Blockchain: Smart Contracts

Lecture 9

Stablecoins & Lending Protocols

Recap: Solidity

Everything is a contract:

- Contracts manage state variables
- Contracts have functions that can be called externally
- Can inherit code from other contracts (`contract A is B,C`)
- Types of contracts: contract, interface, abstract, library

Global objects: `block`, `msg`, `tx`

An example: ERC20 tokens

- https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

- A standard API for fungible tokens.   (ERC-721 for non-fungible tokens)

- An ERC20 token is itself a smart contract that maintains all user balances:

  mapping(address => uint256) internal _**balances**;

- A standard interface allows other contracts to interact with every ERC20 token.   No need for special logic for each token.

ERC20 token interface

function **transfer**(address _to,   uint256 _value) external returns (bool);

function **transferFrom**(address _from,   address _to,   uint256 _value) external returns (bool);

function **approve**(address _spender, uint256 _value) external returns (bool);

function **totalSupply**() external view returns (uint256);

function **balanceOf**(address _owner) external view returns (uint256);

function **allowance**(address _owner, address _spender) external view returns (uint256);

An example …

Consider two ERC-20 tokens: say USDC and WETH

- USDC is a contract that maintains a **_balances[]** mapping
- WETH is a different contract that also maintains **_balances[]**

Say Bob owns 5 USDC and 2 WETH. This is recorded as:

In USDC contract: **_balances[Bob's address] == 5**

In WETH contract: **_balances[Bob's address] == 2**

Wallet software shows all the coins associated with Bob's address

Anyone can read ERC20 _balances[]

Transaction Hash: 0x6b85ca95e484d94503d1276456bfc32cc55f6fdb8bb231ff83....

Tells the USDC contract to transfer 10,010.00 USDC from Circle's account to 0x7656159E42209A95b77aD374d...

Storage Address: 0x4d3e7741e6c98c0c469419fcfe58fa7ec622d7b26345802d22d17415768760f8

Before: Hex → 0x0000000000000000000000000000000000000000000000000000000000000000

After: Hex → 0x00000000000000000000000000000000000000000000000000000002540be400

recipient's entry

Storage Address: 0x57d18af793d7300c4ba46d192ec7aa095070dde6c52c687c6d0d92fb8532b305

Before: Hex → 0x000000000000000000000000000000000000000000000000000266988cda8061

After: Hex → 0x0000000000000000000000000000000000000000000000000002669638ce9c61

Circle's entry

(Circle's balance after)

(etherscan.io)

Calling other contracts

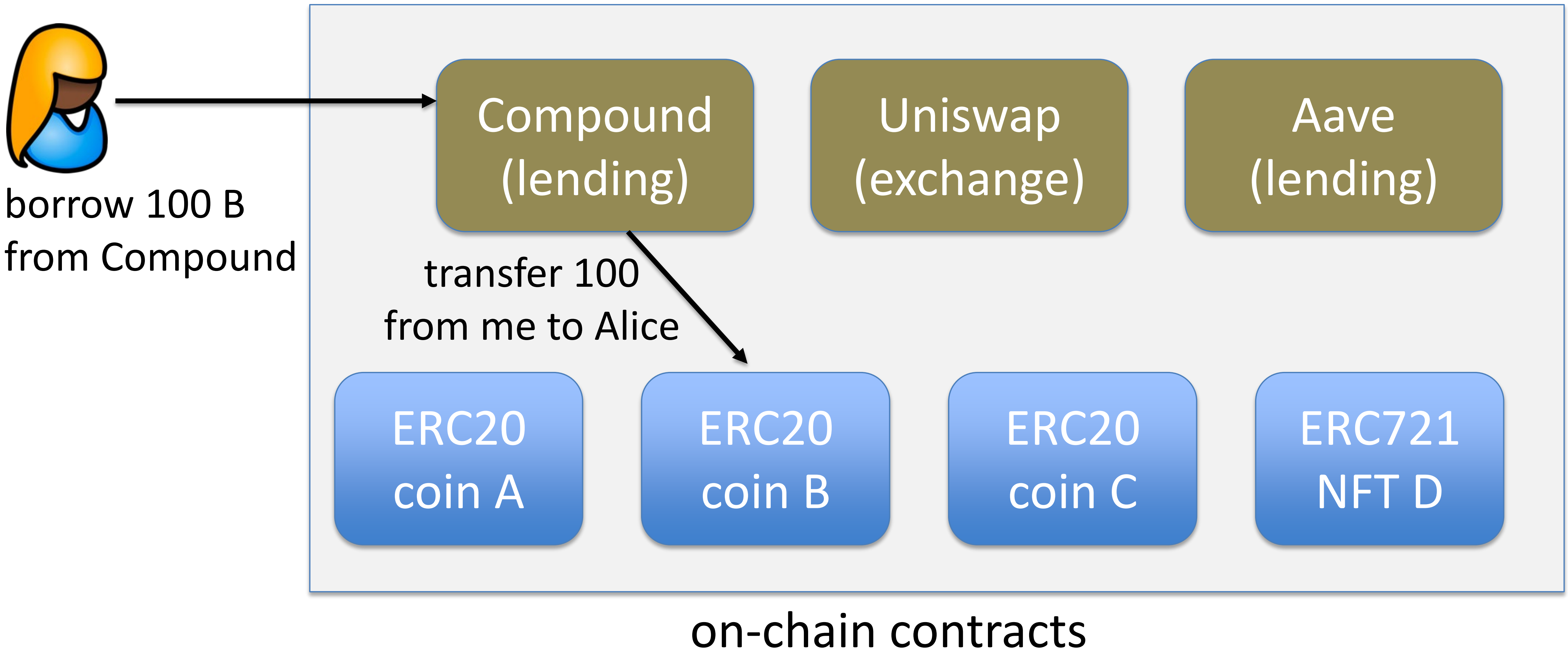
Addresses can be cast to contract types.

```
address _usdc = 0x7656159E42209A95b77aD374d…;
ERC20Token usdcContract = ERC20Token(_usdc);
```

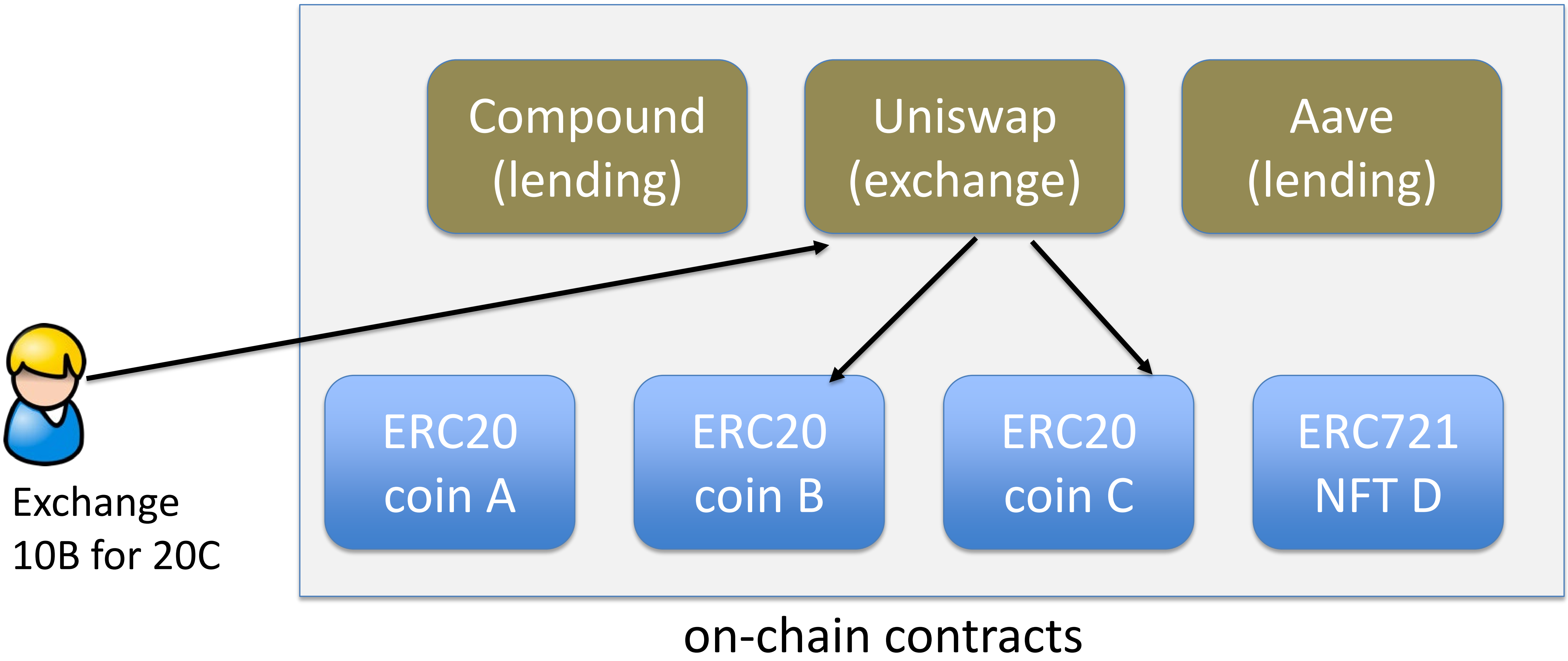
To call the “transfer” function of contract at address _usdc:

```
usdcContract.transfer(_to,  _value);
```

