Oracles, tokens and off-chain networks

Things build with smart contracts

Leander Jehl

Oracles

Oracle

Access data outside the blockchain

An **Oracle** is a smart contract that publishes information about real world data on the chain.

Oracle
Example: Rain ensurance

- Ensurance contract needs weather data to

  - Pay out policies

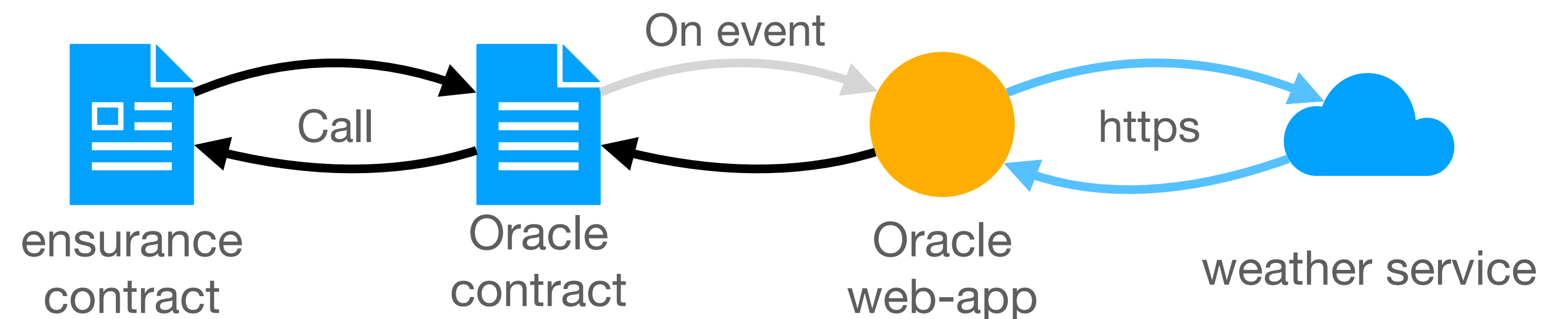 - Determine prices

ensurance contract

weather service (yr.no)
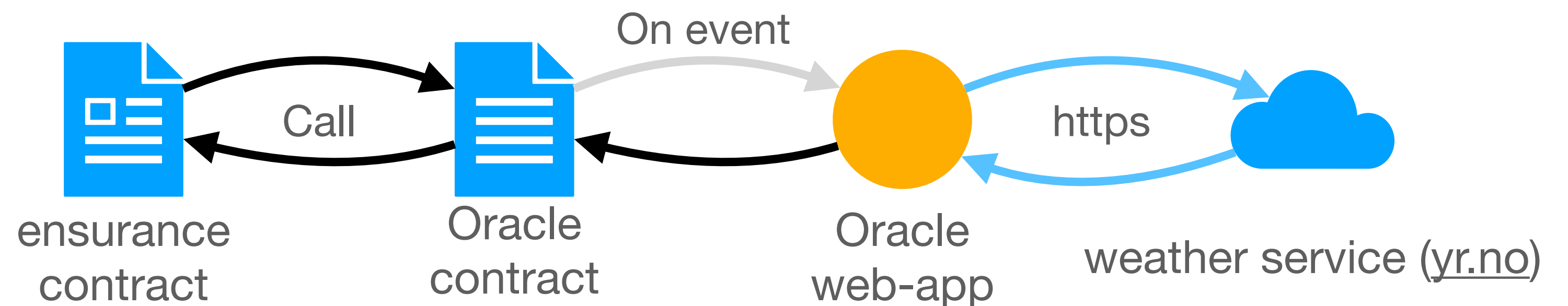
Oracle
Example: Rain ensurance

- Ensurance contract needs weather data to

  - Pay out policies

  - Determine prices

Oracle
Example: Rain ensurance

- Ensurance contract calls oracle contract

- Oracle contract emits event

- Oracle web app listens to event

- Web app gets data from api

- Web app invokes contract

ensurance contract — Call — Oracle contract — On event — Oracle web-app — https — weather service (yr.no)

Oracle
Example: Rain ensurance

- Ensurance contract calls oracle contract

- Oracle contract emits event

- Oracle web app listens to event

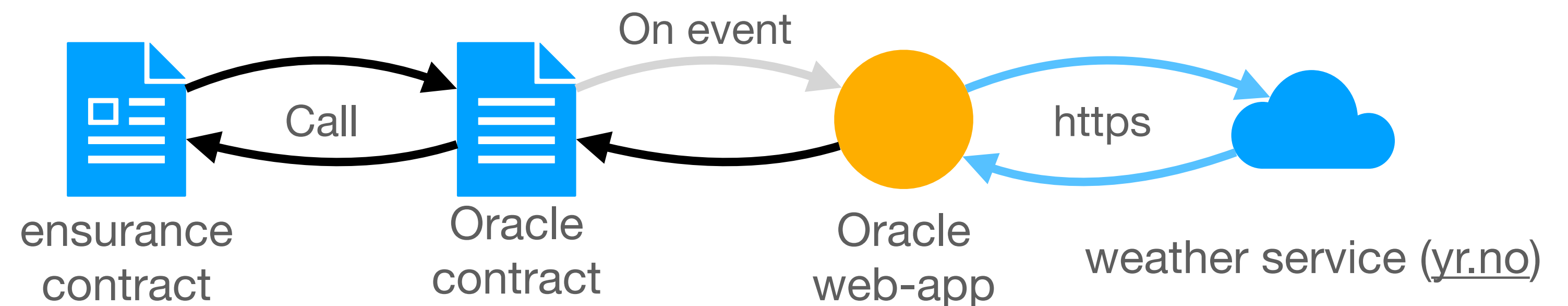- Web app gets data from api
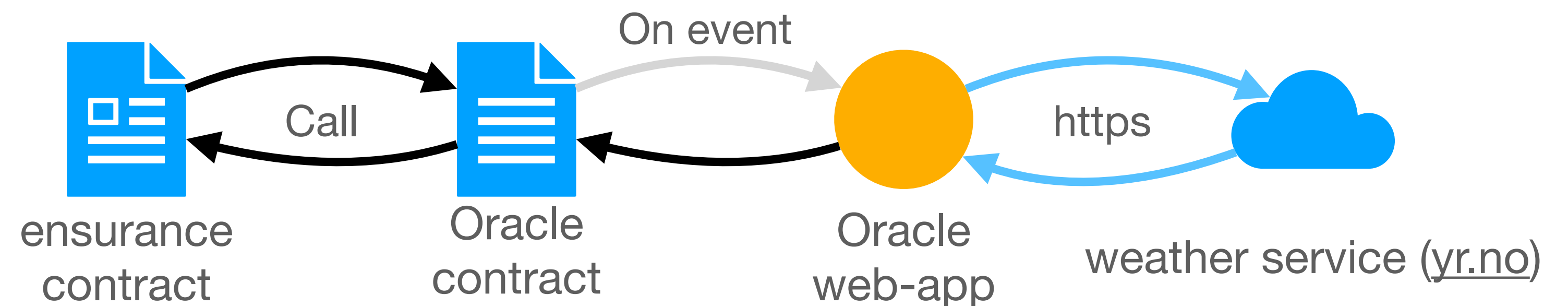
- Web app invokes contract

Check cryptozombies tutorial

ensurance contract — Call — Oracle contract — On event — Oracle web-app — https — weather service (yr.no)

Oracle
Example: Rain ensurance

- Why should we use an extra oracle contract?

  - Can update if we need to update oracle

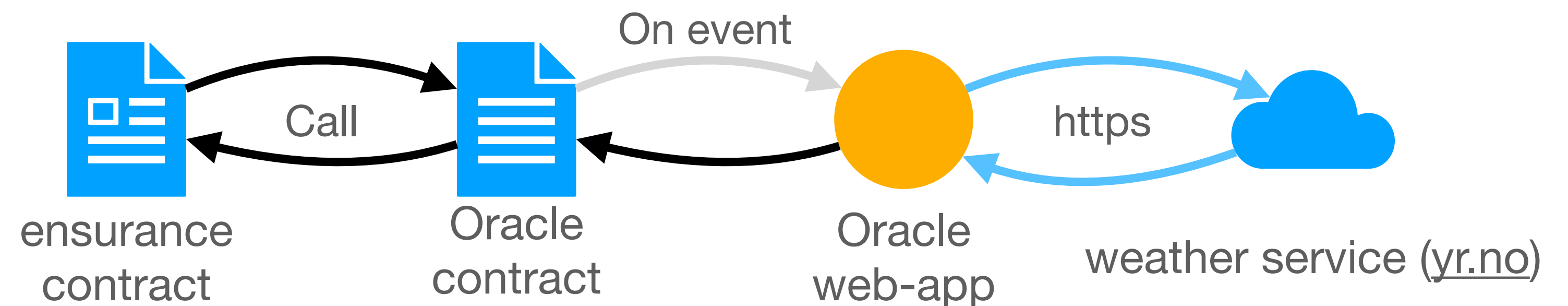- Who do we need to trust?

