Blockchains & Cryptocurrencies

DeFi

Instructor: Matthew Green & Abhishek Jain
Spring 2023

News?

News?

MakerDAO’s Plan To Enable MKR As Collateral For DAI Draws Comparisons To UST

Proposed Change Forms Part Of Co-Founder Rune Christensen’s Controversial ‘Endgame’ Plan

By: **Owen Fernau** February 28, 2023

News?

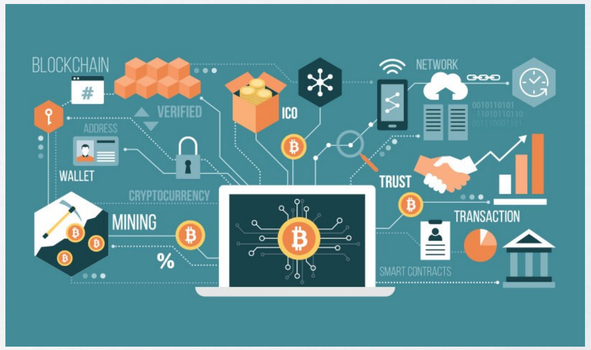
MakerDAO’s Plan To Enable MKR As Collateral For DAI Draws Comparisons To UST

Proposed Change Forms Part Of Co-Founder Rune Christensen’s Controversial ‘Endgame’ Plan

By: **Owen Fernau** February 28, 2023

Today

- We're basically going to talk about DeFi
- With a couple of finish-up points re: Solidity



Contract upgrades

- Ethereum contracts are not (natively) upgradeable
  - Once a contract is deployed, it can self-destruct
  - But its code cannot be changed
  - But some contracts need to be upgraded (bug fixes, etc.)
    - How are we going to handle this?

Upgrade approaches

- There are several:
  - Don't allow upgrades ever! (e.g., deploy all new contracts)
    - Downside: must convince users to migrate,
      will need to manually copy over any existing state

Upgrade approaches

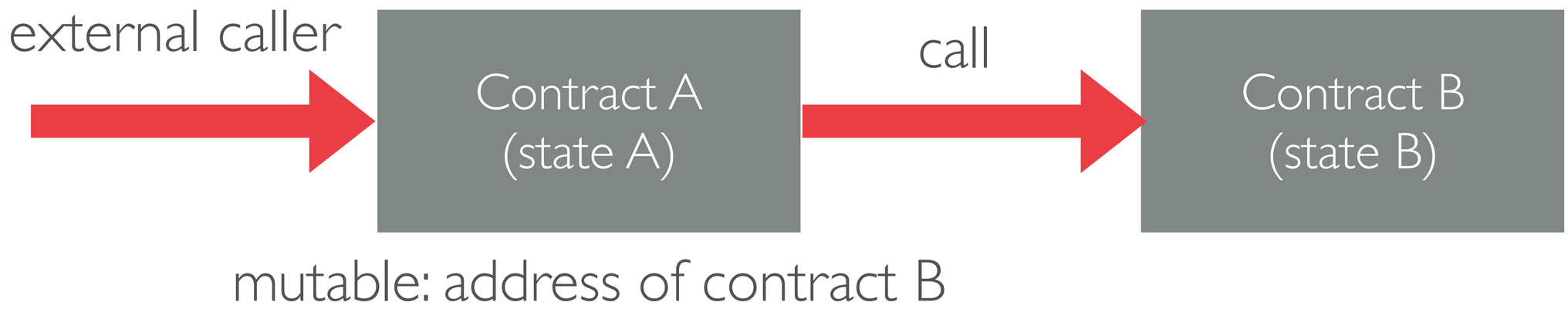
- There are several:
  - Implement upgrade capability in your program logic (somehow)
  - How do we do this?

Proxy pattern

- Based on two standard features of Ethereum
  - An Ethereum contract (A) can call another contract (B) as a "subroutine"

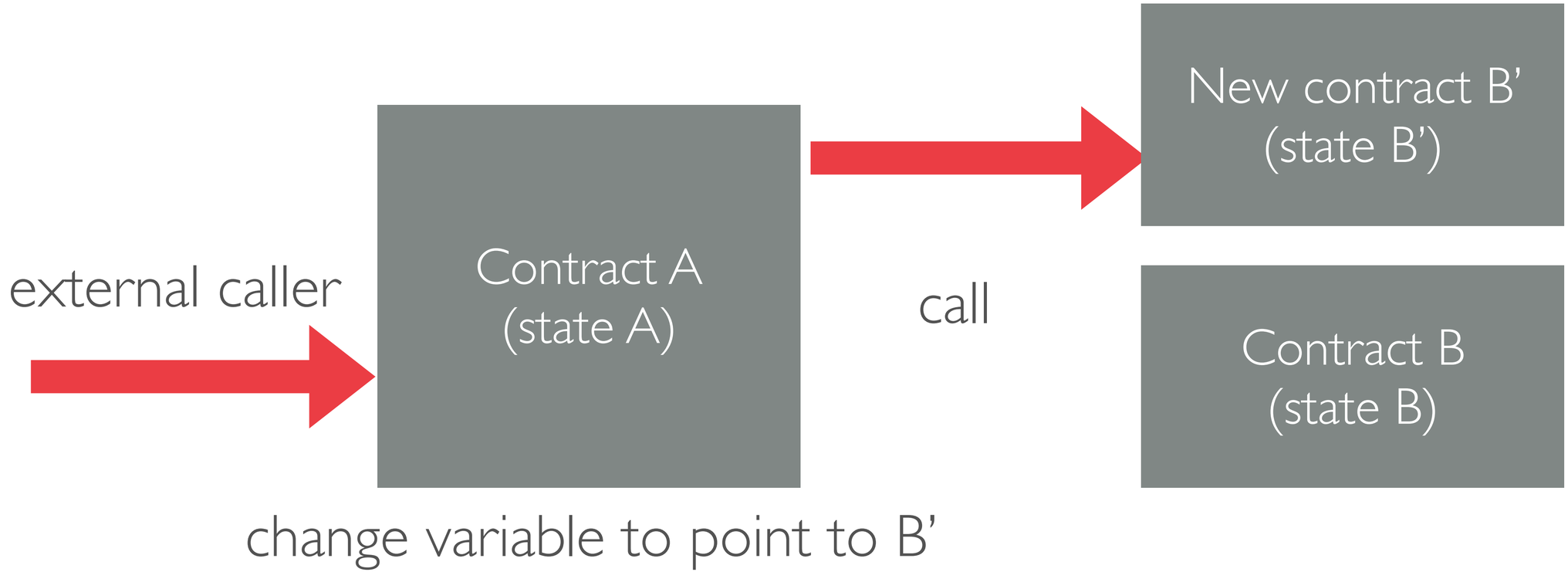
Proxy pattern

- Based on two standard features of Ethereum
  - An Ethereum contract (A) can call another contract (B) as a "subroutine"
  - The location of that second contract can be stored in a mutable variable



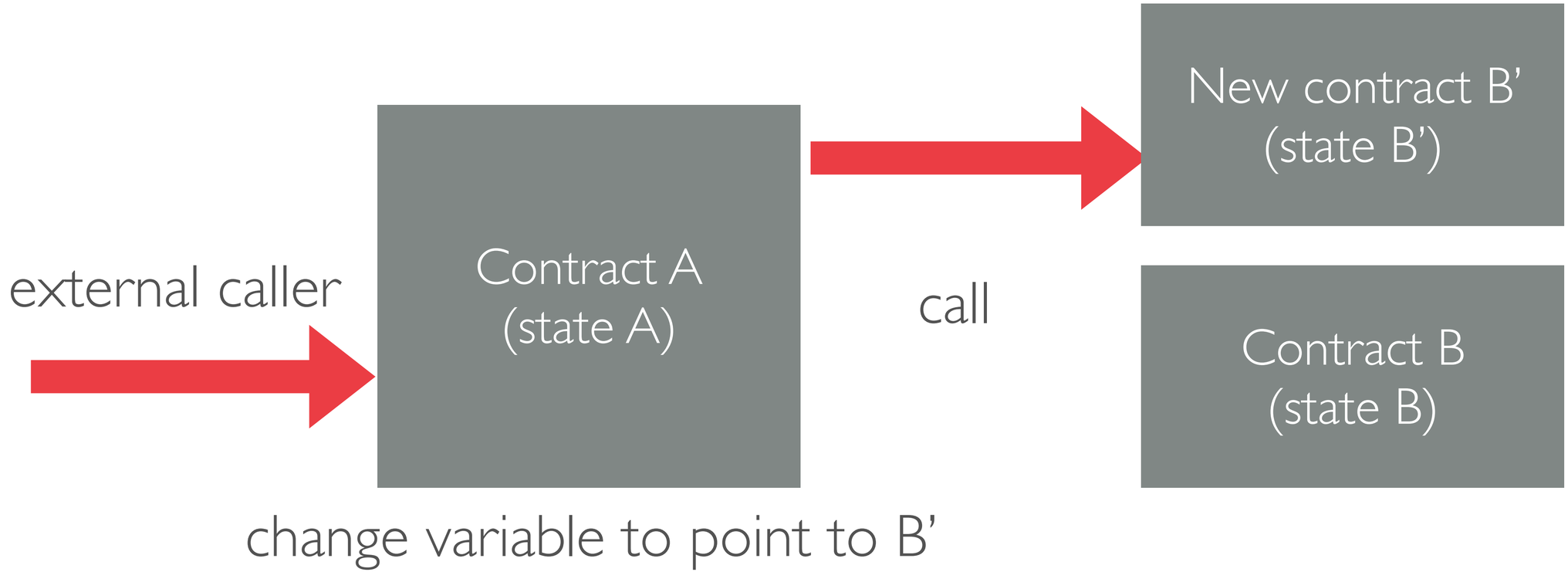
Proxy pattern

- Based on two standard features of Ethereum
  - We can always update that variable (using some special contract call we implement ourselves.)