The world of DeFi



The world of DeFi



DeFi app #1: Stablecoins

Stable Coins

A cryptocurrency designed to trade at a fixed price

- Examples: **1 coin = 1 USD**, 1 coin = 1 EUR, 1 coin = 1 USDX

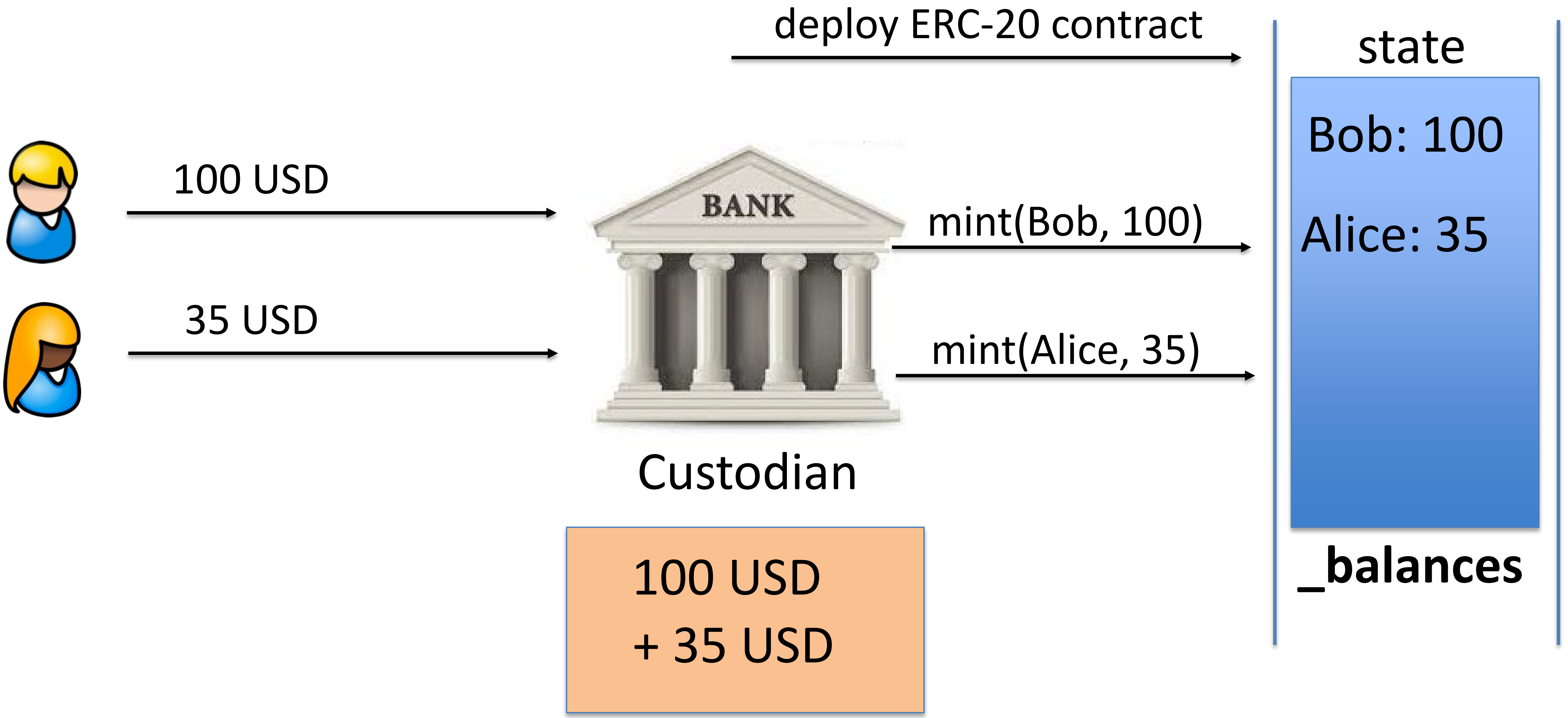
Goals:

- Integrate real-world currencies into on-chain applications
- Enable people without easy access to USD,
  to hold and trade a USD-equivalent asset

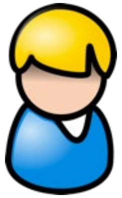
Types of stable coins

	centralized	algorithmic
collateralized	custodial stablecoins (USD Coin)	synthetics (DAI, RAI)
Un(der)collateralized	central bank (digital) currency	Undercollateralized stablecoins

Custodial stablecoins: minting

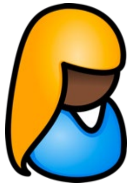


Custodial stablecoins: transfers



pay Carol 15$: transfer(Bob → Carol, 15)

(and gas fee)



Transfers are done on-chain
(custodian is not involved)

135 USD

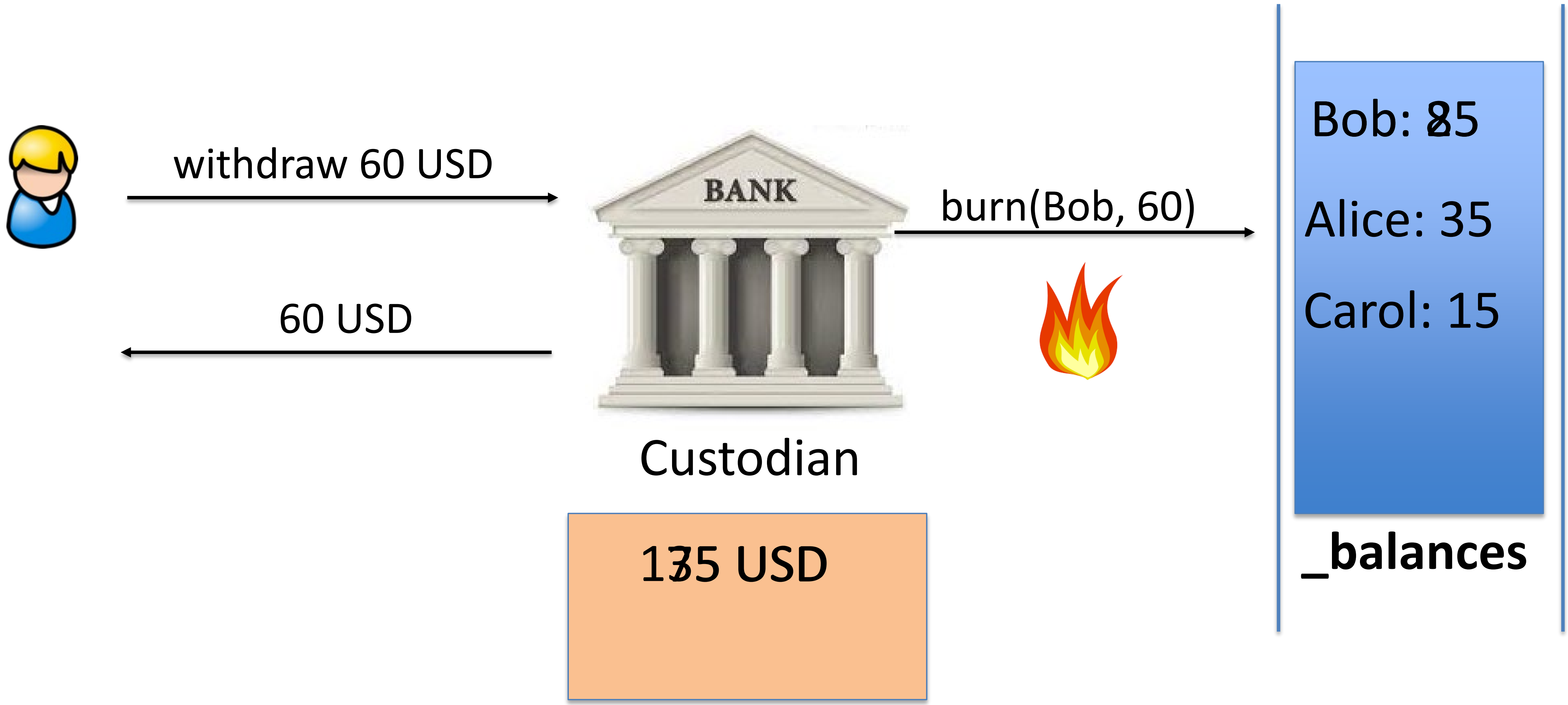
Bob: 85

Alice: 35

Carol: 15

_balances

Custodial stablecoins: withdrawal



Two Examples

	Coins issued	24h volume
USDC	25.3 B	4.6 B
USDT	83.7 B	20.8 B

Some issues

Custodian keeps treasury in a traditional bank

- Must be audited to ensure treasury is available
- Earns interest on deposits

Custodian has strong powers:

- Can freeze accounts / refuse withdrawal requests
- Custodian can remove funds from user balances

Collateralized Decentralized Stablecoins

Goal: a stablecoin with no trusted parties

Examples: DAI, RAI, and others.

