netherlands eScience center

Enabling Collaborative Multi-Domain Applications:

A Blockchain-Based Solution with Petri Net Workflow Modeling and Incentivization

Reggie Cushing, Xin Zhou, Adam Belloum,

Paola Grosso, Tom van Engers and Cees de Laat

IEEE TPS 2023

NWO Netherlands Organisation for Scientific Research

ciena

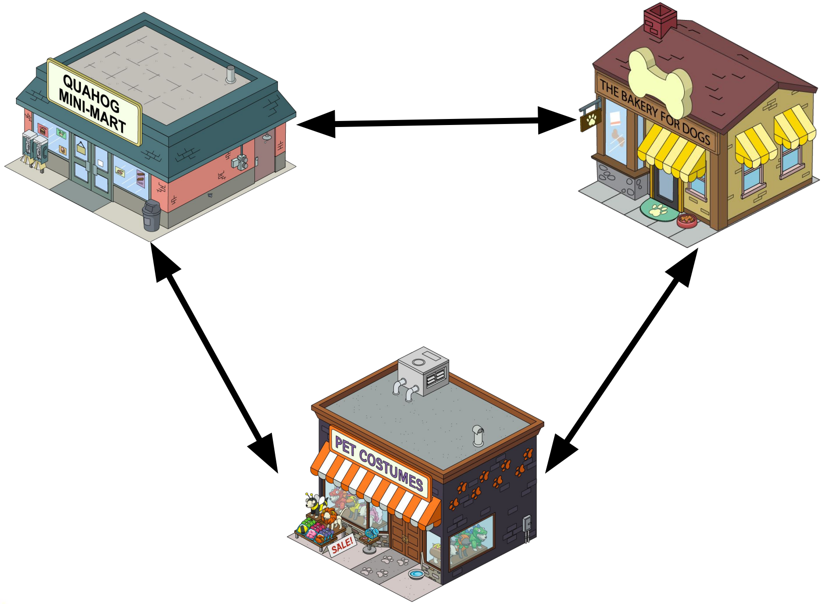
Logistics Data $\mathcal{D}_{\mathcal{L}}4^{\mathcal{L}}\mathcal{D}$ Data Logistics

netherlands eScience center

Data logistics 4 logistics Data project

Creation of innovative solutions that allow stakeholders to agree on how data is **stored**, **accessed**, **shared** and **transformed** in a **controllable**, **enforceable**, **accountable**, **auditable** and goal-oriented fashion.

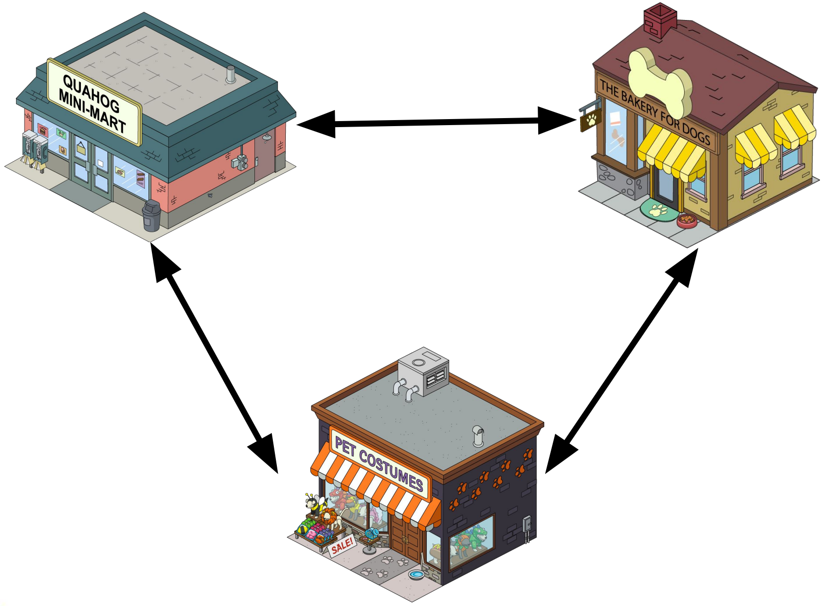
Motivation



Multi-domain applications are characterized by applications such as **workflows** that **cross domain boundaries.**

Examples include **airline**, **healthcare** and **smart cities.**

Digital Data Marketplace

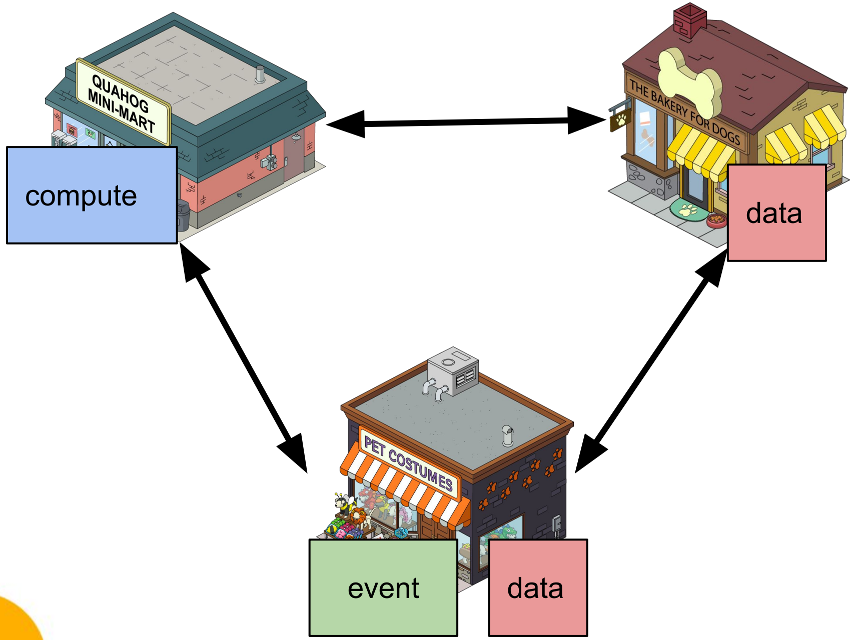


Organizations **agree** to **share data** and **compute** because of mutual benefit to all parties.

We refer to such a platform as a **Digital Data Marketplace** or DDM

netherlands eScience center

Digital Data Marketplace

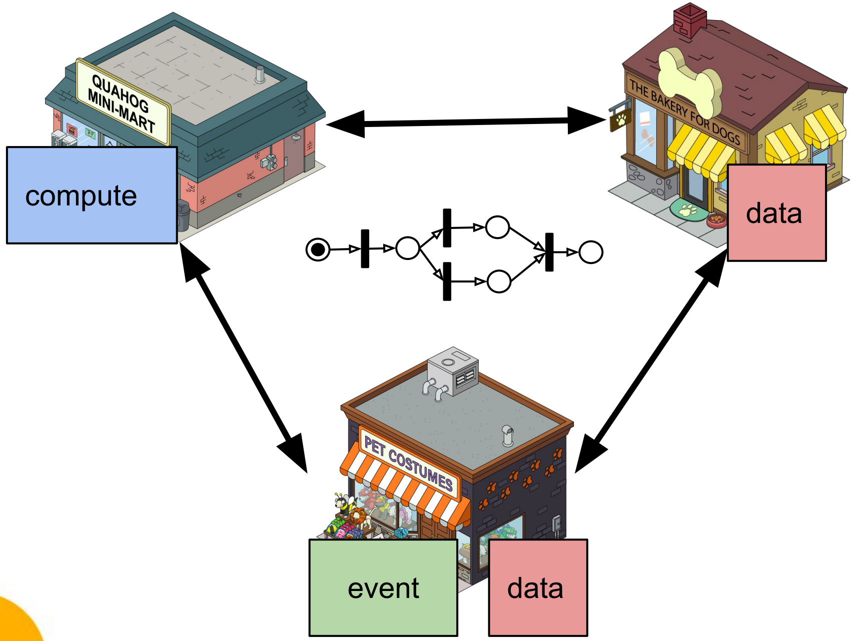


The question we address here is:

How to coordinate **workflows** without a *trusted* 3rd party and **incentivize** collaboration in a multi-domain setting?