  - Oracle provider, and API provider

Oracle
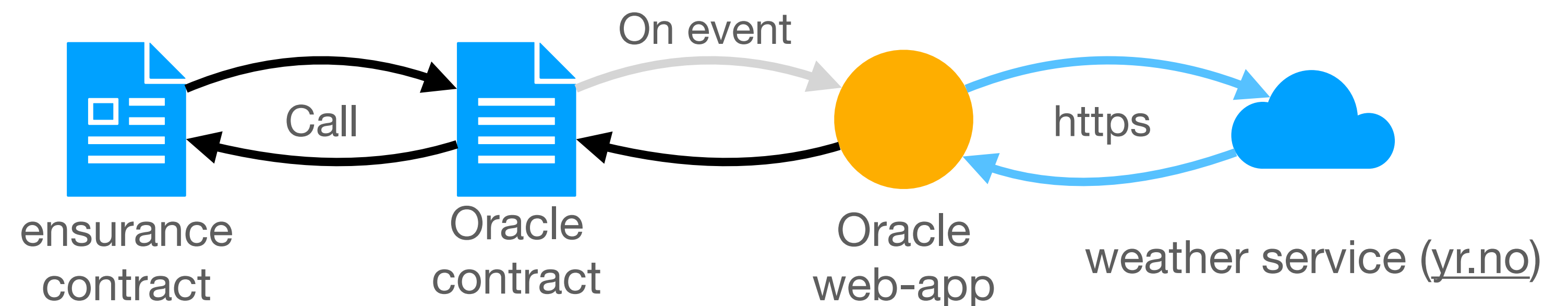## Example: Rain ensurance

- Can we avoid trusting the oracle?

  - Yes, run oracle web-app in trusted execution (Intel SGX)

Oracle
Variations

- Access private data, e.g. using login

    - Yes, run oracle web-app in trusted execution (Intel SGX)

- Provide oracle service, that anyone can use

ensurance contract — Call → Oracle contract — On event → Oracle web-app — https → weather service (yr.no)

Tokens

Tokens

Native and non-native tokens

- Native tokens is the base currency of a blockchain

  - Bitcoin, ether, …

- Non-native tokens are similar but they are build using smart contracts

Tokens
Non-native tokens

- A smart contract keeps token balances

- Limited supply?

- Holders get benefits?

  - Voting rights

  - Discount

  - Etc

```solidity
contract TokenBank {
    mapping(address => uint) private balances;
    address public owner;
    uint public price;

    // function SimpleBank() deprecated syntax for
    constructor(uint tprice) public {
        owner = msg.sender;
        price = tprice;
    }

    function buy() public payable returns(uint) {
        balances[msg.sender] += msg.value/price;
        return balances[msg.sender];
    }

    function transfer(uint amount, address receiver) public returns (){
        if (balances[msg.sender] >= amount){
            balances[msg.sender] -= amount;
            balances[receiver] += amount;
        }
    }

    function balance() view public returns (uint) {
        return balances[msg.sender];
    }
}
```

Tokens
Non-fungible tokens

- A smart contract keeps token balances

- Fungible tokens: All the same

- Non-fungible tokens: each token is different

```solidity
contract NFToken {
    mapping(uint256 tokenid => address) private owners;
    address public owner;

    // function SimpleBank() deprecated syntax for
    constructor() public {
        owner = msg.sender;
    }

    function create(uint tid) public returns() {
        require(msg.sender == owner);
        require(owners[tid] == address(0));
        ownsers[tid] = owner;
    }

    function transfer(uint tid, address receiver) public returns (){
        if (owner[tid] == msg.sender){
            owner[tid] =receiver;
        }
    }

}
```
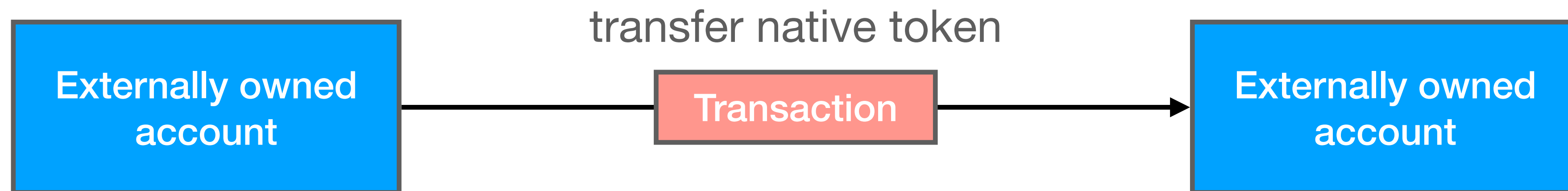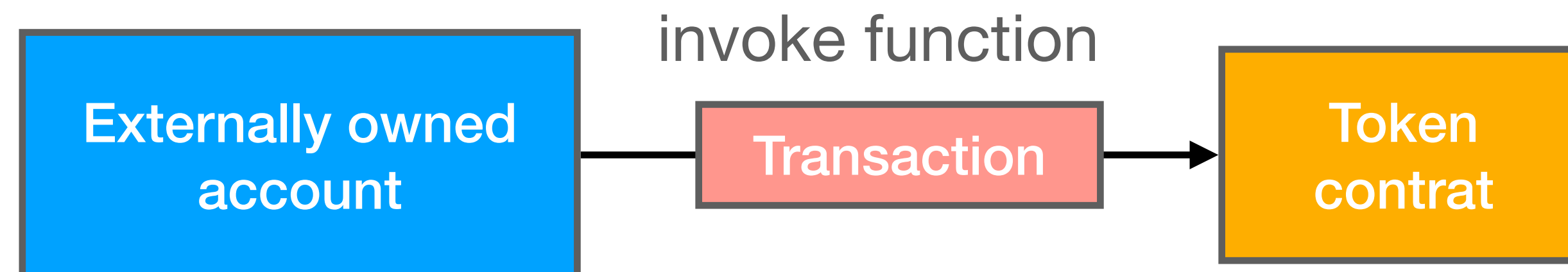
Tokens
Transfer tokens
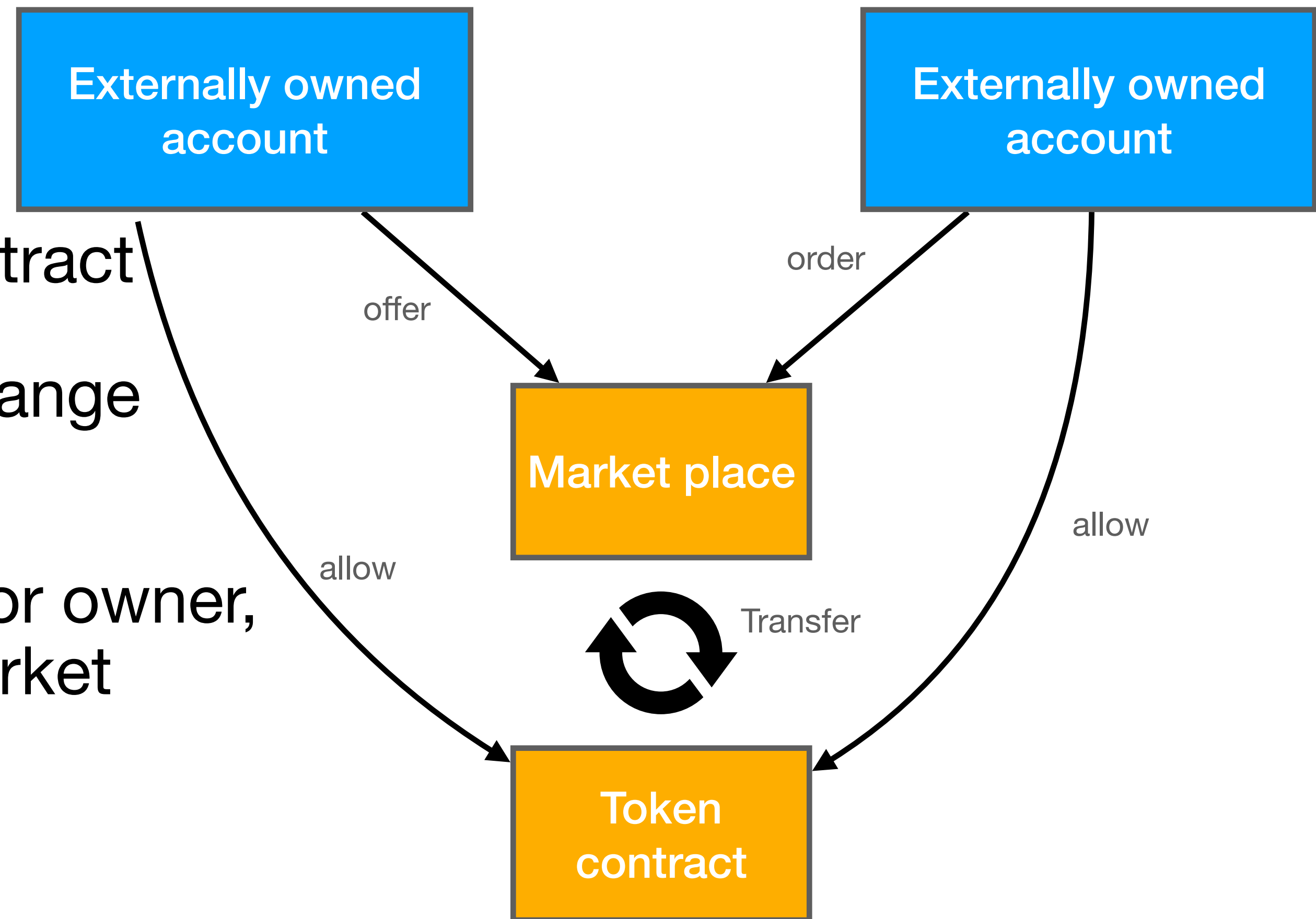
- Transfer native token by transaction

