



# Ethereum on Python

中国Python开发者大会

[jan@cryptape.com](mailto:jan@cryptape.com)

@janx  
peatio  
pyethapp  
pyethereum  
ruby-ethereum  
cita



# Bitcoin Decentralized **Data** Layer



# Ethereum Decentralized **Computation** Layer





# Pattern

Cryptography

P2P Network

Consensus

Authenticated Data Structure



# DEVP2P

## Node Discovery (UDP)

- Kademlia
- Bootstrap nodes

## Secure Transport (TCP)

- Handshake
- Transport
- Multiplexer
- Flow Control

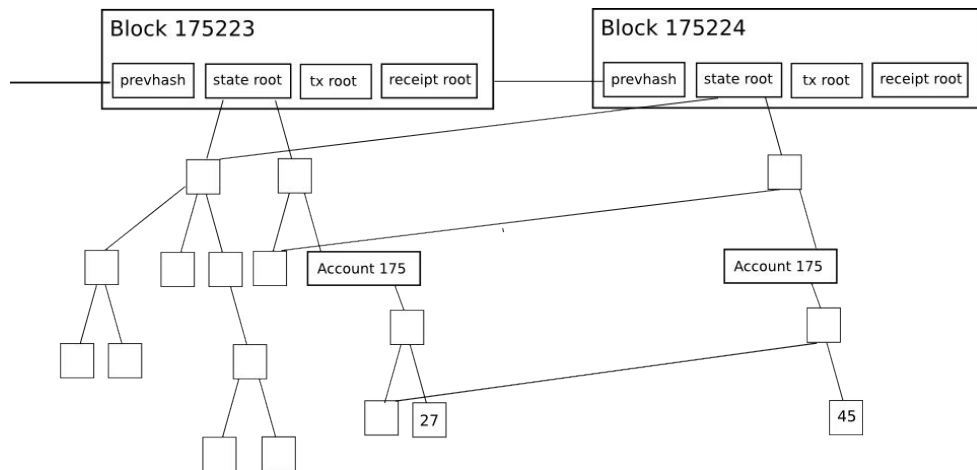


# Authenticated Data Structure

Hash

List

Tree



[github.com/ethereum/  
pyethereum](https://github.com/ethereum/pyethereum)





```
class Block(rlp.Serializable):  
  
    fields = [  
        ('header', BlockHeader),  
        ('transactions', CountableList(Transaction)),  
        ('uncles', CountableList(BlockHeader))  
    ]
```



```
class BlockHeader(rlp.Serializable):  
  
    fields = [  
        ('prevhash', hash32),  
        ('uncles_hash', hash32),  
        ('coinbase', address),  
        ('state_root', trie_root),  
        ('tx_list_root', trie_root),  
        ('receipts_root', trie_root),  
        ('bloom', int256),  
        ('difficulty', big_endian_int),  
        ('number', big_endian_int),  
        ('gas_limit', big_endian_int),  
        ('gas_used', big_endian_int),  
        ('timestamp', big_endian_int),  
        ('extra_data', binary),  
        ('mixhash', binary),  
        ('nonce', binary)  
    ]
```



```
class Account(rlp.Serializable):  
  
    fields = [  
        ('nonce', big_endian_int),  
        ('balance', big_endian_int),  
        ('storage', trie_root),  
        ('code_hash', hash32)  
    ]
```



```
class Transaction(rlp.Serializable):  
  
    fields = [  
        ('nonce', big_endian_int),  
        ('gasprice', big_endian_int),  
        ('startgas', big_endian_int),  
        ('to', utils.address),  
        ('value', big_endian_int),  
        ('data', binary),  
        ('v', big_endian_int),  
        ('r', big_endian_int),  
        ('s', big_endian_int),  
    ]
```



# Account Model

Contract Account

External Account

Machine

Human

Program

Private Key



# PoW Ethereum

15s

Uncles

ASIC Resistance



# EVM

## Ethereum Virtual Machine

Minimal

Safe

Deterministic

Bounded Turing-complete



# Gas Mechanism

Halting Problem

Opcode Pricing

Example:

$\text{Tx cost} = 21000 \text{ gas} * 20 \text{ shannon per gas} = 0.00042 \text{ ether}$



```
opcodes = {  
    0x00: ['STOP', 0, 0, 0],  
    0x01: ['ADD', 2, 1, 3],  
    0x02: ['MUL', 2, 1, 5],  
    0x03: ['SUB', 2, 1, 3],  
    0x04: ['DIV', 2, 1, 5],  
    0x05: ['SDIV', 2, 1, 5],  
    0x06: ['MOD', 2, 1, 5],  
    0x07: ['SMOD', 2, 1, 5],  
    0x08: ['ADDMOD', 3, 1, 8],  
    0x09: ['MULMOD', 3, 1, 8],  
    0x0a: ['EXP', 2, 1, 10],  
    0x0b: ['SIGNEXTEND', 2, 1, 5],  
}
```





