Information Technology Fundamentals

Mohammad Hossein Manshaei

manshaei@gmail.com

Web Systems: Bitcoin and Smart Contract
Module 5: Part 9

Contents

Bitcoin Transactions

A Bitcoin Transaction

metadata

input(s)

output(s)

```
{
        "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
        "ver":1,
        "vin_sz":2,
        "vout_sz":1,
        "lock_time":0,
        "size":404,
        "in":[
          {
            "prev_out":{
              "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
              "n":0
            },
          "scriptSig":"30440..."
        },
        {
          "prev_out":{
            "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
            "n":0
          },
          "scriptSig":"3f3a4ce81...."
        }
      ],
      "out":[
        {
          "value":"10.12287097",
          "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
        }
      ]
}
```

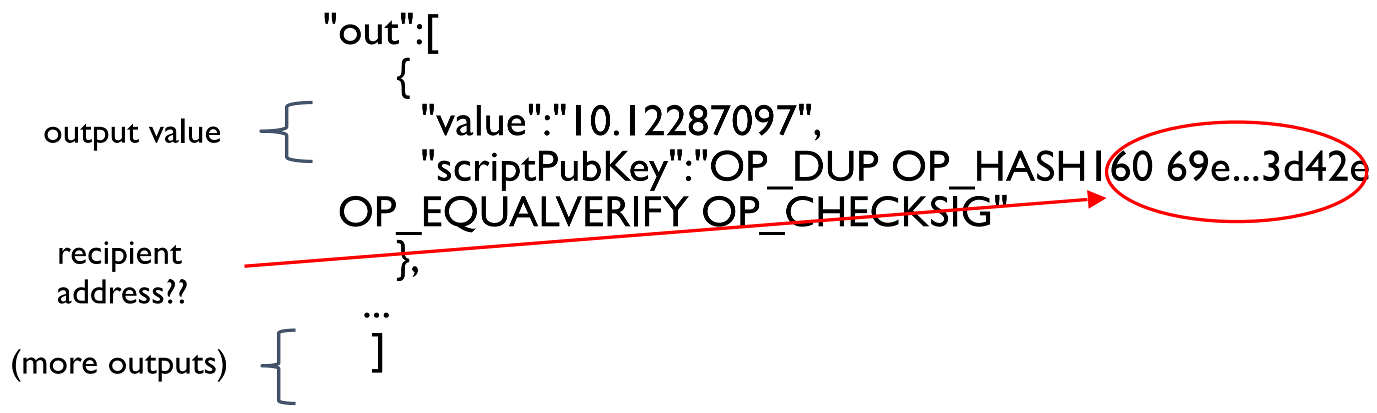
Transaction Metadata

```
                          {
transaction hash      {       "hash":"5a42590...b8b6b",     also serves as a
                                                            unique ID
                      {         "ver":1,
housekeeping          {         "vin_sz":2,
                      {         "vout_sz":1,
"not valid before"    {         "lock_time":0,
housekeeping          {         "size":404,
                          ...
                          }
```

Transaction Inputs

```
                          "in":[
                               {
previous                         "prev_out":{
transaction                 {      "hash":"3be4...80260",
                                   "n":0
                                 },
signature                   {  "scriptSig":"30440....3f3a4ce81"
                               },
                              ...
(more inputs)               { ],
```

Transaction Outputs



```
"out":[
     {
       "value":"10.12287097",
       "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
     },
   ...
   ]
```

Sum of all output values less than or equal to sum of all input values!