netherlands eScience center

Digital Data Marketplace



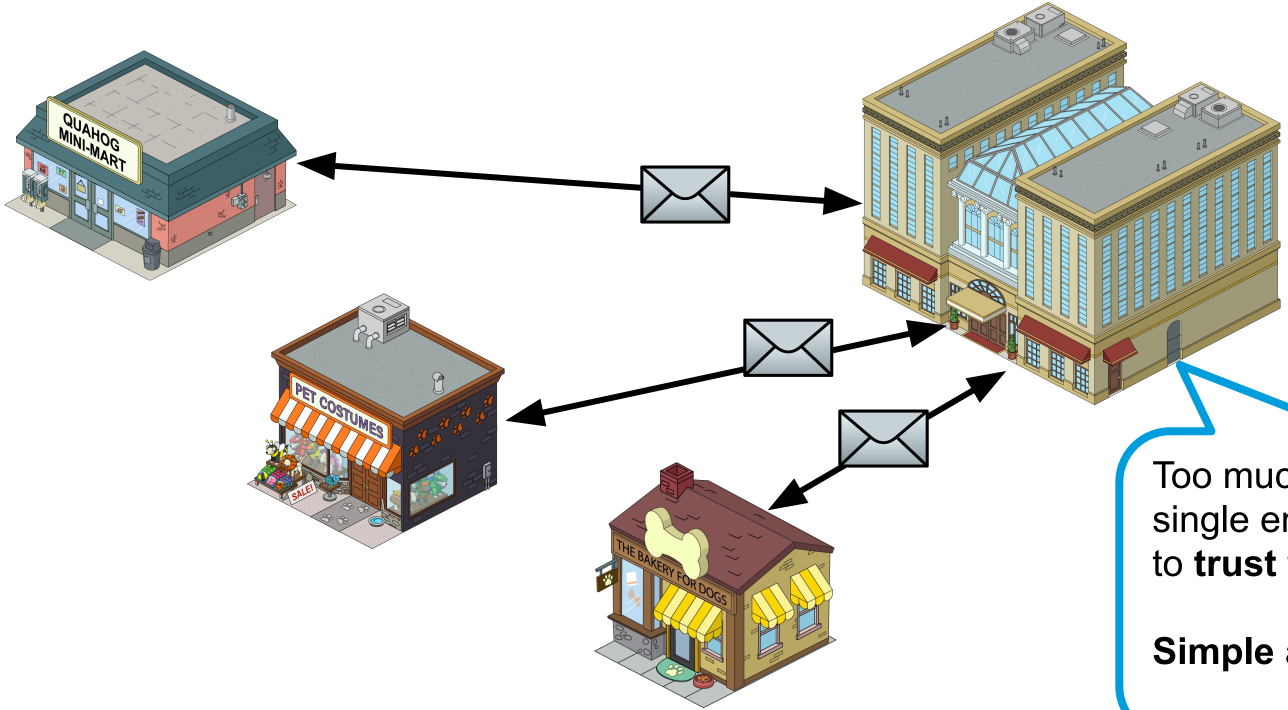
What we propose is:

A Petri net based coordination layer on top of a blockchain layer.

Overarching challenges of a DDM

- Multi-domain **Policy Enforcement** and **Auditing**
- Multi-domain **Identity** and **Trust** Management
- Multi-domain **Application/Workflows** Management
- Multi-domain Collaborative **Infrastructure**

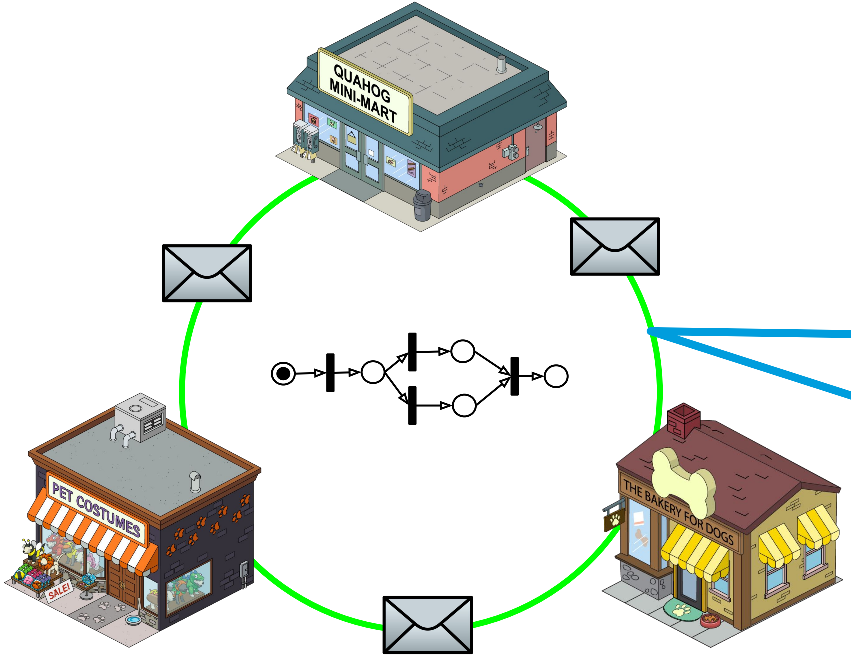
Coordination - Centralized Authority



Too much power given to a single entity. Everyone needs to **trust** the authority.

Simple and **efficient** design.

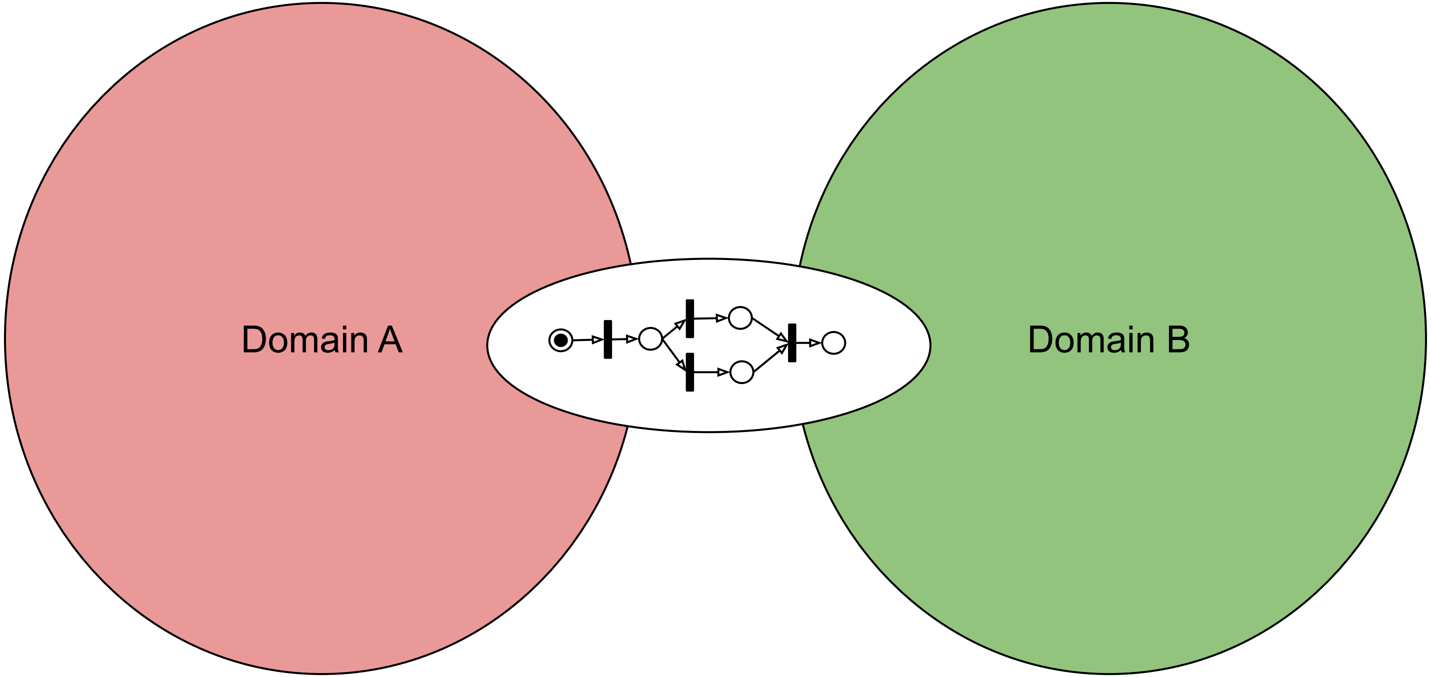
Coordination - Decentralized Authority



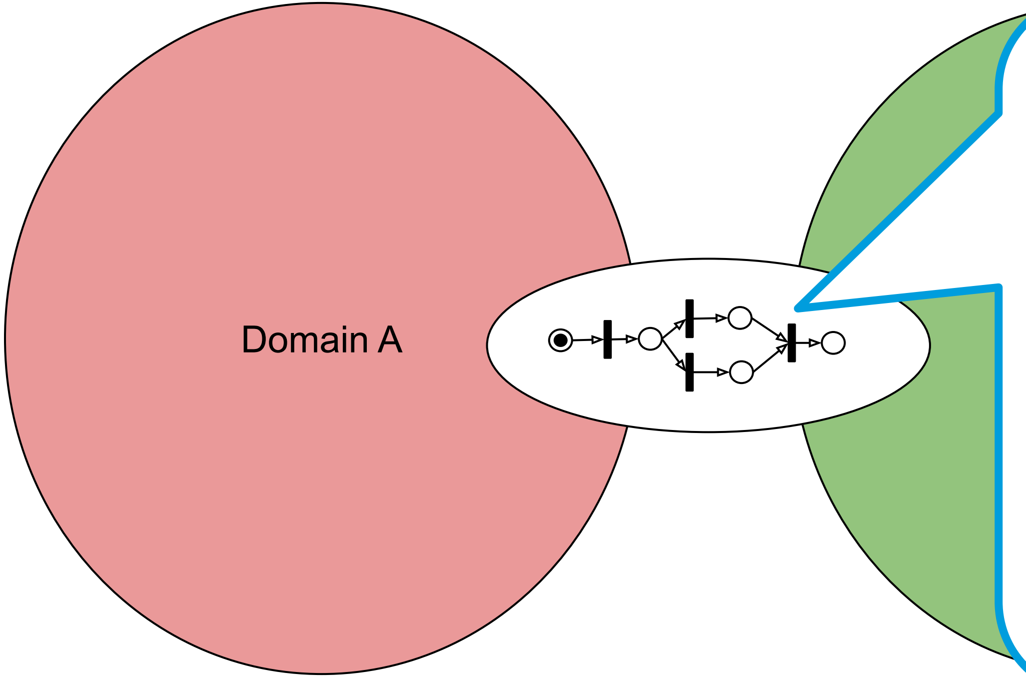
No central authority, coordination is distributed, **trust** is distributed.