Not as widely used as centralized stablecoins

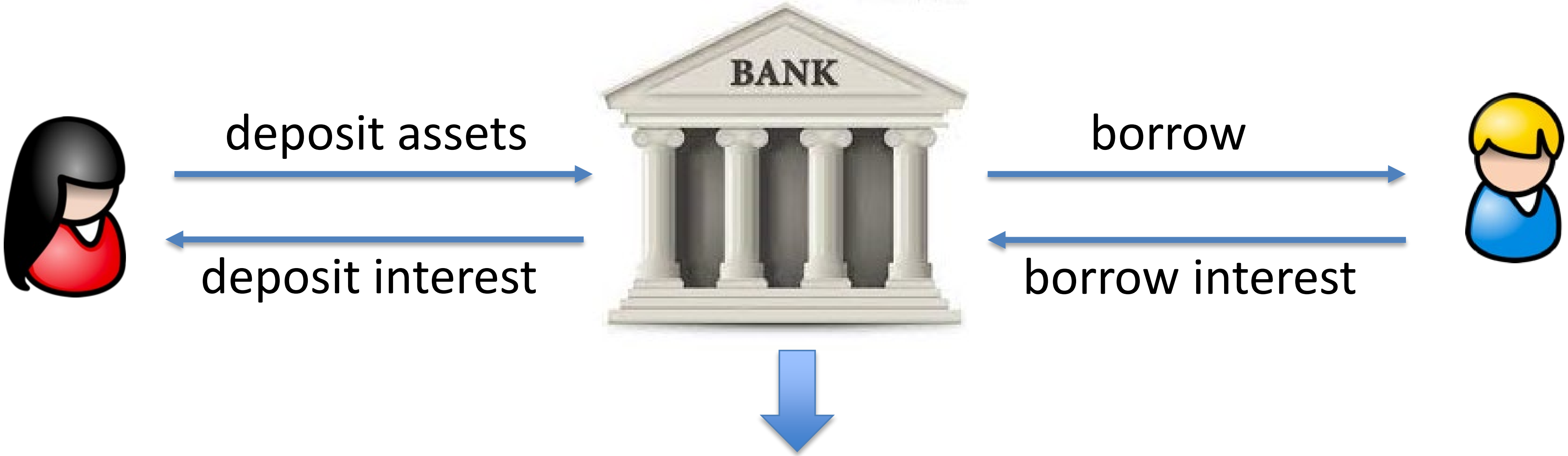
DeFi app #2: Lending Protocols

Goal: explain how decentralized lending works

This is not investment or financial advice

The role of banks in the economy

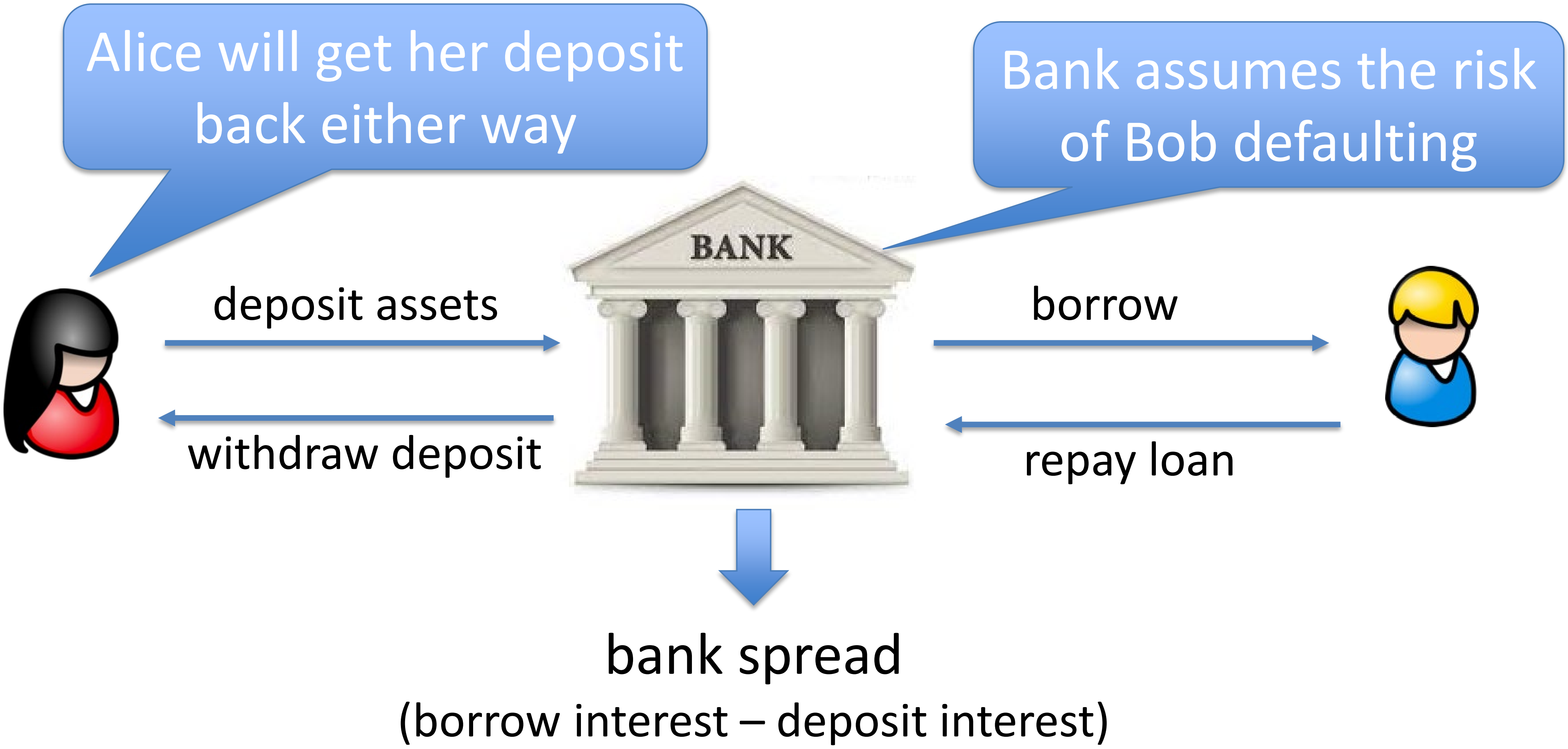
Banks bring together lenders and borrowers



bank spread

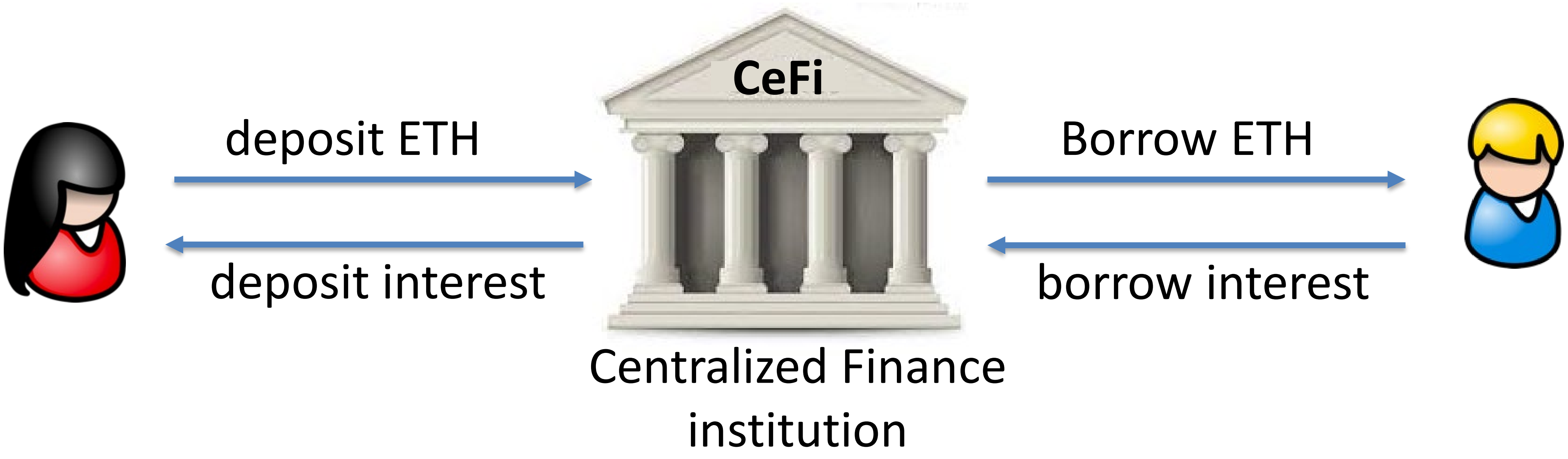
(borrow interest – deposit interest)

The role of banks in the economy



Crypto: CeFi lending (e.g., Blockfi, Nexo, …)

Same as with a traditional bank:



Alice gives her assets to the CeFi institution to lend out to Bob

The role of collateral

(1 ETH = 100 UNI)

CeFi's concern: what if Bob defaults on loan?