If sum of all output values less than sum of all input values, then difference goes to miner as a transaction fee

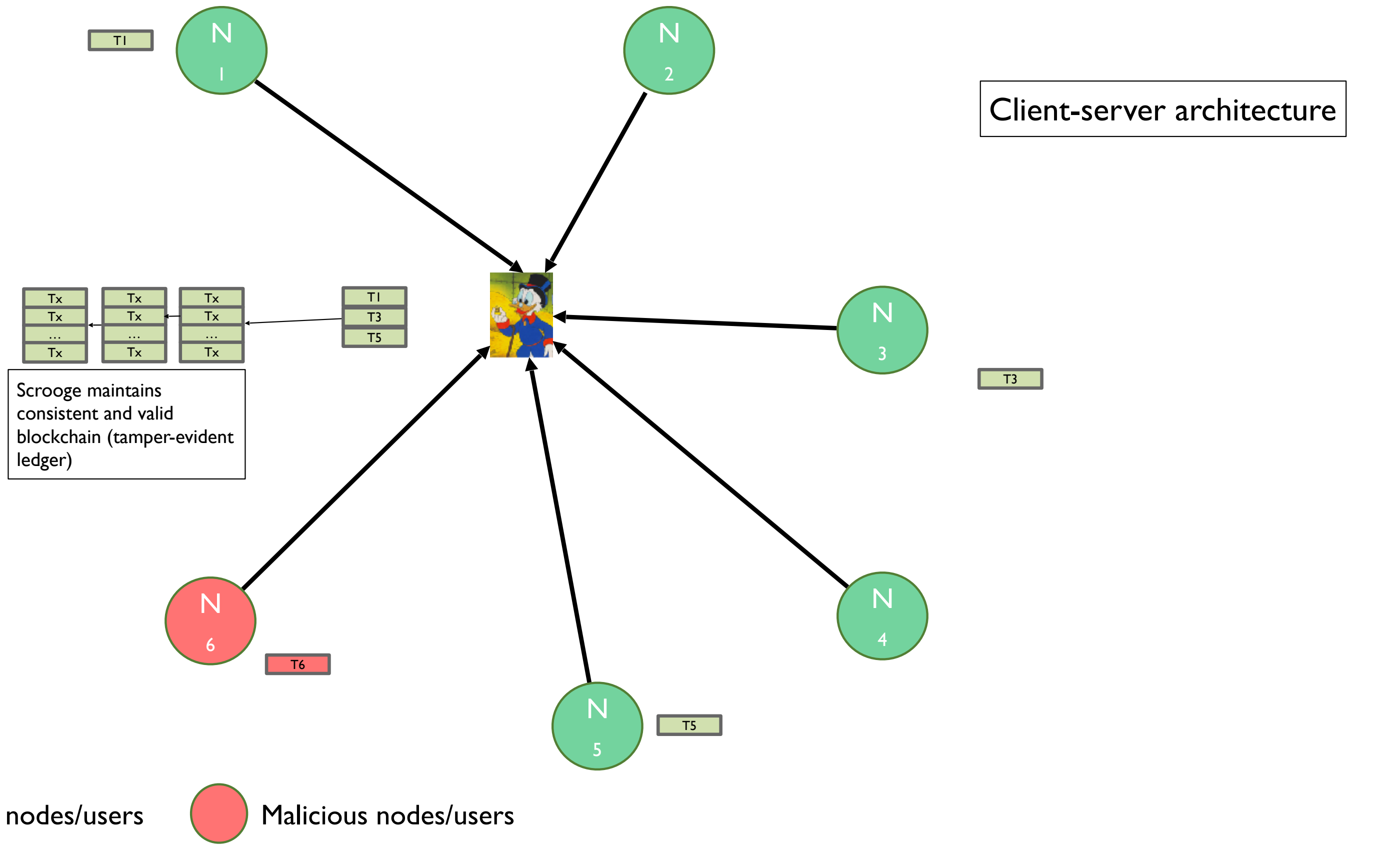
Bitcoin Consensus

- Bitcoin Consensus – most important functionality of the Bitcoin P2P network.

- What do Bitcoin nodes need to reach a consensus on?
  - Which <u>transactions</u> were broadcast on the network.
  - <u>Order</u> in which these transactions occurred.
  - Transactions are valid (output<=input and not double spent).