Complex and **cumbersome** design.

Shared states, shared truth



Shared states, shared truth



Encoding multi-domain **agreement** as a Petri net.

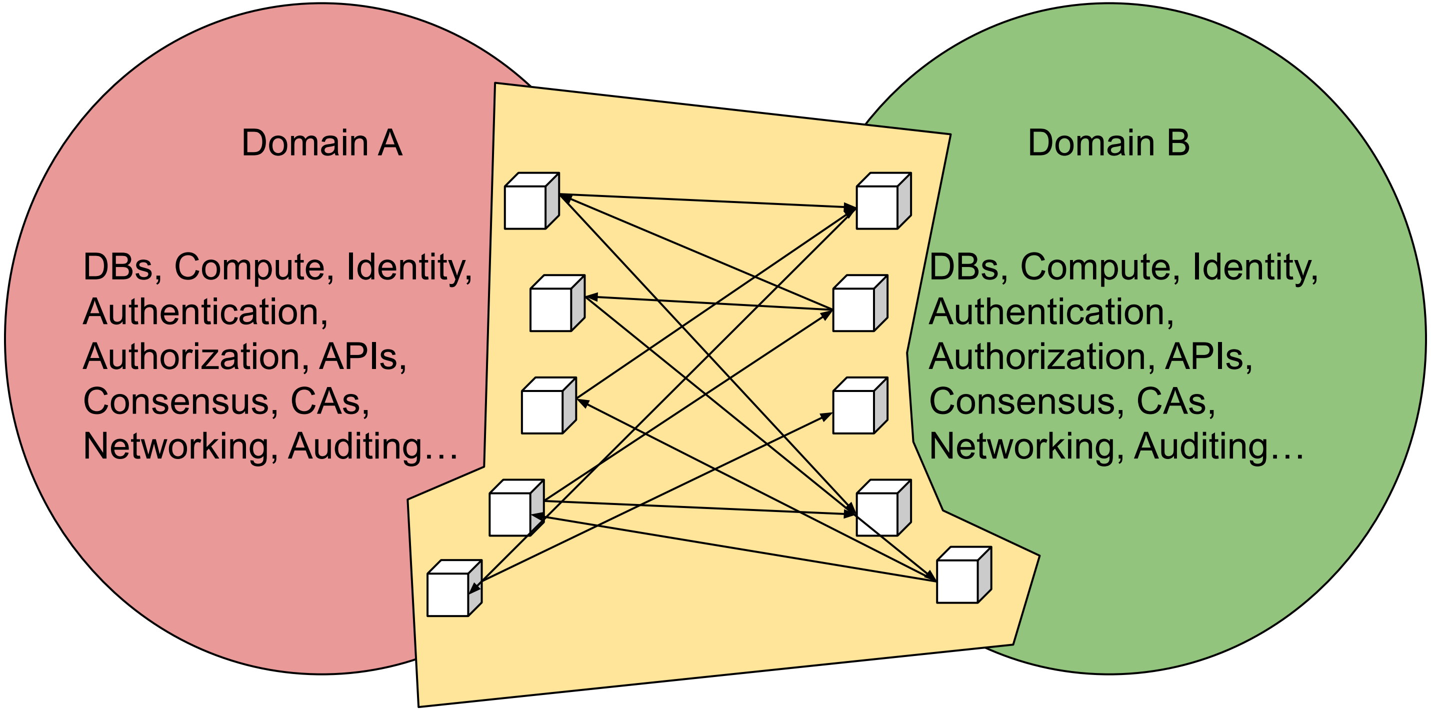
Petri net markings denote the current **state** of the agreement.

The Petri net models the **obligations** (tasks) at different states.

