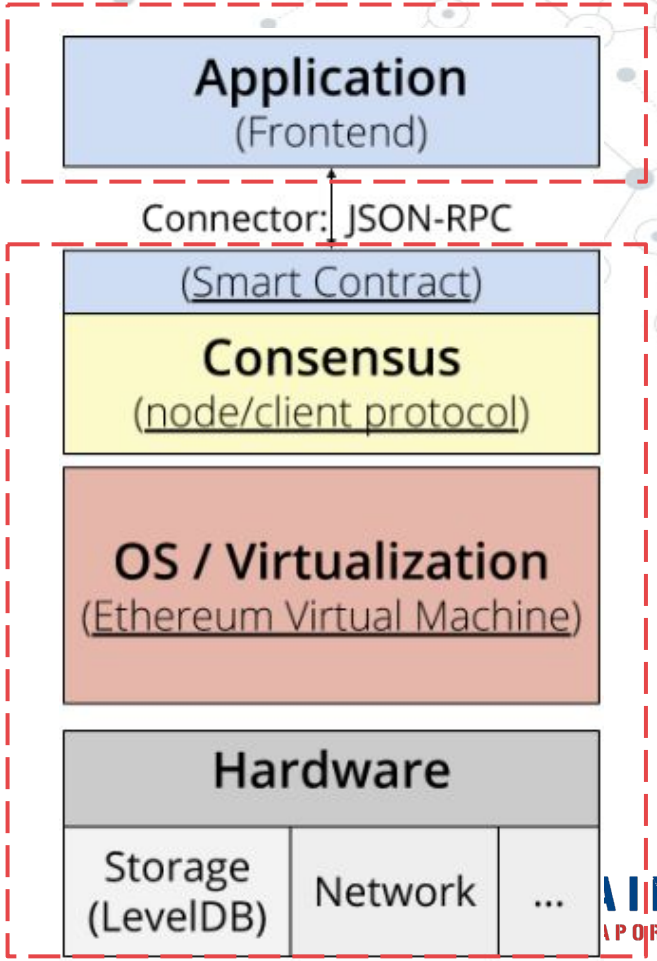


Software Application Stack



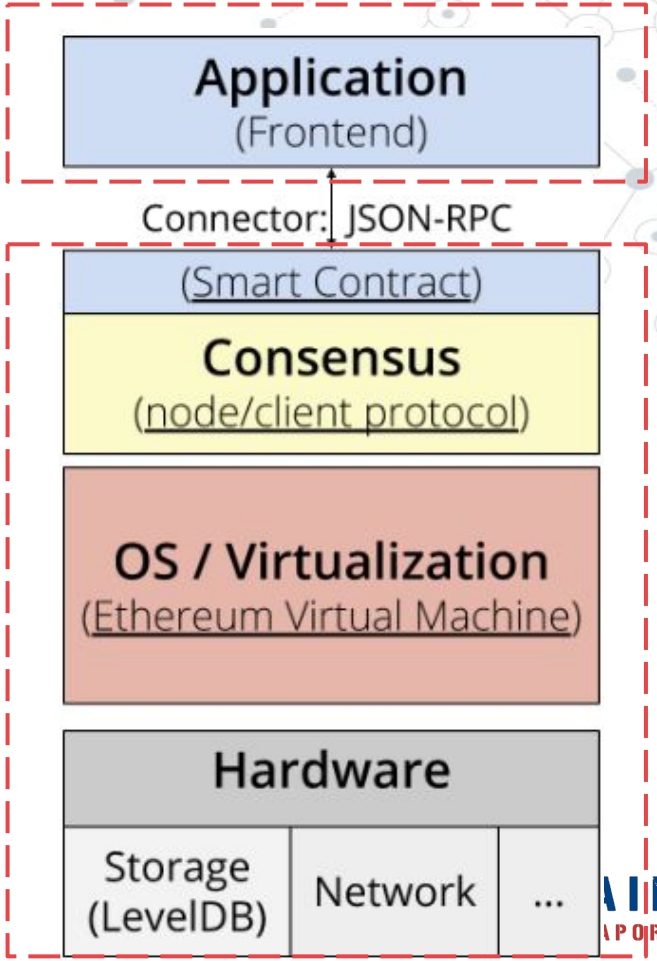
ÐApp Stack

- *"Smart Contract"* in High Level Language
  - Solidity (today's focus)
  - Vyper
  - LLL (Lisp-like, low level)
- *Connectors*: JSON-RPC
  - ≅ Ajax
  - Javascript API (web3.js, ethjs)
  - Python API (web3.py)