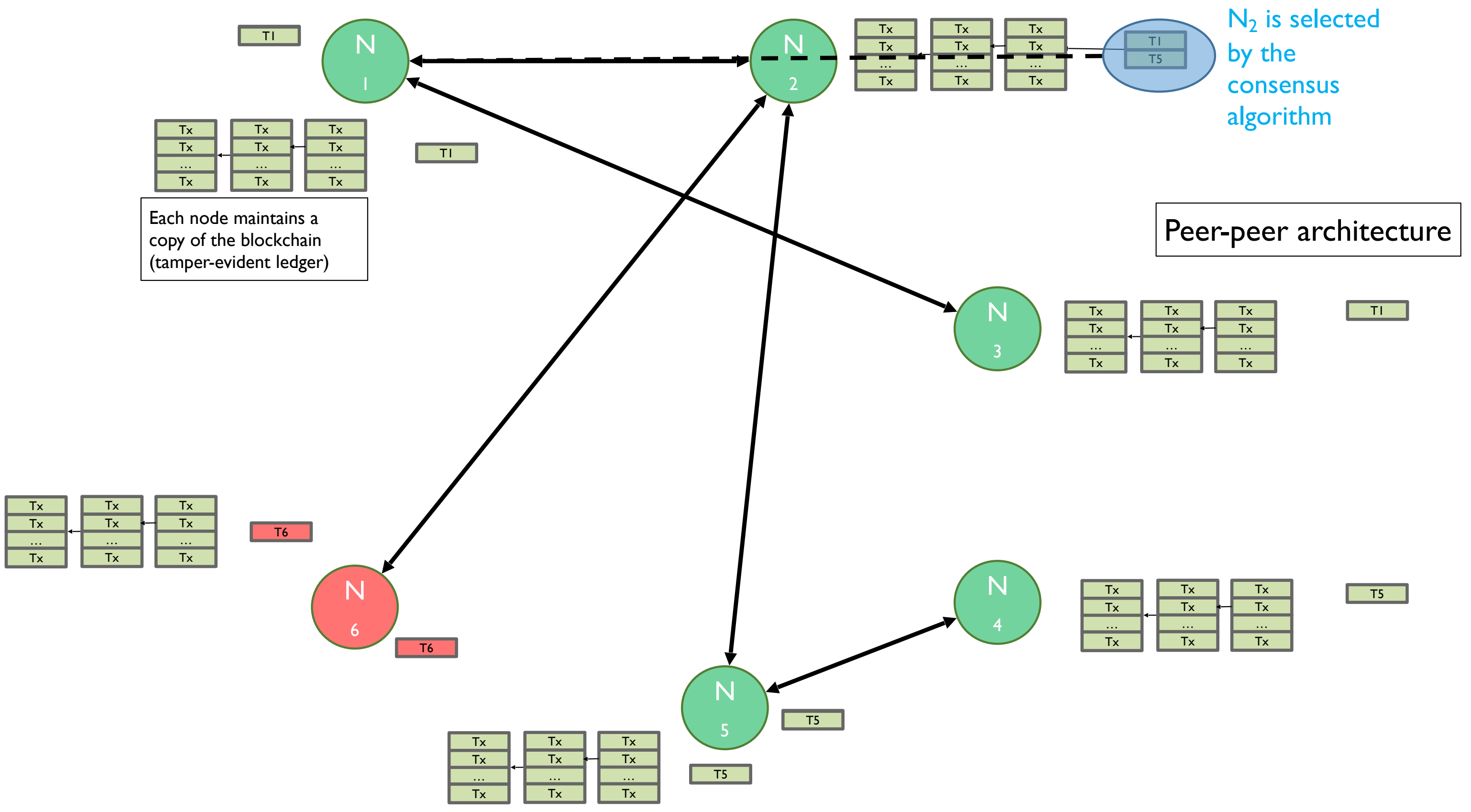
Result of the consensus protocol: Consistent, valid and immutable global transaction ledger.