What needs to be done **when** and by **whom.**

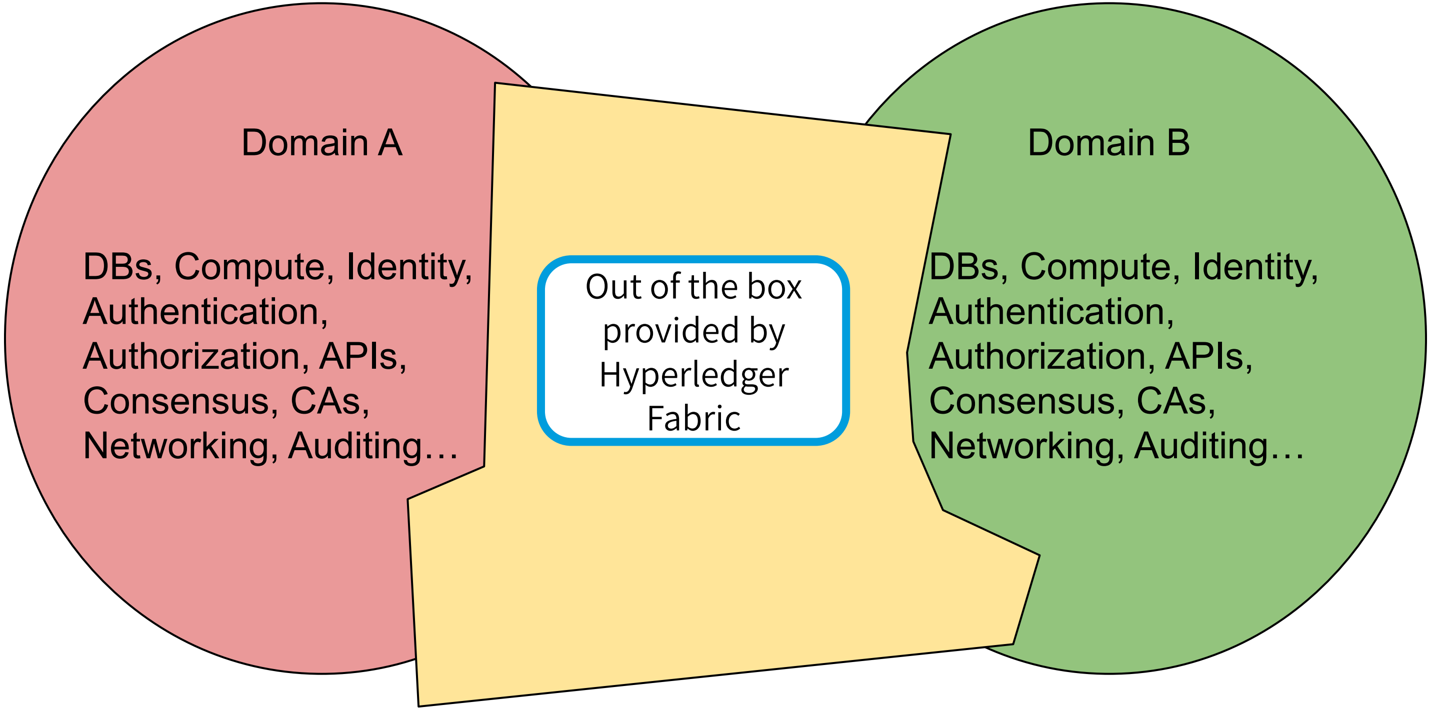
netherlands eScience center

Operationalize components

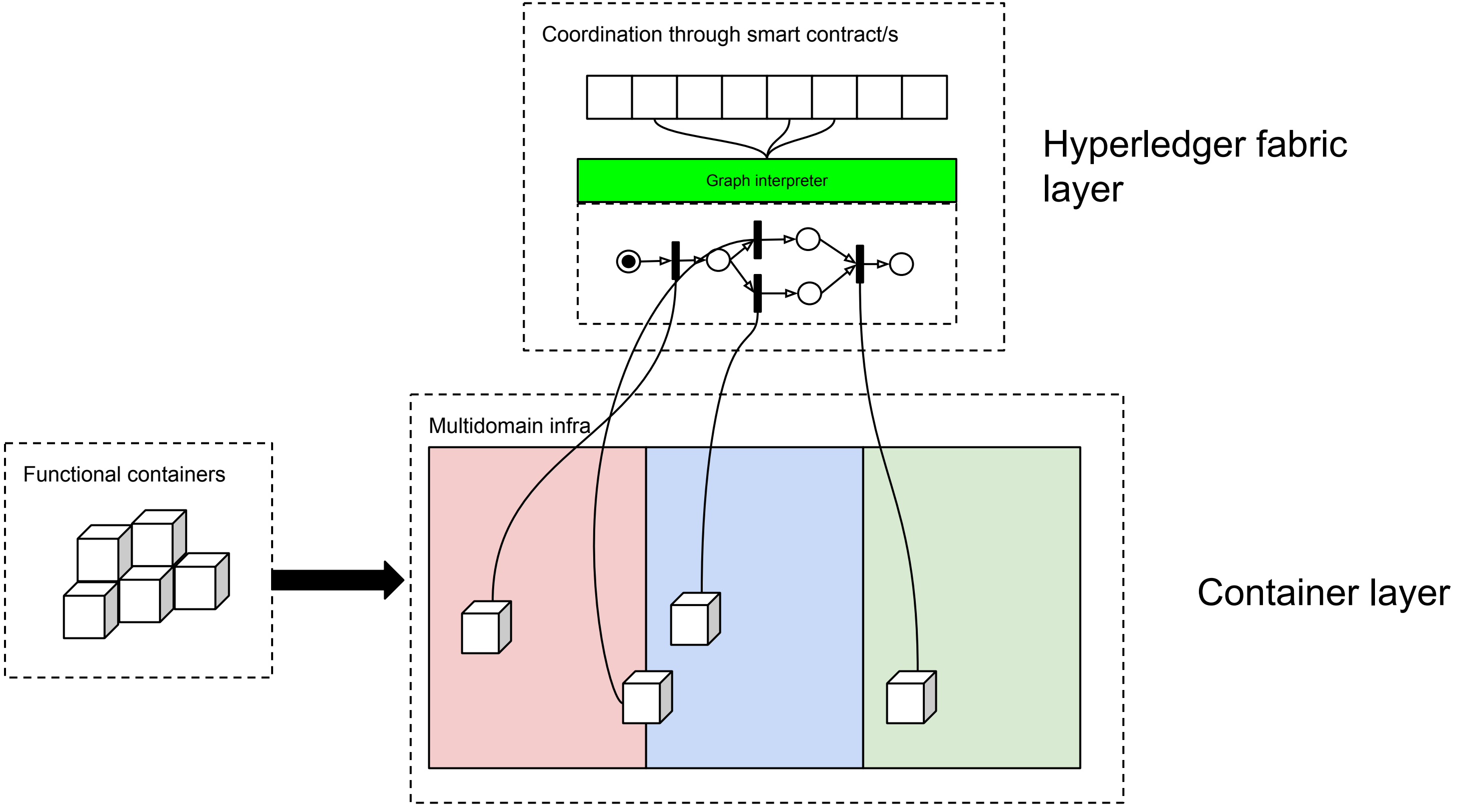


netherlands eScience center

Hyperledger Fabric

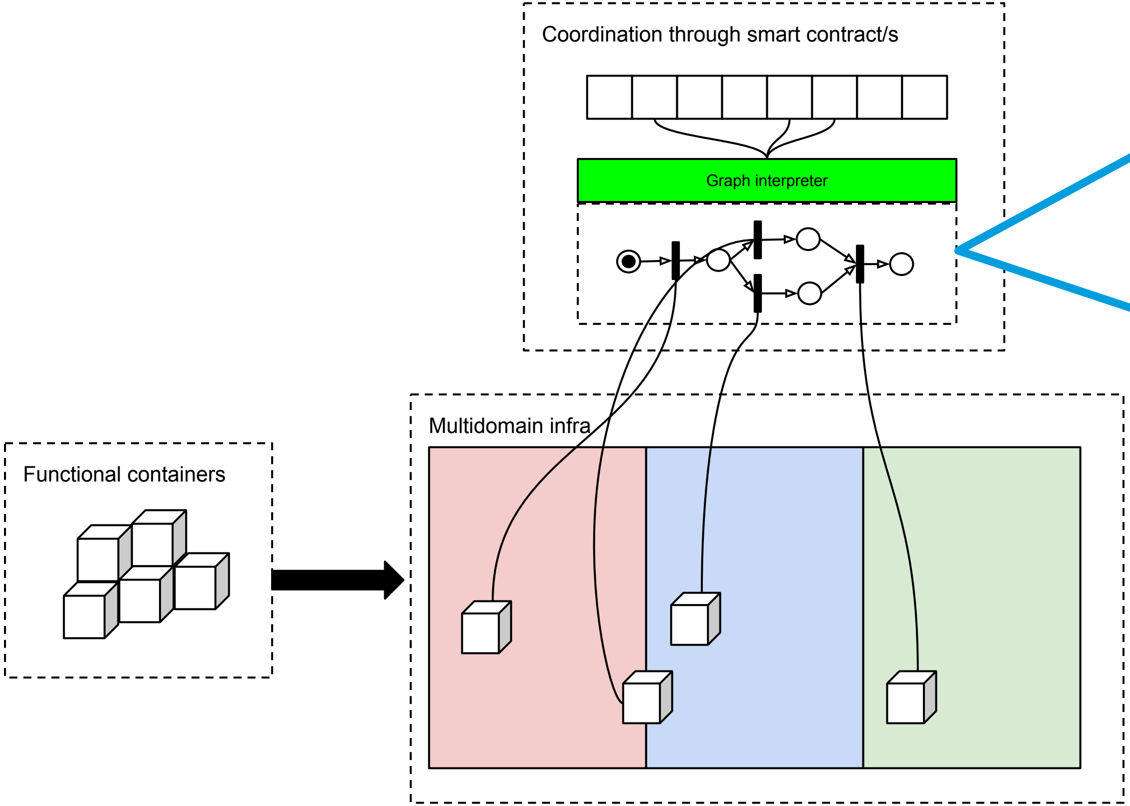


Petri net workflow on blockchain



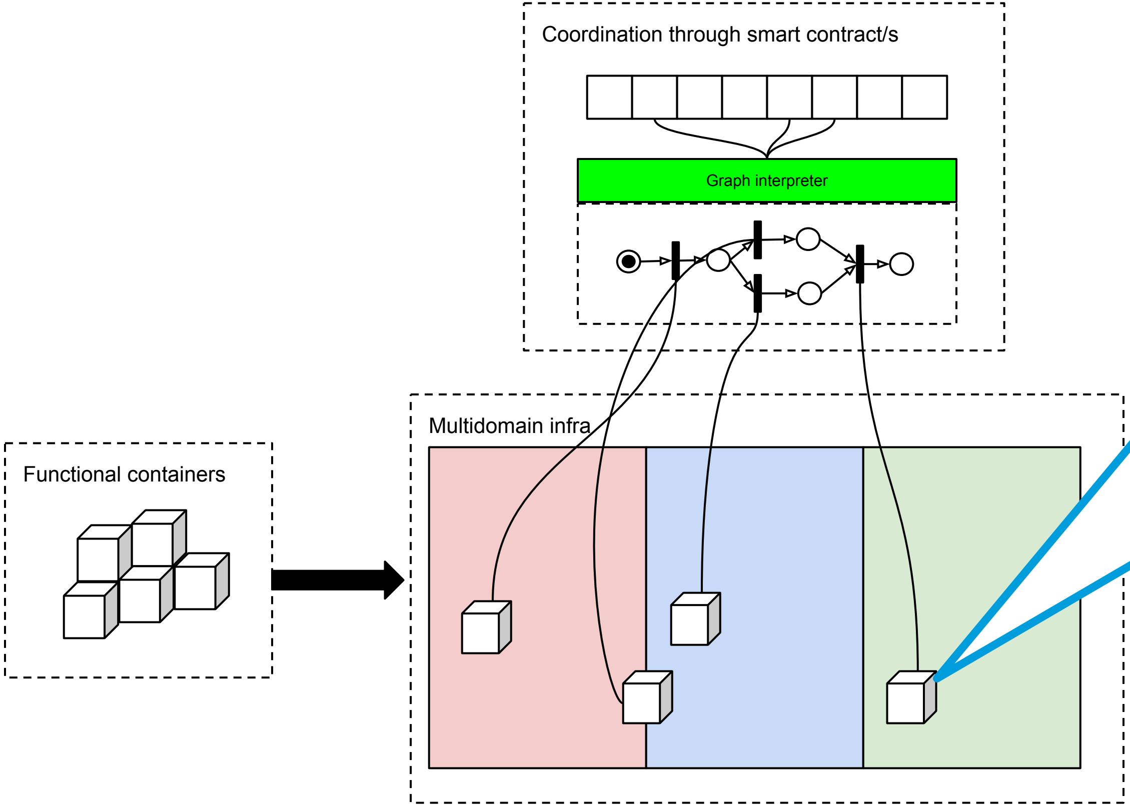
netherlands eScience center

Petri net workflow on blockchain



- **Generic** Petri net **interpreter** running on a blockchain i.e. every peer is running the executor.
- A task needs certain amount of tokens to **fire**
- Blockchain transactions **move** tokens.
- When a task has enough input tokens it will **fire** which in turn generates blockchain events.