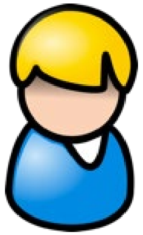
$\Longrightarrow$ CeFi will absorb the loss

Solution: require Bob to lock up collateral

collateral

deposit 500 UNI

Borrow 1 ETH



debt position:

+ 500 UNI
− 1 ETH

interest deducted from collateral

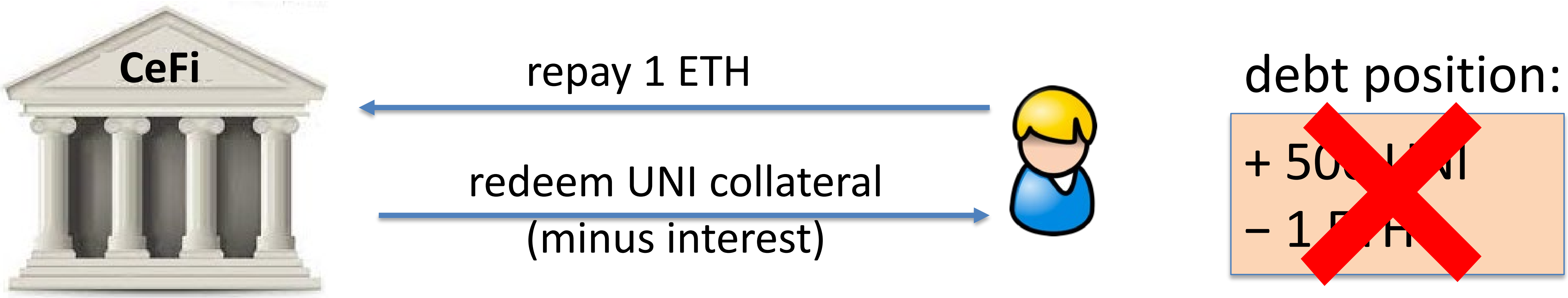
over collateralized loan

The role of collateral

(1 ETH = 100 UNI)

Several things can happen next:

(1) Bob repays loan



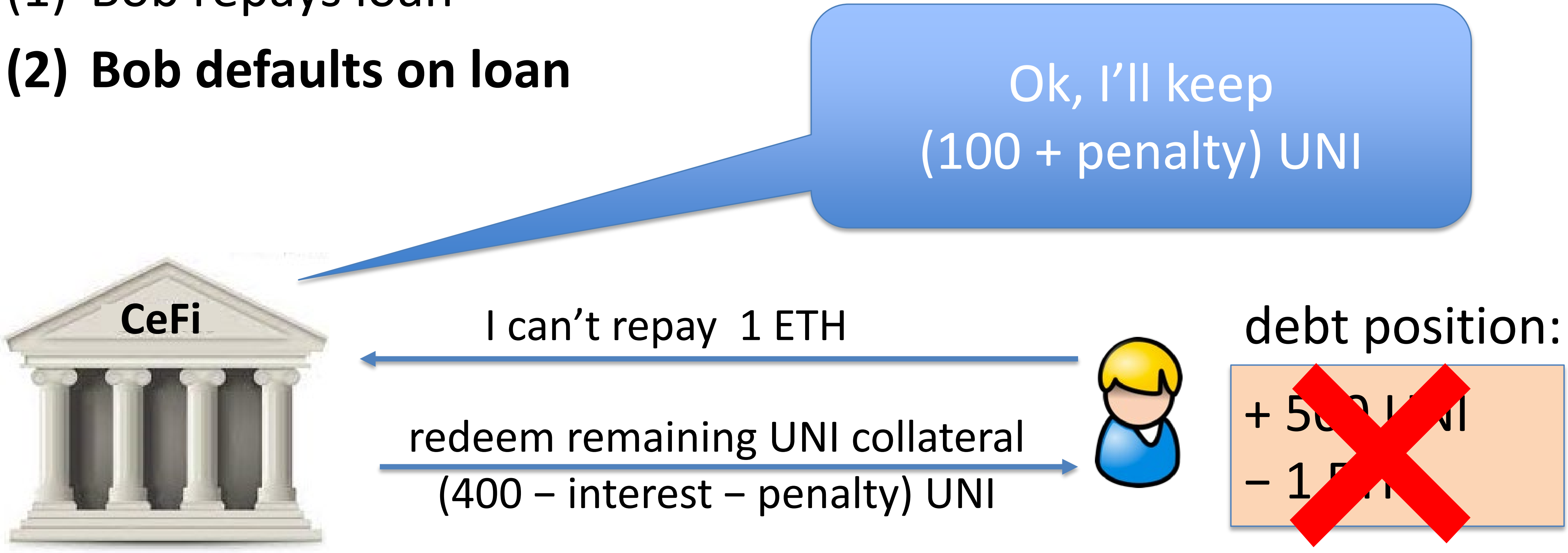
The role of collateral

(1 ETH = 100 UNI)

Several things can happen next:

(1) Bob repays loan

(2) Bob defaults on loan



The role of collateral

(1 ETH = 400 UNI)

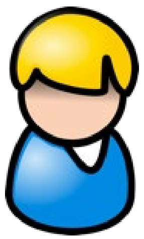
Several things can happen next:

(1) Bob repays loan

(2) Bob defaults on loan

(3) Liquidation: value of loan increases relative to collateral

I need to liquidate
your collateral
(and charge a penalty = 20 UNI)



debt position:

+ 80 UNI
− 0 ETH

lender needs to liquidate **before** value(debt) > value(collateral)

Terminology

Collateral: assets that serve as a security deposit

Over-collateralization: borrower has to provide

value(collateral) > value(loan)

Under-collateralization: *value(collateral) < value(loan)*

collateral factor

Liquidation:

if value(debt) > 0.6 × value(collateral)

then collateral is liquidated until inequality flips

(liquidation reduces both sides of the inequality)

Collateral factor

CollateralFactor $\in$ [0,1]

- Max value that can be borrowed using this collateral

- High volatility asset $\implies$ low collateral factor
- Relatively stable asset $\implies$ higher collateral factor

<u>Examples:</u> (on Compound)

ETH, DAI: 83%, UNI: 75%, MKR: 73%

Health of a debt position

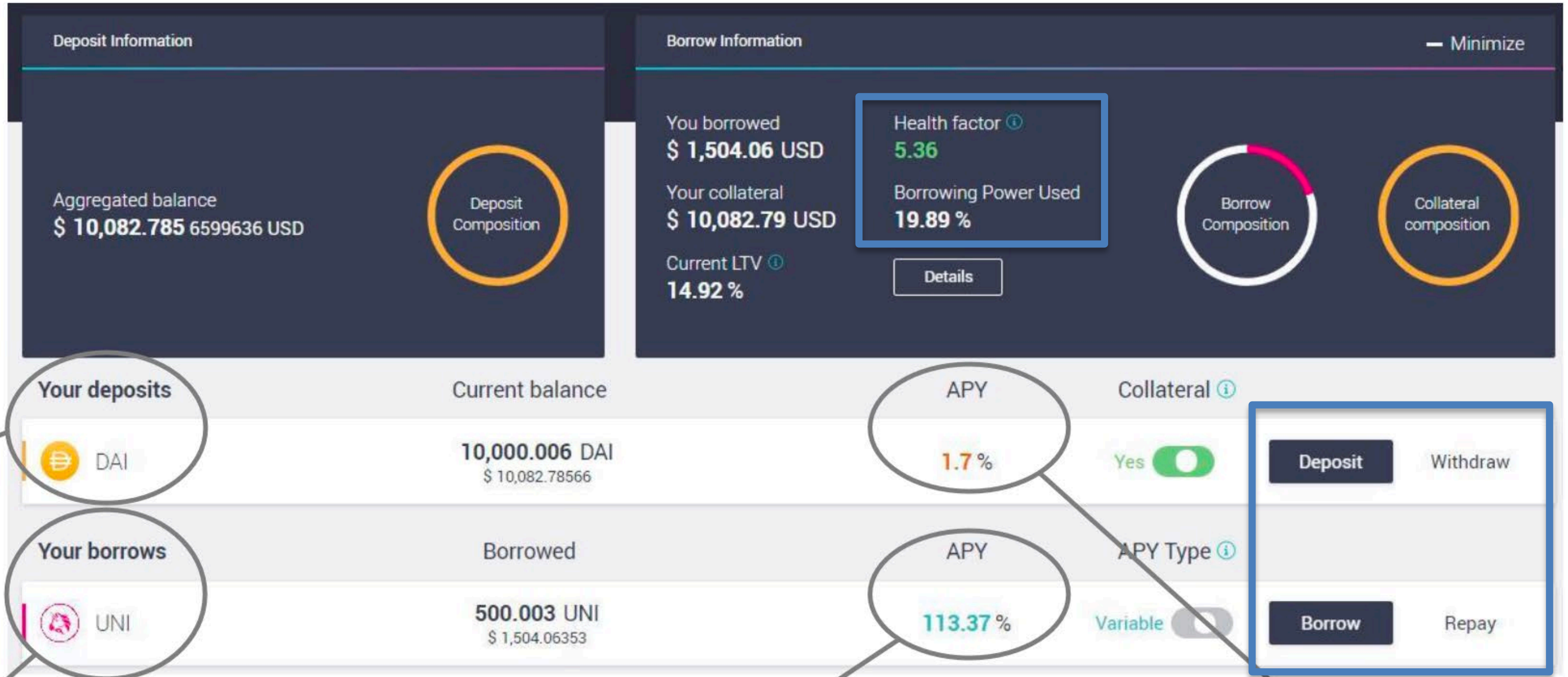
$$\text{BorrowCapacity} = \sum_i \text{value}(\text{collateral}_i) \times \text{CollateralFactor}_i$$