How Centralized Consensus Works



How Bitcoin Consensus Works

Nodes implicitly accept (reject) this block by building on top of it (or not) when they are selected next by the consensus algorithm



N_2 is selected by the consensus algorithm

Each node maintains a copy of the blockchain (tamper-evident ledger)

Peer-peer architecture

New block forwarded to all nodes who update their blockchains

How consensus <u>could</u> work in Bitcoin

At any given time (in the bitcoin peer-to-peer network):

- All nodes have a sequence of <u>blocks of transactions</u> (called, ledger or block chain) they've reached consensus on
- Each node has a set of outstanding transactions it's heard about (but not yet included in the block chain)
  - For these transactions consensus has not yet happened
  - Each node may have a slightly different outstanding transaction pool

Consensus algorithm (simplified)

1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. In each round a <u>random</u> node gets to broadcast its block
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
5. Nodes express their acceptance of the block by including its hash in the next block they create

Why consensus in Bitcoin is hard?

1. Nodes may crash or become offline.
2. Peer-to-peer network is imperfect.
   - Not all pairs of nodes connected (and may participate).
   - Faults in network.
   - Latency.

No notion of global time ==> constraints the set of consensus algorithms that can be used

3. Nodes may be malicious.

Contents

- Motivation: E-Cash, Credit, Digital Currency, CryptoCurrency, …
- Public Ledger and Blockchain
- BitCoin: Main Idea and Framework
  ✓Transactions
  ✓Proof of Work
  ✓Attacks and Trust
  ✓Block Reward
- Smart Contract: A Blockchain Approach

Proof of Work

Remaining Problems

1. How to pick a random node?

2. How to avoid a free-for-all system due to rewards?
   - Everybody may want to run a bitcoin node in order to get free rewards.

3. How to prevent Sybil attacks?
   - An adversary may create a large number of Sybil nodes to subvert the consensus process.

Solution: Mining using Proof-of-Work (PoW).

Proof of Work (PoW)

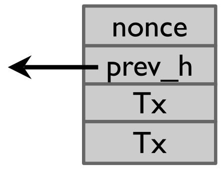
To approximate selecting a random node: *select nodes in proportion to a resource that no one can monopolize (we hope):*

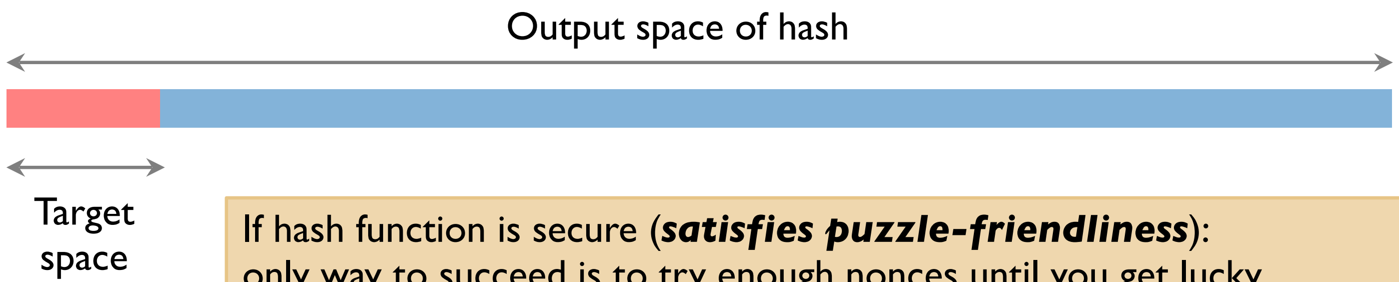
- In proportion to computing power: **proof-of-work** *(Used in Bitcoins).*
- In proportion to ownership of the currency: **proof-of-stake** (*Not used in Bitcoins – but a legitimate model used in other cryptocurrencies*).

Hash Puzzles

To create block, find nonce s.t.
H(nonce ǁ prev_hash ǁ tx ǁ … ǁ tx) is very small.

In other words, *H(nonce ǁ prev_hash ǁ tx ǁ … ǁ tx) < target .*





If hash function is secure (***satisfies puzzle-friendliness***):
only way to succeed is to try enough nonces until you get lucky

Mining Bitcoins in 6 Easy Steps

1. <u>Join</u> the network, listen for transactions.
    a. Validate all proposed transactions.

. <u>Listen</u> for new blocks, maintain blockchain.
    a. When a new block is proposed, validate it.

