SEC-205: Distributed Ledger and Blockchain

Lecture 2 – Ethereum and Smart Contracts

Instructed By:

Dr. Charnon Pattiyanon

Assistant Director of IT and Instructor
CMKL University

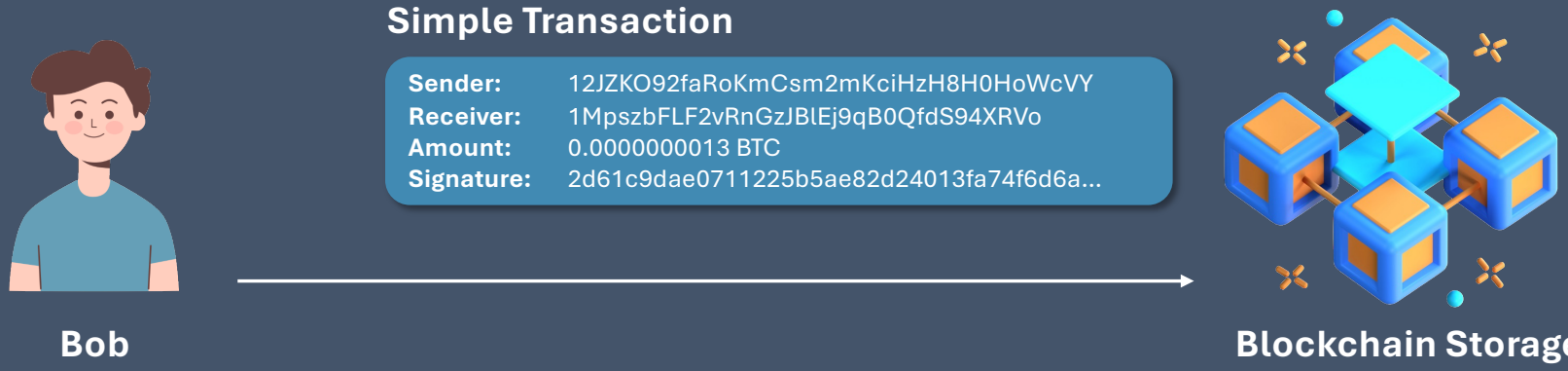
Artificial Intelligence and Computer Engineering (AICE) Program

Today's Outline

- In today's lecture, we will explore and learn about:
  - Limitations and Difficulties of Bitcoin's Blockchain.
  - What are Ethereum and Smart Contracts?
  - Background on Smart Contract Development

Limitations and Difficulties of Bitcoin's Blockchain

- In bitcoin, consensus in a network is met by the mining mechanism *without the need for a trusted third party* or an authority.
- However, consensus in Bitcoin is limited to simple transactions, e.g., money transfers.