Petri net workflow on blockchain



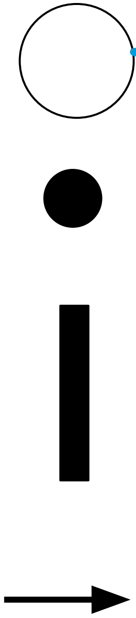
- Containers monitor the ledger for **transition firings** to trigger a process inside a container (the task).
- The container will make blockchain transactions to signal the task is completed and Petri net **places** are updated with **tokens.**

Highlevel Fabric primitives

- **Participants**
  - Users with an x509 certificate given by one of the organizations CA.
- **Assets**
  - User defined data structs owned by participants.
- **Transactions**
  - Read/Write to ledger.
  - Change asset ownership.
- **Chaincode**
  - Javascript/go/java programs that run on Fabric network to implement **smart contracts.**

netherlands eScience center

Petri net elements to Fabric mapping



- A place receives is a placeholder for tokens.
- It is owned by a domain.
- Represented as an asset.

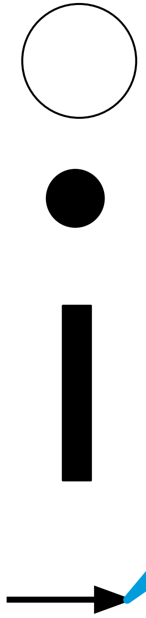
Petri net elements to Fabric mapping

- Tokens are passed between places.
- They are owned by domains.
- Ownership is transferable.
- They are represented as assets.
- Tokens are typed e.g. data, authorization.

Petri net elements to Fabric mapping

- Transitions model off-chain actions.
- Transitions are what move tokens between places.
- They are represented as an asset.
- They are owned by domains.
- They map to container functions.
- A transition fire implies a container function execution.

Petri net elements to Fabric mapping



- Edges connect the Petri net elements.
- The list of edges are represented as an **asset**.
- They indicate the required input tokens for a transition and the number of output tokens.
- A transition (container function) fires when the required input tokens are ready.

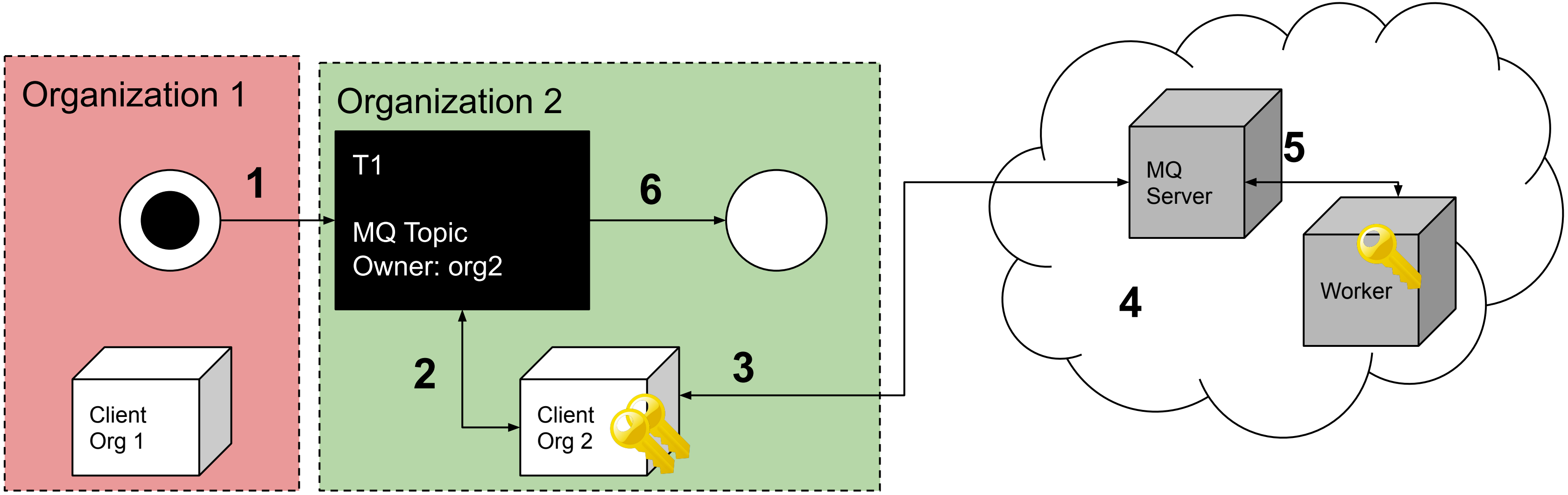
Petri net contract API

- Create|Update|Delete Token
- Create|Update|Delete Place
- Create|Update|Delete Transition
- Create|Update|Delete Net
- AcceptNet
  - Organization sign the Petri net.
- PutToken
  - Moves a token, checks for transitions to **fire.**

Client interface application

- Off-chain interface
- Builds wallet
  - User keys from organization CA.
  - Enrolls user.
- Connect to a Fabric node
- Calls contract API.
- Listen for ledger events.
- Calls **container** functions
  - Signs command with wallet keys

Container functions

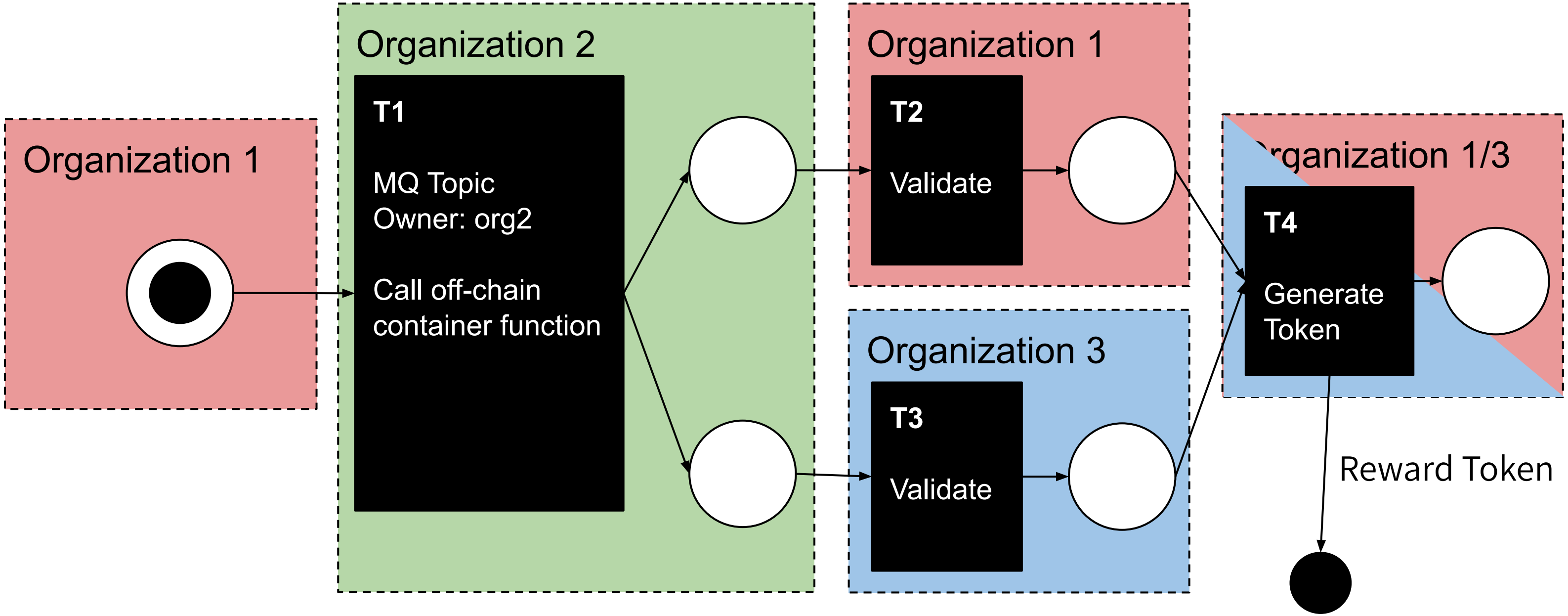


1. **T1** fires, event generated on blockchain.
2. **Client Org 2** reads event; is owner of **T1** (has keys).
3. Encrypts and signs message with **wallet** keys.
4. Publishes message on message queue server.
5. Worker reads message. Decrypts using wallet public key, (**white list**) performs action.
6. **Client Org 2** updates ledger.

