Oracles between blockchains and the outside world

Matthieu BETTINGER

Summary

- Motivation
- Oracles: Theory and practice
- Standardization & Interoperability
- Data provisioning trade-offs

Motivation

- **Blockchains are closed systems**

  → they cannot read/write data of off-chain systems

  → off-chain entities provide data

  → Need for data provisioning systems

- **Use cases for smart contracts in IoT (Industry 4.0, Smart Cities,...), Education, Health and more**
  - Relying on critical off-chain data or events

    → Need for ***trustworthy*** data provisioning systems

Oracles

- **Link blockchains to off-chain data and events (even to other blockchains) and vice-versa**

- **Two components:**
  - Off-chain: data collection, processing, transmission wrt. requests
  - On-chain: interfaces requesting/accepting data + storage

- **The proportion of on/off-chain computation depends on the architecture**

Types of oracles (Mühlberger et al. 2020)

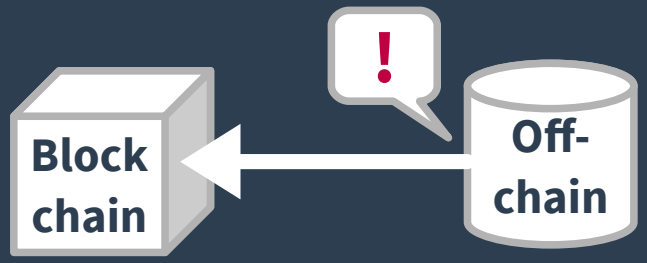
- **Is it the data recipient who requests the data transfer ?**
  - Yes → Pull-based
  - No → Push-based (another entity is responsible)

- **What is the data transfer path ?**
  - Blockchain to outside world → Outbound
  - Outside world to blockchain → Inbound

Example for showcasing oracle types

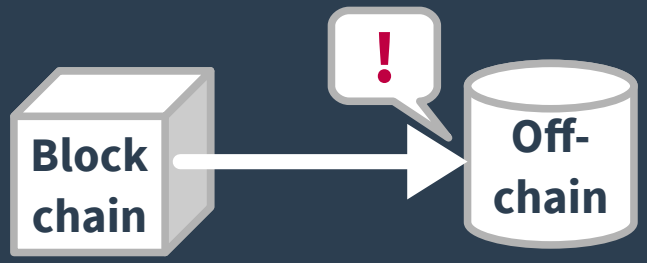
- **Blockchain-based betting platform**
  - Users populate events (e.g. sports matches) on-chain
  - Any interested user can lookup current betting opportunities and bet on them
  - The platform requests the results from a trustworthy oracle system
  - Once a result is known, the platform broadcasts winners of the corresponding bet

Push-inbound oracles



- **Goal: Feed relevant off-chain data to the smart contract**
- **Behaviour:**
  - Off-chain states/events are monitored
  - EOAs (Externally Owned Accounts) send transactions
  - Passive recipient smart contract
- **Example:**
  - Add a sports event on which to bet in the platform

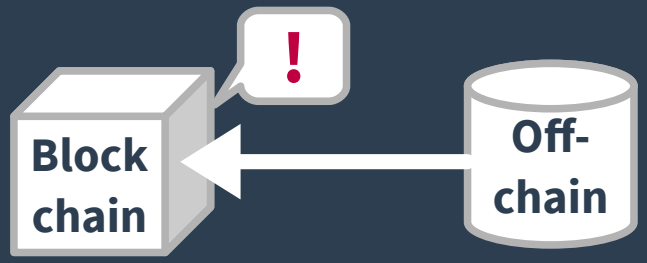
Pull-outbound oracles



- **Goal: Retrieve relevant data from the smart contract**
- **Behaviour:**
  - Queries: off-chain systems read the state of the contract
  - Passive data source smart contract

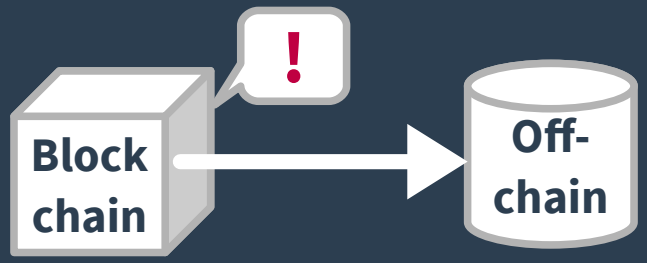
- **Example:**
  - Lookup current standings for a bet