- Transfer non-native token by function call on token contract

Tokens

ERC20 Tokens

- Standart interface for token contract

- Allows markets to sell and exchange all compliant tokens

- Token contract needs support for owner, to allow/delegate transfer to market place
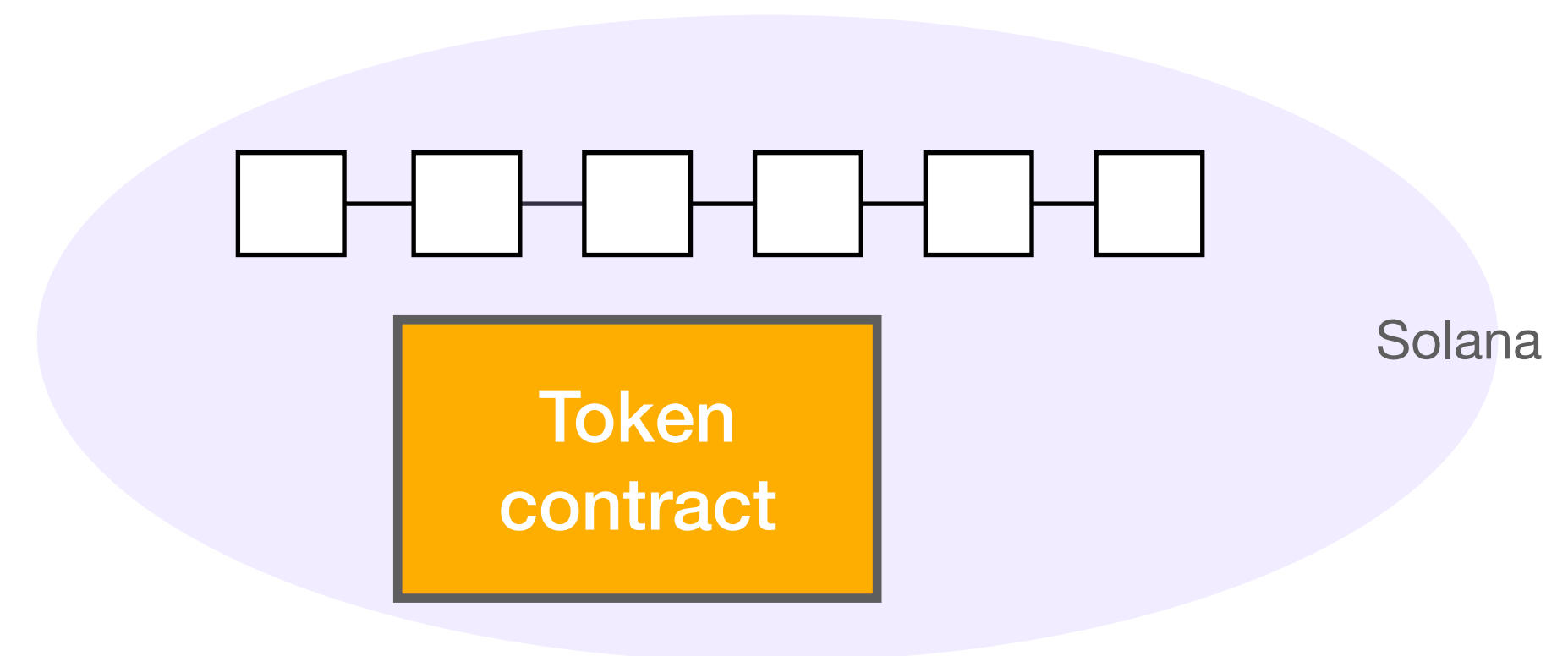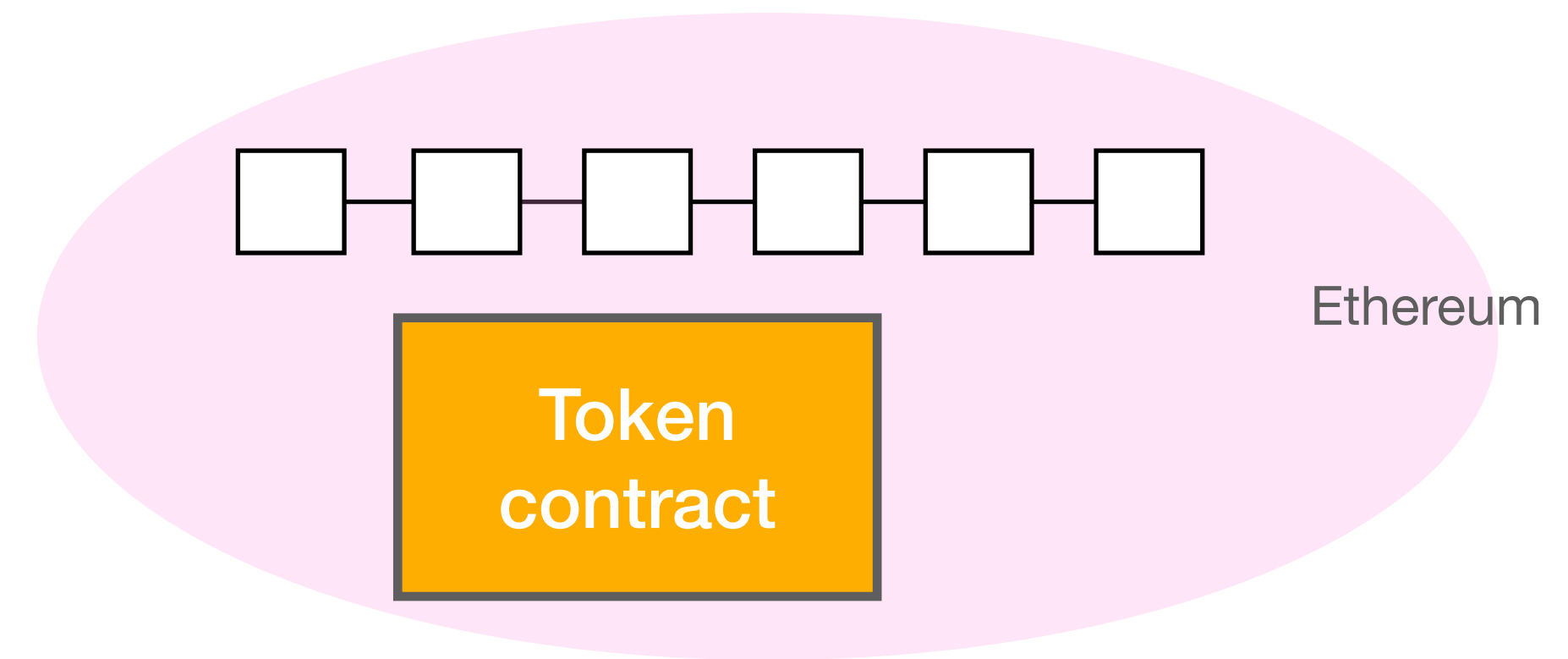

Token contract

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v5.5.0-rc.1/contracts/token/ERC20/ERC20.sol

Tokens

ERC20 Tokens

- Can you move an ERC20 token from one blockchain to the other?