ĐApp Stack

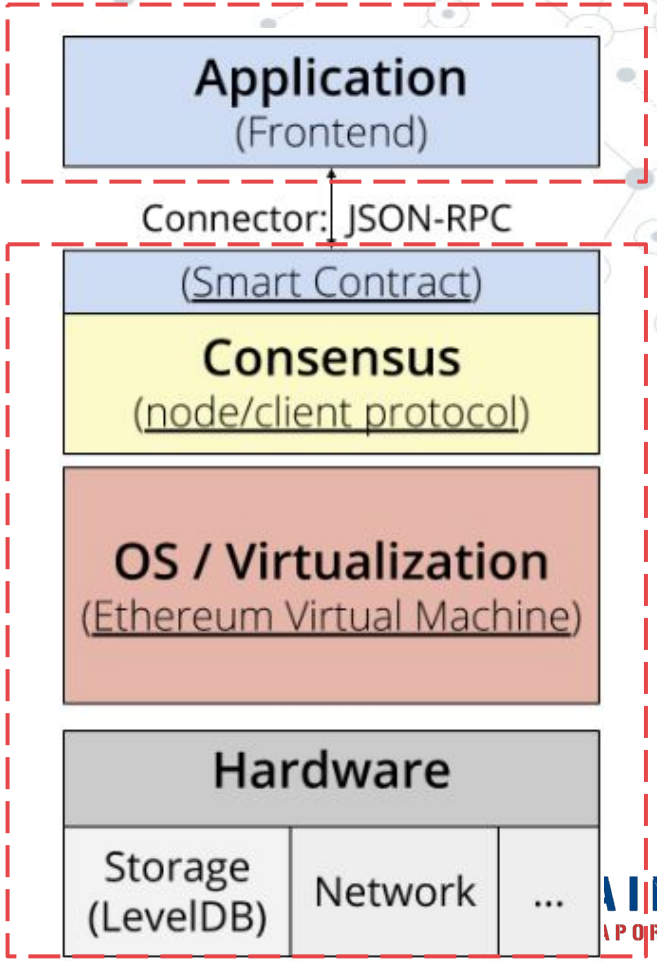
- ***Ethereum Client* / Node**
  - Why? -- Consensus in P2P Network (libp2p)
  - Cpp Client, Geth (Golang Client), Parity (Rust Client), Pyethereum (Python Client)
- ***Storage***
  - key-value pair
  - LevelDB by Geth
  - RocksDB by Rust



ĐApp Stack

- ***Ethereum Virtual Machine***
  - Specification: Yellow Paper
  - **Bytecode** to machine code
  - **Host environment independent**

Value	Mnemonic	δ	α	Description
0x00	STOP	0	0	Halts execution.
0x01	ADD	2	1	Addition operation. $\mu'_s[0] \equiv \mu_s[0] + \mu_s[1]$
0x02	MUL	2	1	Multiplication operation. $\mu'_s[0] \equiv \mu_s[0] \times \mu_s[1]$
0x03	SUB	2	1	Subtraction operation. $\mu'_s[0] \equiv \mu_s[0] - \mu_s[1]$
0x04	DIV	2	1	Integer division operation. $\mu'_s[0] \equiv \begin{cases} 0 & \text{if} \quad \mu_s[1] = 0 \\ \lfloor \mu_s[0] \div \mu_s[1] \rfloor & \text{otherwise} \end{cases}$



Developer: *"what is ÐApp?"*

- **An application whose <u>partial or entire "backend" logic resides on Blockchain</u>**
  - Traditional Front End (React, HTML, Electron, etc.)
  - Partial backend functionalities achieved via smart contracts whose execution is enforced by p2p network.
  - Frontend ←connector (web3)→ Blockchain

Why ĐApp ? (Benefits)

- **Veridical (Honest) Computing**
  - Virtual poker? Voting? no Uber surcharge?
- **Tamper-proof : code & states**
- **No Single Point of Failure**
  - High uptime/availability
  - Robust from malicious compromised or accidental mishandle or censorship
- **Network-enforced Execution / Autonomous**
  - AWS is not cloud, it's Amazon

Motivated by now?