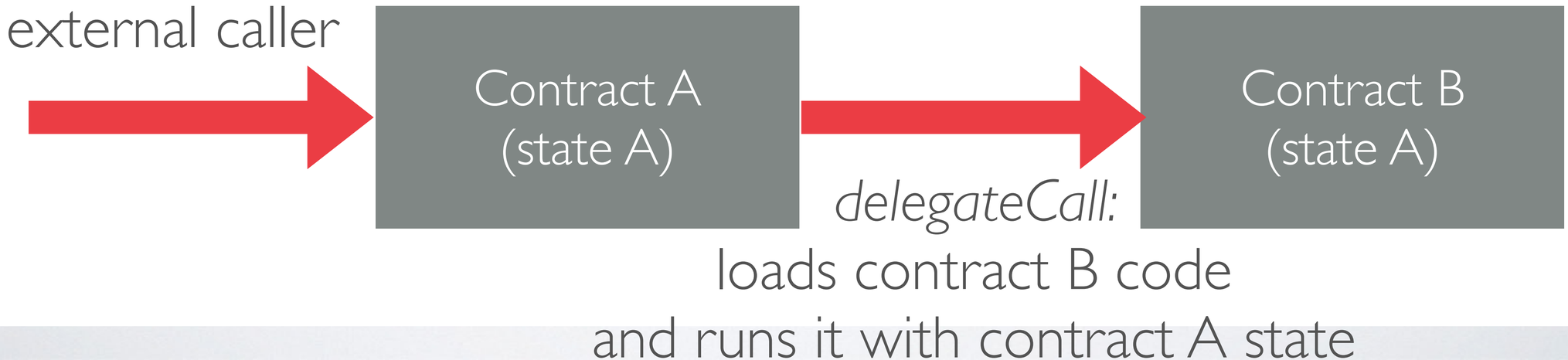
Proxy pattern

- **What are some problems with this solution (so far)?**

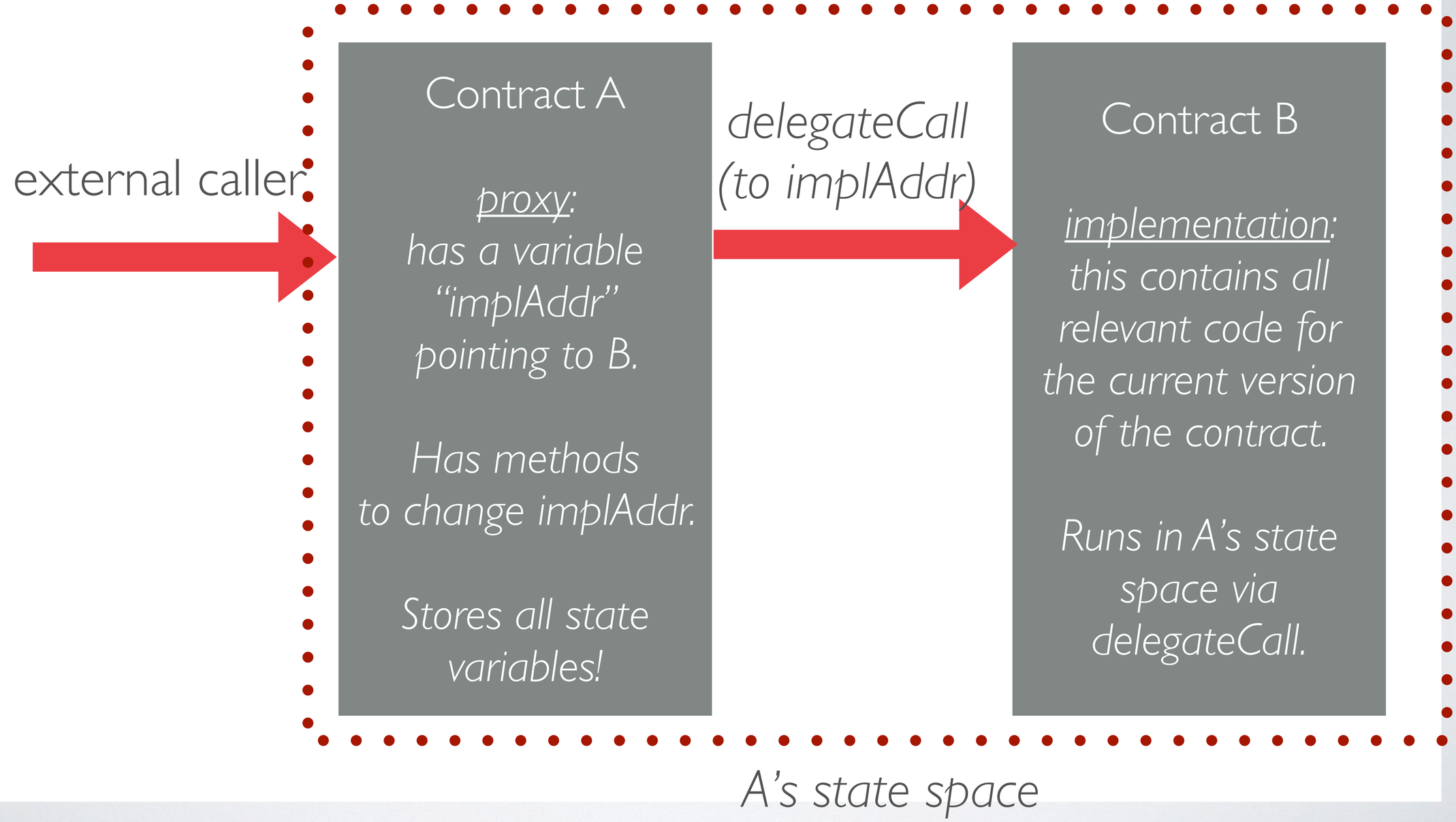


Delegate-Call

- Ethereum offers a nice "optimization"
  - A contract A can "load the code" from contract B into its own state context, run B on A's state
  - This is called delegateCall
    - It allows other contracts to become "code libraries"



Proxy pattern



Proxy upgrade

contract update
"please upgrade to
contract C"

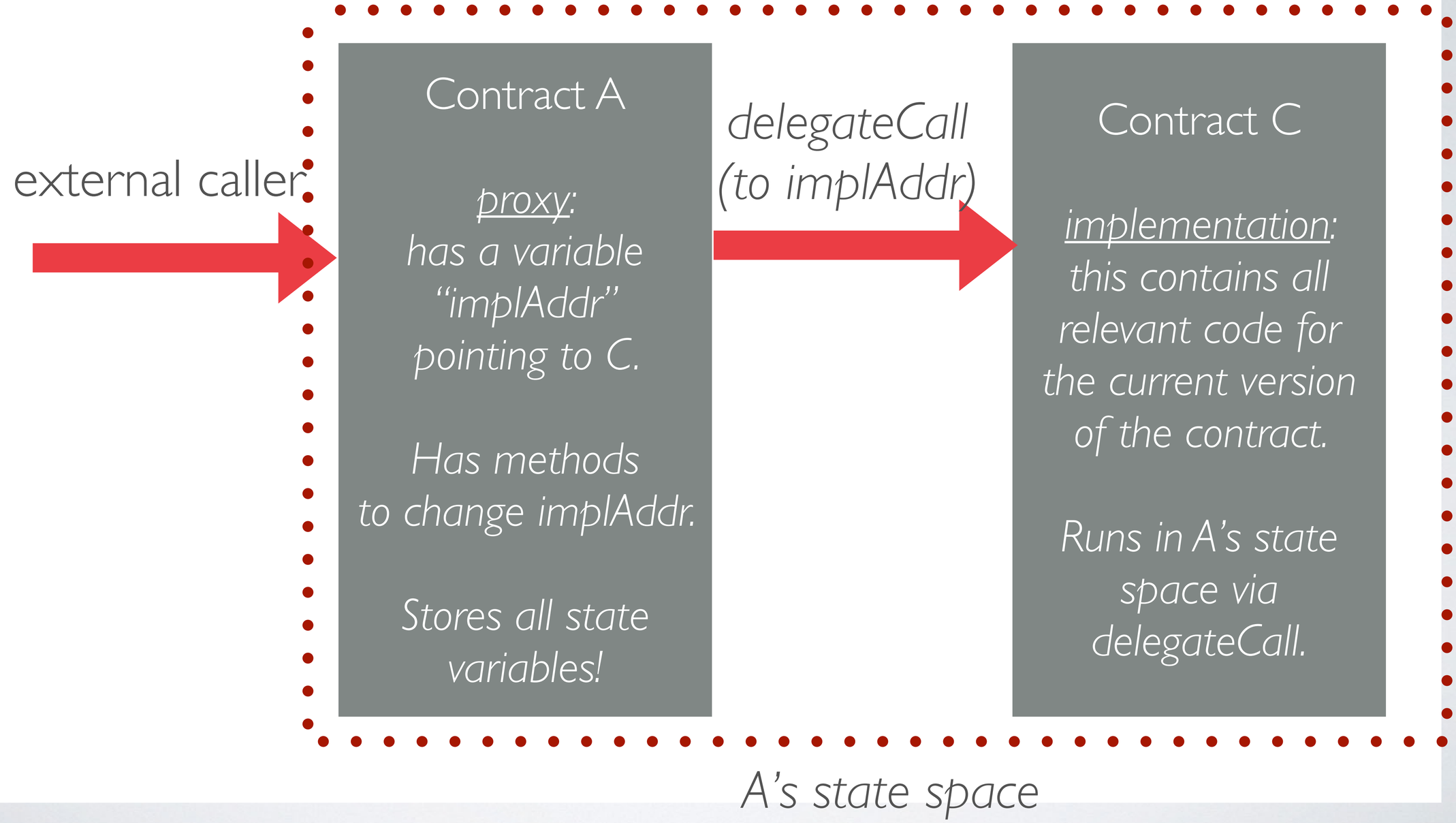
*(e.g., called
by the "administrator"
of this contract)*

Contract A

proxy:
check that upgrade
is authorized,
change "implAddr"
to point to new
Contract C

Contract C

After upgrade



Standards Track: ERC

ERC-2535: Diamonds, Multi-Facet Proxy

Create modular smart contract systems that can be extended after deployment.

