Introduction to dApps

Alex Stokes

10/20/20

UC Davis guest lecture

Who am I?

- Alex Stokes
- @ralexstokes
- Research and development at the Ethereum Foundation
- Working on blockchain scalability
    - Ethereum 2.0

- Researching/studying blockchains since ~2012
- Full time since early 2017

Goals

- What is a dApp?
- Why are they written in Solidity?
- How does Ethereum fit into all of this?

Agenda

- Why the hype?
- Ethereum as “programmable” blockchain
- Smart contracts / dApps are “subprotocols” of Ethereum
- Examples of dApps
- High-level overview of the EVM
- Intro to Solidity
- Look at the code for some smart contracts!

Why the hype?

Internet : information :: Blockchains : value

What is money?

- Money is a way to transfer “chunks” of value between us
  - Cowrie shells
  - Rai stones in Micronesian island of Yap
  - Animal pelts
  - Cigarettes
- Standardized units of value
  - Currency

- Debt, David Graeber

What is money?

- Historically, hard to build money on the internet

- Can cheaply copy a digital artifact…
  - “You wouldn’t copy a car”

What is money?

- Blockchain
  - the technical construction that implements a cryptocurrency

- Cryptocurrency
  - Digital “thing” we can use as money

Blockchain construction

- “Proofs and promises”

- Cryptography
  - “Property rights”, ownership
- Economics / game theory
  - Incentives to behave in certain ways

- Latter bit was more the breakthrough in Bitcoin
  - No trusted third party w/ coin incentive

Bitcoin: a shared ledger

- Get one “application”
- We make a coin/token (BTC) and we record ownership on a shared ledger
- Incentives in the system to maintain the ledger
- Ownership is protected with cryptography
  - Very similarly to how e-commerce transactions are secured
- “Alice sends 10 BTC to Bob”
- “Bob sends 4.5 BTC to Charlie”

Ethereum: the ledger can do whatever you want

- Single purpose => general purpose
- All applications on top of the base layer share network security
- Entries on the ledger can be application-specific
  - Not just coin balances
  - Mapping account address to strings
    - E.g. Ethereum Name Service (ENS)

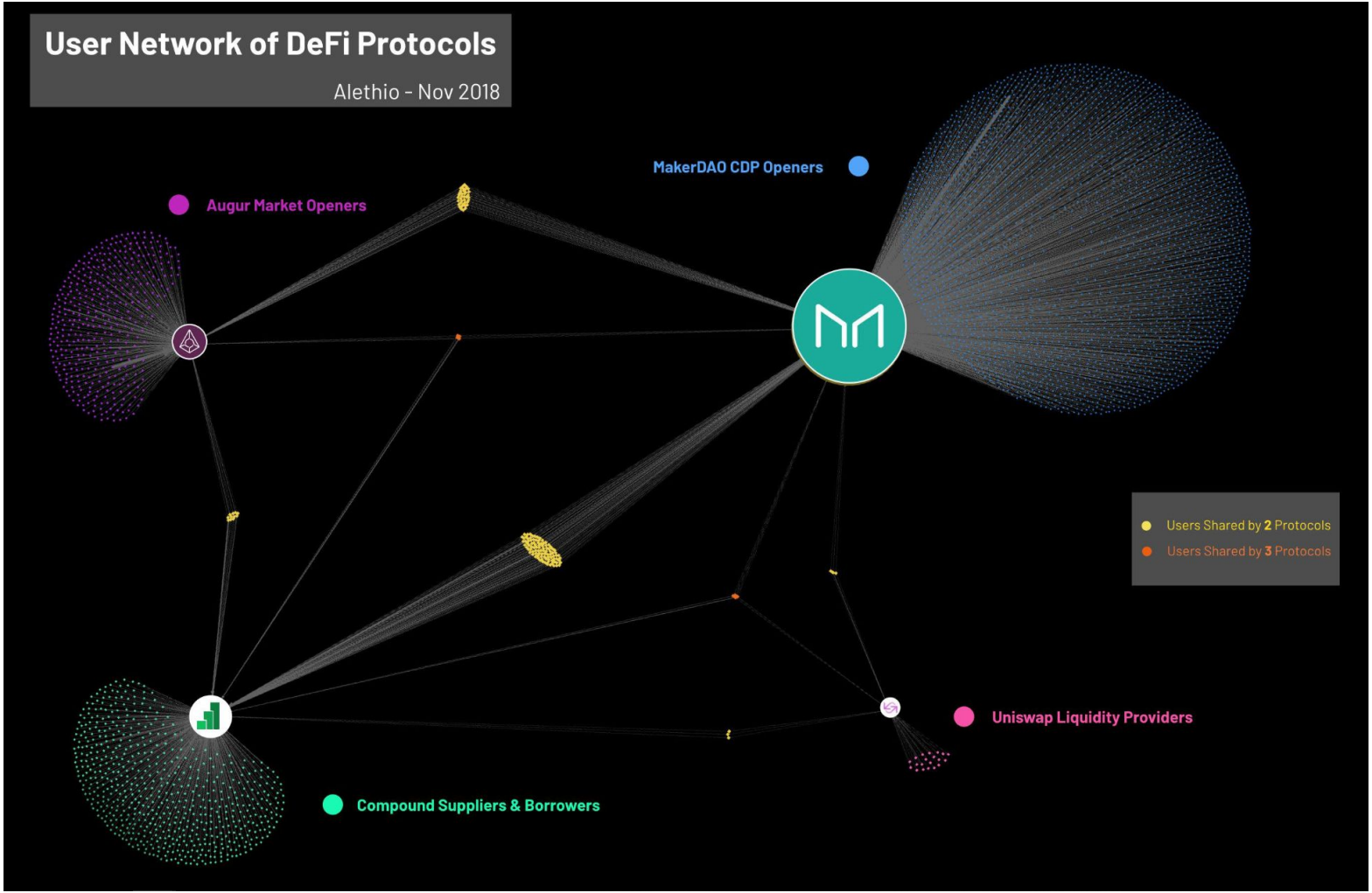
- Importantly, all applications can easily talk to each other
  - See: “DeFi” lecture later in syllabus