Incentivization and reward

- Off-chain (**container functions**) tasks are hard to track.
- **Token economy** to modify behaviour.
  - Reward correct off-chain execution of tasks.
- Implemented as part of the Petri net.
  - Peer audits validate the off-chain task.
  - On agreement an authorization token is generated.
  - This token is exchanged to invoke contracts.
- The more you help the more you can ask for help

Incentivization and reward



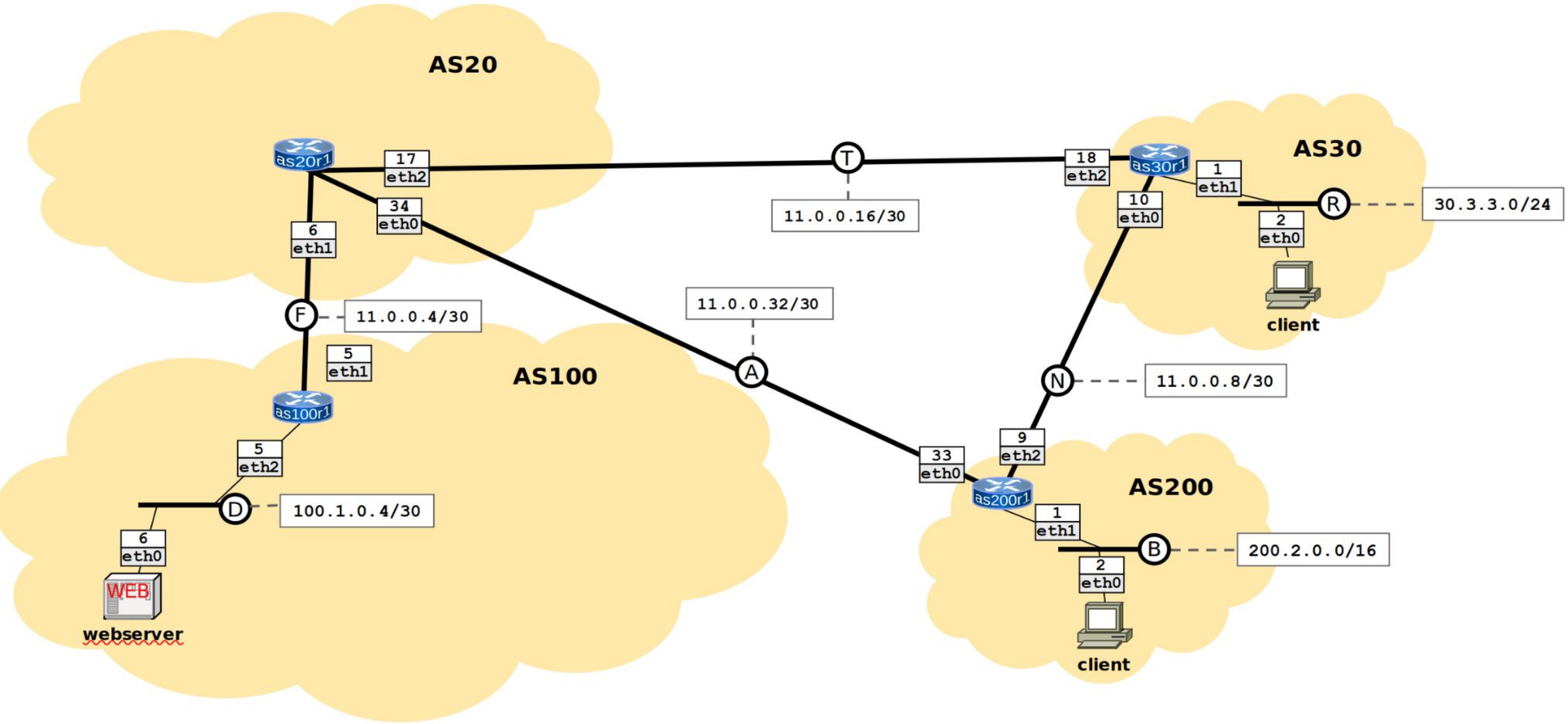
Use Case

- Model collaboration between Internet domains.
- We emulate a simple Internet with 4 ASs.
- We create a Hyperledger across the 3 domains.
- The application says that:

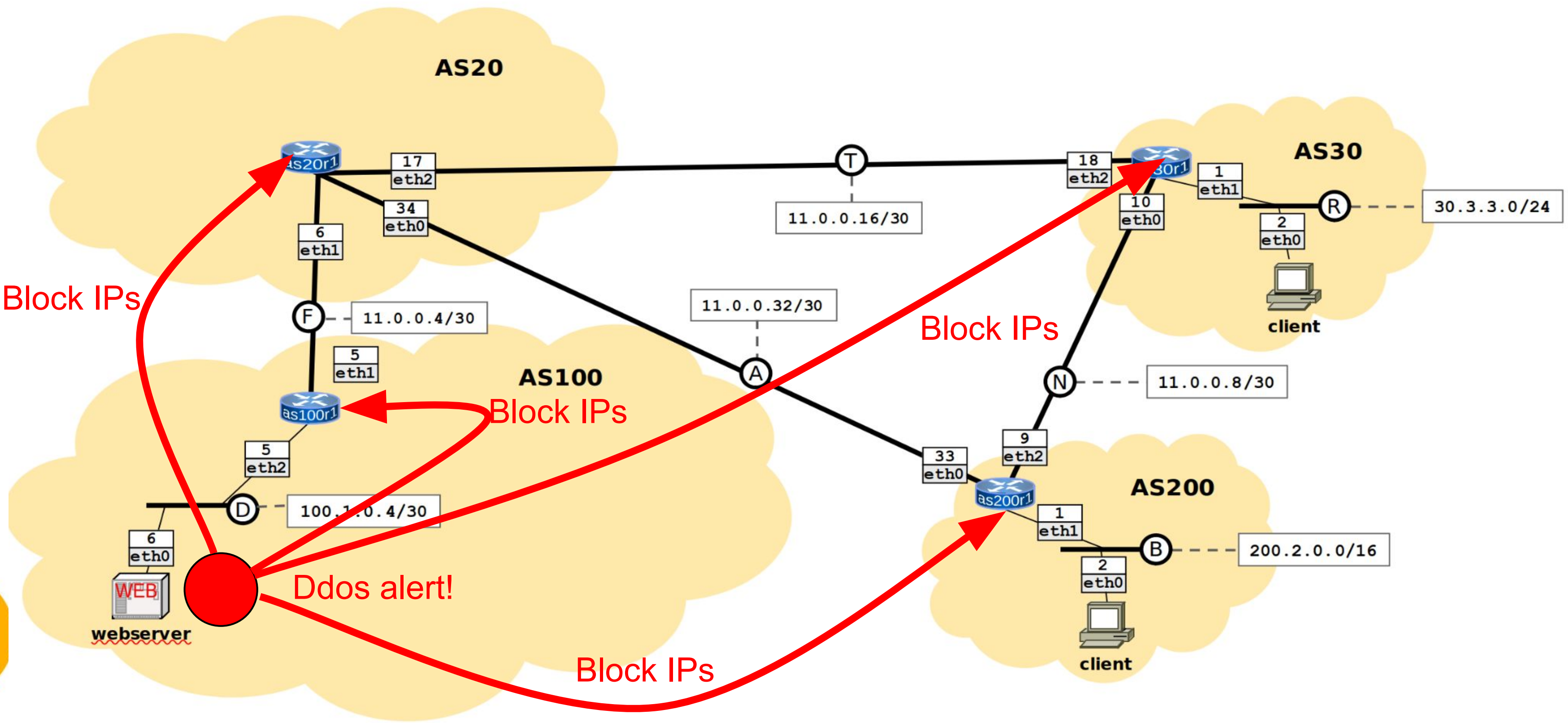
  "If any domain detects a DOS it ask others for help. The others are obliged by contract to block offending IPs."
- This is encoded as a Petri Net using smart contracts.