Authors Nick Mudge (@mudgen)

Created 2020-02-22

Table of Contents

As state space

Proxy costs

- There is always some overhead to proxy calls
  - Each call through the proxy involves (at least):
    1. A check to see if this is an upgrade call
    2. A variable load (lookup) to find *implAddr*
    3. A delegateCall

    Modern proxy contracts use EVM bytecode to minimize this

Proxy	Gas cost	Gas overhead
OpenZeppelin Transparent	29815	2770
Dharma Beacon	29752	2707
EIP-1882 UUPS	28679	1634
Storageless Beacon	28629	1584

Proxies: advantages/disadvantages?

Proxies: advantages/disadvantages

- The ability to upgrade contract code is useful (bug fixes, etc.)
  - Counterargument: it can be very risky
  - If someone "hacks" your upgrade mechanism, they can steal all your money
  - What are some ways we can secure this upgrade process?

Proxies: advantages/disadvantages

- The ability to upgrade contract code is useful (bug fixes, etc.)
  - Counterargument: it can be very risky
  - If someone "hacks" your upgrade mechanism, they can steal all your money
  - What are some ways we can secure this upgrade process?
    - **Multi-sig** (require many signatures to upgrade contract)
    - **Voting** (require many token-holders to vote for an upgrade)
    - **Timelocks** (put a delay between start & end of an upgrade)

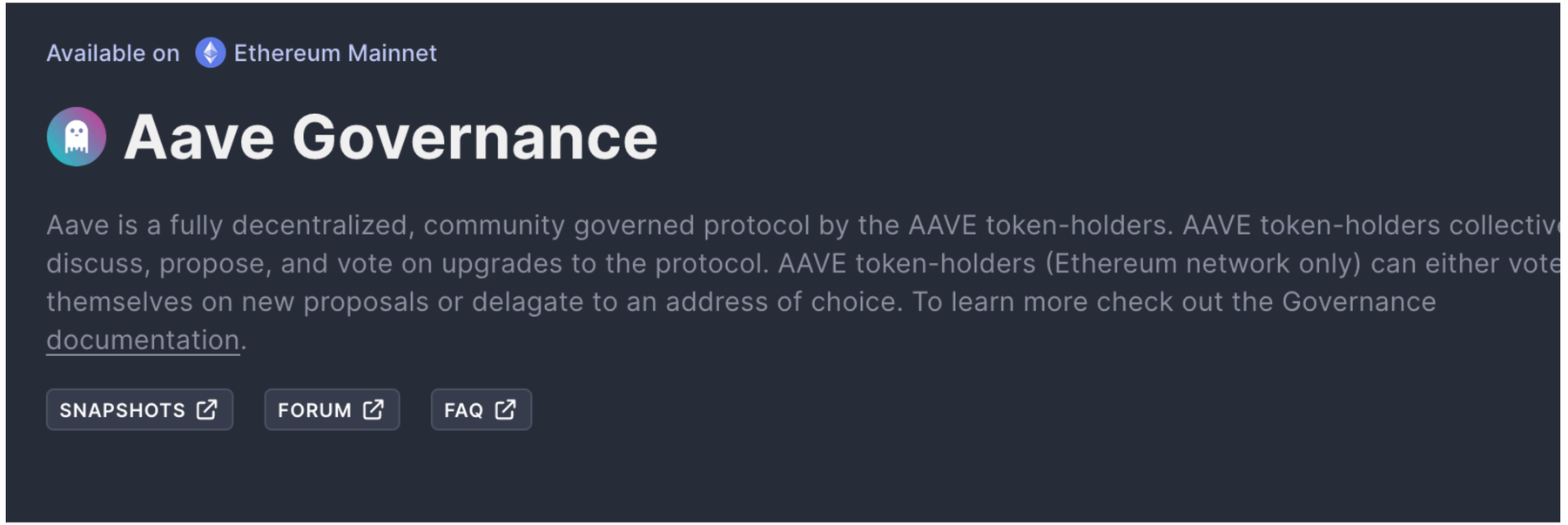
Contract governance

- The ability to upgrade contracts can be even more powerful
  - Enables the notion of "voting"-based contract governance

Contract governance

- The ability to upgrade contracts can be even more powerful
  - Enables the notion of "voting"-based contract governance
- **Idea**: issue a "governance token" (e.g., ERC20)
  - People might get the token in exchange for depositing real money, or for other reasons
  - This token gives holders a "vote" on contract governance
  - People can propose new features for the contract (in practice, this is essentially a contract upgrade proposal)
  - Voters can then review/vote/adopt the new features

Contract governance



Contract governance: dark side

- An observation in practice:
  - Contract governance is sometimes a form of "regulatory arbitrage"
  - Often: the contract developers want to argue that they are simply a software developer, and do not control this "decentralized protocol"
  - However they (or their friends/partners/VCs) may in fact control the bulk of the voting tokens
    - Hence one should always be a little bit skeptical of these claims

Contract governance: dark side II

News ▸ Technology

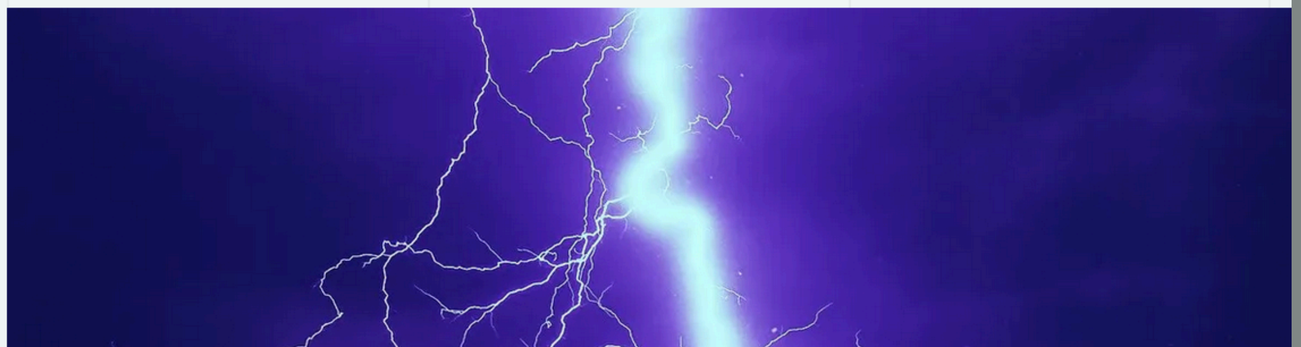
Group Uses Flash Loan to Game Maker Protocol Governance

BProtocol used a DeFi arbitrage tool to push through a governance vote it had proposed on the Maker protocol.

By Jeff Benson

Oct 29, 2020

2 min read



I intend to return to this (if there is time)…

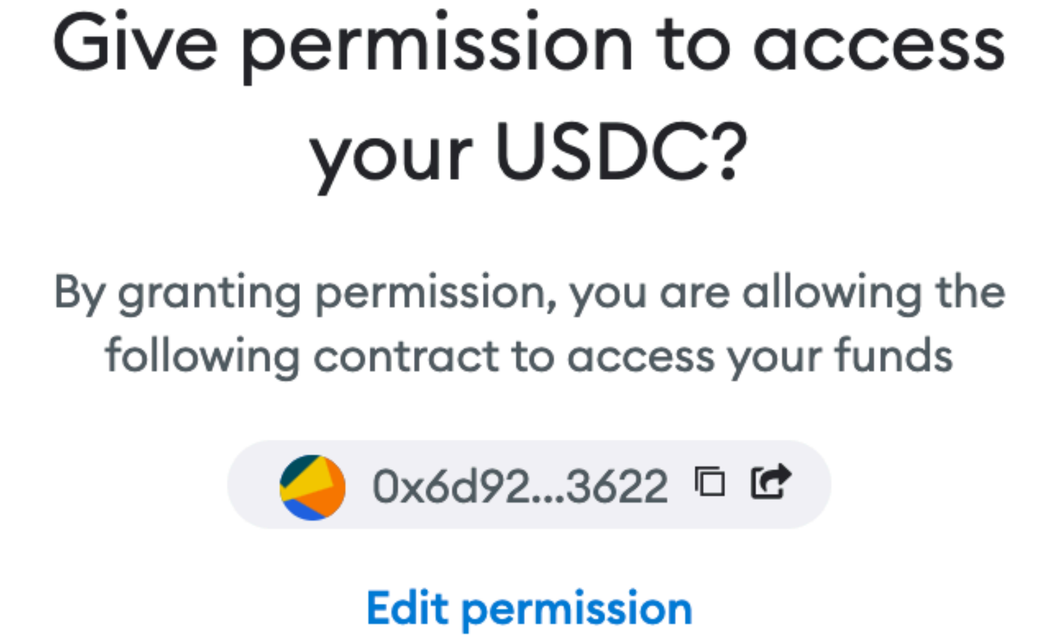
A warning about ERC20s

- Most (Ethereum) DeFi apps use ERC20 tokens
  (or native tokens, like ETH. Sometimes they wrap this into an ERC20 like wETH.)
  - When you send a native (ETH) token to a contract,
    it gets notified that it has received a token
  - When you send an ERC20 token using the transfer() method,
    the contract does not get notified that it has received a token
    - Why not?

