CSBC2000

Week 1 | Class 4

Smart Contracts

Recap

- We saw the technical differences between a blockchain and a DB
- Learnt of different kinds of distributed databases
- Learnt how p2p networks work
  - DHTs: Chord/Kad
- Bitcoin, ETH networking
- IPFS network

This class

- Smart contracting!
- Review what smart contracts are
- DApps
- Remix: Solidity
- Local env: Move

State

- Sum total of variables in a program at a given snapshot of its execution
- In blockchain, it represents data on chain (e.g. UTXO on Bitcoin)
- State transitions are functions that map one state to another state
- Smart contracts define the logic of state transitions in the blockchain
- A state machine (in blockchain) is a VM that can represent the state of a collection of programs running on it

Types of smart contracts

- Smart legal contracts
  - The "end goal"
  - Represent consititutional/municipal/etc laws encoded into logic
  - Can enable decision making

DAOs

- Corporations that have interactions encoded into smart contracts
- Recall Steemit
- Open source (e.g. Redhat) has this informally
- DAO Hack

Application Logic

- Contracts you can use to store crucial information in an app
- Typically required when there is a *trust problem*
- E.g. CryptoBike

What is a contract anyway?

- "an agreement consisting of a promise and consideration enforceable by law"
- Informally, a meeting of minds
- Agreement, execution, enforceability
- Smart contract cuts a lot of these steps