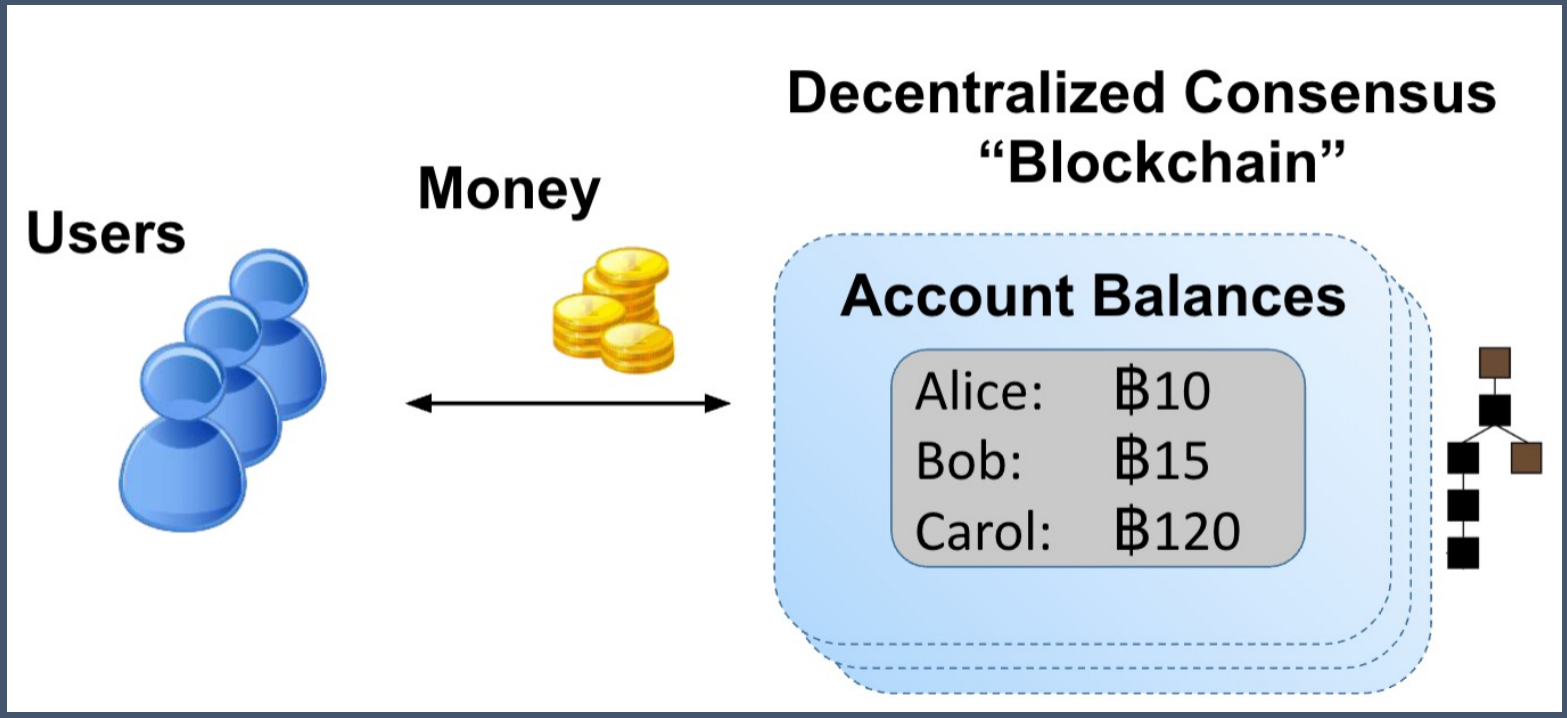


Limitations and Difficulties of Bitcoin's Blockchain

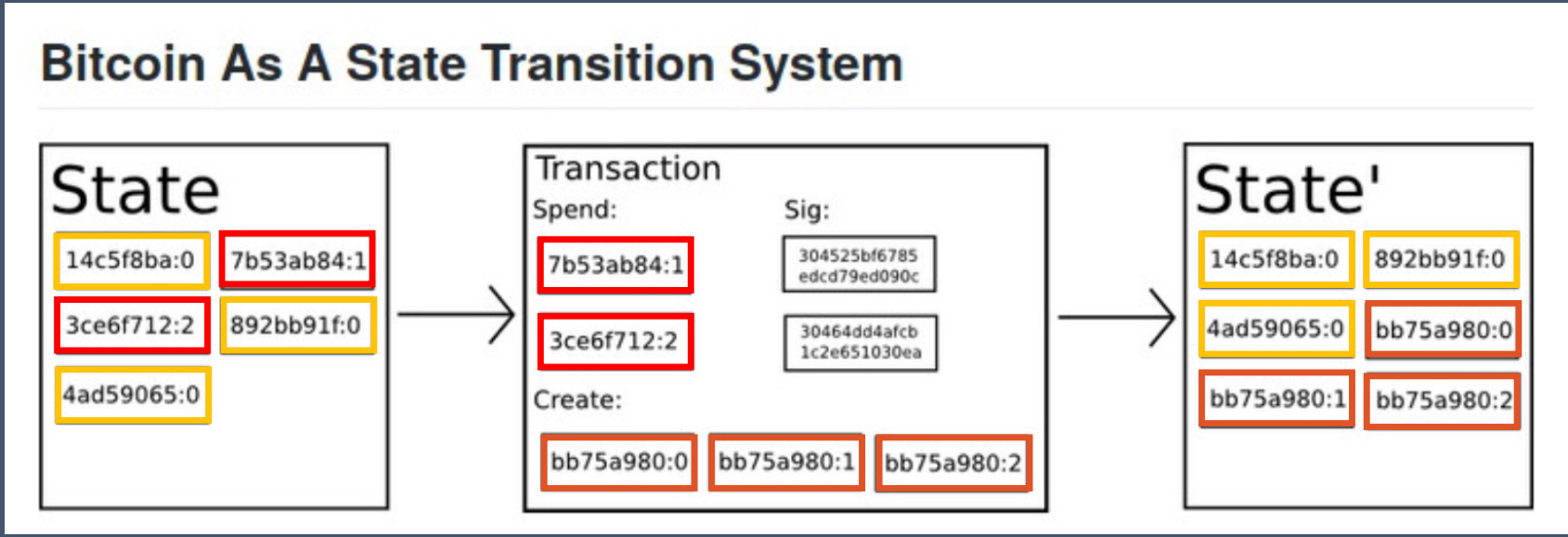
- In bitcoin, consensus in a network is met by the mining mechanism *without the need for a trusted third party* or an authority.
- However, consensus in Bitcoin is limited to simple transactions, e.g., money transfers.