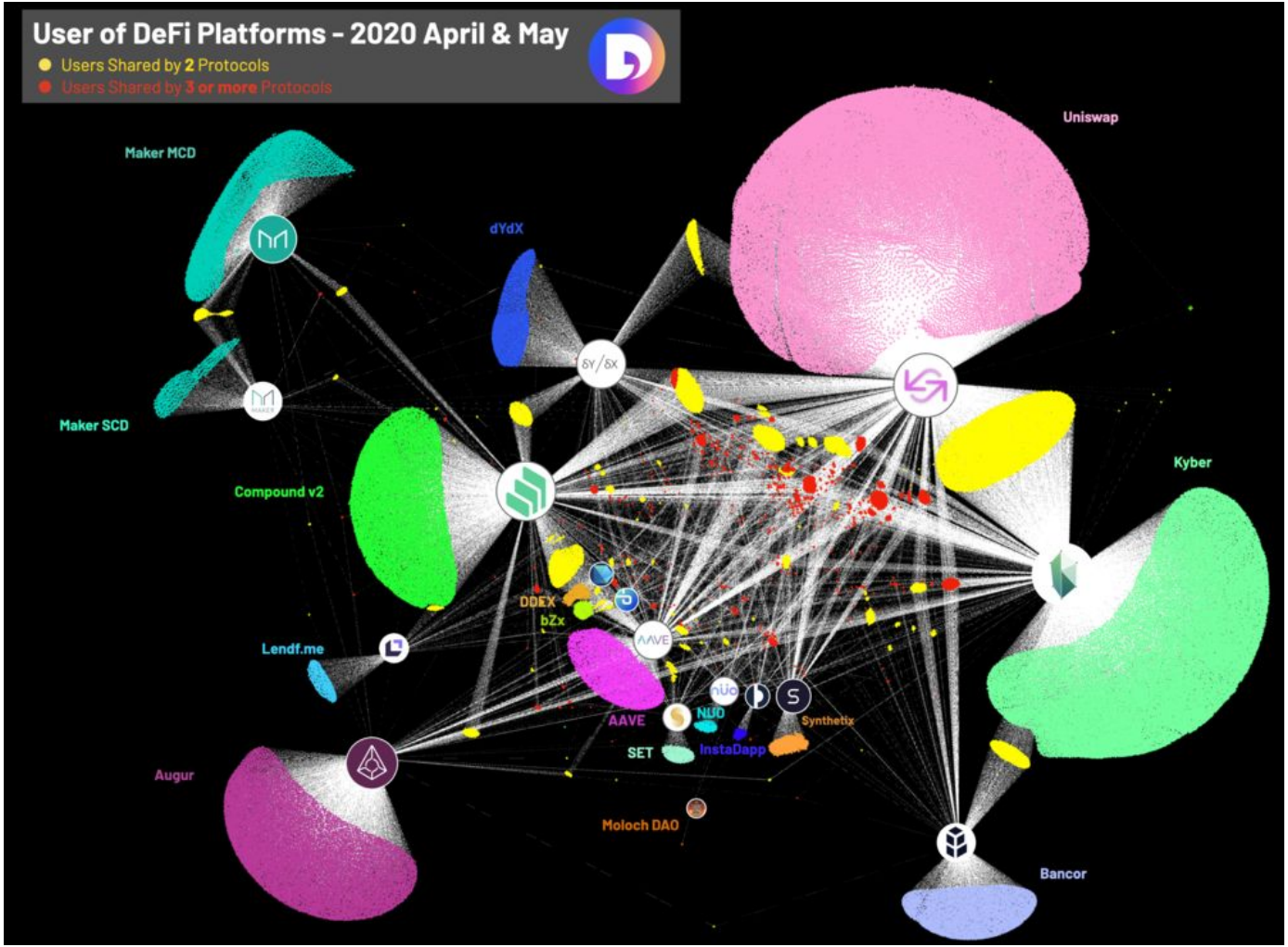
dApps

- Decentralized applications -- “dApps”

- Rather than building your own blockchain, can re-use shared primitives
  - Networking
  - Consensus
  - Security

- Now, barrier to entry is lower and we are off to the races!


User Network of DeFi Protocols
Alethio - Nov 2018
MakerDAO CDP Openers
Augur Market Openers
Users Shared by 2 Protocols
Users Shared by 3 Protocols
Uniswap Liquidity Providers
Compound Suppliers & Borrowers


User of DeFi Platforms - 2020 April & May
Users Shared by 2 Protocols
Users Shared by 3 or more Protocols
Maker MCD
Maker SCD
dYdX
Uniswap
Compound v2
Kyber
DDEX
bZx
Lendf.me
AAVE
Augur
NUO
Synthetix
SET
InstaDapp
Moloch DAO
Bancor

Analogy to Web today

- “Subprotocols”

- The Web is mostly an absurd amount of HTTP transactions
  - Running on TCP/IP
- Twitter is a “subprotocol” of the HTTP protocol

Aside: blockchain scalability

- Early but, likely to have layers of blockchain stack
- Web
  - HTTP / TCP / IP/ Ethernet
- Crypto
  - L2 / L1
    - “Layer 3” to implement operations across L2?
    - ???
  - Base layer: L1
  - Protocols that “sit on top of L1”: L2
    - Roll-ups, Plasma, State channels
- Can further decompose to optimize/specialize
  - Data availability layer (eth2 phase 1)
  - Data validity layer (eth2 phase 2)

Examples of dApps

- Tokens
  - ERC-20
- Loans
  - MakerDAO, “decentralized credit facility”
- Stablecoins