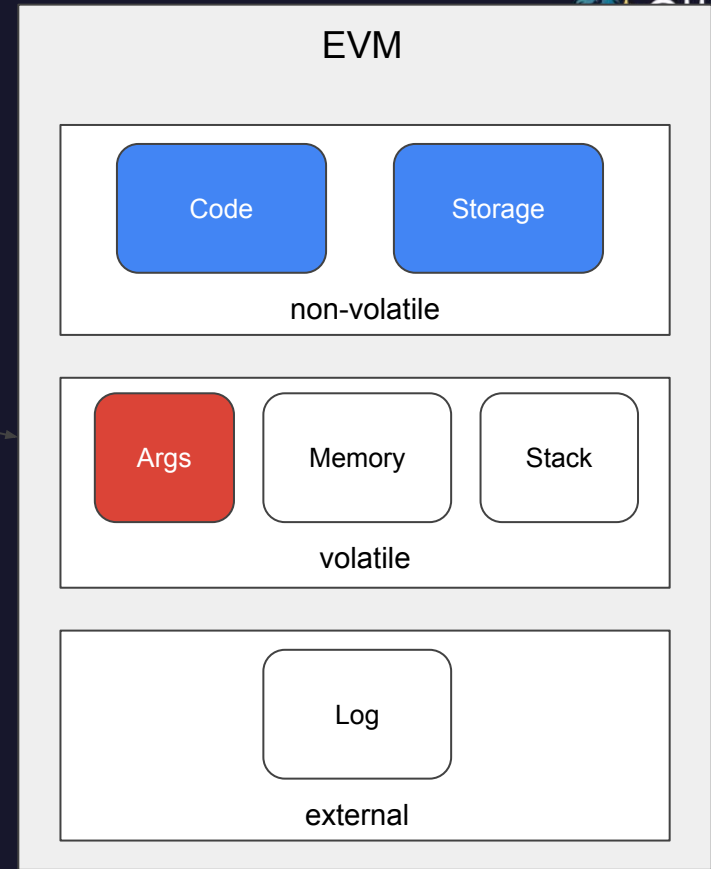
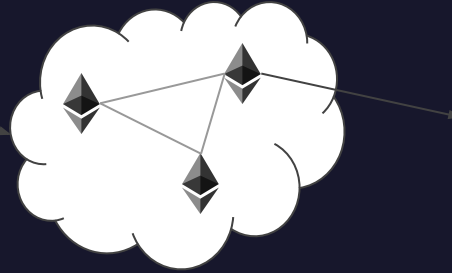
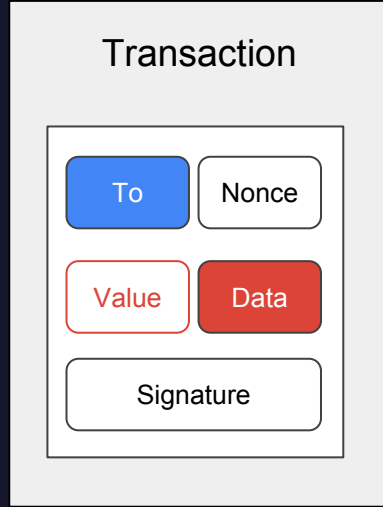
# Precompile Contract

## “System Applications”

- 0x01 - ecrecover
- 0x02 - sha256
- 0x03 - ripemd160
- 0x04 - identity

## Metropolis

- Bigint modular exponentiation
- Addition and scalar multiplication on alt\_bn128
- Ate pairing check on alt\_bn128



Submit & Transaction Execution

```
# Applies the block-level state transition function
def apply_block(state, block):
    # Pre-processing and verification
    snapshot = state.snapshot()
    cs = get_consensus_strategy(state.config)
    try:
        # Start a new block context
        cs.initialize(state, block)
        # Basic validation
        assert validate_header(state, block.header)
        assert cs.check_seal(state, block.header)
        assert cs.validate_uncles(state, block)
        assert validate_transaction_tree(state, block)
        # Process transactions
        for tx in block.transactions:
            apply_transaction(state, tx)
        # Finalize (incl paying block rewards)
        cs.finalize(state, block)
        # Verify state root, tx list root, receipt root
        #print('std', state.to_dict())
        assert verify_execution_results(state, block)
        # Post-finalize (ie. add the block header to the state for now)
        post_finalize(state, block)
    except (ValueError, AssertionError) as e:
        state.revert(snapshot)
        raise e
    return state
```



```
def apply_transaction(state, tx):
    ...
    validate_transaction(state, tx)
    ...

    message_data = vm.CallData([safe_ord(x) for x in tx.data], 0, len(tx.data))
    message = vm.Message(
        tx.sender,
        tx.to,
        tx.value,
        tx.startgas -
        intrinsic_gas,
        message_data,
        code_address=tx.to)

    # MESSAGE
    ext = VMExt(state, tx)

    if tx.to != b'':
        result, gas_remained, data = apply_msg(ext, message)
    else: # CREATE
        result, gas_remained, data = create_contract(ext, message)

    ...
    gas_used = tx.startgas - gas_remained
    ...
```



```
def apply_transaction(state, tx):
    ...
    validate_transaction(state, tx)
    ...

    message_data = vm.CallData([safe_ord(x) for x in tx.data], 0, len(tx.data))
    message = vm.Message(
        tx.sender,
        tx.to,
        tx.value,
        tx.startgas -
        intrinsic_gas,
        message_data,
        code_address=tx.to)

    # MESSAGE
    ext = VMExt(state, tx)

    if tx.to != b'':
        result, gas_remained, data = apply_msg(ext, message)
    else: # CREATE
        result, gas_remained, data = create_contract(ext, message)

    ...
    gas_used = tx.startgas - gas_remained
    ...
```



```
def _apply_msg(ext, msg, code):
    # Transfer value, instaquit if not enough
    snapshot = ext.snapshot()
    if msg.transfers_value:
        if not ext.transfer_value(msg.sender, msg.to, msg.value):
            log_msg.debug('MSG TRANSFER FAILED', have=ext.get_balance(msg.to),
                          want=msg.value)
            return 1, msg.gas, []

    # Main loop
    if msg.code_address in ext.specials:
        res, gas, dat = ext.specials[msg.code_address](ext, msg)
    else:
        res, gas, dat = vm.vm_execute(ext, msg, code)

    if res == 0:
        log_msg.debug('REVERTING')
        ext.revert(snapshot)

    return res, gas, dat
```