Ethereum

Token contract

Solana

Token contract

Stable coins

Tokens with a stable value

- Designed for stable price 1token = 1$

  - Keep reserve to back value of tokens

    - Fiat currency held in traditional banking (e.g. Tether, USDC, …)

    - Different cryptocurrency locked in smart contracts (e.g DAI)

    - Custom investement token and mechanisms to affect the stable price, e.g. printing when too high, buy back when too low.

Stable coins
Tokens with a stable value

- Designed for stable price 1token = 1$

  - Keep reserve to back value of tokens

    - Fiat currency held in traditional banking (e.g. Tether, USDC, …)
      Problem: Liquidity, and how the reserve is structured,
      e.g. fiat, obligations, …

    - Different cryptocurrency locked in smart contracts (e.g DAI)
      Problem: strong price fluctuations in locked cryptocurrency

    - Custom investement token
      Problem: what if investement token looses value?

Off chain networks / Layer 2

Off-chain / Layer 2
General idea

- **Not all data needs to be on the chain.**

- **Save on transaction fees**

- **Get some guarantees from chain.**

  - **Payment channels**

  - **Sidechains**

  - **Commit chains**

Payment channels

Paiment channels
General idea

- **Idea:** If two parties agree, they can do a transaction outside of the chain without paying fees.

  - Once they disagree, they can use the chain to settle the dispute.

  - Can increase transaction throughput

  - Can give low fees

Payment Channels
Example: Uni-directional payment channel

- **Idea:** Allow any number of payments from A to B within given limit

- A creates contract with balance.

- A can send signed statements of B's balance to B

- B can cash in his balance with the contract

- If B does not cash in, A can terminate the contract and get back the balance, after expiration date.

Check example
https://solidity-by-example.org/app/uni-directional-payment-channel/

Payment Channels
Example: Bi-directional payment channel

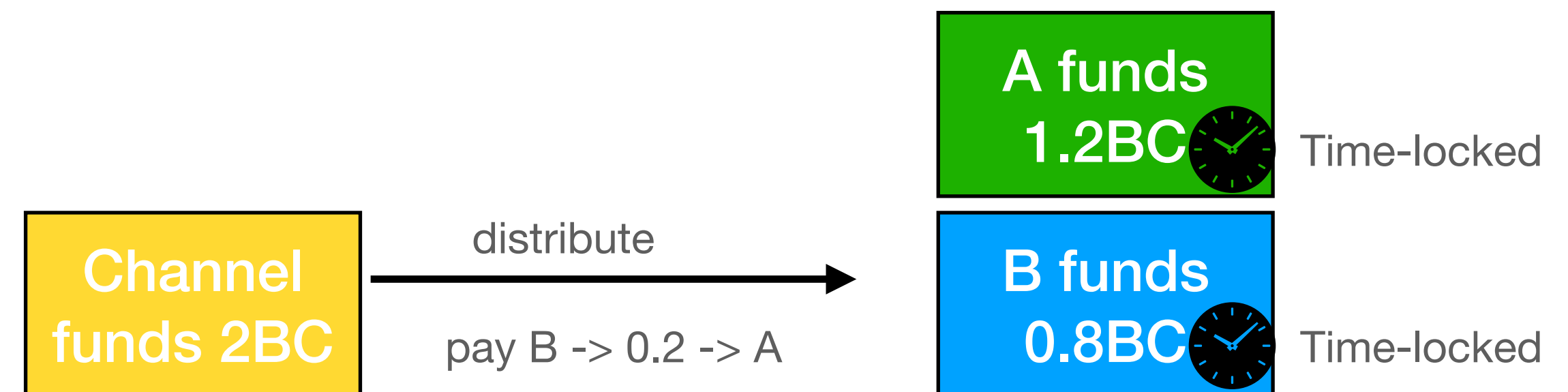- **Idea:** Allow any number of payments between A and B within given limit,

- A and B both pay a balance to contract

- A and B can send signed statements of their balances to each other, with increasing nonces

- A or B can submit balance, signed by both to contract. This triggers countdown

- If other party does note submit a balance with larger nonce, balances are payed out.