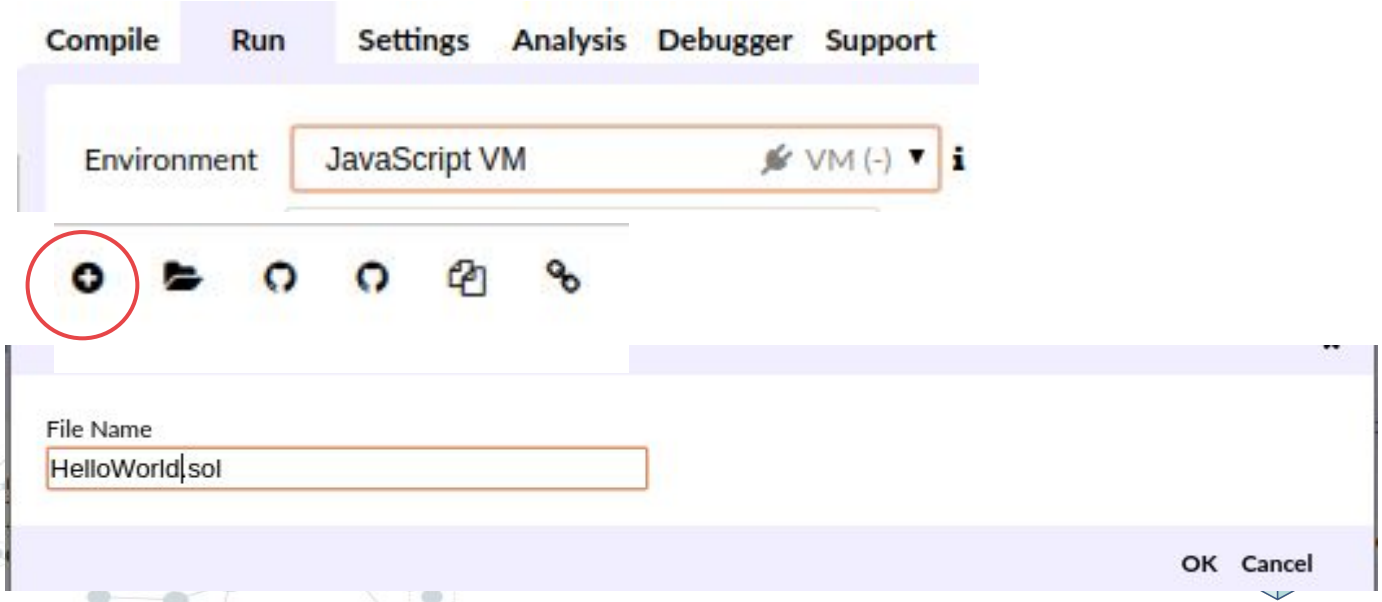
Solidity Language

- Contract-oriented
- Javascript-like

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

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

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

“Hello, World” Contract

- **Remix** Online IDE: http://remix.ethereum.org/



“Hello, World” Contract

https://bit.ly/blockchain-ntusg-code1

```
pragma solidity ^0.4.20;

contract HelloWorld {
    // Declare greeting message
    string public greeting;

    function HelloWorld () public {
        greeting = "Hello, Wolrd! This is my first smart contract!";
    }

    // a setGreeting function to set the variable "greeting" to a new message.
    function setGreeting (string _newGreeting) public {
        greeting = _newGreeting;
    }

    // a redundent getGreeting function which returns the variable state
    // since greeting is a public variable, upon contract deployment, a getter function will be
    // automatically created and used for querying the state of the variable.
    function getGreeting ()
        public
        constant
        returns (string){
        return greeting;
    }
}
```

“Hello, World” Contract

```
creation of HelloWorld pending...

[vm] from:0xca3...a733c, to:HelloWorld.(constructor), value:0 wei, data:0x60
6...30029, 0 logs, hash:0x7b0...30d48

call to HelloWorld.getGreeting

[call] from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c, to:HelloWorld.getGre
eting(), data:fe50c...0cc72, return:
{
        "0": "string: Hello, Wolrd! This is my first smart contract!"
}
```

Details Debug

Details Debug

from	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to	HelloWorld.getGreeting() 0x0dcd2f752394c41875e259e00bb44fd505297caf
transaction cost	23220 gas (Cost only applies when called by a contract)
execution cost	1948 gas (Cost only applies when called by a contract)
input	fe50cc72
decoded input	{}
decoded output	{ "0": "string: Hello, Wolrd! This is my first smart contract!" }
logs	[]

Ethereum, revisit.

- **Ethereum Blockchain**
  - ↔ consensus-base, globally executed virtual machine
  - ↔ decentralized computers containing millions of "objects" -- **account**

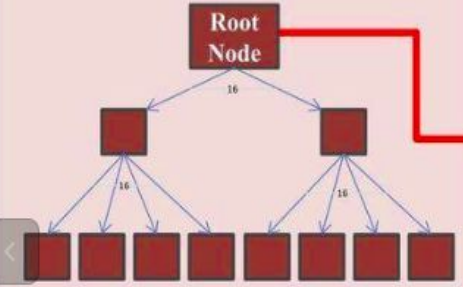
Ethereum, revisit.

- Accounts
  - Externally Owned Account (**EOA**) :
    - Controlled by a private key
    - Able to send ether and signed messages
    - example
  - **Contract** Account :
    - Controlled by code
    - Associated storage for contract related variables, contents.
    - example

Information required to derive Block Header

Account storage contents Trie

A mapping between integer keys (KEC) and integer values (RLP)



Account, σ[address]

RLP data structure

nonce, $\sigma[address]_n$
scalar; the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account.