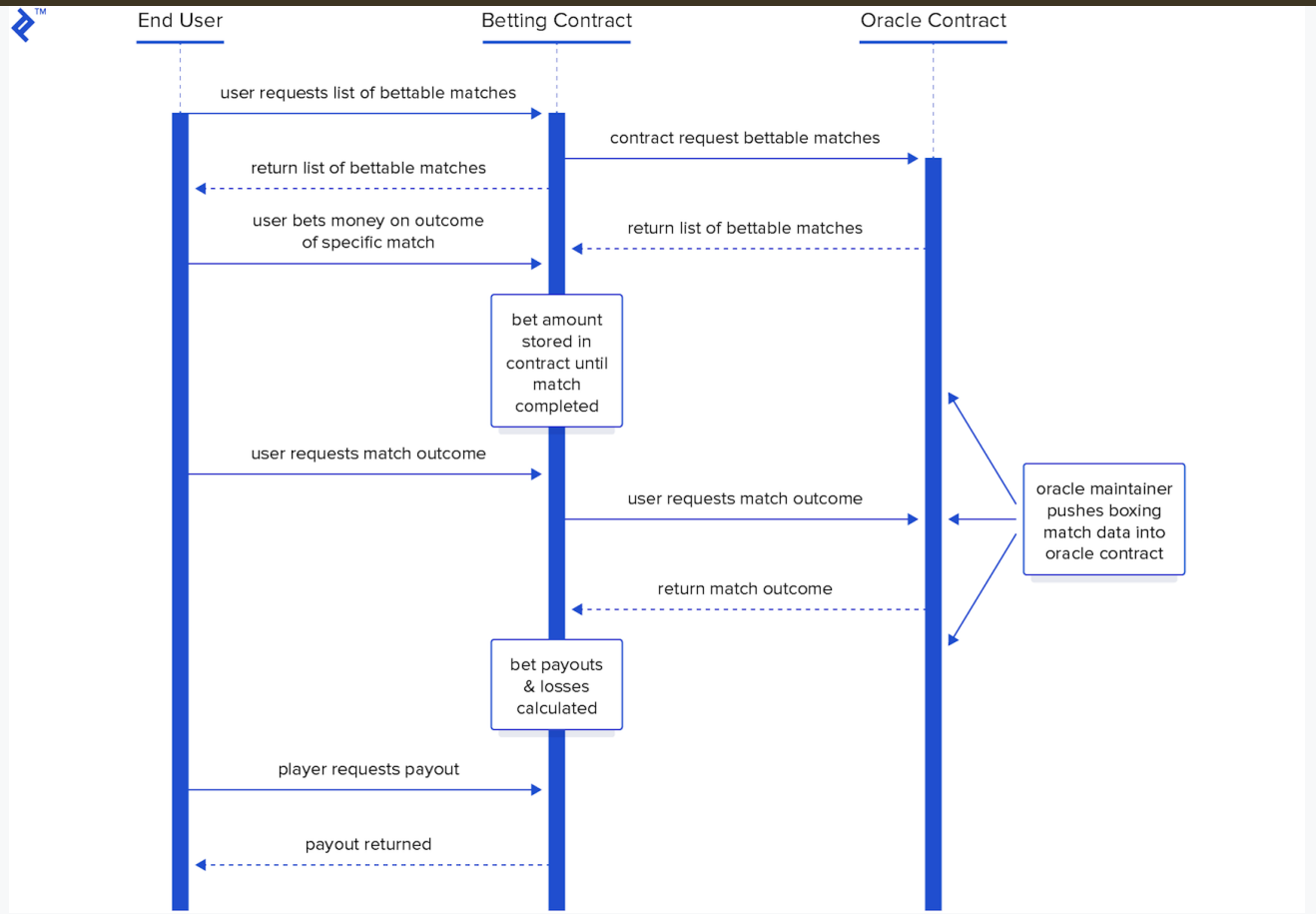
Imperative smart contracts

- Parties come into agreement and clearly define the nature of their agreement
- Then, a trusted party codes these into agreements
- Executed and placed into blockchain
- Need enforceability

Oracles

- "A means for smart contracts to access data from the world outside the blockchain. A type of smart contract themselves, oracles take data from the outside world and put it into the blockchain for other smart contracts to consume."
- Representing reality in a digitized format is hard; can compromise depending on the use case (e.g. sports)

Oracles

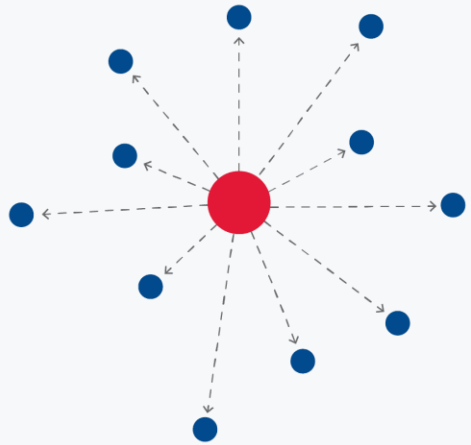

End User
Betting Contract
Oracle Contract
user requests list of bettable matches
contract request bettable matches
return list of bettable matches
user bets money on outcome of specific match
return list of bettable matches
bet amount stored in contract until match completed
user requests match outcome
user requests match outcome
oracle maintainer pushes boxing match data into oracle contract
return match outcome
bet payouts & losses calculated
player requests payout
payout returned

DApps

- Recall networking: traditional App sits on a host and uses horizontal scaling to service requests
- They combine this with a CDN, etc. To help content delivery using caching
- Cloud based deployments are further centralizing

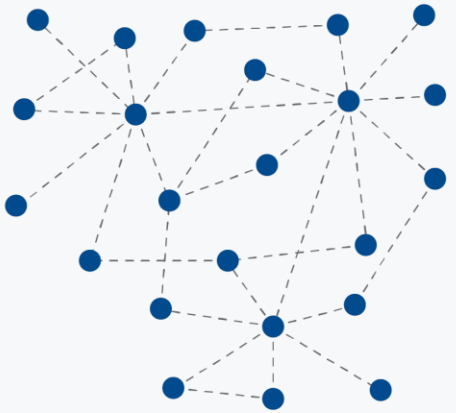
application deployments

- Persistent outages



Dapps

- Decentralized architecture
- Important events are recorded on chain
- Other stuff is stored on a decentralized CDN (e.g. Swarm, IPFS, etc.)
- No owner, no single point of failure
- Premise:



I. There is no single company, program, or member that controls the flow of data in the network.

II. Every participant has access to the data stored in the blockchain through their own node.

Types of Dapps

- Type I

- Supported by their own blockchain

- e.g. Bitcoin, Ethereum, EOS, Steem

- Type II

- Dependent on the blockchain of Type 1 DApps. They are primarily driven as protocols and generate tokens to run (like the one we'll make)

- e.g. Omni, Golem

- Type III

- Use the protocol of Type 2 DApps to perform their functions

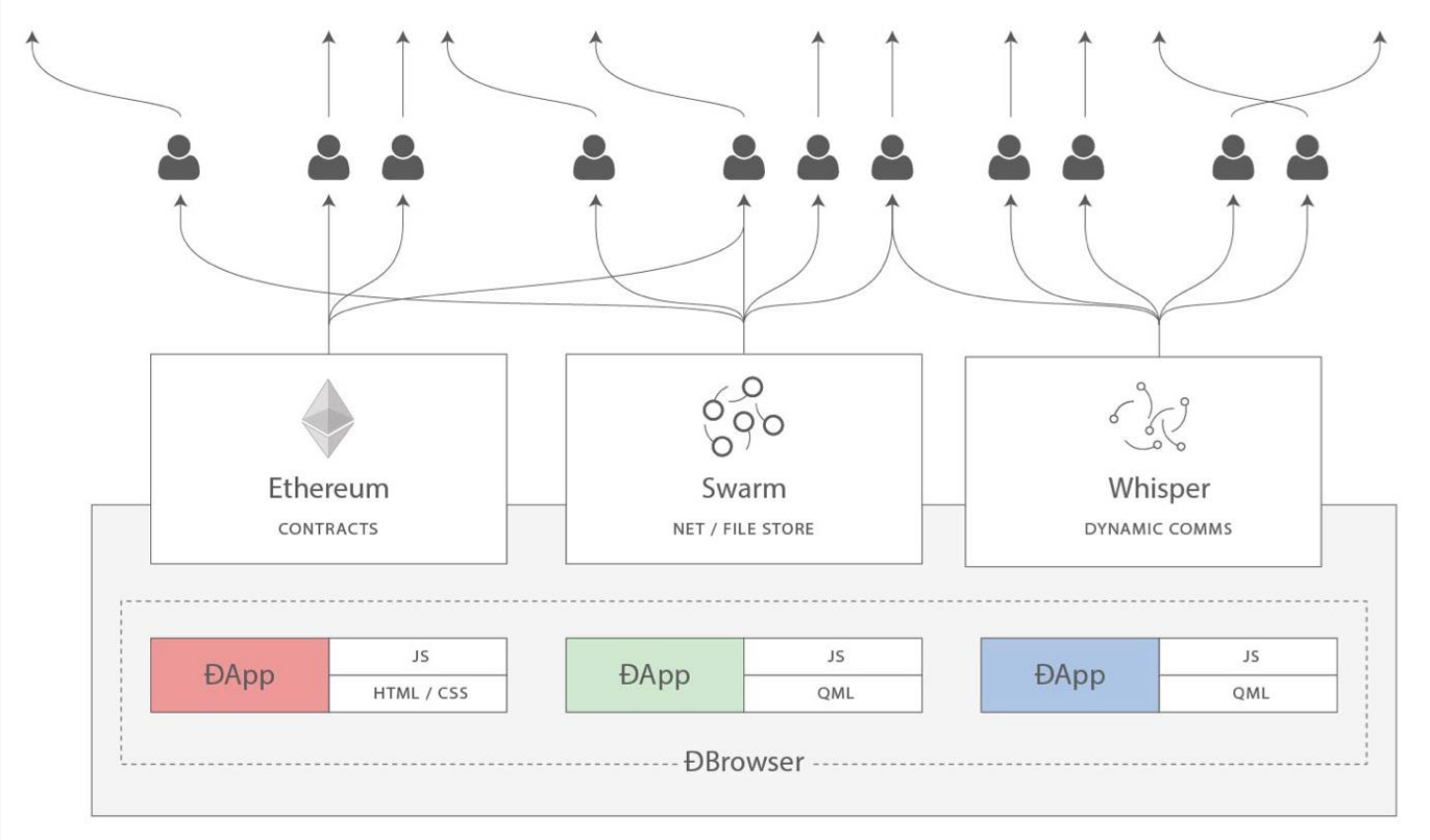
- SAFE network, which uses the Omni protocol to issue SafeCoins