Payment Channels
Example: Bi-directional payment channel
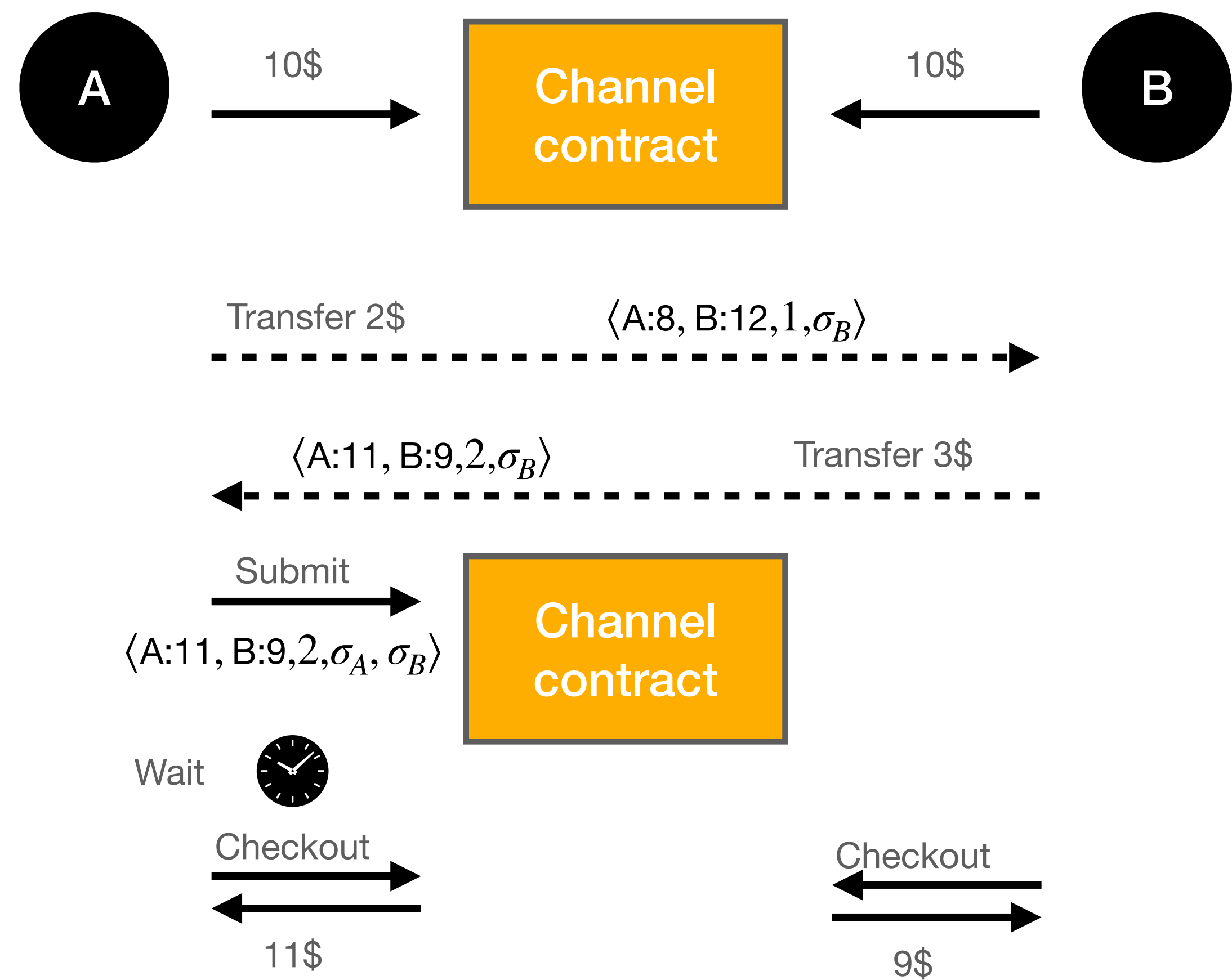
Payment Channels
Example: Bi-directional payment channel
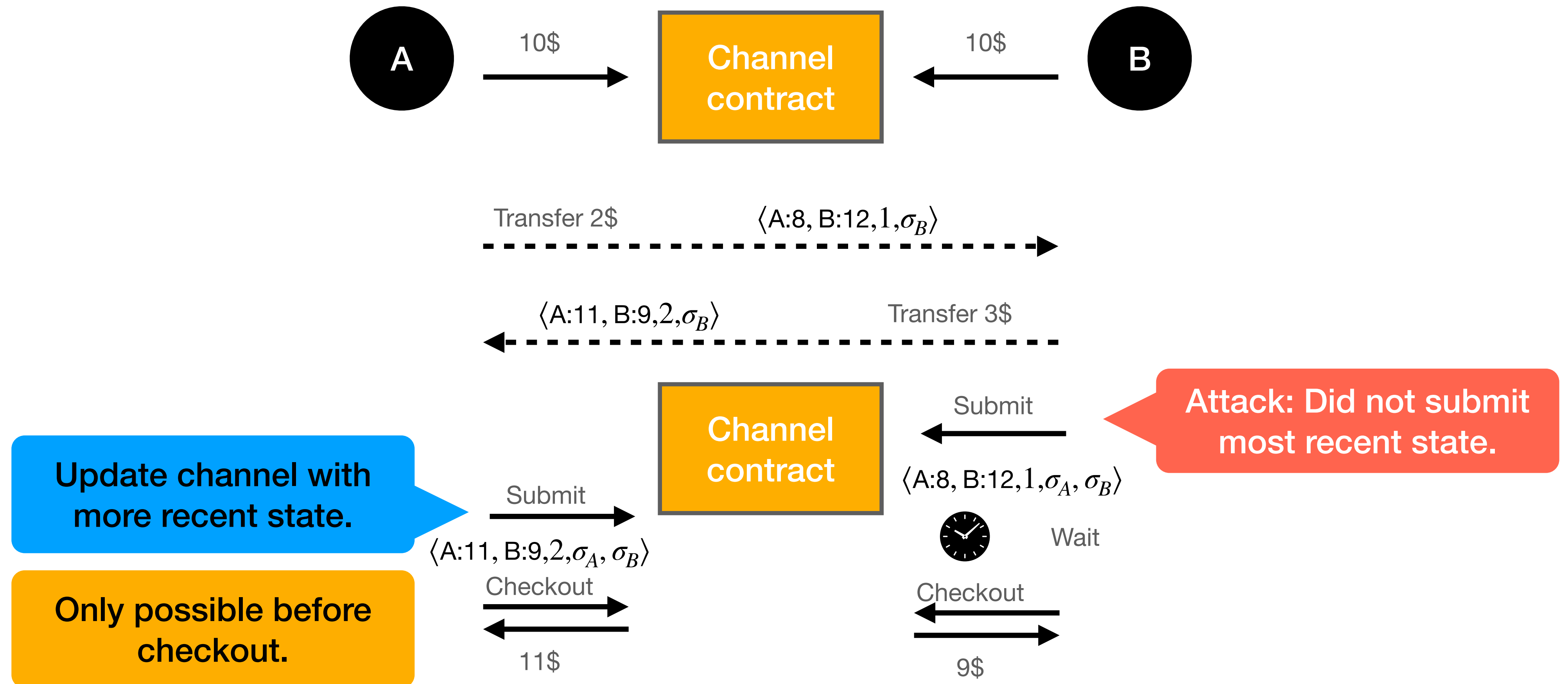
Payment Channels
Example: Bi-directional payment channel

- **Idea:** Allow any number of payments between A and B within given limit,

Problem:

- Timeout

- Locked funds

Payment Channels
Example: Lightning channels on UTXO

- **Funds are locked in one Output**

- **First channel payment:**
  Create valid transaction,
  to distribute funds
  (not submitted)

- **Second payment**
  Create valid transaction to
  distribute funds, and
  revocation transaction

A funds
1.2BC   Time-locked

B funds
0.8BC   Time-locked

Channel
funds 2BC

distribute

pay B -> 0.2 -> A

Payment Channels
Example: Lightning channels on UTXO

- **Funds are locked in one Output**

- **First channel payment:**
  Create valid transaction,
  to distribute funds
  (not submitted)

- **Second payment**
  Create valid transaction to
  distribute funds, and
  revocation transaction
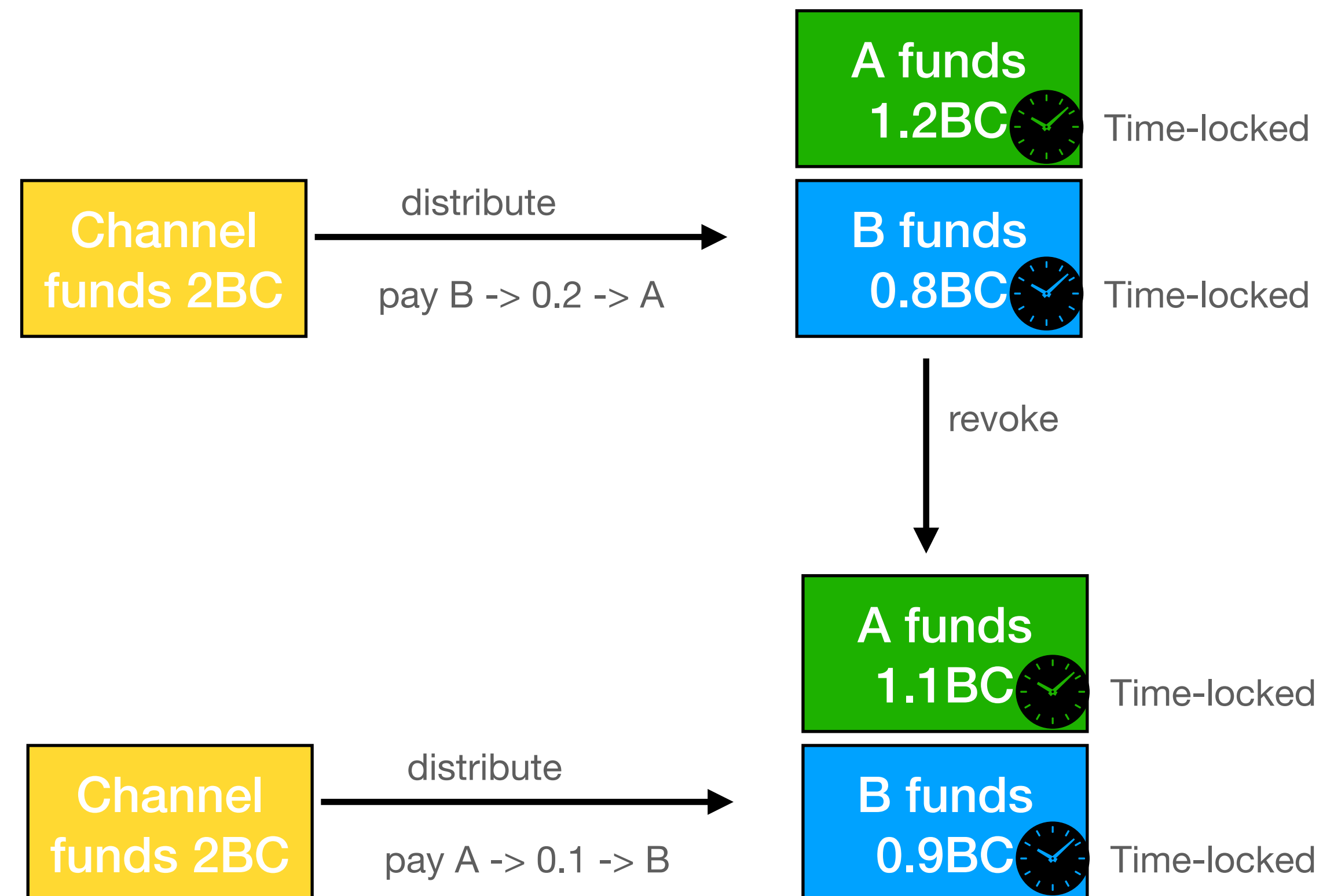
Payment Channels
Example: Multi hop payment

- **Idea:** payment across multipe channels

- Pay fees to intermediates (X and Y)

Payment

Channel

- **How:** Conditional payments, with secret known to A

- B -> Y; Y -> X; … s. t. payment is only valid if participants know the secret.