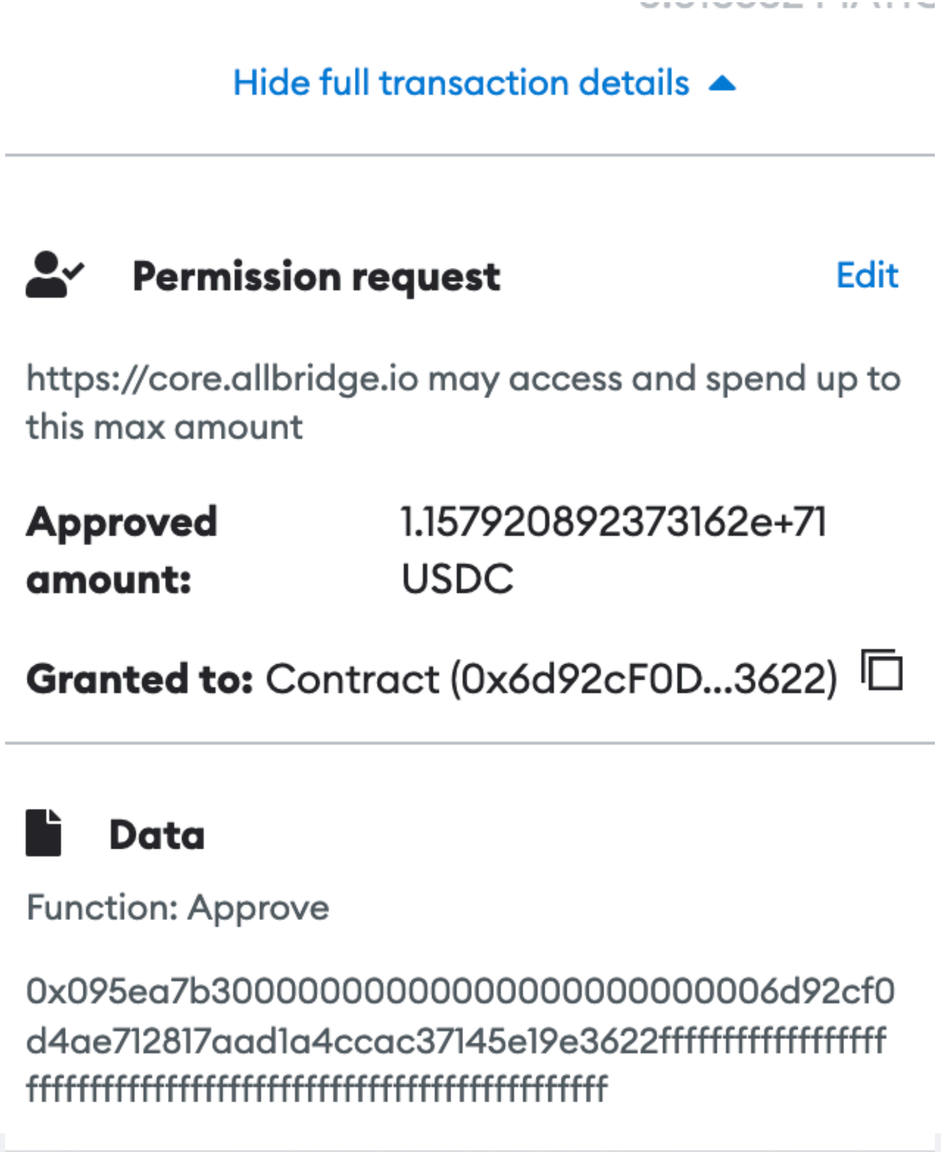
A warning about ERC20s

- A common pattern when sending tokens to contracts
  - First: run the ERC20's approve() method to allow the contract to "withdraw" some money from your ERC20 account
  - Next call the smart contract
    - The smart contract will then call transferFrom() to transfer money from your account
    - Then the contract will complete whatever actions it's supposed to do for you

A warning about ERC20s



A warning about ERC20s



A warning about ERC20s

Demo Main Account Balance 6.00027100 USDC

Spend limit permission

Allow https://core.allbridge.io to withdraw and spend up to the following amount:

◉ Proposed approval limit

Spend limit requested by https://core.allbridge.io

1.1579208923731 62e+71 USDC

○ Custom spend limit

Enter max spend limit

DeFi applications

Legal note: I am going to talk about and show you some things that may be legally uncertain. I am doing this for educational purposes so that you don't have to. Don't put money into these services, you will probably lose it all to theft, scams, or the IRS. Johns Hopkins University does not endorse the use of any cryptocurrency or DeFi protocol.

For any questions, contact Darren Lacey, head of JHU IT.

A warning about ERC20s

- A common pattern when sending tokens to contracts
  - First: run the ERC20's approve() method to allow the contract to "withdraw" some money from your ERC20 account
  - Next call the smart contract
    - The smart contract will then call transferFrom()

Decentralized Exchanges

- Given multiple assets (ERC20s) on Ethereum, can we build platforms to exchange them?
  - Breaks down into two-subproblems:

    1. Price discovery / order matching
    2. Execution (swap)

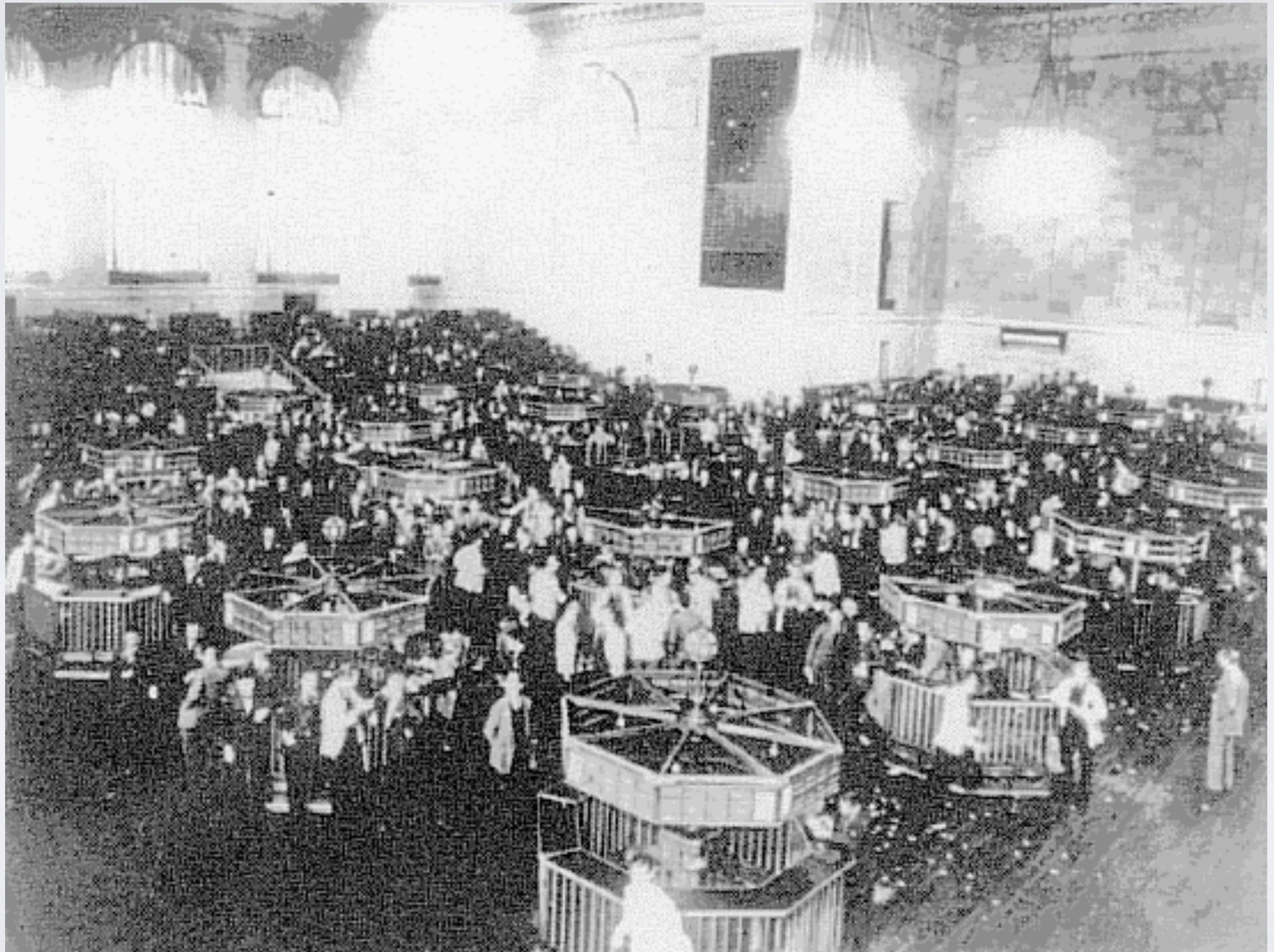
Decentralized Exchanges

- Given multiple assets (ERC20s) on Ethereum, can we build platforms to exchange them?
  - Breaks down into two-subproblems:

    1. Price discovery / order matching
    2. Execution (swap)

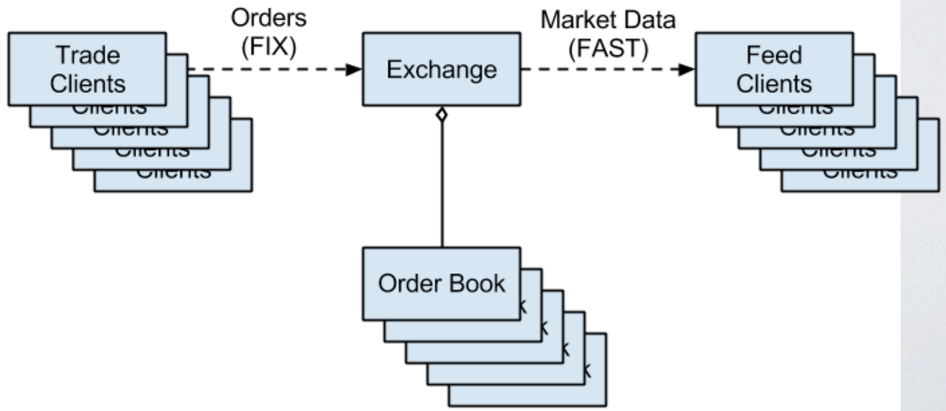
This part is relatively easy

Traditional (centralized) exchange

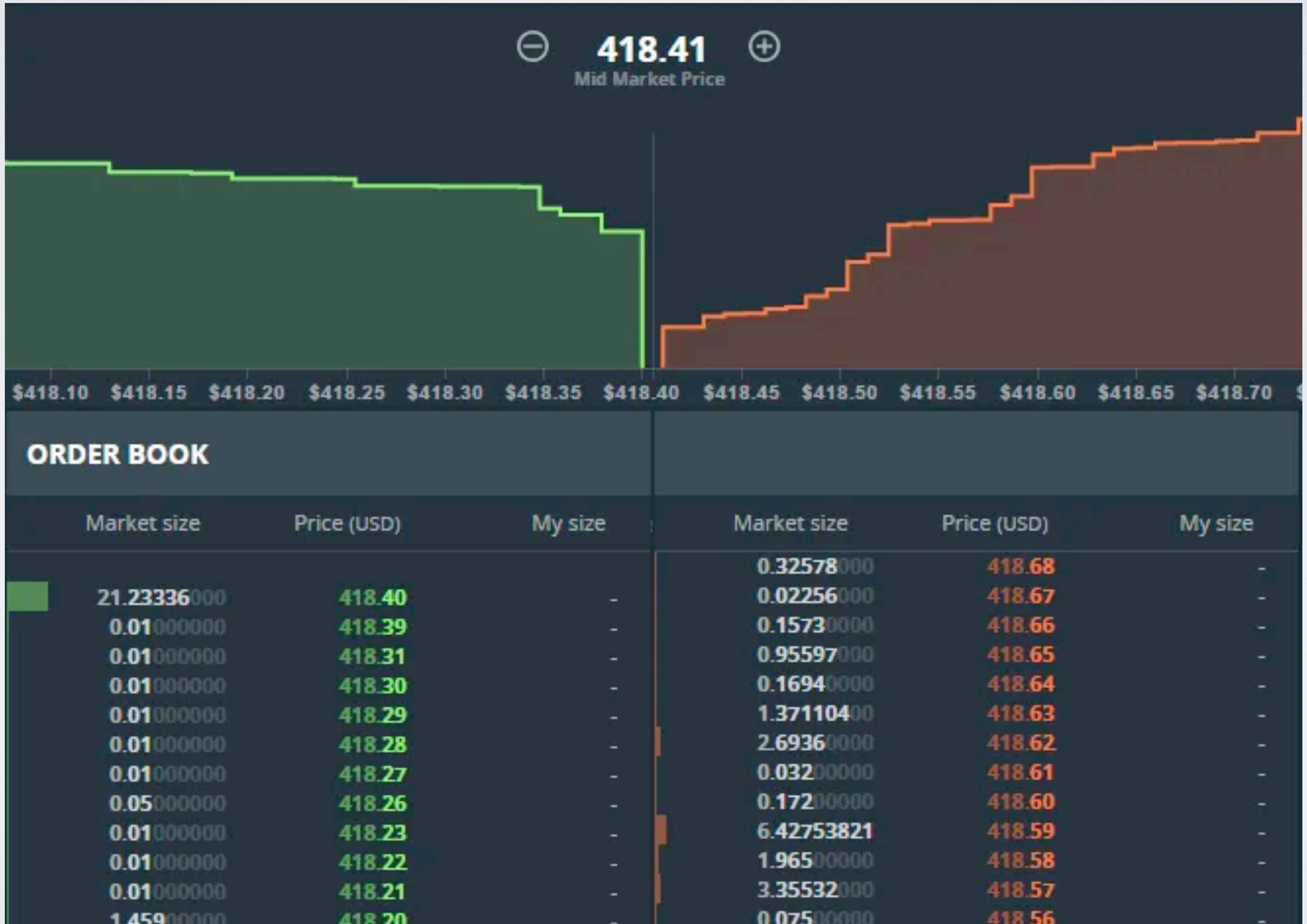


Electronic centralized exchange

- Maintains an "electronic order book"
  - Receives order of various types (e.g., market, limit, etc.)
  - Implements a matching engine to reconcile orders
  - Executes trades (swaps)
  - Produces asset price data (as a side effect)



Electronic order book

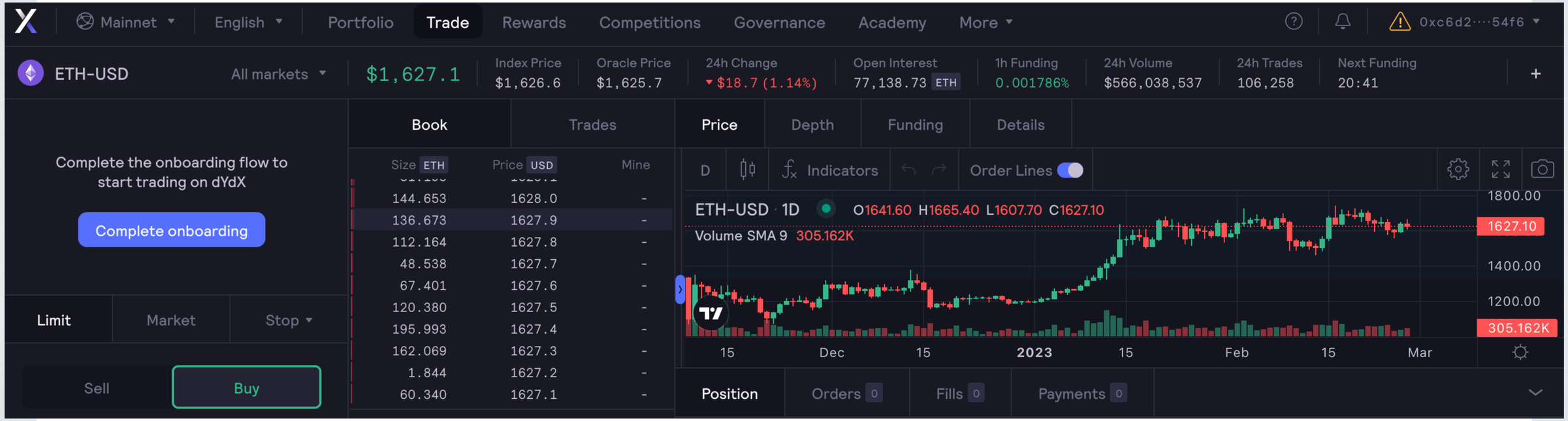


Electronic order book

- Can we do this purely on Ethereum?
  - Why/why not?

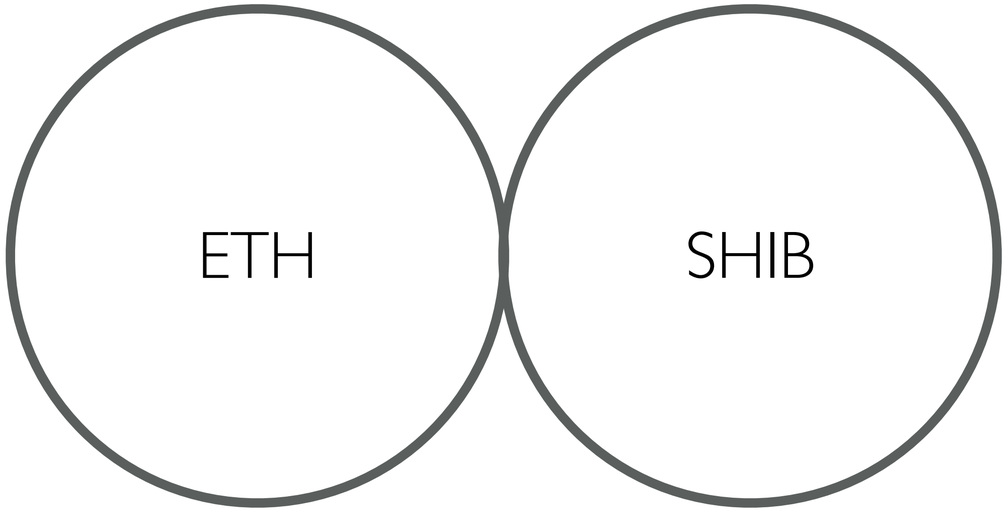
Electronic order book

- Can we do this on Ethereum?
  - Why/why not?