Data Privacy

- Centralized apps control your data
- They are bound by legislation, but do first and ask for forgiveness later
- All the Big 5 companies are undergoing/have undergone an anti-trust
- DApps give control of data back to the user, since they can perform all their interactions anonymously

- Barring cryptoforensics

3 Layer Decentralized Architecture



3 Layer Decentralized Architecture

- Smart contract: Allows state to be stored and updated
- Decentralized storage: allows assets to be stored with high integrity and availability
- Dynamic communications: Allow Dapps to seamlessly communicate with each other
- Many solutions exist for these layers

Solidity

- Created for developing DApps on Ethereum

- "Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state"

OpenZeppelin

- Define smart contract standards
- In Solidity, they represent interfaces
- Each token is its own mini-database of who owns what
- Tokens are variables defined in a smart contract that are written by humans that intend some value assignment

ERC-20

- "Fungible" tokens: can be swapped for another asset
- Can be broken down into smaller components
- Token type for things that are money-like

ERC-721

- "Fungible" tokens: cannot be swapped for another asset
- Cannot be broken down into smaller components
- Token type for things that are like collectibles (e.g. pokemon cards, limited edition things)
- CryptoKitties

Move

- Created by Facebook Libra (now Diem) "for secure, sandboxed, and formally verified programming"
- Not in production and still in experimental stage
- Move: `let x = y` (C++: `auto x = std::move(y)`)

Cardano

- Uses Haskell
- Functional programming is a safer paradigm
- Allows one to define *compositions* for safer usage of state

Future of DApps

- Autonomous agents

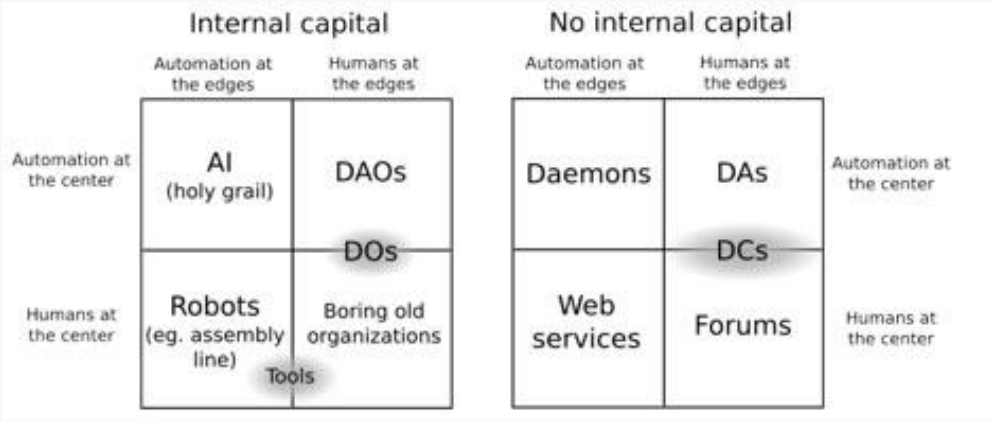
- Set of smart contracts that automatically perform the role of a business

- e.g. Automated banks, automated taxes

- Perfect Oracles

- DAOs at scale

- Buzzword convergence: Quantum-Blockhain-AI



Questions/Comments

Lets code!