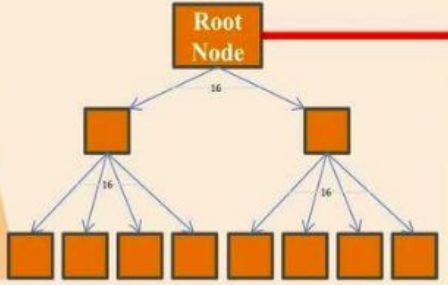
balance, $\sigma[address]_b$
scalar; the number of Wei owned by this address

storageRoot, $\sigma[address]_s$
256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values.

codeHash, $\sigma[address]_c$
Hash of the EVM code of this account - the code that gets executed should this address receive a message call; it is immutable and thus, unlike all other fields, cannot be changed after construction. All such code fragments are contained in the state database under their corresponding hashes for later retrieval.

World State Trie, σ

A mapping between addresses and account states. Stored as a merkle-partrica tree



Transaction, T

nonce, T_n
Scalar; the number of transactions sent by the sender

gasPrice, T_p
scalar; the number of Wei to be paid per unit of gas for all computation costs incurred as a result of execution of this transaction

gasLimit, T_g
scalar; the max amount of gas that should be used in executing this transaction. Paid before any computation is done, may not be increased later.

to, T_t
160-bit address of the message call's recipient or, for a contract creation transaction, 0 (zero bytes).

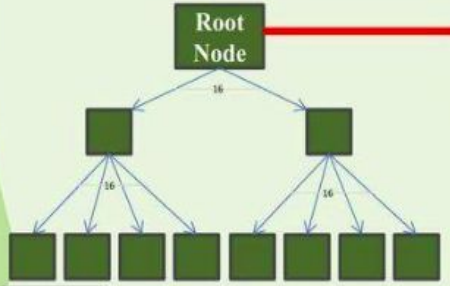
value, T_v
scalar; the number of Wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account

init, T_i
Contract creation transaction only: unlimited size byte array specifying the EVM-code for

data, T_d
Message call transaction only: unlimited size byte array specifying the input data of the

Transaction Trie, T

A merkle-partrica tree of transactions to include



Transaction Receipts Trie

Block, B

Block Header, H or B_H

parentHash, H_p
Keccak256 hash of parent block header

ommersHash, H_o
Keccak256 hash of the ommers list portion of this block

beneficiary, H_c
160-bit address for fees collected from successful mining

stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied

transactionsRoot, H_t
Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list

receiptsRoot, H_e
Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list

logsBloom, H_b
Bloom filter from indexable info (logger address & log topics) contained in each log entry from the receipt of each tx in the tx list

difficulty, H_d
Scalar value corresponding to the difficulty level of this block

number, H_i
scalar value equal to the number of ancestor blocks (genesis block = 0)

gasLimit, H_l
scalar value equal to the current limit of gas expenditure per block

gasUsed, H_g
scalar value equal to the total gas used in transactions in this block

timestamp, H_s
scalar value equal to the reasonable output of Unix's time() at this block's inception

extraData, H_x
arbitrary byte array (<32 bytes) containing data relevant to this block

mixHash, H_m
256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block

nonce, H_n
64-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block

Solidity: the Ballot Contract (Voting)

- One contract per ballot
- Different proposals as options/candidates
- Chairperson give rights to “account” to vote
- Each account could
  - either vote on only one proposal
  - or delegate this vote to someone else

https://bit.ly/blockchain-ntusg-code2

Ballot Contract

- **pragma is a preprocessor directive as version annotation**
  - **^** means accepting any compiler version no lower than 0.4.16
- **contract is similar to *Class* in OOP (inheritance).**
  - contains functions, state variable (permanently stored)
- **function is an executable unit of code.**
  - Internal or external call
  - Different visibility (**public**, **private**, **internal**, **external**)

```
pragma solidity ^0.4.16;
contract Ballot {

    address public chairperson;

.
    function Ballot(bytes32[] proposalNames) public {

    }

    function delegate(address to) public {

    }

    function vote(uint proposal) public {

    }
}
```

Ballot Contract

Data Type

- **Boolean**
- **Integer** (currently no floating point, fixed point number support is coming)
- **address** (has members)
- **Fixed-size byte array**
- **Dynamic-sized byte array**

```
bool isHappy = true;
bool isDoomed = false;
```

```
/*Unsigned integer*/
uint cakeEaten = 6666; // uint256
uint8 examTaken = 255;
/*Signed integer*/
int8 score = -50;
```

```
// fixed-sized
bytes32 msg = "NTU to the moon!";
bytes1 foo = "O";

// dynamic-sized
bytes howeverLong = "blahblahblahblah";
string str = "whatever message I have~";
bytes[] byteArray;
```

Test at 0x8c1...401f5 (memory)

foo	0: bytes1: 0x4f
howeverLong	0: bytes: 0x626c6168626c6168626c6168626c6168
msg	0: bytes32: 0x4e545520746f20746865206d6f6f6e210000000000000000000000000000000000
str	0: string: whatever message I have~

Ballot Contract

Data Type

- **Mapping** (= dictionary)

```
mapping (_keyType => _valueType) mappingName;

mapping (address => uint) deanlistGPA;
```