3. <u>Assemble</u> a new valid block.

4. <u>Find</u> the nonce to make your block valid.

. Hope everybody <u>accepts</u> your new block.

. <u>Profit</u>!

Useful to Bitcoin network

Incentivize miners to do above

Advantage of a PoW System?

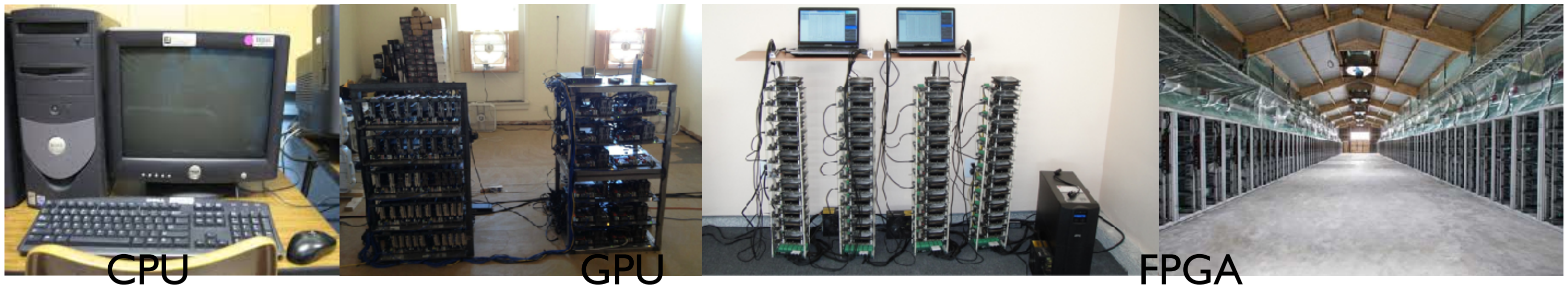
- It completely does away with the problem of magically picking a random node (to propose a block).

- Nodes independently compete by attempting to solve hash puzzles.
  - Once in a while, one (randomly) will succeed and propose the next block.
  - Result: Completely decentralized system ==> No one gets to decide which node proposes the next block.

- Other advantages:
  - Not a free-for-all system ==> Nodes need to work to get paid.
  - Creating new (Sybil) identities is useless without creating new computing power (to solve PoW) to go along with it!

Evolution of Mining

2009 → **2018**



Gold pan

Sluice box

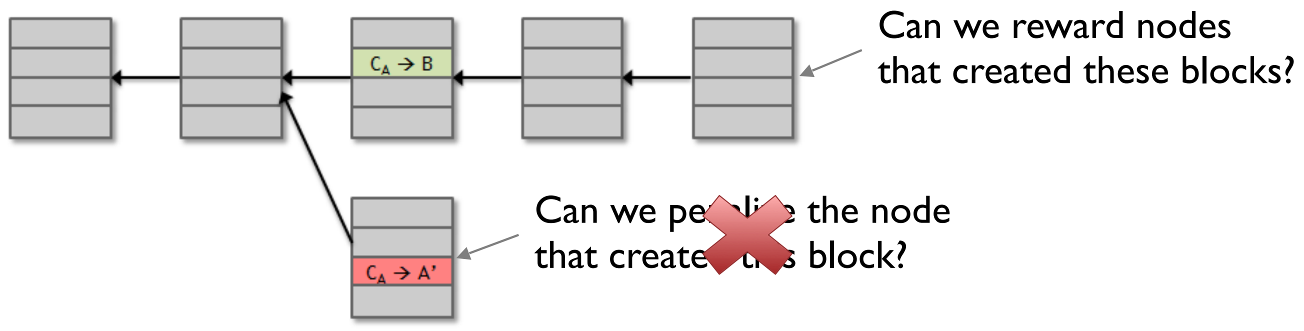
Placer mining

Pit mining

Contents

Attacks and Trust

Assumption of Honesty is Problematic


C_A → B
C_A → A'
Can we reward nodes that created these blocks?
Can we penalize the node that created this block?