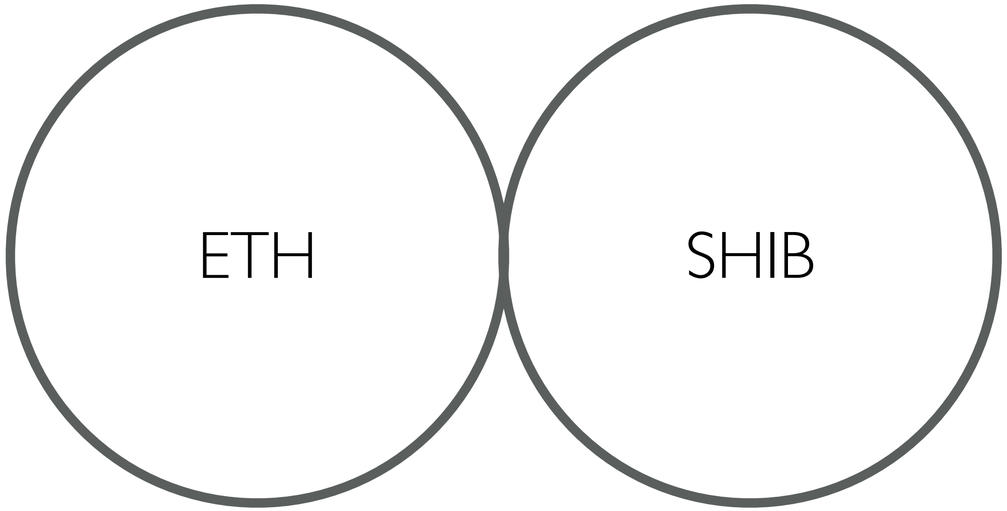
Constant Function Market Makers (AMM)

- A simpler (and much less efficient) type of exchange
  - Built from pairs of assets (e.g., ETH/SHIB)
  - Liquidity providers can deposit and withdraw asset pairs (in some ratio), forming a "liquidity pool"



Constant Function Market Makers (AMM)

- A simpler (and much less efficient) type of exchange
  - Built from pairs of assets (e.g., ETH/SHIB)
  - New LPs can increase/reduce the overall size of both pools, without changing the ratio

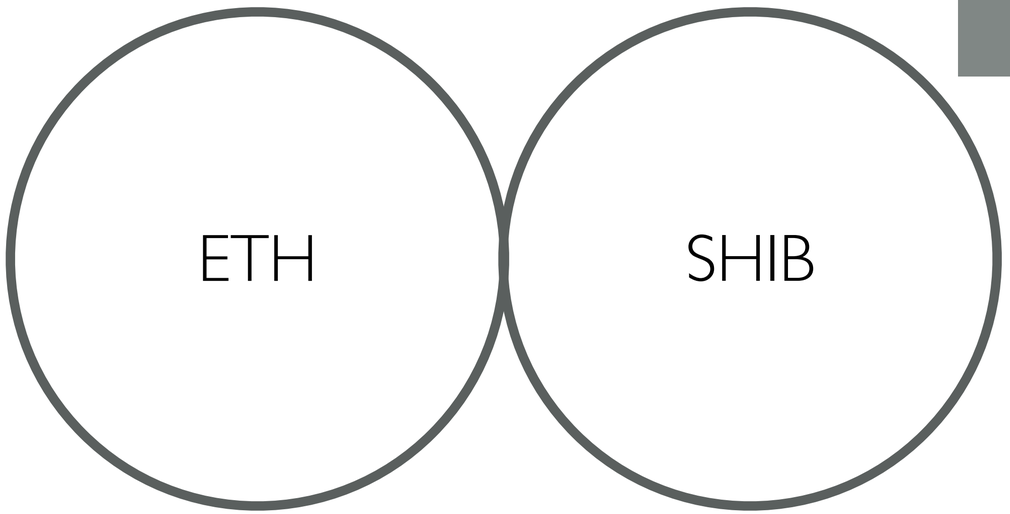


Constant Function Market Makers (AMM)

- A simpler (and much less efficient) type of exchange
  - Built from pairs of assets (e.g., ETH/SHIB)
  - Users can also "swap" (deposit one asset, withdraw the other), in exchange for fees

Overall goal:
Preserve a constant function (e.g., product, sum). E.g.,:

$$A * B = K$$

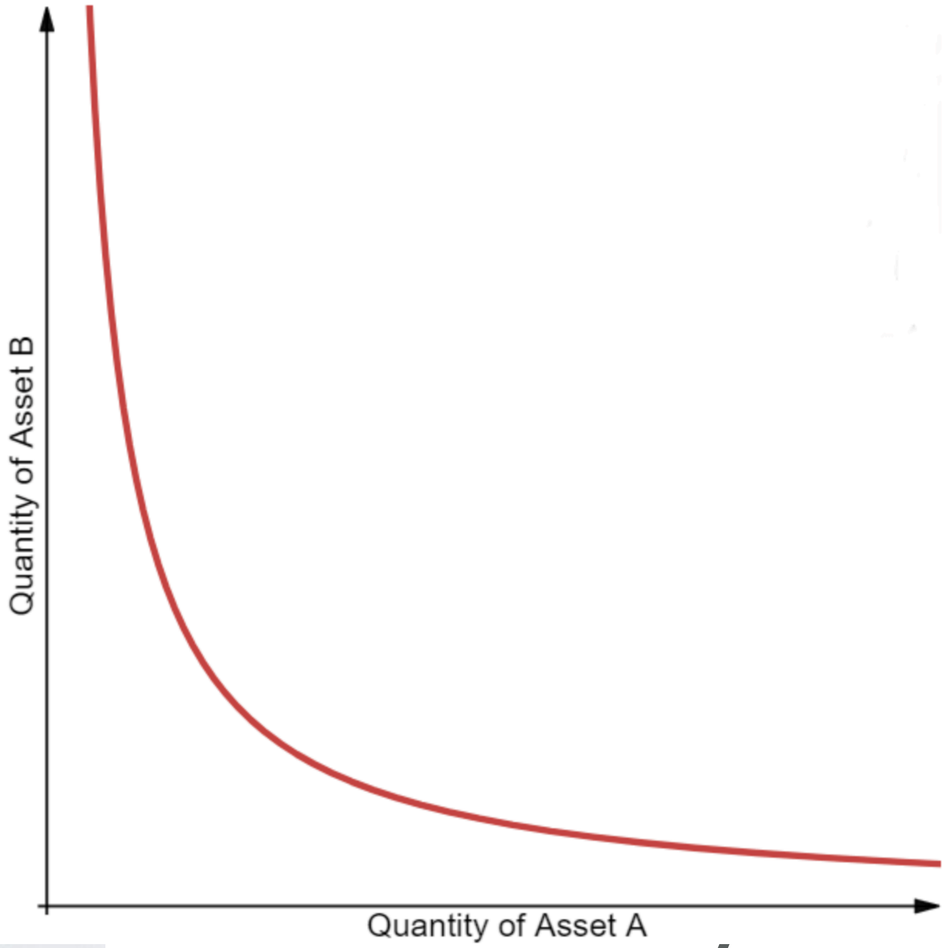


ɔn Market Makers (AMM)

ess efficient) type of exchange

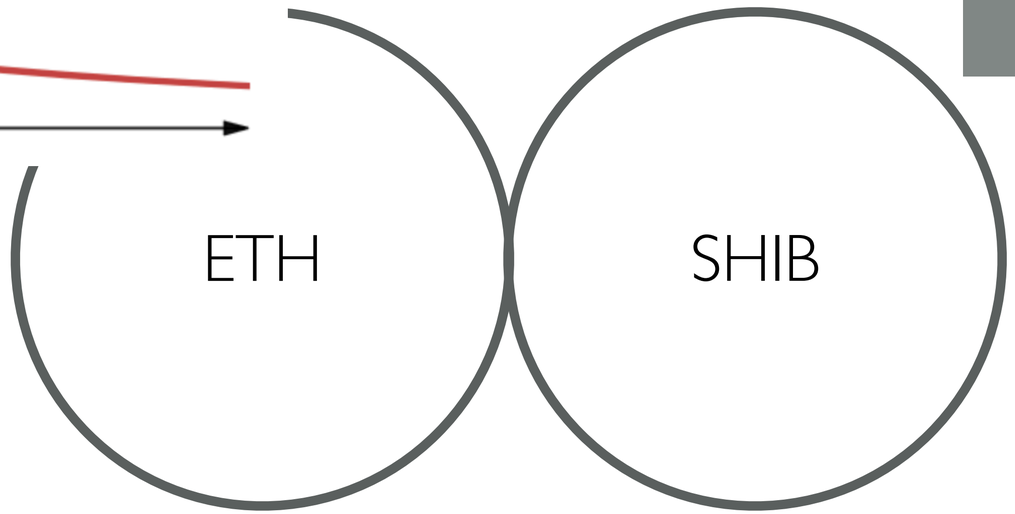
sets (e.g., ETH/SHIB)

d withdraw assets (in vari
ɔol”