- **Struct** (self-defined data type with members)

```
struct Voter {
    uint weight; // weight is accumulated by delegation
    bool voted;  // if true, that person already voted
    address delegate; // person delegated to
    uint vote;   // index of the voted proposal
}
```

Ballot Contract

declaring Voter data type

declaring Proposal data type

chairperson who creates this ballot

map an account to its Voter profile

a list of proposals/candidates

```
contract Ballot {
    // This declares a new complex type which will
    // be used for variables later.
    // It will represent a single voter.
    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    // This is a type for a single proposal.
    struct Proposal {
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    // This declares a state variable that
    // stores a `Voter` struct for each possible address.
    mapping(address => Voter) public voters;

    // A dynamically-sized array of `Proposal` structs.
    Proposal[] public proposals;
```

Ballot Contract

- **Constructor**
  - optional
  - MUST have the same name as the contract
  - called only once during creation to initialize some global variables
- **Special variables**

```
/// Create a new ballot to choose one of `proposalNames`.
function Ballot(bytes32[] proposalNames) public {
    chairperson = msg.sender;
    voters[chairperson].weight = 1;

    // For each of the provided proposal names,
    // create a new proposal object and add it
    // to the end of the array.
    for (uint i = 0; i < proposalNames.length; i++) {
        // `Proposal({...})` creates a temporary
        // Proposal object and `proposals.push(...)`
        // appends it to the end of `proposals`.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

- `msg.data` (`bytes`): complete calldata
- `msg.gas` (`uint`): remaining gas - deprecated in version 0.4.21 and to be replaced by `gasleft()`
- `msg.sender` (`address`): sender of the message (current call)
- `msg.sig` (`bytes4`): first four bytes of the calldata (i.e. function identifier)
- `msg.value` (`uint`): number of wei sent with the message

Ballot Contract

- **require is for error/exception handling.**
  - Equivalent to "*catch-then-throw*"
  - Will revert all state changes within this function call

```
function giveRightToVote(address voter) public {
    require(
        (msg.sender == chairperson) &&
        !voters[voter].voted &&
        (voters[voter].weight == 0)
    );
    voters[voter].weight = 1;
}
```

Ballot Contract

- **storage specifies the data location of certain variable.**
  - 3 options: persistent **storage**, temporary **memory**, limited on-stack **calldata**. (FAQ)
  -

```
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted);
    sender.voted = true;
    sender.vote = proposal;

    // If `proposal` is out of the range of the array,
    // this will throw automatically and revert all
    // changes.
    proposals[proposal].voteCount += sender.weight;
}
```

Ballot Contract

- **view function allows read-only operation.**
  - Keyword used to be **constant**
  - No state modification allowed !!
  - Free of charge

```
function winningProposal() public view
        returns (uint winningProposal_)
{
    uint winningVoteCount = 0;
    for (uint p = 0; p < proposals.length; p++) {
        if (proposals[p].voteCount > winningVoteCount) {
            winningVoteCount = proposals[p].voteCount;
            winningProposal_ = p;
        }
    }
}
```

The following statements are considered modifying the state:

1. Writing to state variables.
2. Emitting events.
3. Creating other contracts.
4. Using `selfdestruct`.
5. Sending Ether via calls.
6. Calling any function not marked `view` or `pure`.
7. Using low-level calls.
8. Using inline assembly that contains certain opcodes.

Solidity literacy, get ✔

- Solidity [0.4.21 Documentation]

CODENINJAS

Compile & Migrate

- **Solidity Compiler (installation guide)**
  - Compile Solidity to Ethereum bytecode
  - Returns metadata in JSON format (incl. ABI)
  - Remix, Solc-js, Binary package (apt-get, homebrew)

Ballot.sol Metadata: https://bit.ly/blockchain-ntusg-abi

Application Binary Interface (ABI)

- **To facilitate and standardize contract interactions**
  - EOA ↔ Contract + Contract A ↔ Contract B
  - Data encoding/decoding based on input/output type
  - Function selection based on function signatures

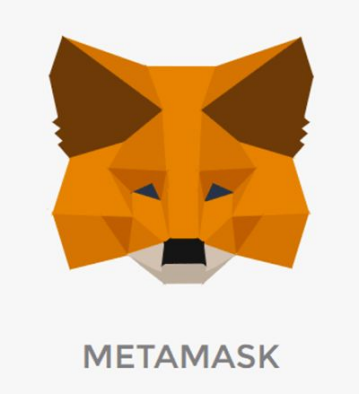
```
function vote(uint proposal) public {