  - Price-stable tokens: DAI, USDC
- Decentralized exchange (DEX)
  - Uniswap, trade arbitrary ERC20 pairs
- Prediction markets
  - Augur, Omen
- Games, NFTs, collectibles
  - Cryptokitties, Rarible, Dark Forest
- DAOs
  - Quadratic voting/funding

Let’s dig in

How do we compute?

- Include a Turing-complete virtual machine for interpreting blockchain transactions

- Blockchain
  - “Chain of blocks”
  - Each block:
    - includes a reference to its parent (the hash)
    - Includes an *ordered* bundle of transactions
  - Replay all transactions in the exact same order to derive state of the network
  - E.g. look at your bank account statement

Bitcoin computer

- Script
- Relatively simple stack-based language
- I can move coins from one address to another
- Multi-signatures (m-of-n)
- Some very basic computation
- But, e.g., can’t write a for loop

Bitcoin computer

Standard Transaction to Bitcoin address (pay-to-pubkey-hash)

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

To demonstrate how scripts look on the wire, here is a raw scriptPubKey:

```
  76       A9             14
OP_DUP OP_HASH160    Bytes to push

89 AB CD EF AB BA AB BA AB BA AB BA AB BA AB BA AB BA   88         AC
                      Data to push                     OP_EQUALVERIFY OP_CHECKSIG
```

Note: scriptSig is in the input of the spending transaction and scriptPubKey is in the output of the previously unspent i.e. "available" transaction.