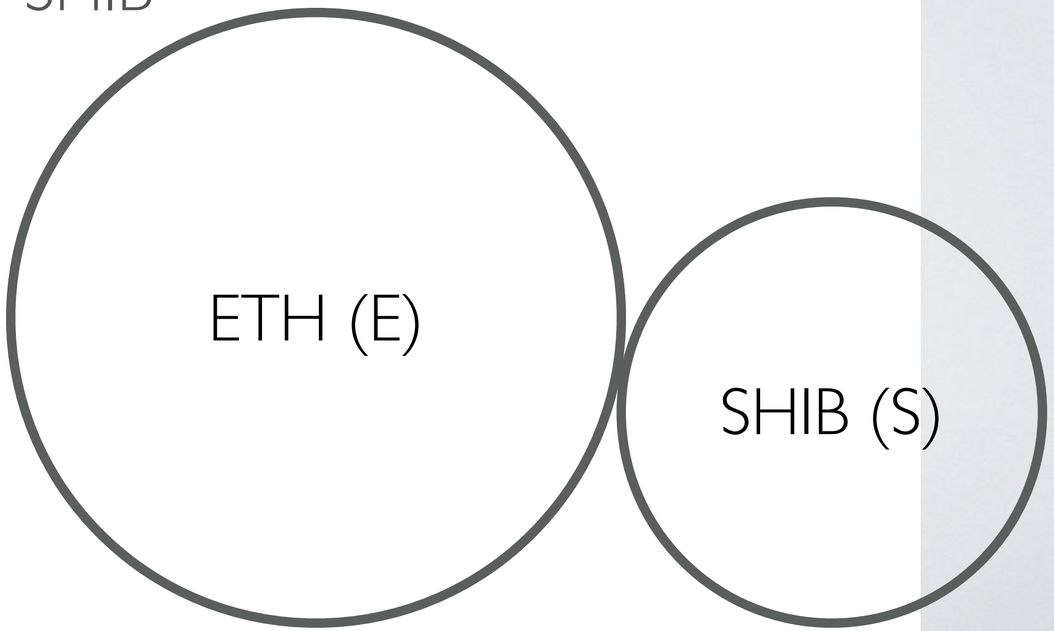
Overall goal:
Preserve a constant function (e.g., product, sum). E.g.,:

$A * B = K$

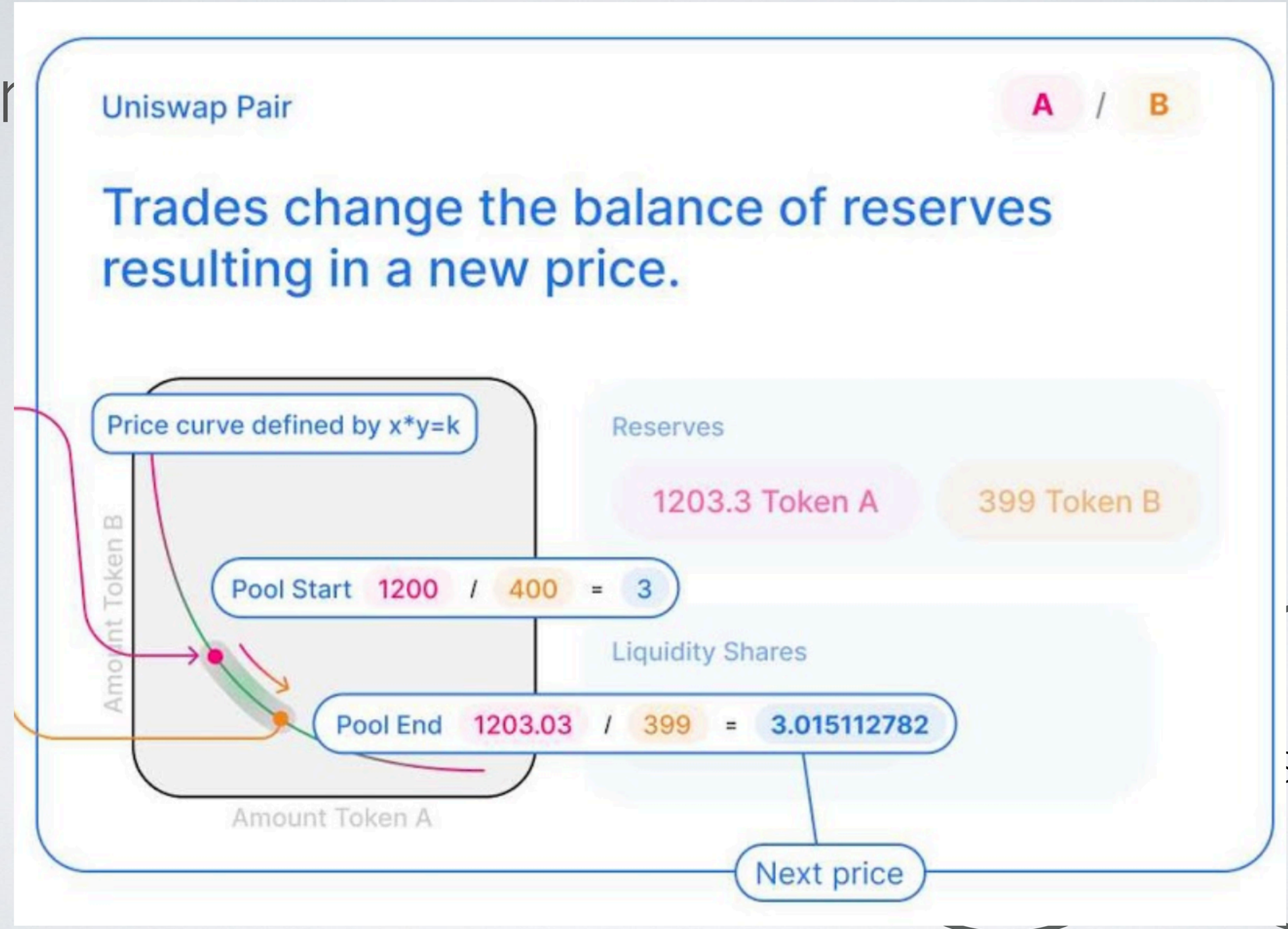


Constant Function Market Makers (AMM)

- User A deposits ETH, wants to “buy" SHIB
  - Initial equation: E * S = K
  - Defines an implicit price (ratio) between the assets
  - They can now deposit ETH & withdraw SHIB at that price (plus pay some fees)
    - This trade changes the market price
    - New price ratio for SHIB: (K / E’) / S’



Cor M)

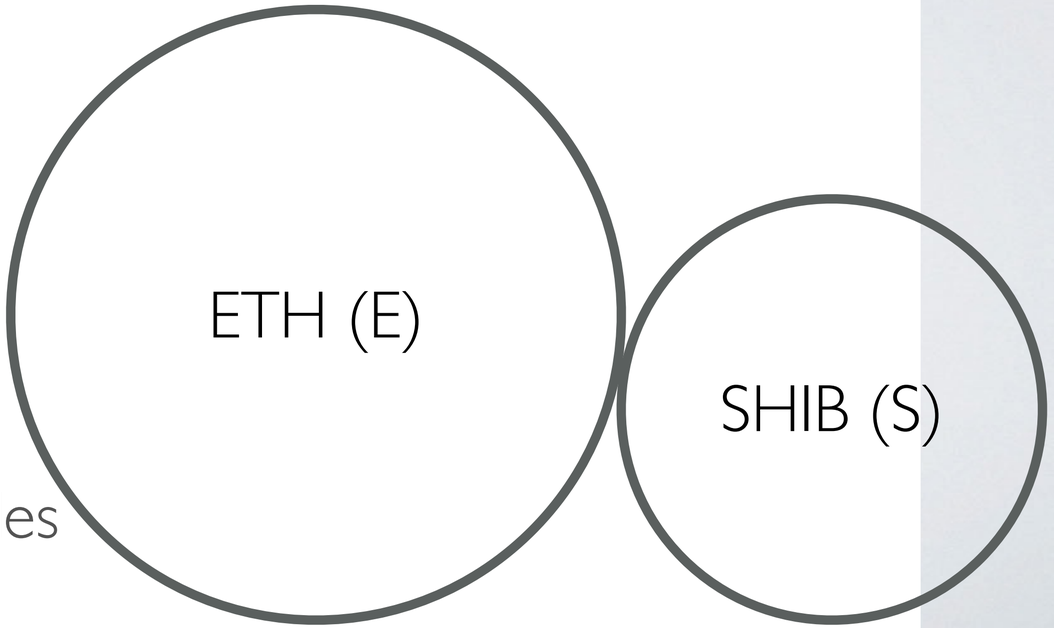


SDC (U)

https://academy.shrimpy.io/lesson/what-is-an-amm-automated-market-maker

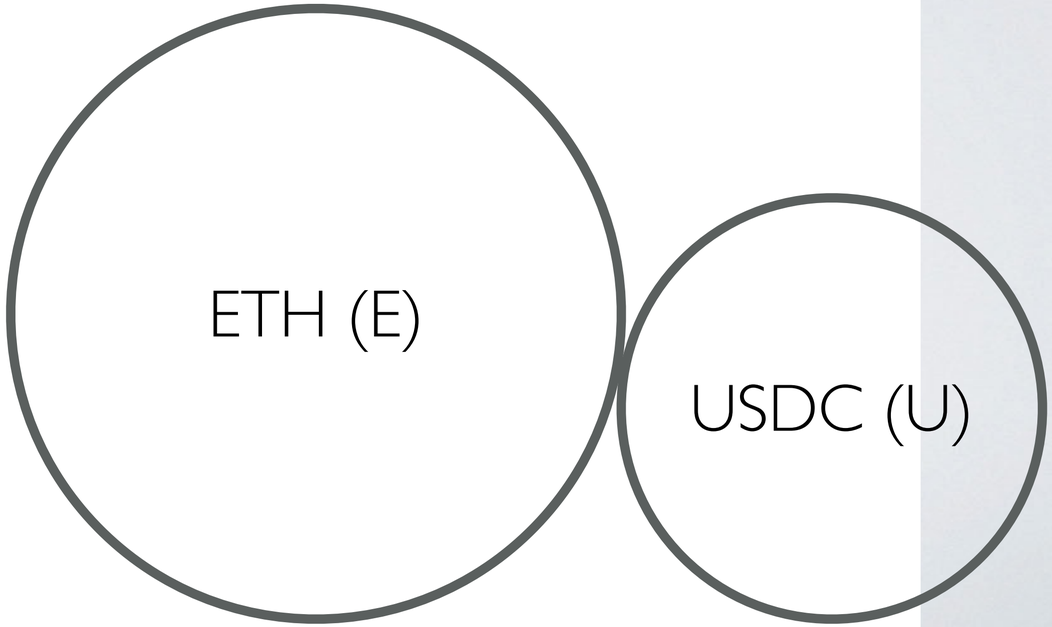
Arbitrage / MEV

- In principle, the price of assets should stay "close" to the true price (whatever that is)
  - If the ratio gets far out of whack, hungry traders will "adjust" the pool sizes by executing profitable swaps
  - If there is no pool for an asset pair, systems can "route" trades through multiple pools with more liquidity
    e.g., ETH->USDC, USDC->SHIB
  - Slippage can be quite large
  - Relatively easy to "front-run" public trades