}
```

```
keccak256("vote(uint256)");
```

```
======= Ballot.sol:Ballot =======
Function signatures:
2e4176cf: chairperson()
5c19a95c: delegate(address)
9e7b8d61: giveRightToVote(address)
013cf08b: proposals(uint256)
0121b93f: vote(uint256)
a3ec138d: voters(address)
e2ba53f0: winnerName()
609ff1bd: winningProposal()
```

Migration / Contract Deployment

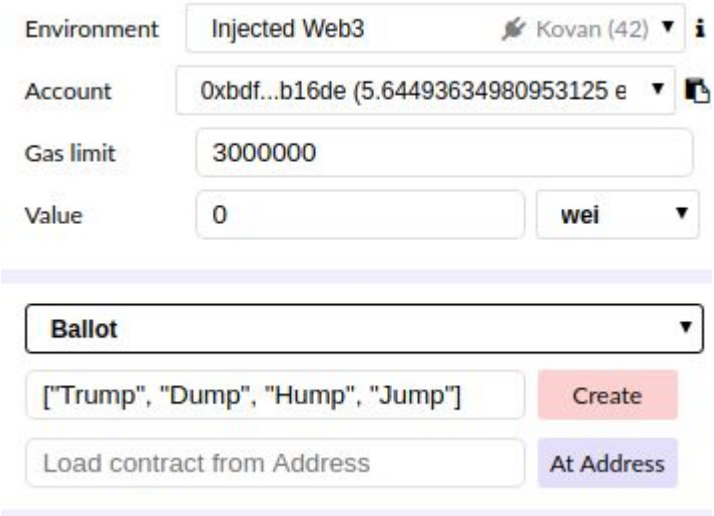
- Connected to Ethereum Network, and broadcast a transaction to create contract
  - Parity, Geth, **Metamask** (installation guide)

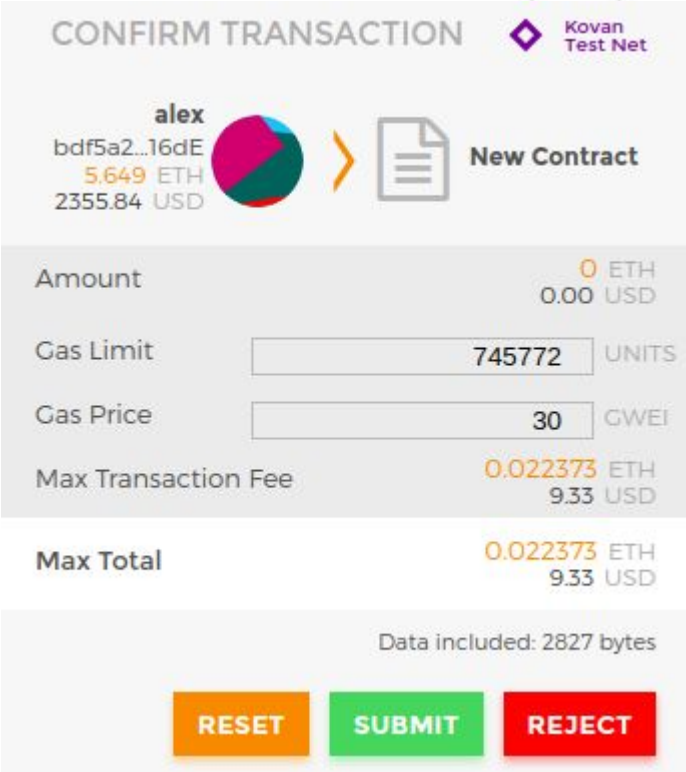


Contract Deployment: Testnet

- **Alternative Blockchain for testing (a.k.a *Sandbox*)**
  - Ether on testnet has no real value
  - Maintained by few trusted validators (PoA instead of PoW)
  - Testnet Faucet : where you get your "fake" ether for testing (e.g. Kovan faucet)
  - Ropsten, Rinkeby, Kovan

MetaMask + Remix Demo





BLOCKCHAIN @ NTU SINGAPORE

MetaMask + Remix Demo

Overview

Transaction Information

Tools & Utilities

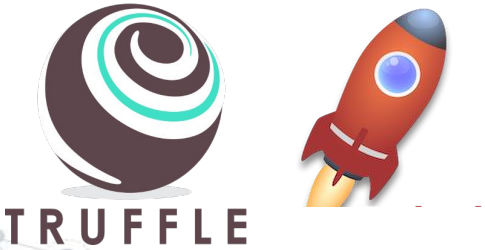
TxHash:	0xaae824ac01954cec3a3f1a3ce1b110f5243e7557fb35acfeb1304e70067a36aa
Block Height:	6821134 (2 block confirmations)
TimeStamp:	46 secs ago (Apr-11-2018 05:22:08 AM +UTC)
From:	0xbdf5a292de1c15cd3daaa636dddf043a02ab16de (AlexXiong97)
To:	[Contract 0xc99802e79e5e5560d751915dda693e226d893f69 Created]
Value:	0 Ether ($0.00)
Gas Limit:	886722
Gas Used By Txn:	886722
Gas Price:	0.000000005 Ether (5 Gwei)
Actual Tx Cost/Fee:	0.00443361 Ether ($0.000000)
Nonce:	231
Input Data:	


