Agenda

1. Few words about Soramitsu
2. Overall description of Iroha
3. Possible use cases
4. Workshop case description
Contacts

Vadim Reutskiy

- Project lead in Japanese office
- Email: reutskiy@soramitsu.co.jp
- Telegram: @vreutskiy
About Soramitsu

• Fintech software development company
• Founded in 2016 in Tokyo, Japan
• 55+ employees

• Worldwide locations:
  • Tokyo (Japan) JP
  • Innopolis (Russia) RU
  • Phnom Penh (Cambodia) KH
  • Astana (Kazakhstan) KZ
  • Zug (Switzerland) CH*

*In progress
Who are we?

Creator of Hyperledger Iroha and an active member of the Linux Foundation’s Hyperledger Project

We are creating a payment system based on Hyperledger Iroha for the central bank and regulator of the Kingdom of Cambodia

We are a proud member of the Japan Blockchain Association
Key features of Iroha

• Command-driven architecture
  • Asset management
  • Identity management
• Support of linux, macOS, Windows environment
• Byzantine fault-tolerant ordering service and consensus
• Role-based access control

• Client libraries, including example apps for iOS, JS (Vue.JS), Android (Java 8)
• Universal peer role and easy scripted deployment with Docker and Ansible
• Multi-signature transactions
Why Iroha?

• Distributed ledger platform for simple use-cases of payments and identity storage.

• Uses fixed set of commands 16 in total (e.g. asset creation, transfers, account creation) and 11 queries (e.g. get account detail, get account balance) in its client API layer.

• Has modular design of its core components: storage, consensus, etc. which allows fundamental changes and opens up possibilities for contribution.
Why Iroha?

• Unique consensus and ordering service algorithms compared to other platforms with BFT class reliability.

• Has modern C++ design which lowers maintenance efforts and increases simplicity of use for API consumers (designers of blockchain-powered applications).

• Successful proof of concept projects include: cross-chain exchange, bank settlements, asset tokenization in the blockchain, digital identity, and supply chain scenarios.
## Comparison of DLT

<table>
<thead>
<tr>
<th>Factor per platform</th>
<th>Hyperledger Fabric (and IBM blockchain)</th>
<th>Hyperledger Iroha</th>
<th>Hyperledger Sawtooth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regional awareness</strong></td>
<td>China! and the rest of the world</td>
<td>Asia, especially Japan</td>
<td>USA</td>
</tr>
<tr>
<td><strong>Differentiators</strong></td>
<td>Extendable deployment architecture, «channels»</td>
<td>Universal peer role, SQL state, linearly scalable consensus</td>
<td>Transaction processors, pluggable components</td>
</tr>
<tr>
<td><strong>Is this a blockchain?</strong></td>
<td>Yes (although it stores invalid transactions)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>gRPC &amp; REST</td>
<td>gRPC</td>
<td>gRPC</td>
</tr>
<tr>
<td><strong>Business logic layer</strong></td>
<td>Smart contracts in Go, Java &amp; Solidity</td>
<td>Commands and queries</td>
<td>Transaction families and processors</td>
</tr>
<tr>
<td><strong>Contributing companies</strong></td>
<td>IBM</td>
<td>Soramitsu</td>
<td>Intel</td>
</tr>
<tr>
<td><strong>BFT</strong></td>
<td>—</td>
<td>+</td>
<td>+?</td>
</tr>
<tr>
<td><strong>In production?</strong></td>
<td>+</td>
<td>+-</td>
<td>+?</td>
</tr>
</tbody>
</table>
# Comparison of DLT

<table>
<thead>
<tr>
<th>Factor per platform</th>
<th>Corda</th>
<th>Hyperledger Iroha</th>
<th>Ethereum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regional awareness</td>
<td>UK, India, USA</td>
<td>Asia, especially Japan</td>
<td>The world</td>
</tr>
<tr>
<td>Differentiators</td>
<td>Scalability</td>
<td>Universal peer role, SQL state, linearly scalable consensus</td>
<td>Turing-complete smart contacts, same codebase for public and private</td>
</tr>
<tr>
<td>Is this a blockchain?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>API</td>
<td>JSON-RPC?</td>
<td>gRPC</td>
<td>JSON-RPC</td>
</tr>
<tr>
<td>Business logic layer</td>
<td>Transactions processors in Kotlin?</td>
<td>Commands and queries</td>
<td>Solidity smart contracts</td>
</tr>
<tr>
<td>Contributing companies</td>
<td>R3</td>
<td>Soramitsu</td>
<td>Ethereum foundation</td>
</tr>
<tr>
<td>BFT</td>
<td>—</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>In production?</td>
<td>+</td>
<td>+</td>
<td>+?</td>
</tr>
</tbody>
</table>
Iroha Architecture