(in ETH)

$$\boxed{\text{health} = \frac{\text{BorrowCapacity}}{\text{value}(\text{TotalDebt})}}$$

helath < 1 $\implies$ triggers liquidation until (health ≥ 1)

Example: Aave dashboard (a DeFi lending Dapp)



DAI is deposited as collateral

actions

UNI is borrowed

The borrowing interests the borrower needs to pay

In Aave, the collteral is also lent out. Hence the borrower can also earn interests.

Why borrow ETH?

If Bob has collateral, why can't he just buy ETH?

- Bob may need ETH (e.g., to buy in-game assets),
  but he might not want to sell his collateral (e.g., an NFT)

- As an investment strategy: using UNI to borrow ETH gives Bob exposure to both

The problem with CeFi lending

Users must trust the CeFi institution:

- Not to get hacked, steal assets, or miscalculate
- This is why traditional finance is regulated

- Interest payments go to the exchange, not liquidity provider Alice

- CeFi fully controls spread (borrow interest – deposit interest)

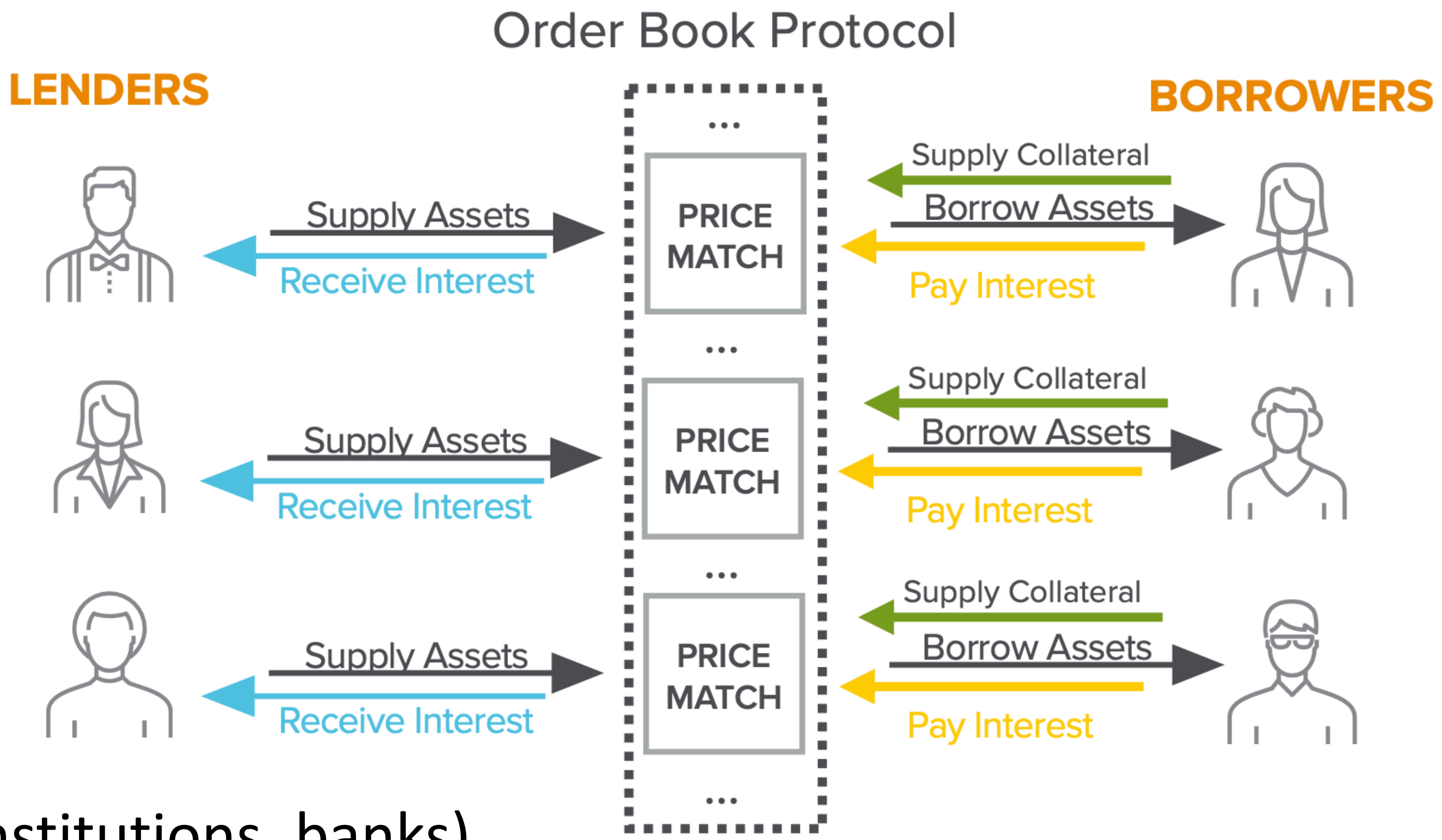
DeFi Lending

Can we build an on-chain lending Dapp?

$\Longrightarrow$ no central trusted parties

$\Longrightarrow$ code available on Ethereum for inspection

A first idea: an order book Dapp



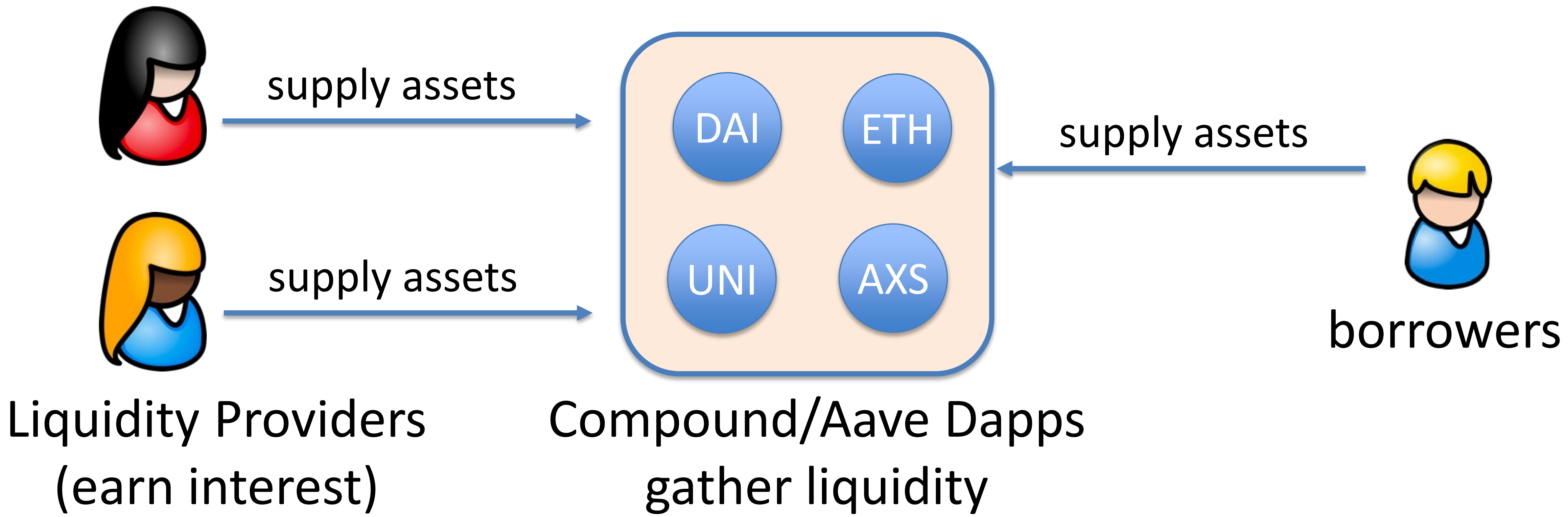
(large institutions, banks)