```
0x6060604052341561000f57600080fd5b604051610b0b380380610b0b833981016040528080518201919050506000336000806101010
00a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff1602
179055506001806000806000905490610100a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffff
ffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060000181905 5
```


Convert To Ascii

Development Framework

- **Why?**
  - Avoid repetitive configuration, boilerplate == automation
  - Encapsulation and Abstraction
  - Default best practice, structure ...
- **What?**
  - Traditional : React, Django, Rails ...
  - Ethereum: **Truffle**, Embark ...



Truffle Framework + Ganache CLI

- **Local Blockchain Instance**
- **Ethereum Client Simulation**
  - Accounts
  - TestRPC
- **Testing, debugging, logging →faster, easier development**
  - Time manipulation (fast forward) etc.

Truffle, Ganache Installation

```
npm install -g truffle
```

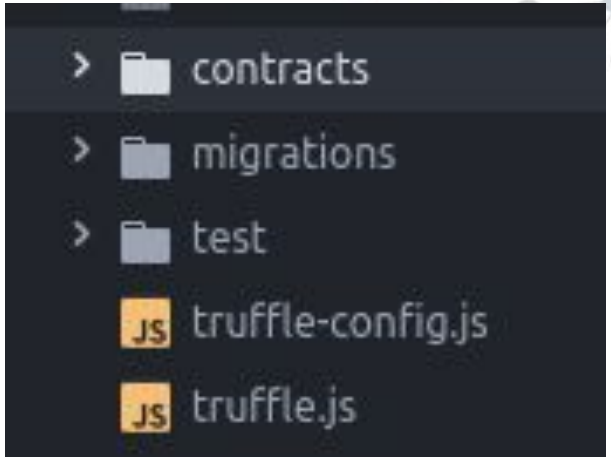
REQUIREMENTS

- NodeJS 5.0+ recommended.

```
// Make sure to have npm v5.3.0 and node v8.3.0 installed
$ npm install -g truffle

// Command line interface, github.com/trufflesuite/ganache-cli/blob/master/README.md
$ npm install -g ganache-cli

// Ganache installation: truffleframework.com/ganache/
```

Truffle Layout

- **<u>contracts</u> folder:**
  - contains contract source code
- **<u>migrations</u> folder:**
  - staging deployment tasks
- **<u>test</u> folder:**
  - unit testing script



Truffle Console (Demo)

```
truffle(development)> contract.chairperson()
'0x83741a953fb24cb49f2bae0cb4928c4d99b75306'
truffle(development)>
```

```
Ganache CLI v6.0.3 (ganache-core: 2.0.2)

Available Accounts
==================
(0) 0x83741a953fb24cb49f2bae0cb4928c4d99b75306
(1) 0x666233719ad845e6bdaefbc58866b5eaf72a317e
(2) 0x72dfdfaeffe96e8b7627f8cb0379ae849ac74b01
(3) 0xc4221816929a0236f7de33b12f1f9b172ee92796
(4) 0x12e0ecbdaa27d2e3b9f2d34ef0ed9ee5a2390774
(5) 0x4df2df6ea5dc1a247198d92d5fb26c43334ac514
(6) 0x4b836445c950baf016551da0cbfabdb20c67c8d4
(7) 0xba1202a554722d05c80d8e43d8c9a20b6925aa1d
(8) 0xa49a1740f336e1ae453d9e910b46a404c2d8352f
(9) 0x2ab2fff516b34debea08ad71175c0b74ef16a7cc

Private Keys
==================
(0) 500067d3899a2c49e536d3d11f53a085d0d184814ca4fa19a2ec245cf901ed12
(1) 96cf98ce95daa97f7a25d4d3cb58016c599722fc47c6f91f56872e43c00c2912
(2) b306c343b06904cc560de93e8686b2f3aa03c1ce86e41235ff740f20e7cdf1ee
(3) 86ac06e61e14e6b888f46c5669cfb8b2f37ebacd68f26349a3a634073b674b3c
```

BLOCKCHAIN @ NTU SINGAPORE

Truffle Testing Script (Demo)

- Mocha testing framework + Chai assertion library
- Unit testing
  - automate the manual truffle console process
  - contract, describe, it (individual test case)

Truffle Testing Script (Demo)

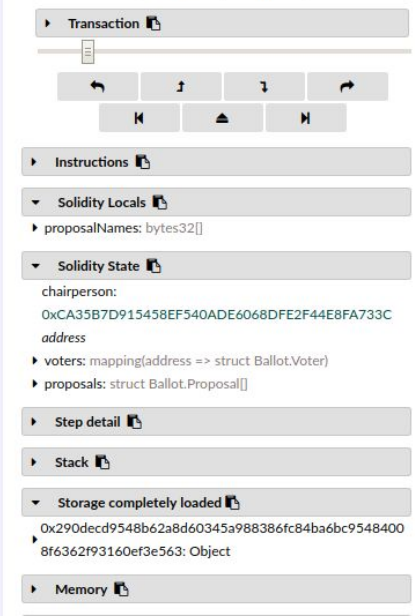
```
const Ballot = artifacts.require("./Ballot.sol");

contract('Ballot', function(accounts){
  describe('constructor should the sender', () => {
    it('should instantiate the correct chairperson', () => {
      return Ballot.deployed().then(instance => {
        return instance.chairperson()
      }).then(address => {
        assert.equal(address, accounts[0], 'chairperson address is wrong!');
      })
    });
  });
});
```

Debugging (Demo)

- Truffle debugger