https://en.bitcoin.it/wiki/Script

Bitcoin computer

- Simple on purpose
- Easier to secure a less-complex thing
- Don’t pay the complexity cost for something you don’t need

- … but, it is pretty obvious there is demand for a more general computer beyond token transfer

Ethereum computer

- Rather than build N blockchains for N applications
- Build 1 blockchain for N applications
  - Requires transaction semantics are general purpose

- Ethereum Virtual Machine (EVM)

EVM

- Stack-based VM
  - Also has ephemeral memory, persistent memory (storage)
- Basic arithmetic, logic, control flow

- Transactions are either:
  - Ether transfers (just ETH)
  - EVM computations (maybe ETH + EVM bytecode)
- Smart contracts are “stored programs”
  - EVM bytecode that has been deployed to a particular address
- “Call a contract”
  - Transaction to some address (with bytecode) that receives transaction payload as input data

EVM

- How to stop an “infinite loop” transaction?
  - Every transaction has to pay a fee proportional to the resources they consume
- Gas
  - Every bytecode has a gas cost
  - Execution is metered
  - Transaction declares gas price (and max gas)
  - Sender pays a `fee = gas_used * gas_price`

- Example gas schedule
  - https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs/edit#gid=0
    - May be a little stale, but you get the idea

EVM

```
05: 34     CALLVALUE
06: 80     DUP1
07: 15     ISZERO
08: 61     PUSH2 0x0010
0B: 57     JUMPI
0C: 6000   PUSH1 0x00
0E: 80     DUP1
0F: FD     REVERT
10: 5B     JUMPDEST
11: 50     POP
12: 60C7   PUSH1 0xc7
14: 80     DUP1
15: 61001F PUSH2 0x001f
18: 60     PUSH1 0x00
1A: 39     CODECOPY
1B: 60     PUSH1 0x00
1D: F3     RETURN
1E: 00     STOP
```

https://blog.trustlook.com/understand-evm-bytecode-part-1/

Live example on Etherscan

- ETH transfer
  - https://etherscan.io/tx/0xb9656b2d50c67a8a50015ab7e7c77e089417610d5855ba542a713e4e39cfb49a

- Contract interaction
  - https://etherscan.io/tx/0x337f8f9b1677d4868672d998bba7c323b8fb685cfe39def255e6152eb20df5d7

What are smart contracts?

- This stored EVM bytecode on chain
- Link contracts together to build dApps
  - E.g. Uniswap

- One instance of a smart contract has contract-specific state on-chain.
- Moreover, can interact with any other contract on-chain.
- This is a big deal!

Solidity

- EVM is low-level
- You wouldn’t write x86 assembly today
- Write a higher-level language that compiles to your execution target

- Solidity is the premier high-level language for the EVM

Solidity

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.5.0 <0.8.0;

contract C {
    function f(uint a, uint b) public view returns (uint) {
        return a * (b + 42) + block.timestamp;
    }
}
```

https://solidity.readthedocs.io/en/v0.7.4/contracts.html

Solidity example with state

- Move to Remix IDE...

ERC-20 token contract

- Review source code of OpenZeppelin token contract
  - https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

Uniswap

- Trace Etherscan transaction to see a dApp in action
  - https://etherscan.io/tx/0x852fcc5d2d96eed3b2fb14cbd9e01d23796b50ba73a91f14d2f3b0510b889851
  - https://github.com/Uniswap/uniswap-v2-periphery/blob/master/contracts/UniswapV2Router02.sol#L284

SECURITY SECURITY SECURITY

- Immature tooling
- Bugs mean lost $$$
- _literally_ billions of dollars of funds lost or stolen at this point

SECURITY SECURITY SECURITY

- Infamous example:
  - DAO hack
  - Function re-entrancy
  - Refer: https://quantstamp.com/blog/what-is-a-re-entrancy-attack

https://medium.com/coinmonks/protect-your-solidity-smart-contracts-from-reentrancy-attacks-9972c3af7c21

```
function withdraw() external {
    uint256 amount = balances[msg.sender];
    require(msg.sender.call.value(amount)());
    balances[msg.sender] = 0;
}
```

SECURITY SECURITY SECURITY

- “More like launching a rocket, than launching a new photo sharing app”
- Testing, audits, formal verification
- ALWAYS do your own research before putting money into this stuff

- Exciting, but at the same time, high risk environment

Questions?

- Hopefully, you see the potential here and are excited to go further :)

- Feel free to reach out directly, happy to help anyone orient/ideate/etc...
- Twitter: @ralexstokes