- Friendly settlement: Secret forwarded A -> X -> Y

- Unfriendly settlement: A publishes secret on chain, X and Y can see secret.
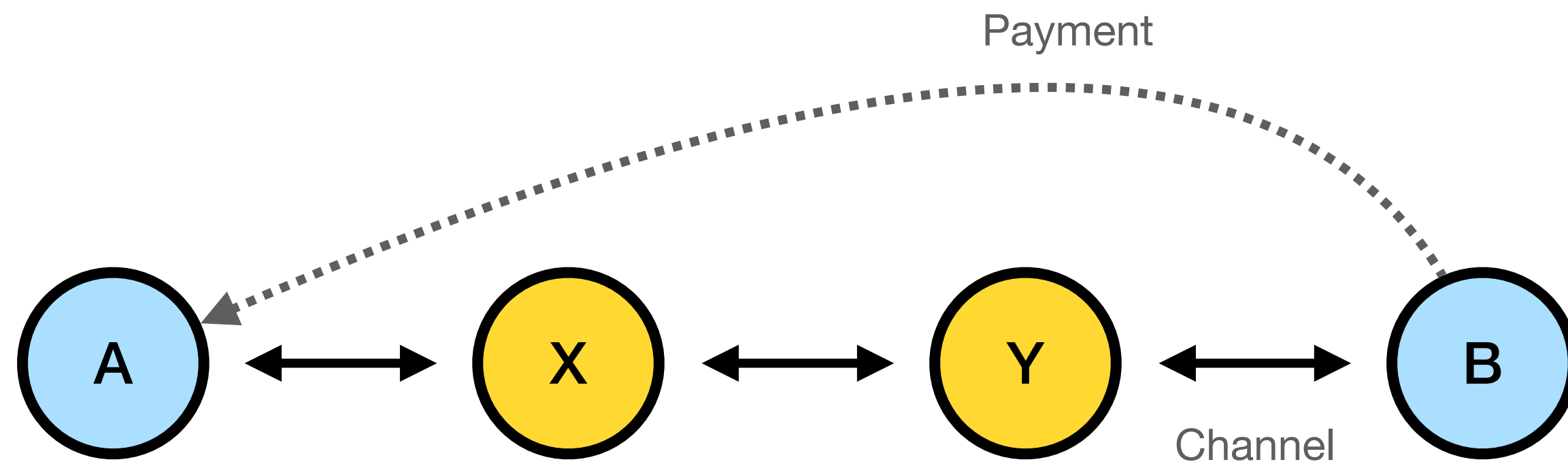
Payment Channels
Example: Payment routing

- **Find path from B to A**

Problem:

- Limitted & changing channel capacity

- Fees play a role

- Privacy of transaction plays a role, *e.g. avoid intermediaries knowing who pays what to whom.*
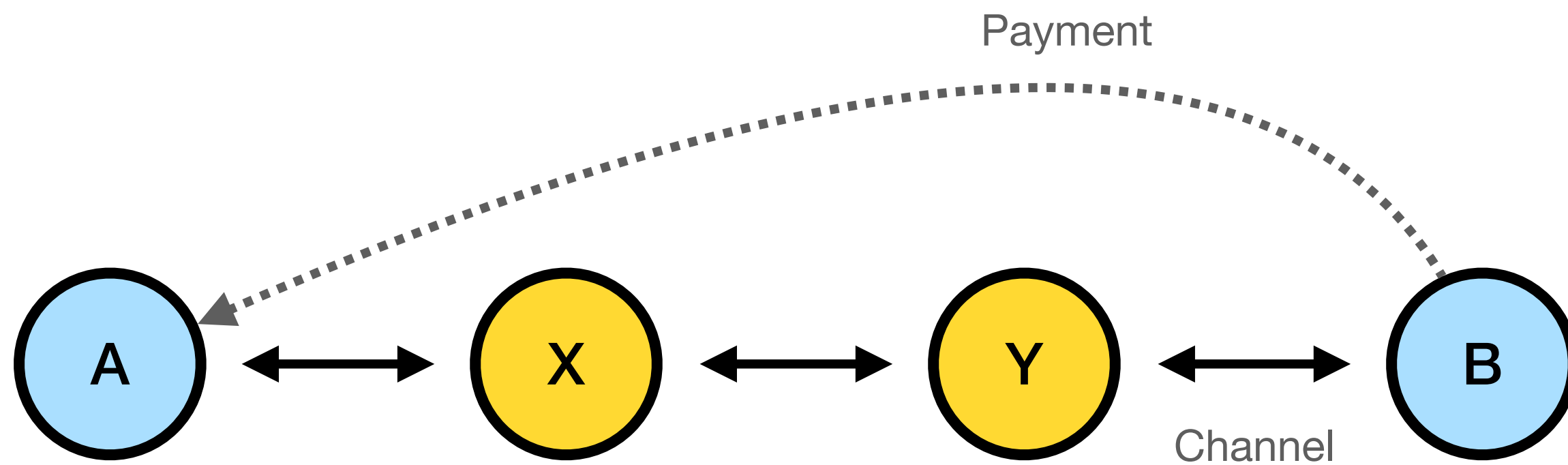
Payment Channels
Example: Other channels

- **Virtual channels:**

  - Given two payement channels A <-> I and I <-> B, create a virtual channel between A <-> B.

  - Intermediate is only involved in opening and closing the virtual channel.

  - Fewer fees

- **State channels:**

  - A channel where we can create smart contracts.

  - Only channel members can interact with these contracts.

Side Chains

Side Chains
General idea

- A separate, smaller blockchain

- State root regularly insterted into main chain
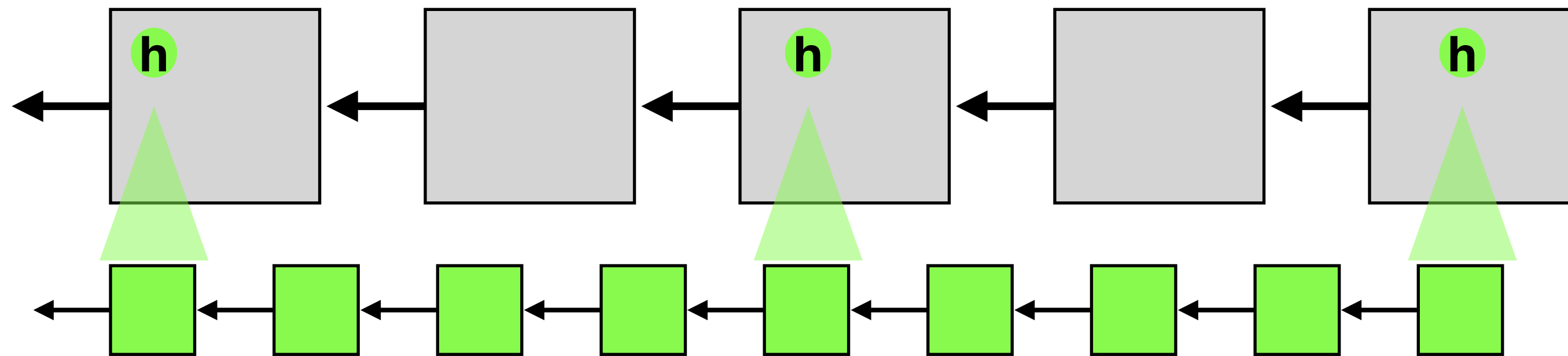
Side Chains
General idea