```
$ truffle debug 0x8e5dadfb921ddddfa8f53af1f9bd8beeac6838d52d7e0c2fe5085b42a4f3ca76
```

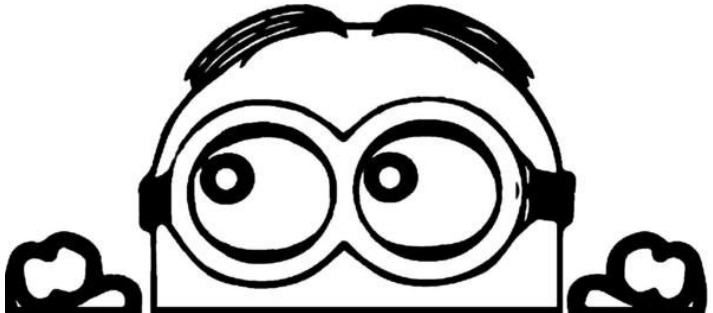
- Remix debugger



BLOCKCHAIN @ NTU SINGAPORE

A lot more to expect… (next semester)

- ERC 20, ERC 721 … Token Standard
- Inter-contract interaction
- Common paradigm (commit-reveal, partial timelock, multisig etc.)
- Smart contract security/best practice
- Web3 library



Blockchain @ NTU Singapore

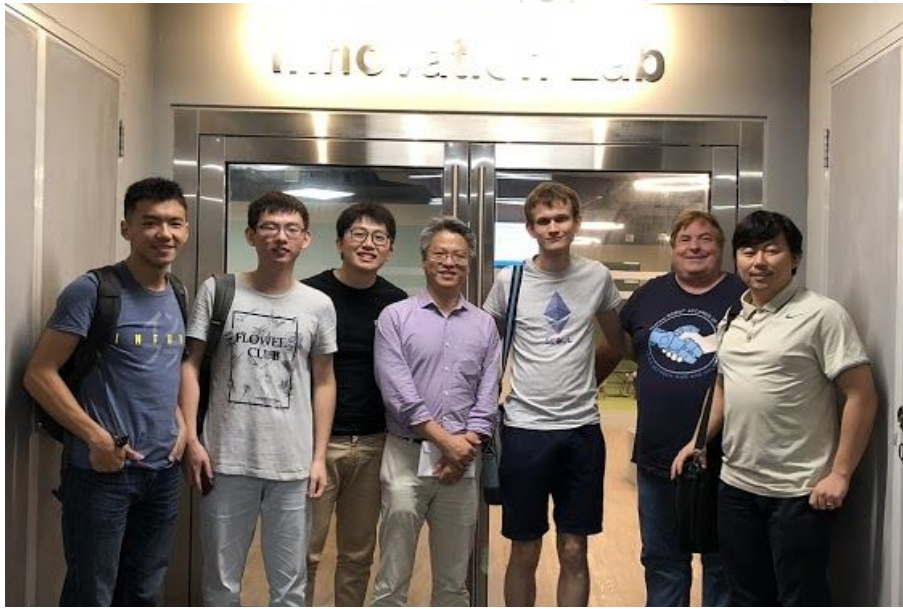
- Research, develop and educate
- Academic research +

  dApp projects +

  Weekly sharing +

  Industrial networking
- Advisor: Prof. Wen (SCSE) and

  Vitalik (Ethereum Foundation)



Penn blockchain

BLOCKCHAIN AT BERKELEY

BLOCKCHAIN @ NTU SINGAPORE

Blockchain @ NTU Singapore

- We are **recruiting** the <u>core team</u> !
  - R & D
  - Outreach
  - Education
  - Consulting

- **Not Really Benefit**: Geek friends, self-initiated projects, "meshy" organization

BLOCKCHAIN
@ NTU SINGAPORE

Thank you so much!!

Slack Channel

(Slides, materials)

BLOCKCHAIN @ NTU SINGAPORE

Reference / Attribution

- Solidity Documentation: http://solidity.readthedocs.io/en/v0.4.21/
- Ethereum Wiki: https://github.com/ethereum/wiki/wiki
- Joseph Chow SV Ethereum Meetup: http://tinyurl.com/ethereumcoding101
- Visual Interpretation of Yellow Paper: https://i.redd.it/vko4yn9gqopx.png