- More complicated rules such as *conditional money transfer* or *online voting* cannot be implemented with Bitcoin's transactions.
- **Ethereum** is another decentralized financial blockchain like Bitcoin, but they have many more features. (Bitcoin=BTC, Ethereum=ETH)
- They develops the idea of complicated rule further to allow **consensus on code**.

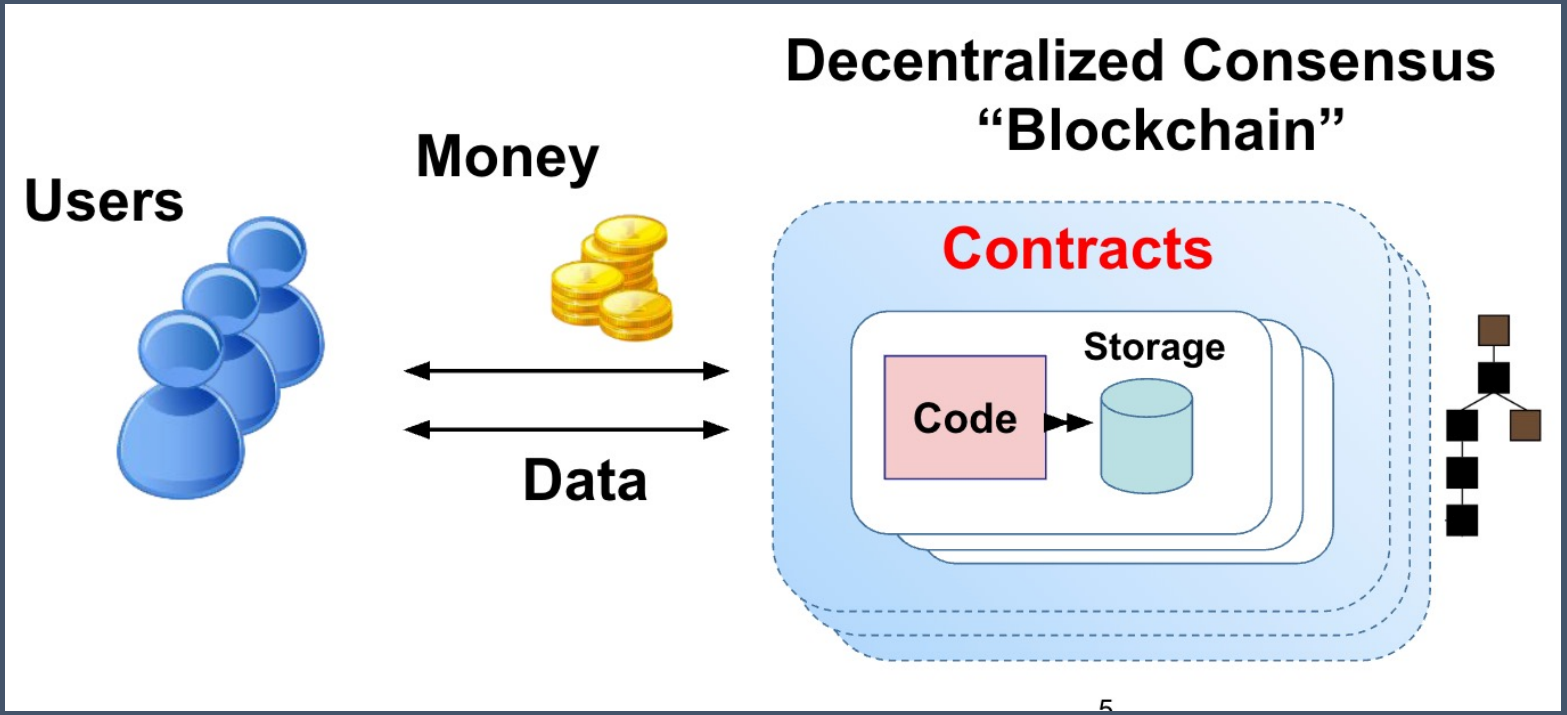
Bitcoin Transactions



Bitcoin As A State Transition System

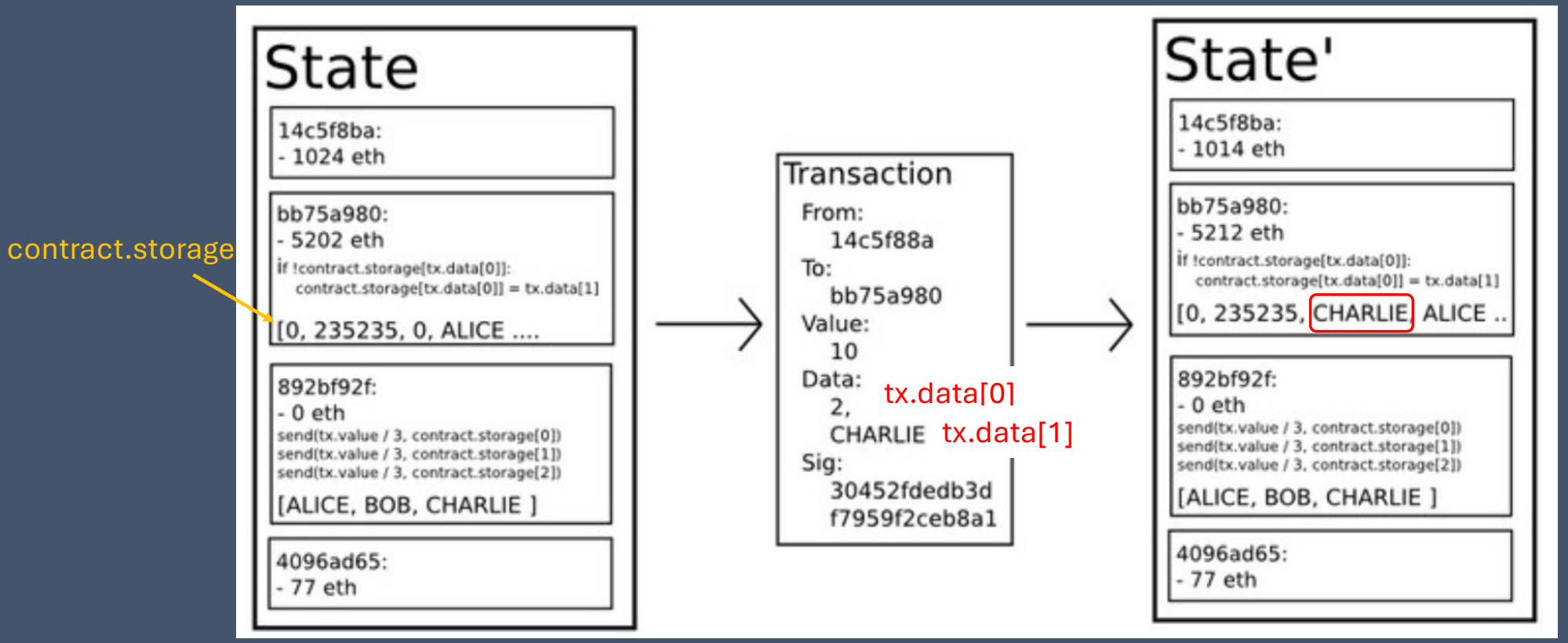


Ethereum and Smart Contracts



Smart Contracts are user-defined programs running on top of a blockchain.