Iroha Peer

Statefull Validator

block

YAC Consensus

agreed block

Block storage

Torii

gRPC

Client

Iroha Peer N

Iroha Peer M

voting

ordered proposal

transaction

Stateless Validator

verified transaction

Ordering Service
Data Structures in the Iroha

- Every user (or entity) has its own account: `{account_name}@{domain}`
- Every account can have several types of assets on board: `{asset_name}#{domain}`
- Every account has a list of roles with corresponding permissions
- Every account can have more than one signatories to perform multi-signature transactions
Commands & Queries in Hyperledger Iroha

Without writing code, asset, identity & supply chain management can be done using prepared commands in the data model. This eases development and increases reliability.
Commands in Hyperledger Iroha

Peer
AddPeer

Domains
CreateDomain

Assets
CreateAsset
AddAssetQuantity
SubtractAssetQuantity
TransferAsset

Account
CreateAccount
AddSignatory
RemoveSignatory
SetAccountQuorum
SetAccountDetail

Permissions
CreateRole
AppendRole
DetachRole
GrantPermission
RevokePermission
Queries in Hyperledger Iroha

**Account**
- GetAccount
- GetAccountAssets
- GetAccountDetail
- GetSignatories

**Transactions**
- GetTransactions
- GetAccountTransactions
- GetAccountAssetTransactions
- GetPendingTransactions

**Assets**
- GetAssetInfo

**Permissions**
- GetRoles
- GetRolePermissions

Diagram:
- **Peer**
  - **Multi assets**
    - Signature
    - Role
  - **Account**
  - **Permissions**
Use Case I: Digital Currency

The main use case for the blockchain based system.

- **Account** corresponds to the **physical user** of the system
- **Asset** corresponds to single **digital currency**
- **Domain** corresponds to particular **Bank or Institution**
- **Quorum** for every account have value 1
Use Case II: Verifiable Claims

Provides trusted distributed source of truth for document verification

- **Account** corresponds to the particular **document**
- **Account details** contains **verification information**
- **Domain** corresponds to the particular **institution**
Use Case III: Decentralized Depository

- Distributed network of trusted digital currency exchange
- **Account** corresponds to the registered user
- **Account details** keeps important information about linked external currencies and logic of synchronization
- **Quorum** is variable and corresponds to amount of nodes inside the system
During the workshop we will create a simple web service, which can be used as middleware between the client application and Iohha Network, as we always do in the real projects.

You can use any client library and any approach, which you prefer, but for simplicity and synchronization I recommend to use prepared wireframe for the Web Service on Kotlin, which uses Iroha Java Client Library.
Workshop steps recommended order

1. Start the Iroha node locally and perform several operations over it using CLI (by following "Getting Started" document).
2. Get familiar with the Client Java Library
3. Obtain the wireframe example web server
4. Implement needed functions by example from the Client Java Library
5. Check that everything works correctly using Postman tool
Thank you for attention!
Now I am open for questions
What we propose

Learn Latest Trends in Fintech Industry
• What is permissioned Blockchain and how to use it
• Learn on real cases but in a safe environment

Open source contribution
• A contribution in known project
• Create web/mobile product based on Blockchain
Project constraints

- Back: Any language that supports Protobuf
- Front: VueJS, React
- Mobile: Native language
- Github / open source
Decentralized permission model

Example of Role Decentralization

- Decentralized RBAC* permission model without single point failure
- Separation of three powers can be created to avoid concentration of authority
- Roles and permissions are set determined in the genesis block

*RBAC=Role Base Access Control
## Comparison of DLT Platforms

<table>
<thead>
<tr>
<th>Hyperledger Fabric</th>
<th>Corda</th>
<th>Hyperledger Iroha</th>
<th>Ethereum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing complete smart contracts</td>
<td>Supports payments and notarization of messages (but through centralized service)</td>
<td>Pre-defined smart contracts called “commands”</td>
<td>Turing complete smart contracts</td>
</tr>
<tr>
<td>Supports privacy via “channels”</td>
<td>Supports privacy via UTXO</td>
<td>Supports privacy via permissions</td>
<td>Private version has been tested by many banks around the world</td>
</tr>
<tr>
<td>Supported by IBM</td>
<td>Supported by R3</td>
<td>SDKs for mobile apps</td>
<td>Supported by the Ethereum open source community</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported by Soramitsu</td>
<td></td>
</tr>
</tbody>
</table>
Proposed technical solution

Shared distributed ledger
- Make a shared distributed ledger with some client information
- Each company only shares part of data to ensure that product is unique