Challenges

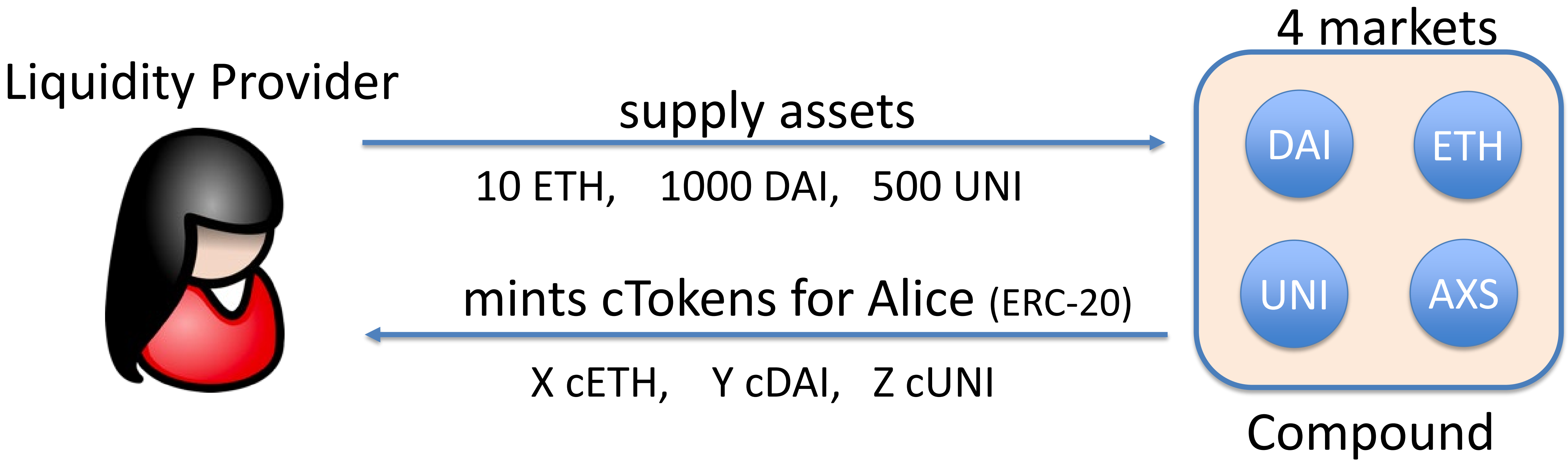
- **Computationally expensive**: matching borrowers to lenders requires many transactions per person (post a bid, retract if the market changes, repeat)
- **Concentrated risk**: lenders are exposed to their direct counterparty defaulting
- **Complex withdrawal**: a lender must wait for their counter-parties to repay their debts

A better approach: liquidity pools

Over-collateralized lending: Compound and Aave



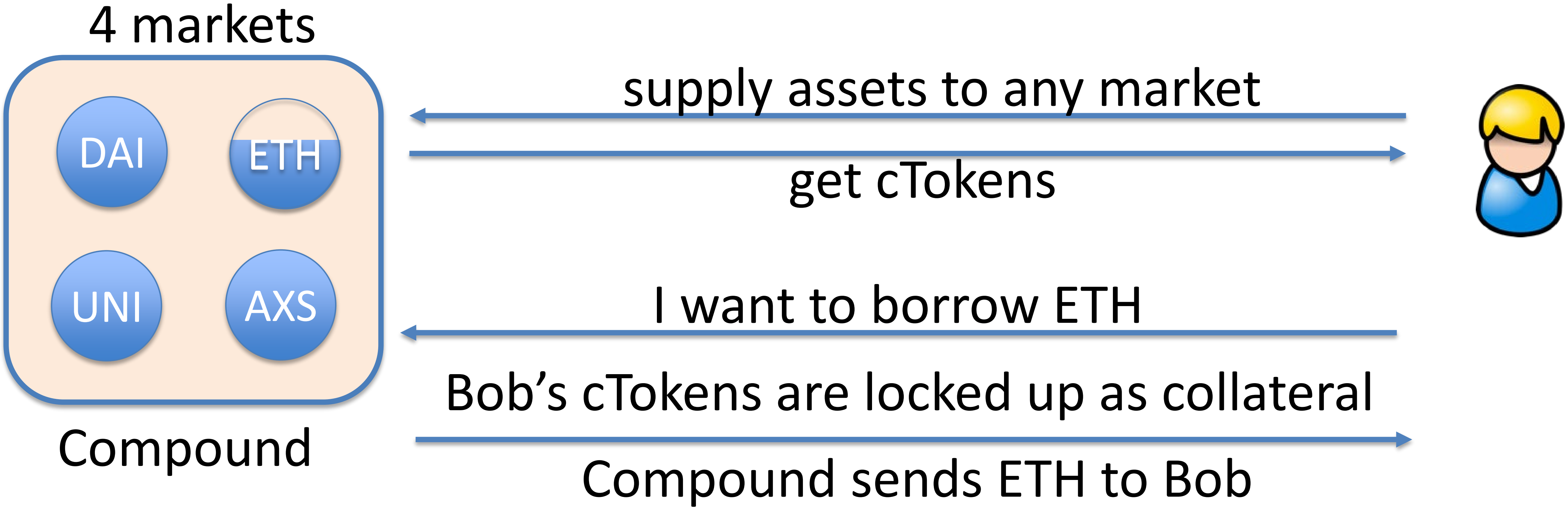
Example: Compound cTokens



Value of X, Y, Z is determined by the current exchange rate:

Token to cToken exchange rate is calculated every block

Borrowers



Bob's accrued interest increases ETH/cETH exchange rate

$\Longrightarrow$ benefit cETH token holders (ETH liquidity providers)

The exchange rate

Consider the ETH market:

Supplying ETH: adds to $\text{UnderlyingBalance}_{\text{ETH}}$

Borrowing ETH: adds to $\text{totalBorrowBalance}_{\text{ETH}}$

Interest: added repeatedly to $\text{totalBorrowBalance}_{\text{ETH}}$

$$\text{ExchangeRate}_{\text{ETH/cETH}} = \frac{\text{UnderlyingBalance}_{\text{ETH}} + \text{totalBorrowBalance}_{\text{ETH}} - \text{reserve}_{\text{ETH}}}{\text{cTokenSupply}_{\text{ETH}}}$$