- A separate, smaller blockchain

- State root regularly insterted into main chain
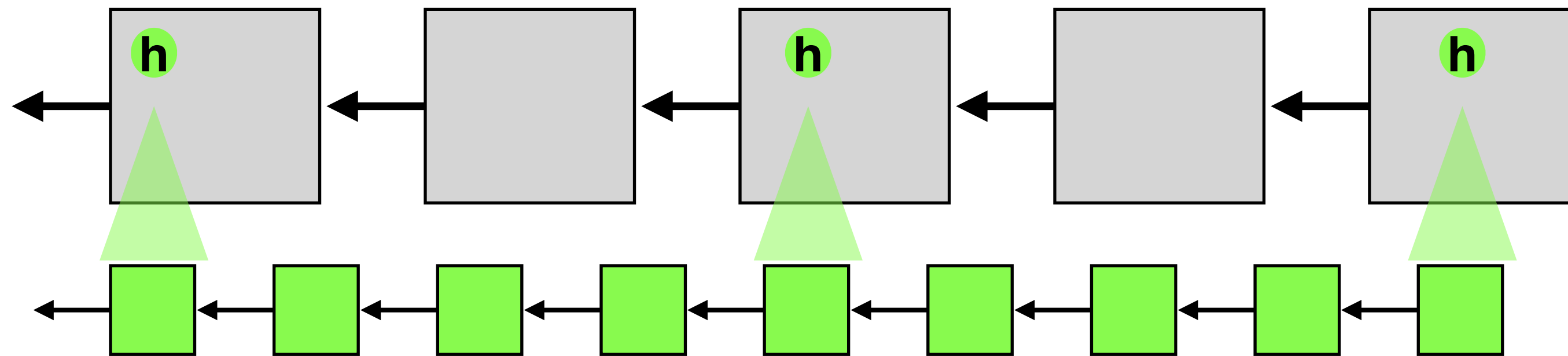
Benefits:

- State fixed by main chain

- Trusted asset transfer

Problem:

- Can a 51% attack on a side chain change state?

Side Chains

General idea

- A separate, smaller blockchain

- State root regularly insterted into main chain
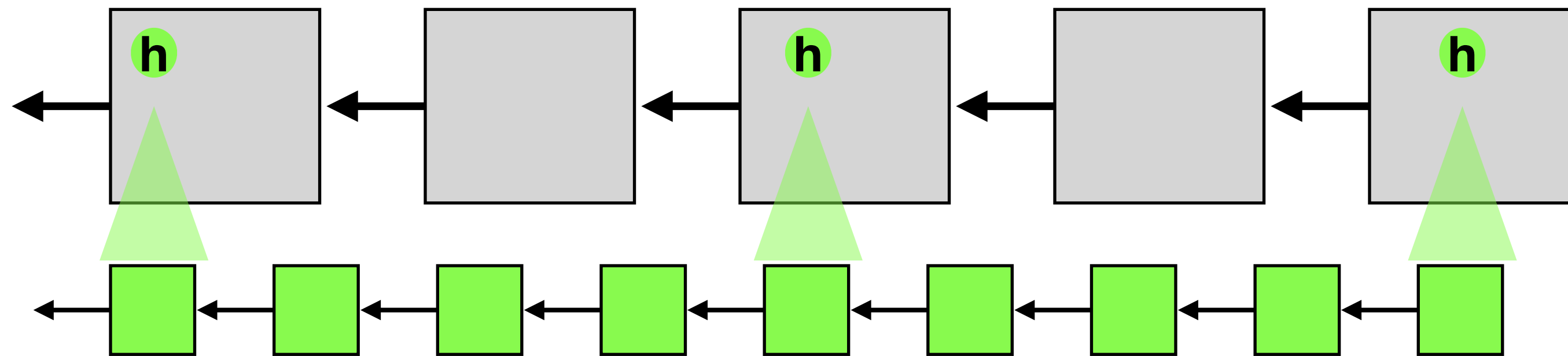
Problem:

- What can happen?
  - Unavailable?
  - Changed?

Commit chains / rollups

Commit chains
Optimistic rollups

- **Idea:** Similar to side chains with single central node (operator) and rules for dispute on chain

- **Dispute:** Can detect and dispute false state updates,
similar to payment channels
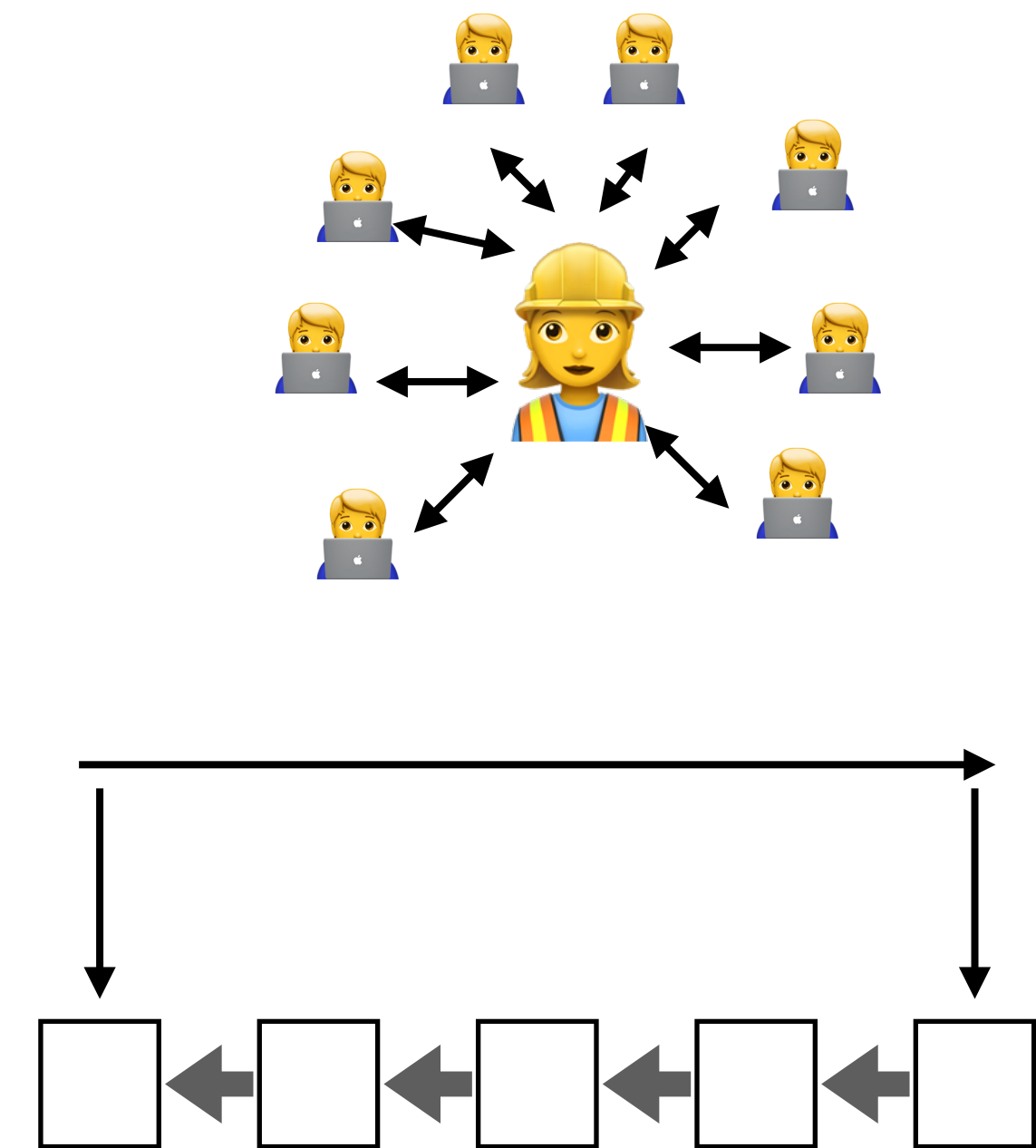
Commit chains

Rollups

- Operator regularly publishes root of state (merkle tree root)

- To finalize operations, need to wait for next state root.

- Can retrieve funds, on chain, according to last state root.