pyethapp  
pyethereum  
pydevp2p  
py-evm







# High Level Languages

Solidity, Serpent, Bamboo, Viper, ...

ABI - Solidity Calling Convention

Formal Verification

```
contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function MyToken(
        uint256 initialSupply
    ) {
        balanceOf[msg.sender] = initialSupply;           // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) {
        require(balanceOf[msg.sender] >= _value);        // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                 // Subtract from the sender
        balanceOf[_to] += _value;                         // Add the same to the recipient
    }
}
```





# DApp

Smart Contract

UI + web3.js

STATE OF THE DAPPS					
<div><div></div> Search</div> <div></div>					
219 dapps listed <span>Sort: Updated</span>					
<div>Trustery</div> <div>Mustafa Al-Bassam</div> <div>Public Key Infrastructure and identity management system</div> <div></div> <div>Working Prototype2016-05-19</div>	<div>EtherDesign</div> <div>Ed</div> <div>AI Design Dapp</div> <div></div> <div>Live2016-05-18</div>	<div>KPCS Ethereum</div> <div>Eric Schulte</div> <div>The Kimberley Process certificate issuing process</div> <div></div> <div>Working Prototype2016-05-17</div>	<div>Zonafide</div> <div>Paul Worrall</div> <div>Trusted Zones: stopping loss due to identity theft</div> <div></div> <div>Working Prototype2016-05-17</div>	<div>Proof of Phone</div> <div>Igor Barinov</div> <div>KYC oracle validates and links your phone number and eth address</div> <div>MIT</div> <div>Live2016-05-16</div>	<div>ETH Digger</div> <div>ETH Digger</div> <div>Investment in Ethereum cloud mining using smart contract</div> <div>MIT</div> <div>Live2016-05-16</div>
<div>elcoin</div> <div>Pavel Usenkov &amp; Sergey Primachik</div> <div>Multifunctional platform based on elCoin - ethereum token</div> <div></div> <div>Live2016-05-14</div>	<div>Crypted RPS</div> <div>WhySo3rious</div> <div>Uncheatable RPS Duels with encrypted hands</div> <div></div> <div>Live2016-05-14</div>	<div>CO-AKT</div> <div>Nick Barba, Kevin Kriss</div> <div>The Future of Crowdsourcing: Rapid Business and Group Formation</div> <div></div> <div>Work in Progress2016-05-12</div>	<div>Decibel.LIVE</div> <div>Shane Loomb</div> <div>Real time Noise Monitoring</div> <div></div> <div>Work in Progress2016-05-11</div>	<div>CAPSOL</div> <div>GoDarko</div> <div>Anonymous virtual gossdaching with cryptos and game assets</div> <div></div> <div>Work in Progress2016-05-09</div>	<div>Ethereum games</div> <div>Ethereum games</div> <div>Site for luck-based ethereum games</div> <div></div> <div>Live2016-05-09</div>
<div>FarmShare</div> <div>William E Bodell III</div> <div>Distributed community-supported agriculture (CSA) platform</div> <div></div> <div></div>	<div>Matching Ethers</div> <div>WhySo3rious</div> <div>Strategy and luck game: Round of 2 teams. Bet upside or flipped</div> <div></div> <div></div>	<div>Etheroll</div> <div>James Britt</div> <div>Ether dice game casino / gamble ether</div> <div>proprietary</div> <div></div>	<div>Eaterra</div> <div>Elliot Yeo</div> <div>Planetary Food Sovereignty</div> <div></div> <div></div>	<div>Last is me!</div> <div>Riccardo Casatta</div> <div>Trustless, time based lottery</div> <div>MIT</div> <div></div>	<div>btcrelay</div> <div>Joseph Chow</div> <div>A bridge between the Bitcoin blockchain &amp; Ethereum smart contracts</div> <div>MIT</div> <div></div>

קהל קטן

CITA on Rust



We're hiring!

[join@cryptape.com](mailto:join@cryptape.com)