$\Longrightarrow$ As totalBorrowBalance increases so does ExchangeRate

The interest rate: constantly updates

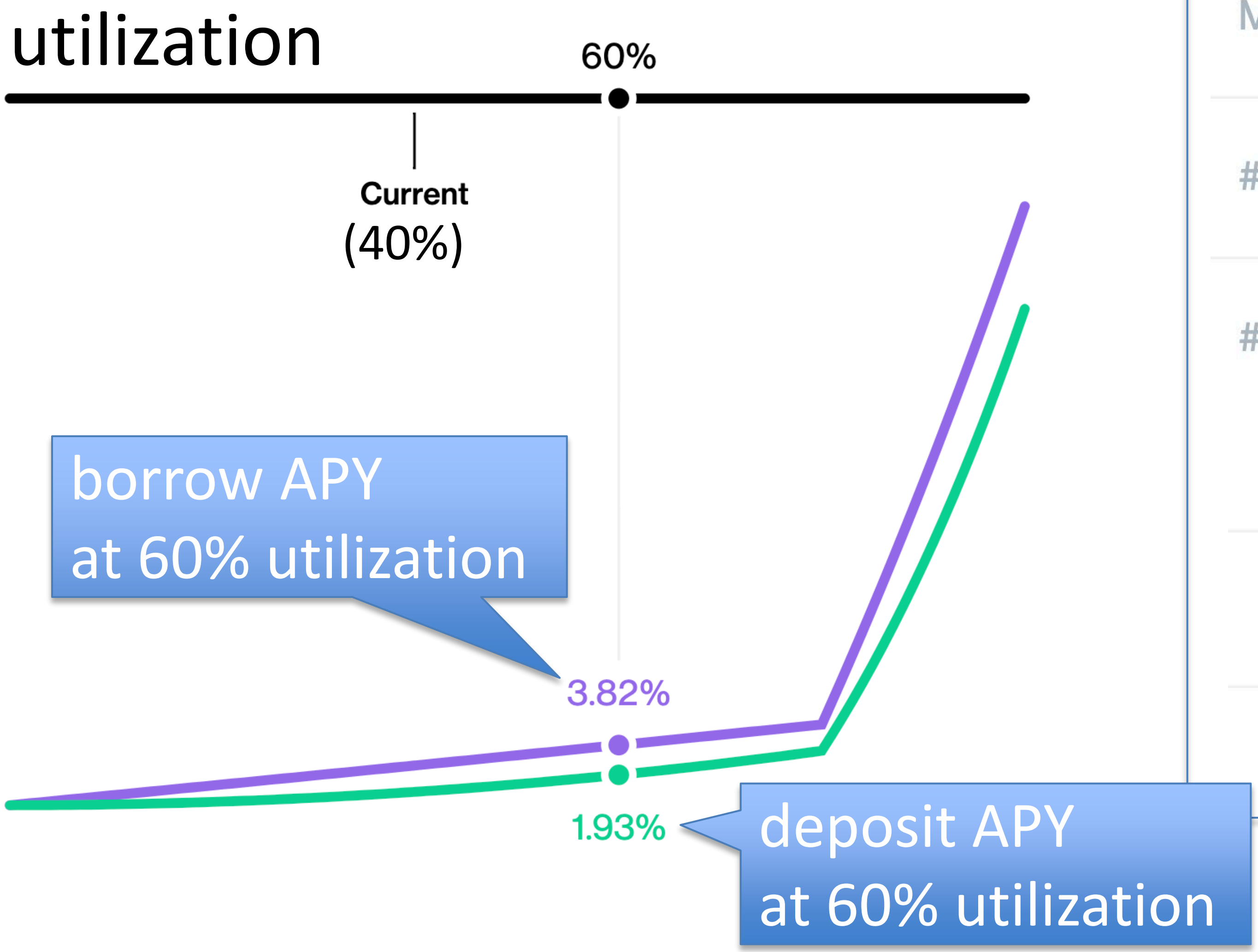
Key idea: determined by demand for asset vs. asset market size

Utilization ratio: $$U_{ETH} = \frac{\text{totalBorrowBalance}_{ETH}}{\text{availableBalance}_{ETH} + \text{totalBorrowBalance}_{ETH}}$$

higher totalBorrowBalance, or
lower availableBalance in contract → higher $\mathbf{U_{ETH}} \in [0,1]$

$$\text{interestRate}_{ETH} = \text{BaseRate}_{ETH} + \mathbf{U_{ETH}} \times \text{slope}_{ETH}$$

Example: Compound DAI market



Market Liquidity	377,443,771 DAI
# of Suppliers	18468
# of Borrowers	2750
Collateral Factor	83%
cDAI Minted	26,810,077,978
Exchange Rate	1 DAI = 45.26986803778856 cDAI

(Oct. 2022)

Liquidation: debt > BorrowCapacity

If user's health < 1 then **anyone** can call:

liquidate(borrower, CollateralAsset, BorrowAsset, uint amount)

address of borrower being liquidated

Liquidator wants cTokens in this asset (e.g., cDAI)

Liquidator is providing this asset (e.g., ETH)

This function transfers liquidator's ETH into ETH market, and gives the liquidator cDAI from user's collateral

Liquidation: debt > BorrowCapacity

If user's health < 1 the **anyone** can call:

liq t)

add

b set

Liquidator is repaying the user's ETH debt
and getting the user's cDAI
[at a discounted exchange rate -- penalty for user]

(e.g., cDAI)

(e.g., ETH)

This function transfers liquidator's ETH into ETH market,
and gives the liquidator cDAI from user's collateral

What is liquidation risk?

Historical DAI interest rate on Compound (APY):

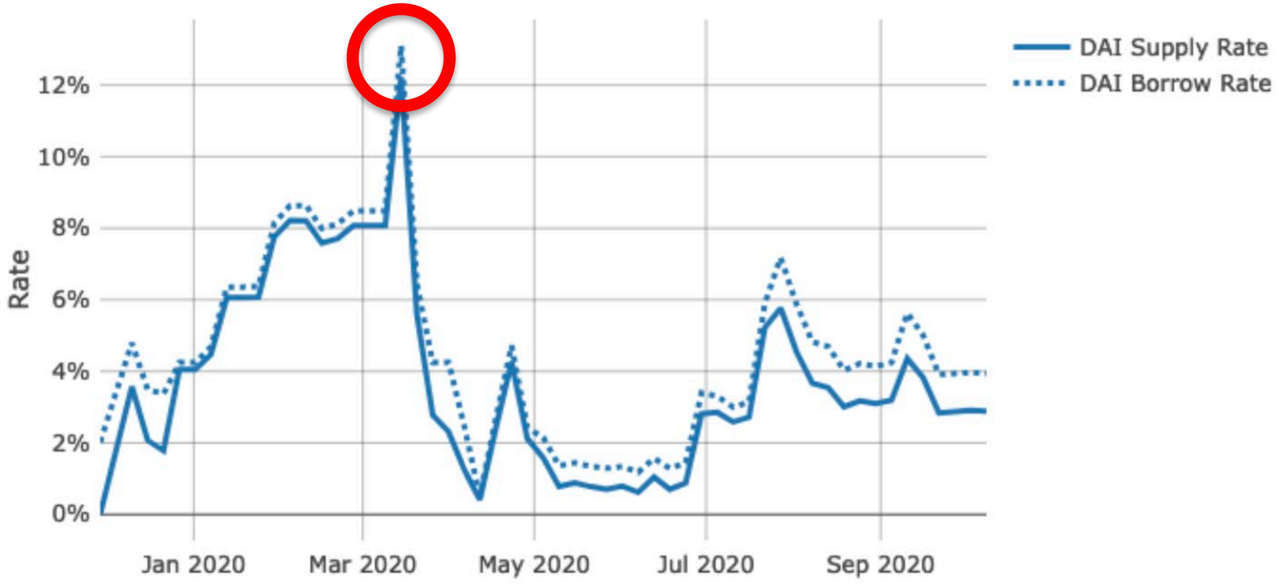
Demand for DAI spikes

$\Longrightarrow$ price of DAI spikes

$\Longrightarrow$ user's debt shoots up

$\Longrightarrow$ user's health drops

$\Longrightarrow$ liquidation …

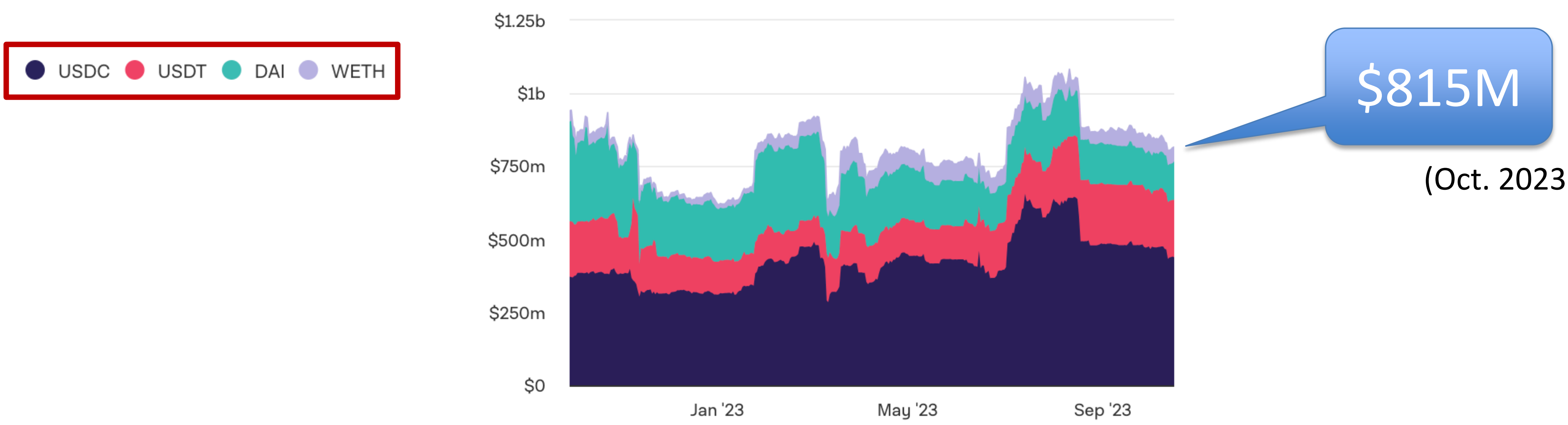


To use Compound, borrower must constantly monitor APY and quickly repay loans if APY goes too high (can be automated)

Summary & stats

- Liquidity providers can earn interest on their assets
- DeFi lending usage:

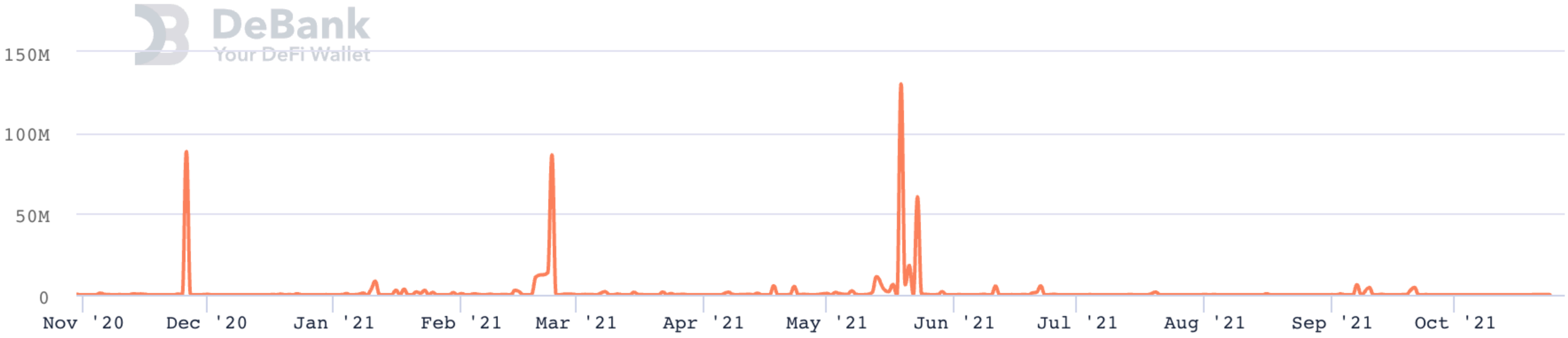
Compund outstanding debt



(Oct. 2023)

Summary & stats

Compound liquidation statistics:



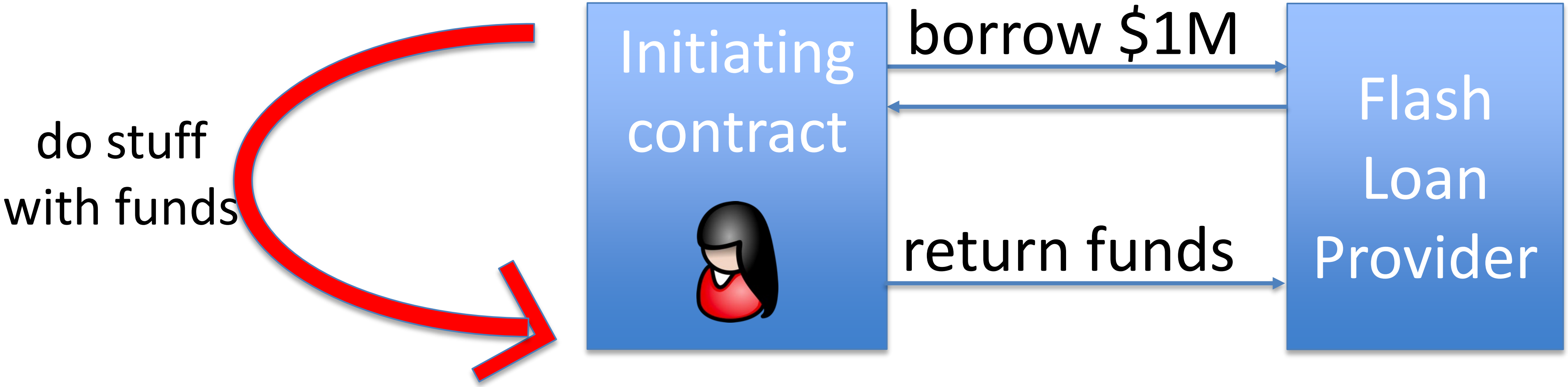
Caused by collateral price drops or debt APY spikes

Flash loans

What is a flash loan?

A flash loan is taken and repaid in a single transaction

$\Longrightarrow$ zero risk for lender $\Longrightarrow$ borrower needs no collateral



(Tx is valid only if funds are returned in same Tx)

"Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit"

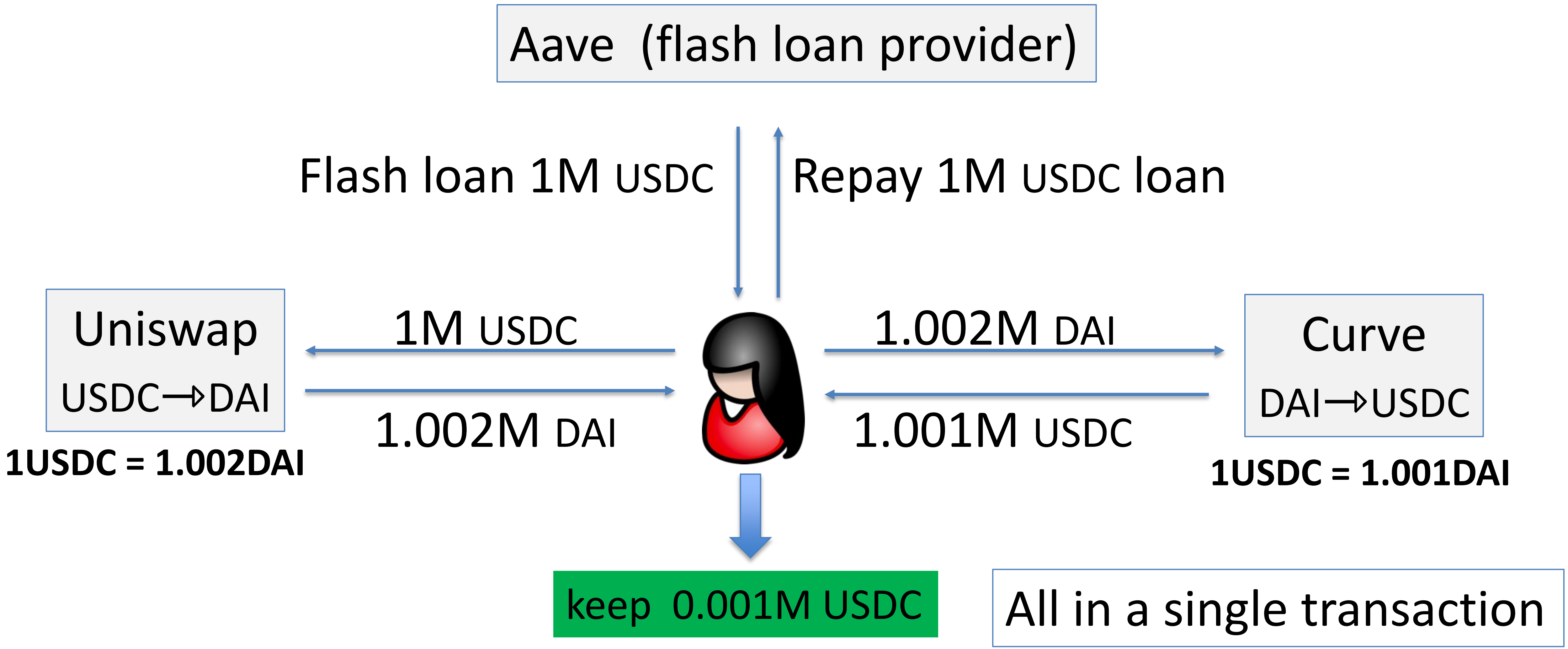
Use cases

- Risk free arbitrage
- Collateral swap
- DeFi attacks: price oracle manipulation

⋮

Risk free arbitrage

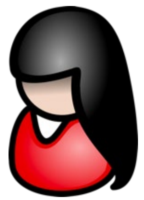
Alice finds a USDC/DAI price difference in two pools



Collateral swap

start:

Alice @Compound



end goal:

Alice @Compound

-1000 DAI
+1 cETH

borrowed DAI using ETH as collateral

Take 1000 DAI flash loan
Repay 1000 DAI debt
Redeem 1 cETH
Swap 1 cETH for 3000 cUSDC
Deposit 3000 cUSDC as collateral
Borrow 1000 DAI
Repay 1000 DAI flash loan

(a single Ethereum transaction)

-1000 DAI
+3000 cUSDC

borrowed DAI using USDC as collateral

Aave v1 implementation

```
function flashLoan(address _receiver, uint256 _amount) {
    …
    // transfer funds to the receiver
    core.transferToUser(_reserve, userPayable, _amount);

    // execute action of the receiver
    receiver.executeOperation(_reserve, _amount, amountFee, _params);
    …
    // abort if loan is not repaid
    require( availableLiquidityAfter == availableLiquidityBefore.add(amountFee),
            "balance inconsistent");
}
```

Flash loans amounts on Aave (in 2021)

Top 5 Days - Loan Amount

Date	FALSHLOAN_USD ▾
May 22	624.5M
May 5	520.9M
May 21	515.0M
May 19	265.7M
Aug 3	163.7M