Ethereum As A State Transition System



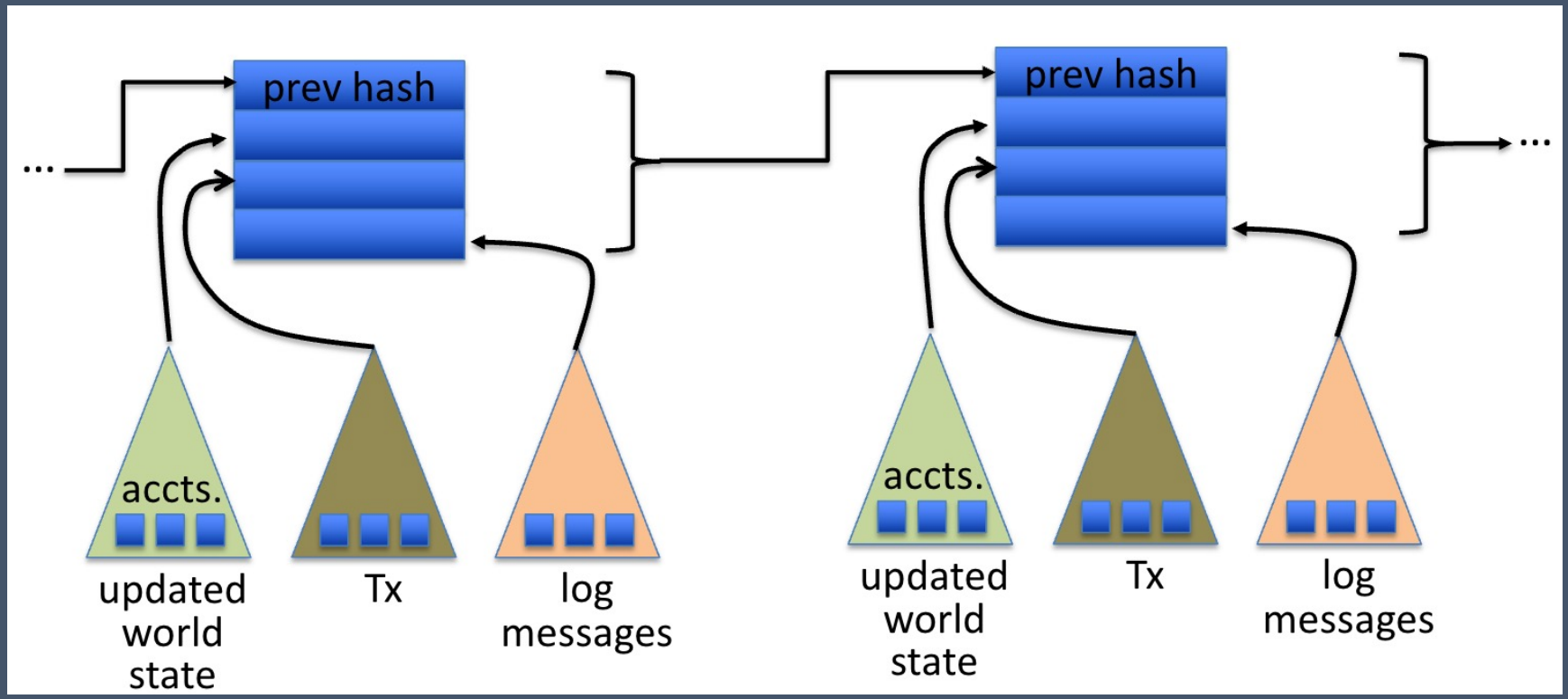
Comparison Between Bitcoin and Ethereum

Technical Features	Bitcoin	Ethereum
• **Cryptography and Timestamped Logs**	**Yes**	**Yes**
○ Cryptographic Hash Function	Yes	Yes
○ Timestamped Append-Only Logs	Yes	Yes
○ Block Header and Merkle Tree	Yes	Yes
○ Asymmetric Key Cryptography and Digital Signatures	Yes	Yes
○ Addresses	Yes	Yes
• **Decentralized Network Consensus**	**Yes**	**Yes**
○ Proof of Work	Yes	Yes
○ Native Currency	Yes	Yes
○ Network	Yes	Yes
• **Transaction Script and Unspent Transaction Output (UTXO)**	**Yes**	**No**
○ Transaction Inputs and Outputs	Yes	State Transitions
○ UTXO Set	Yes	Account Based
○ Script Language	Yes	7 Languages