- Members need to check, that state updates are correct.
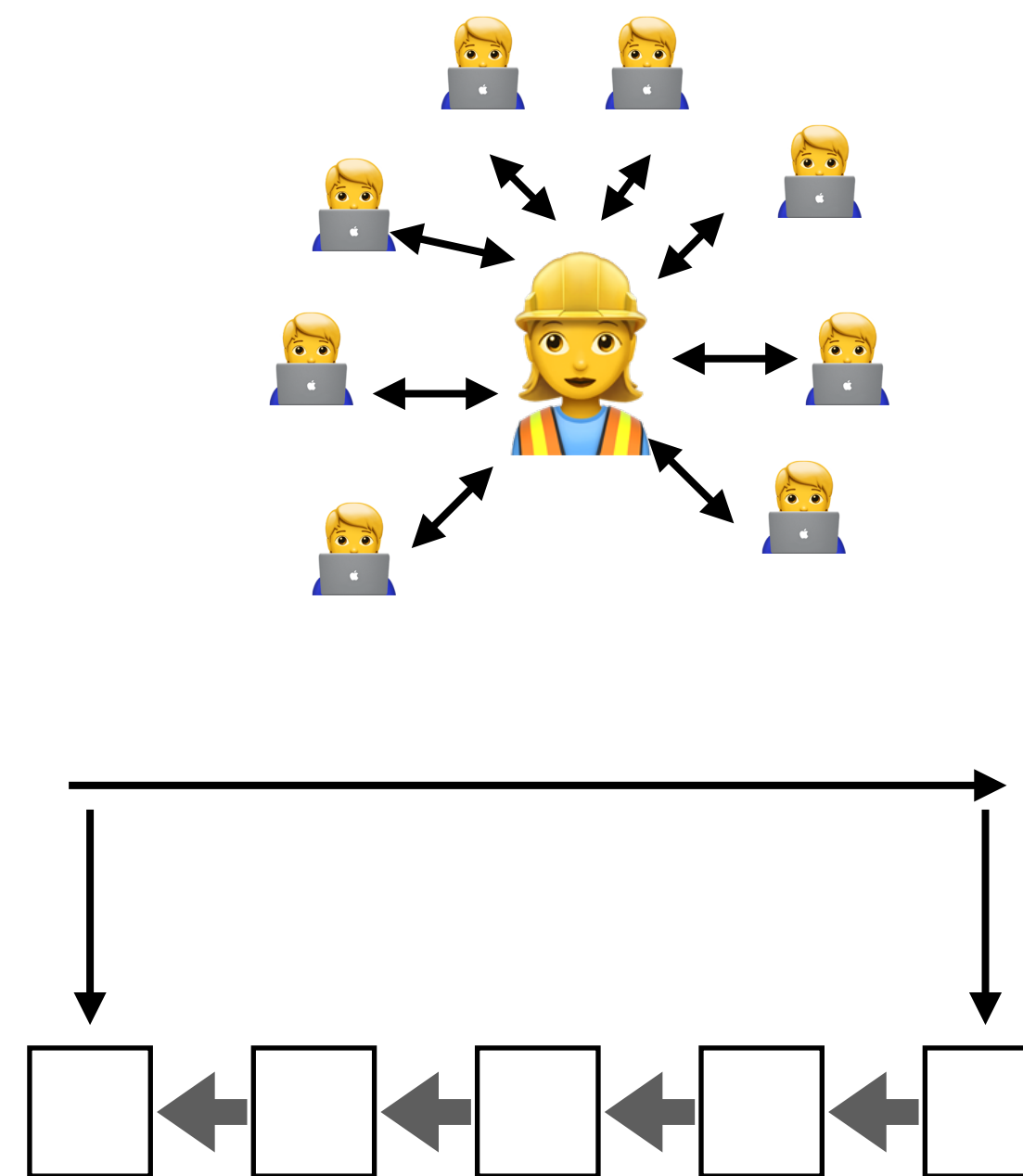
Commit chains

What is submitted to the blockchain?

- Merkle root of new state:
  Need to check that transition is correct
  - Optimistik rollup

- Zero-knowledge proof:
  Ensures correct transition
  Needs to be checked in smart contract
  - zk-rollup
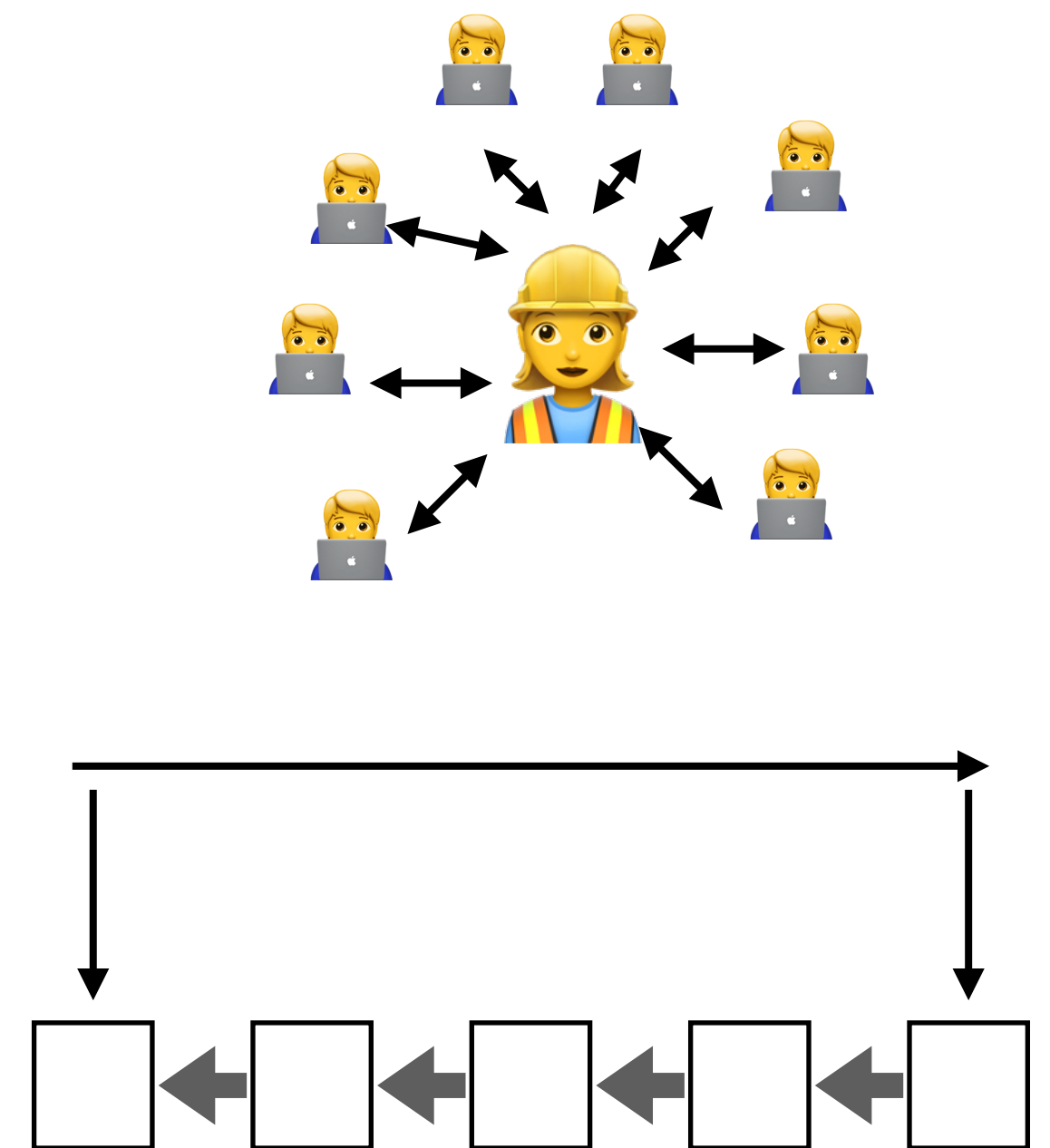
Channels and Commit chains
Assumptions

- **Synchrony:**

  - *Transactions submitted to the blockchain are executed within a max time bound*

  - Needed to submit complaint in time

- **Online:**

  - *Participants need to stay online.*

  - Needed to detect/react to misbehaviour

Off Chain comparison

	On chain transaction	Channel	Commit chain
Cheep fees	❌	✅	✅
Fast confirmation	❌	✅	❌
Can go offline	✅	❌	❌
Unlimitted capacity	✅	❌	✅
Joining	Not necessary	Setup cost	No cost