In other words, can we give nodes incentives for behaving honestly?

✓ We can utilize the fact that Bitcoin (the currency) has value to achieve distributed consensus!

Contents

Block Reward

Incentive 1: Block Reward

Creator of block gets to

- include special coin-creation transaction in the block.
- choose recipient address of this transaction.

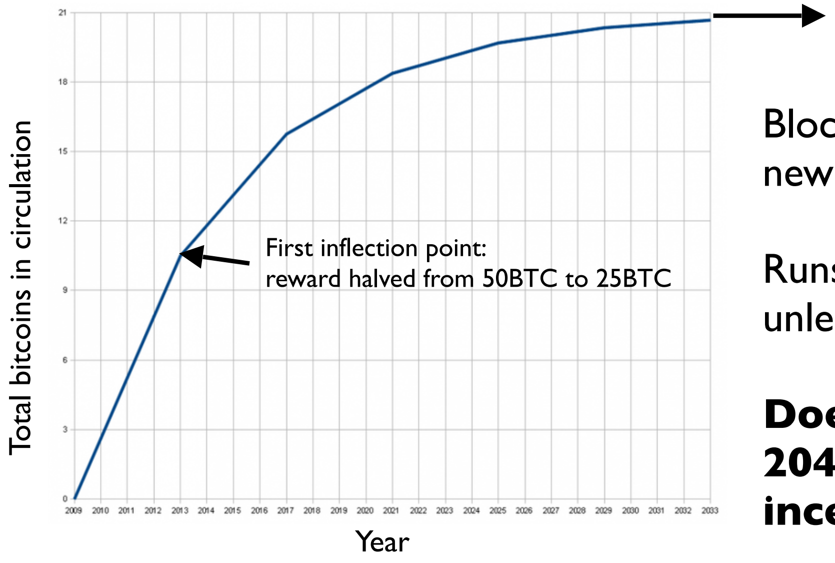
Value is fixed: currently 12.5 BTC, halves every 210,000 blocks created (or every 4 years at the current rate of block creation).

- We are now in the third period – first period block reward was 50 BTC.
- Reward drops to 6.25 BTC on 24th May 2020, 16:11:39 (est).

Block creator gets to “collect” the reward only if the block ends up on long-term consensus branch!

- Subtle but powerful trick: Incentivizes nodes to behave in way that will get other nodes to extend their block.

There's a finite supply of bitcoins



Total supply: 21 million

Block reward is how new bitcoins are created

Runs out in 2040. No new bitcoins unless rules change

Does that mean that after 2040, nodes will no longer have incentive to behave honestly?

Not really!

Incentive 2: Transaction Fees

- Creator of transaction can choose to make output value less than input value.
- Remainder is a transaction fee and goes to block creator.
- Purely voluntary, like a tip.
  - But system will evolve, and will become mandatory, as Block rewards run out.

Contents

- Motivation: E-Cash, Credit, Digital Currency, CryptoCurrency, …

- Public Ledger and Blockchain

- BitCoin: Main Idea and Framework
  ✓Transactions
  ✓Proof of Work
  ✓Attacks and Trust
  ✓Block Reward