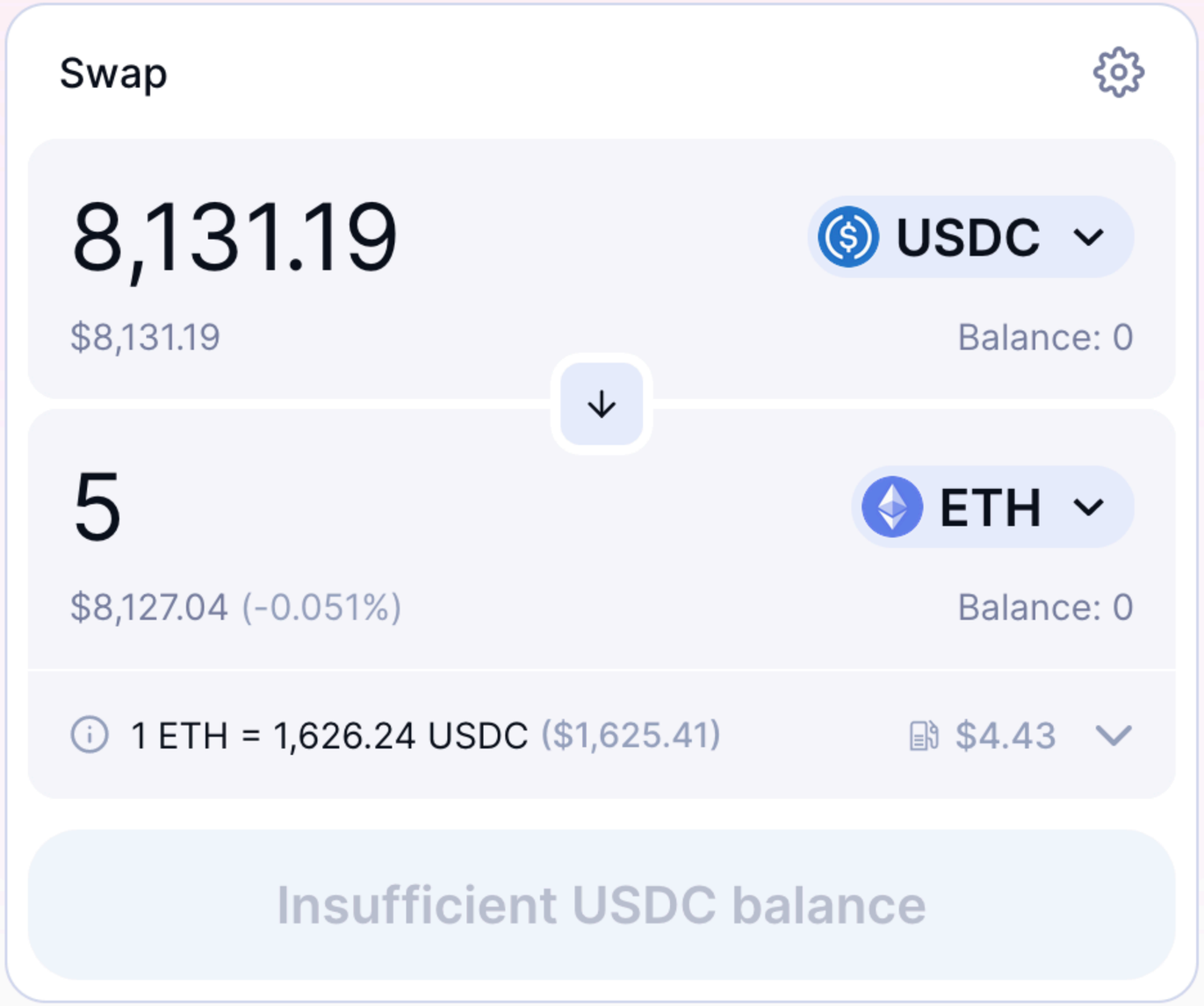


Why put your "liquidity" in these pools?

- The liquidity providers receive fees from the trade (usually split with the DeFi operators, developers)
  - They receive "Liquidity Provider Tokens" (yet another made-up ERC20) that they can later redeem to get money back out (in a new ratio)
  - **Criticism:** being an LP is dumb, because arbitrage is profitable and you are the loser in all those trades
  - Many LPs have been "incentivized" to participate by weird token schemes



W s?

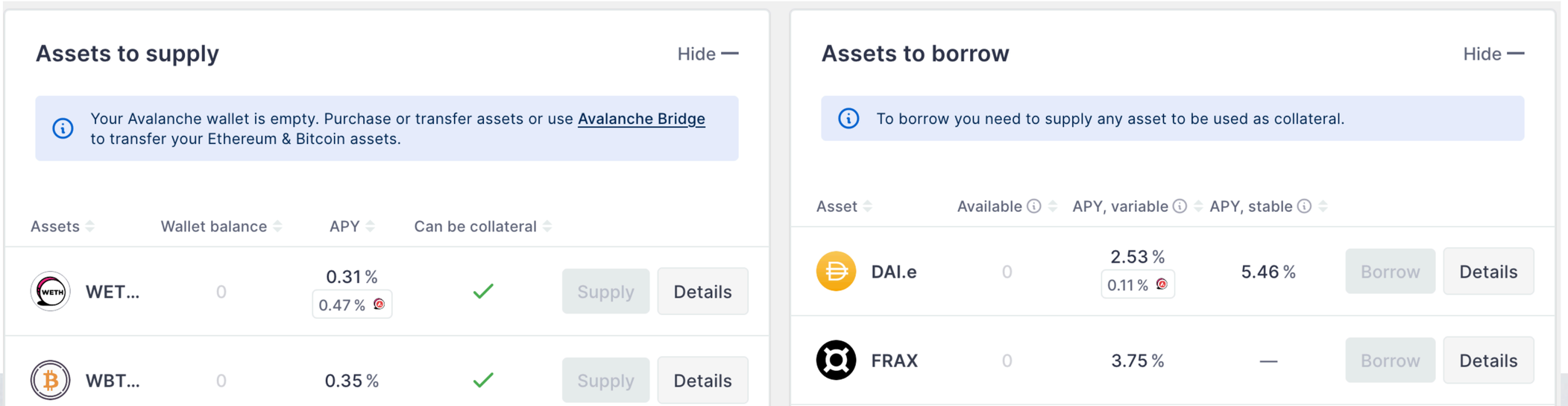

Swap
8,131.19
USDC
$8,131.19
Balance: 0
5
ETH
$8,127.04 (-0.051%)
Balance: 0
1 ETH = 1,626.24 USDC ($1,625.41)
$4.43
Insufficient USDC balance

Asset lending

- Another thing smart contracts can do easily is asset lending
  - **Problem:** smart contracts are not great at getting people to repay asset loans
  - This means you need some way to insure those loans
  - Answer: require (over)collateralization

Asset lending

- Another "popular" DeFi service is collaterized asset lending (borrowing)
  - You wish to borrow some amount of Asset B (for whatever)
  - You can deposit and "lock up" some amount of Asset A as collateral for the loan (and pay interest)



Asset lending

- If lenders don't repay the principal, their collateral is automatically liquidated to repay the lenders
  - Hence collateral value must (substantially) exceed loan value
  - If the value of the collateral drops, the system may automatically liquidate it without warning
  - **Question:** How do these smart contracts know the "value" of anything?

Asset lending

- If lenders don't repay the principal, their collateral is automatically liquidated to repay the lenders
  - Hence collateral value must (substantially) exceed loan value
  - If the value of the collateral drops, the system may automatically liquidate it without warning
  - **Question:** How do these smart contracts know the "value" of anything?
  - **Answer:** services provide price "oracles" to these systems, by sending Txes (to some contract) that contain current prices, e.g., ChainLink.
    (Or they can query AMM contracts!)

Asset lending

Assets to supply

Hide —

Your Avalanche wallet is empty. Purchase or transfer assets or use Avalanche Bridge to transfer your Ethereum & Bitcoin assets.

Assets	Wallet balance	APY	Can be collateral		
WET...	0	0.31 % 0.47 %	✓	Supply	Details
WBT...	0	0.35 %	✓	Supply	Details
WAVAX	0	1.65 % 1.15 %	✓	Supply	Details
AVAX	0	1.65 % 1.15 %	✓	Supply	Details

Assets to borrow

Hide —

To borrow you need to supply any asset to be used as collateral.

Asset	Available	APY, variable	APY, stable		
DAI.e	0	2.53 % 0.11 %	5.46 %	Borrow	Details
FRAX	0	3.75 %	—	Borrow	Details
MAI	0	3.51 %	—	Borrow	Details
USDC	0	2.31 % 0.17 %	5.43 %	Borrow	Details

Asset lending (horror stories)

- There are big risks here, if the collateral asset is badly priced (or thinly traded)
  - Show up with some fake nonsense-coin, that has been "wash traded" into appearing to have value
  - Borrow actual money with it
  - Walk away and never come back
- So these systems are not usually unsupervised… and lenders can lose money

Flash loans!

- It is possible to make loans that must be repaid within the course of a single transaction

Future of Ethereum

- Proof of stake
- Rollups
- Sharding