netherlands eScience center

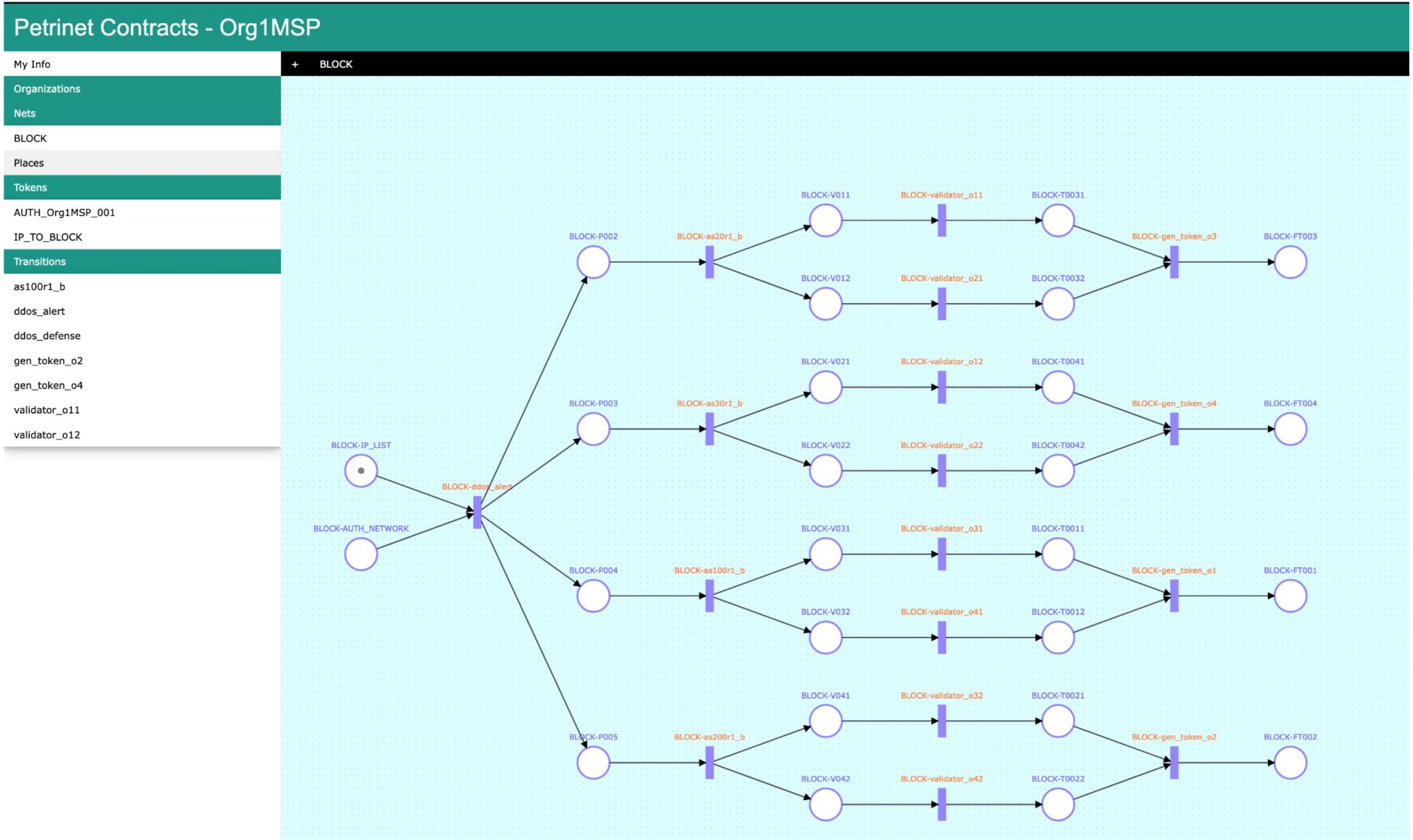
Use Case - network emulator



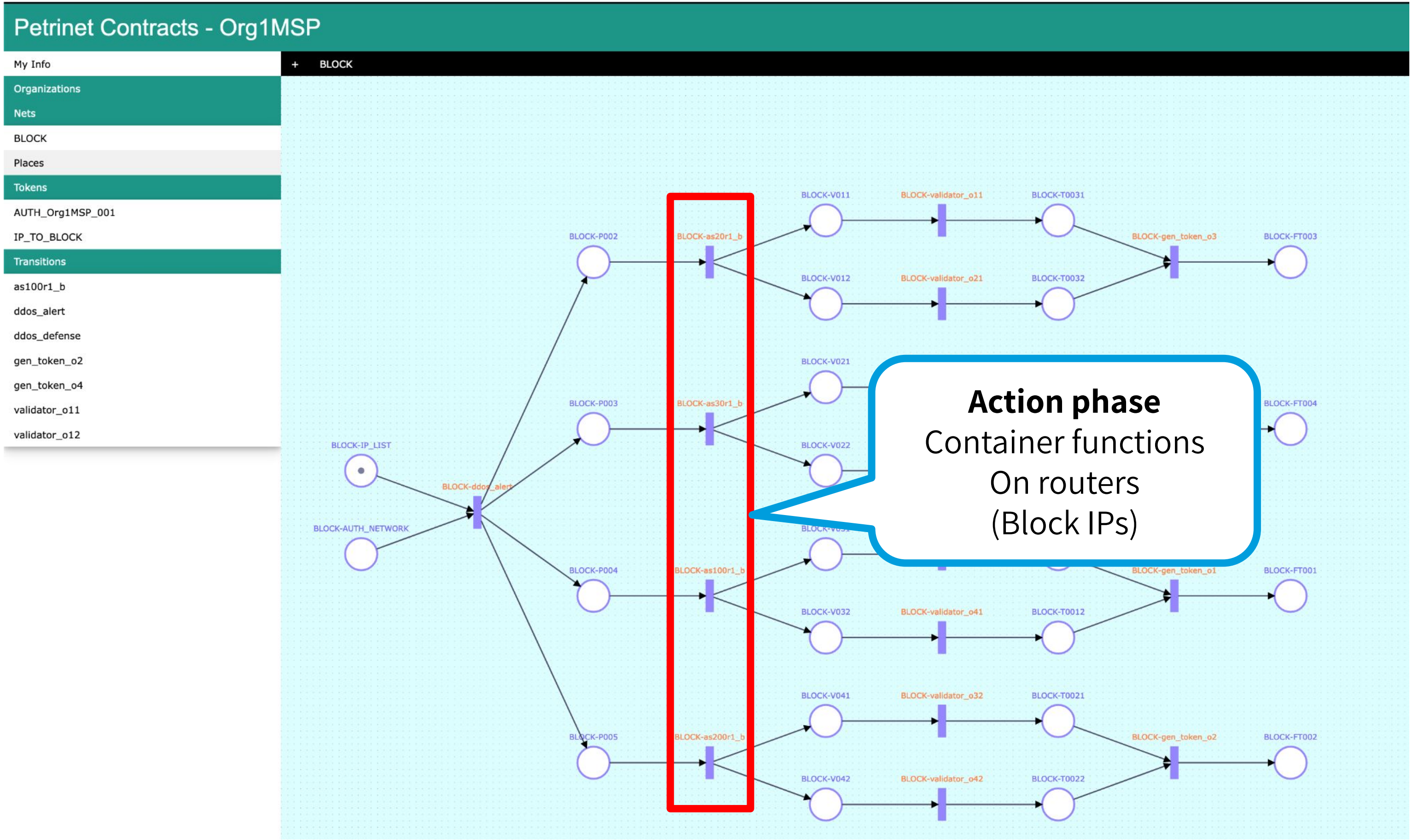
Use Case - network emulator