Comparison Between Bitcoin and Ethereum

	Bitcoin	Ethereum
Founder	Satoshi Nakamoto	Vatalik Buterin
Genesis	January 2009	July 2015
Code	Non Turing (Script)	Turing Complete (Solidity, Serpent, Lisp Like Language, or Mutan)
Ledger	UTXO - Transactions	State – Account Based
Merkle Tree	Transactions	Transactions, State, Storage, Receipts (with Nonces)
Block Time	10 Minutes	14 Seconds
Consensus	Proof of Work	Proof of Work
Hash Function	SHA 256	ETHash
Currency	BTC	ETH
Mining	ASIC	GPU
Hash Rate	54 Exahash/second	260 Terahash/second
Pre-Sale	None	ICO and Prerelease of 72 Million ETH
Reward	12.5 BTC/Block	3 ETH/Block

Abstraction of Ethereum Blockchain



Smart Contract Development: Solidity and EVM

Solidity Program
High Level Language

Ethereum Virtual Machine (EVM) Program
Low Level Bytecode

```
pragma solidity ^0.5.0

contract MyRegistry {

  mapping (string => address) public registry;

  function registerDomain (string memory domain) public
{   //Can only reserve new unregistered domain name
    require(registry[domain] == address(0));

    //Update the owner of this domain
    registry[domain] = msg.sender;
  }
}
```

```
REVERT JUMPDEST POP PUSH2 0x303 DUP1 PUSH2
PUSH1 0x4 CALLDATASIZE LT PUSH2 0x78 JUMPI
0x100000000000000000000000000000000000000
0x7D JUMPI DUP1 PUSH4 0x1D0806AE EQ PUSH2
PUSH2 0xDD JUMPI DUP1 PUSH4 0xD3642A88 EQ
DUP1 REVERT JUMPDEST PUSH2 0x85 PUSH2 0x18
DUP1 REVERT JUMPDEST POP PUSH2 0x9C PUSH2
```

Gas in Ethereum

- Gas in Ethereum is like an operation fee for executing smart contracts.
- More complicated transactions consume more gas, so they cost more.

Every instruction costs a fixed amount of gas

- A counter of gas used is tracked when executing the transaction through smart contracts.

```
pragma solidity ^0.5.0

contract MyRegistry {

  mapping (string => address) public registry;

  function registerDomain (string memory domain) public {
    //Can only reserve new unregistered domain name
    require(registry[domain] == address(0));

    //Update the owner of this domain
    registry[domain] = msg.sender;
  }
}
```

Remaining Gas: 9500

Every instruction costs a fixed amount of gas

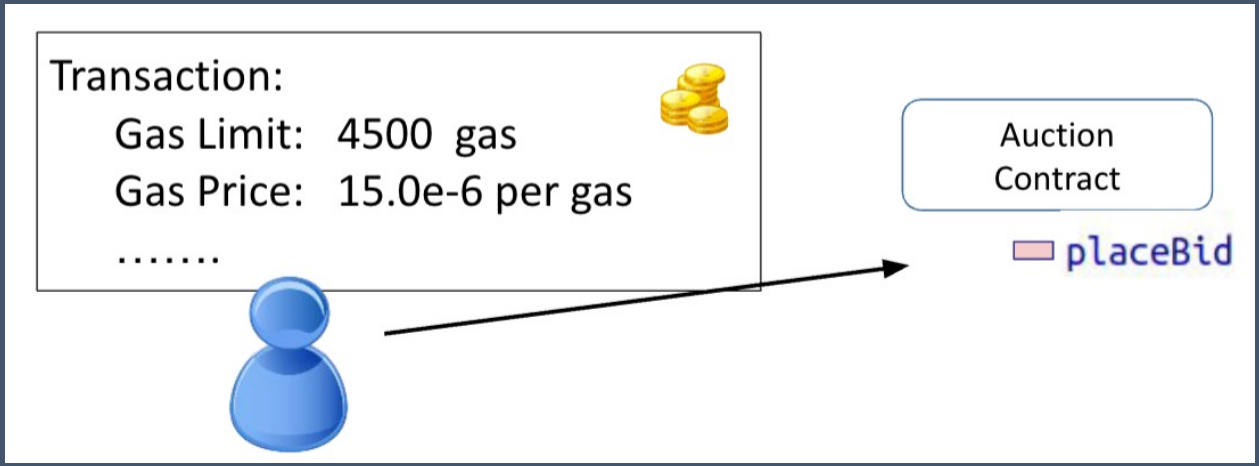
- A counter of gas used is tracked when executing the transaction through smart contracts.

```
pragma solidity ^0.5.0

contract MyRegistry {

  mapping (string => address) public registry;

  function registerDomain (string memory domain) public {
    //Can only reserve new unregistered domain name
    require(registry[domain] == address(0));

    //Update the owner of this domain
    registry[domain] = msg.sender;
  }
}
```