Pull-inbound oracles



- **Goal: Receive relevant off-chain data**
- **Behaviour:**
  - Events (+ requests stored in the smart contract's state)
  - Active smart contract (still needs an EOA to initiate the request's creation)
- **Example:**
  - Request the result of a sports match (if available)

Push-outbound oracles



- **Goal: Signal relevant chain data to offchain systems**
- **Behaviour:**
  - Events (+ requests stored in the smart contract's state)
  - Active smart contract (still needs an EOA to initiate the request's creation)
- **Example:**
  - Broadcast winners of a bet

Real-world oracle systems

- **ChainLink V1: On-chain data aggregation**

  → Provides predetermined feeds of temporal data (e.g. ETH-USD exchange rate)

  - Off-chain nodes periodically fetch new values from an off-chain source
  - Each node pushes its findings on-chain
  - Values are aggregated on a given metric (avg, med, …)
  - Outlier values' nodes are punished

- **Pull-inbound oracle (for periodic updates)**
- **Push-inbound oracle (for value-adaptative updates, e.g. based on relative change)**

Event-based programming

- **In Solidity:**

  event NamedEvent(Type$_1$,T$_2$,T$_3$,...); → declaration in class' body

  emit NamedEvent(param$_1$,p$_2$,p$_3$,...); → call in a function's body

- **Event parameters (up to three) can be *indexed* for easier filtering**

- **In Java/web3j:**

  contractInstance.NamedEventFlowable(...).subscribe(event → { /*event handling*/ })

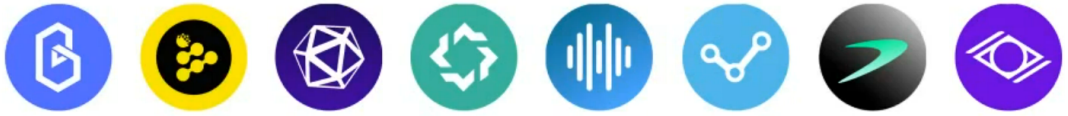
- **In JS/web3js:**

  var namedEvent = constractInstance.NamedEvent();

  namedEvent.watch(function(error, result){ /*event handling*/ });

Standardization

- **EIPs: Ethereum Improvement Proposals**
  - EIP-1154: push-inbound & pull-outbound oracles
    - Abandoned by the community (but still used by iExec)
  - EIP-2362: interface for pull-based price-feed oracles
    - By the Alliance of Decentralized Oracles

      
    - In reaction to the unpopularity of EIP-1154

Interoperability

- **Increasing difficulty**
  - Standard < Manual interop. < Automated interop.

- **Interoperating entities:**
  - smart contracts (SCs) in the same blockchain
  - SCs and oracle systems
  - SCs across distinct blockchains (through oracles)

On-chain VS Off-chain computations

- **Oracle systems have both on- & off-chain parts**

- **On-chain:**
  - Greater security/decentralization → Greater costs/delays
- **Off-chain:**
  - Less costs but loss of blockchain properties

→ Balance of costs and security to be found

On-chain VS Off-chain computations

- **ChainLink V2: Off-Chain Reporting**
  - Rounds of an off-chain BFT algorithm
  - A selected leader generates the new value to report
  - Other nodes agree by signing the report
  - The agreed-upon report is pushed on-chain

Other data acceptance mechanisms

- **Astrea: Voting**
  - A proposition is submitted on-chain
  - Participants check whether the proposition is true or false
  - Participants vote whether the proposition is true or false
  - After a timeout or sufficient votes, voting is closed
  - The minority loses its staked assets

Other data acceptance mechanisms

- **DOS (Decentralized Oracle System): Random selection**
  - **N** participants are randomly dispatched in small groups of size **n**
  - All nodes in a group compute the result of the same task
  - **t** or more out of **n** must sign the same result to attest it
  - This signed result is pushed on-chain
  - Only signers are rewarded