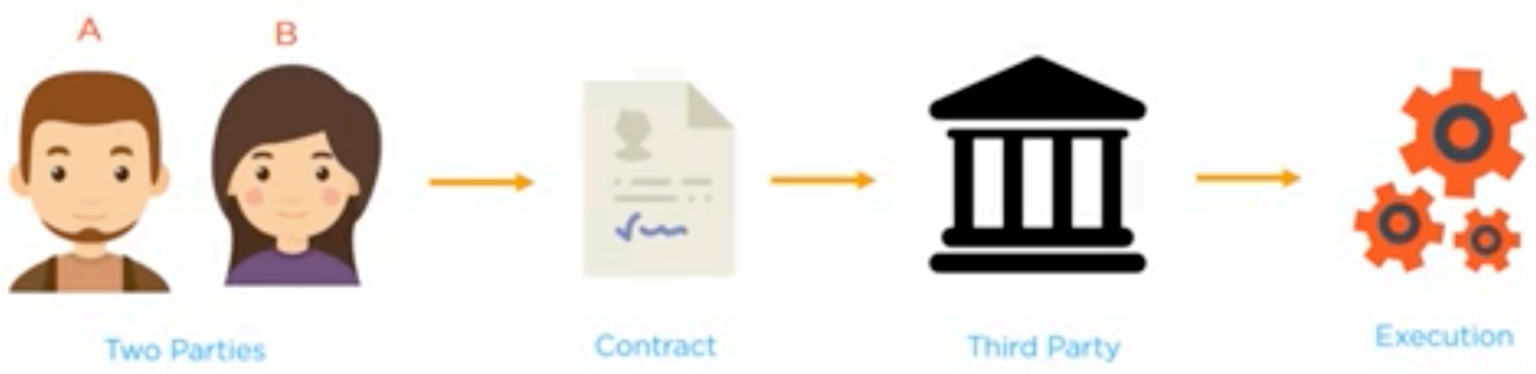
- Smart Contract: A Blockchain Approach

Smart Contract Models

- **Notion of accounts:**
  1. Externally Owned Accounts (governed by users).
  2. Contract Accounts (governed by contracts or code).

- **Smart Contract:** A program that lives on the Blockchain (forever).
  - Written in Solidity – a high-level programming language used by Ethereum (more comes later!).

Traditional Contract vs Smart Contract



- **Smart Contract:** Remove dependency on Third Party (Execution of Contract and Save it)

Smart Contract vs Traditional Contract

Characteristics	Traditional Contract	Smart Contract
Third Party	Yes	No
Execution Time	1-3 Days	Minutes
Remittance	Manual	Automatic
Transparency	Unavailable	Available
Archiving	Difficult	Easy
Security	Limited	Built-in with Cryptography
Cost	Expensive	Cheap

Ethereum: A Blockchain-based Smart Contract Platform

Ethereum

It is a blockchain with expressive programming language, hence programming language makes it ideal for **smart contracts**

- A **smart contract** is a computer program executed in a **secure environment** that directly controls **digital assets** (A program that lives forever on the Blockchain.)
- **Digital Assets:** Domain name, Website, Money, gold, silver, stock share, Game items, Network bandwidth, and computation cycles

How Ethereum Works?

- Two types of account:

1. **Normal account:** like in Bitcoin, has balance and address
2. **Smart Contract account:** like an object: containing (i) code, and (ii) private storage (key-value storage)

   - **Code can**
     - Send ETH to other accounts
     - Read/write storage
     - Call (ie. start execution in) other contracts

Ethereum Language:

1. **Solidity:** An object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum. [Looks like Java]

2. **Serpent:** Looks like Python

Smart Contract Model in Ethereum

- Anyone can create a contract and upload it.
  - Pay a small fee, done by means of a special "transaction".
- Users send "specially crafted" transactions to execute these contracts.
- Miners agree on order of transactions and actually execute contracts using a PoW-based consensus.

- **Ethereum** clients run a special virtual machine, called EVM, which executes the smart contracts.
  - This makes potentials for DoS attack: A malicious miner can DoS attack full nodes by including lots of computation in their Transactions
  - **Solution (GAS):** Charge fee per computational step

Two types of Blockchain Applications

- Public Blockchains:
  - Blockchain participants are not authenticated.
  - Anyone can participate (also referred to as permission-less blockchains).
  - Example: Bitcoin, Ethereum.

- Private Blockchains:
  - Blockchain participants are authenticated.
  - Only authenticated nodes can participate (also referred to as permissioned blockchains).
  - Example: Hyperledger framework by IBM.

Future Directions

- Incentives: Why should users participate?
- Scalability: How to increase number of transactions per second?
- Space: How to reduce the size of the Blockchain?
- Security: How to protect against malicious/vulnerable contracts?
- Applications in other domains/businesses: For authentication, integrity, record-keeping, etc.
  - IoT
  - Supply Chain
  - Financial Services
  - Spectrum Management
  - ……