Remaining Gas: 8000

Gas Limits and Refunds

- Each transaction specifies a gas limit and a price for the gas, in unit of ETHer.
- ETHer value to pay for the gas must be reserved up front.
- At the end of contract execution, unused gas is refunded.



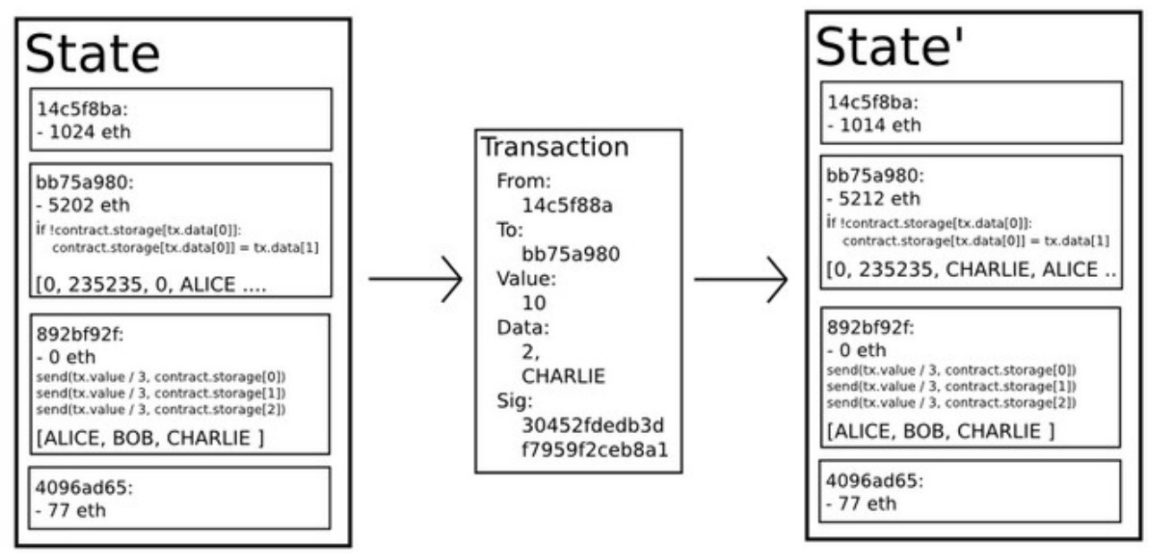
What Happens When Gas Runs Out?

- An **Out-Of-Gas** exception is thrown
- Any changes made to storage variables, any account transfers, are **reverted** to their state before this method call.
- You are ***still charged*** the gas fee for every instruction leading up to the exception.
- Like other exceptions, it can be ***caught*** by a handler function
- Methods can be invoked with just a portion of available gas

"With the use of gas in Ethereum, it gives a good reason to optimize your smart contract code!"

Summary of Today's Lecture

- Upon successful of today's lecture, we have learned about:



Solidity Program
High Level Language

Ethereum Virtual Machine (EVM) Program
Low Level Bytecode

End of the Lecture

Please do not hesitate to ask any questions to free your curiosity,

If you have any further questions after the class, please contact me via email (charnon@cmkl.ac.th).