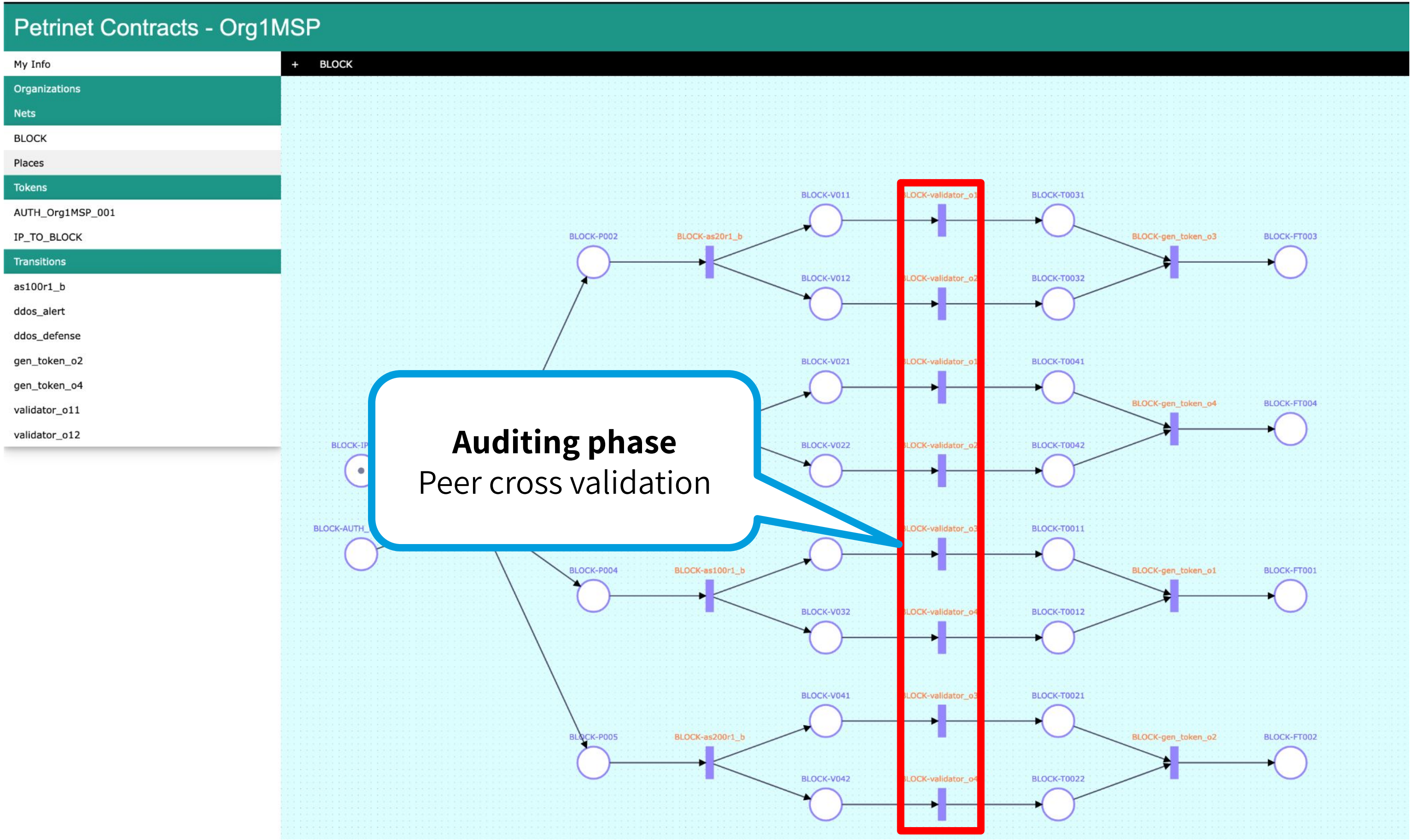


netherlands eScience center



netherlands eScience center

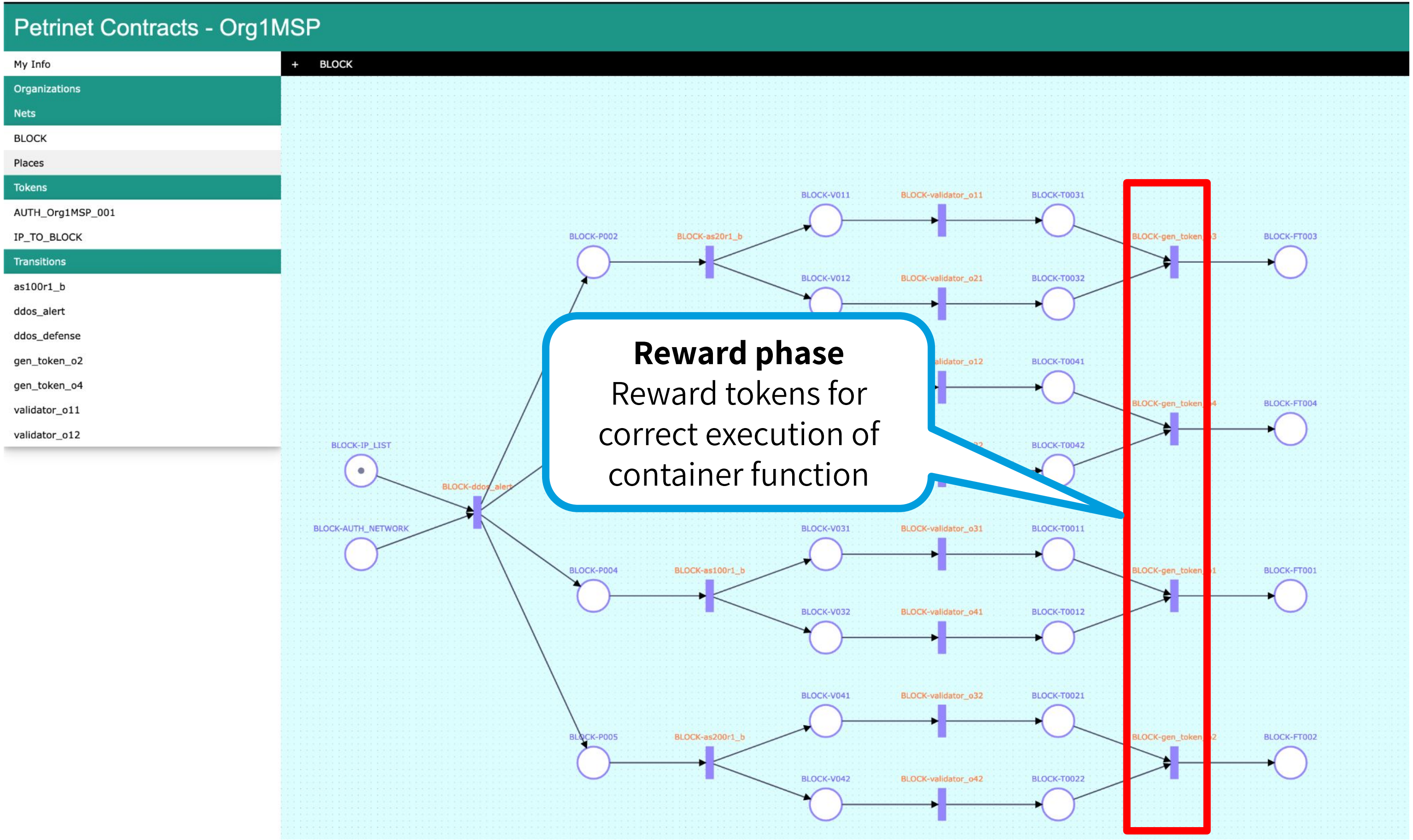




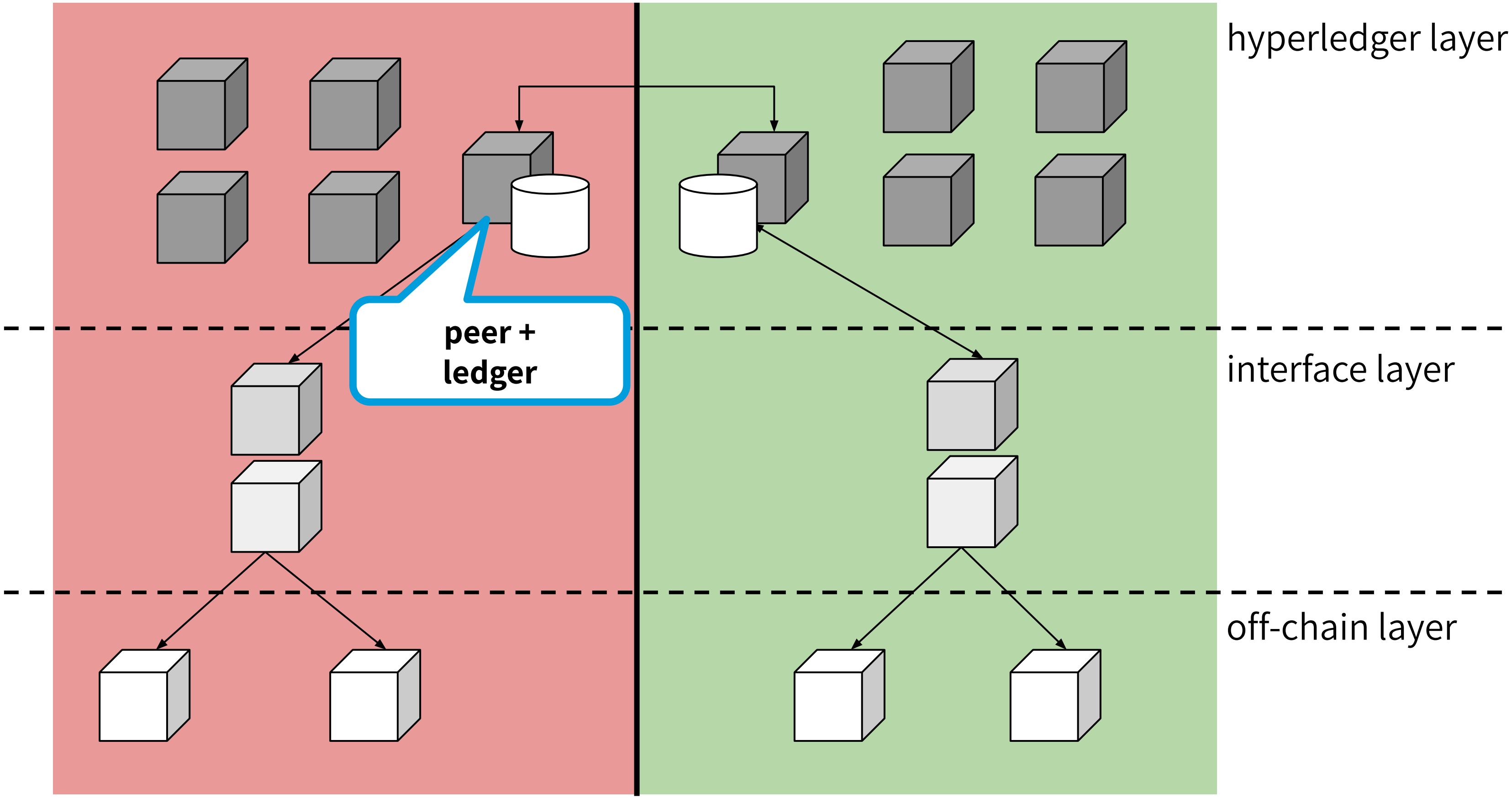
Auditing phase

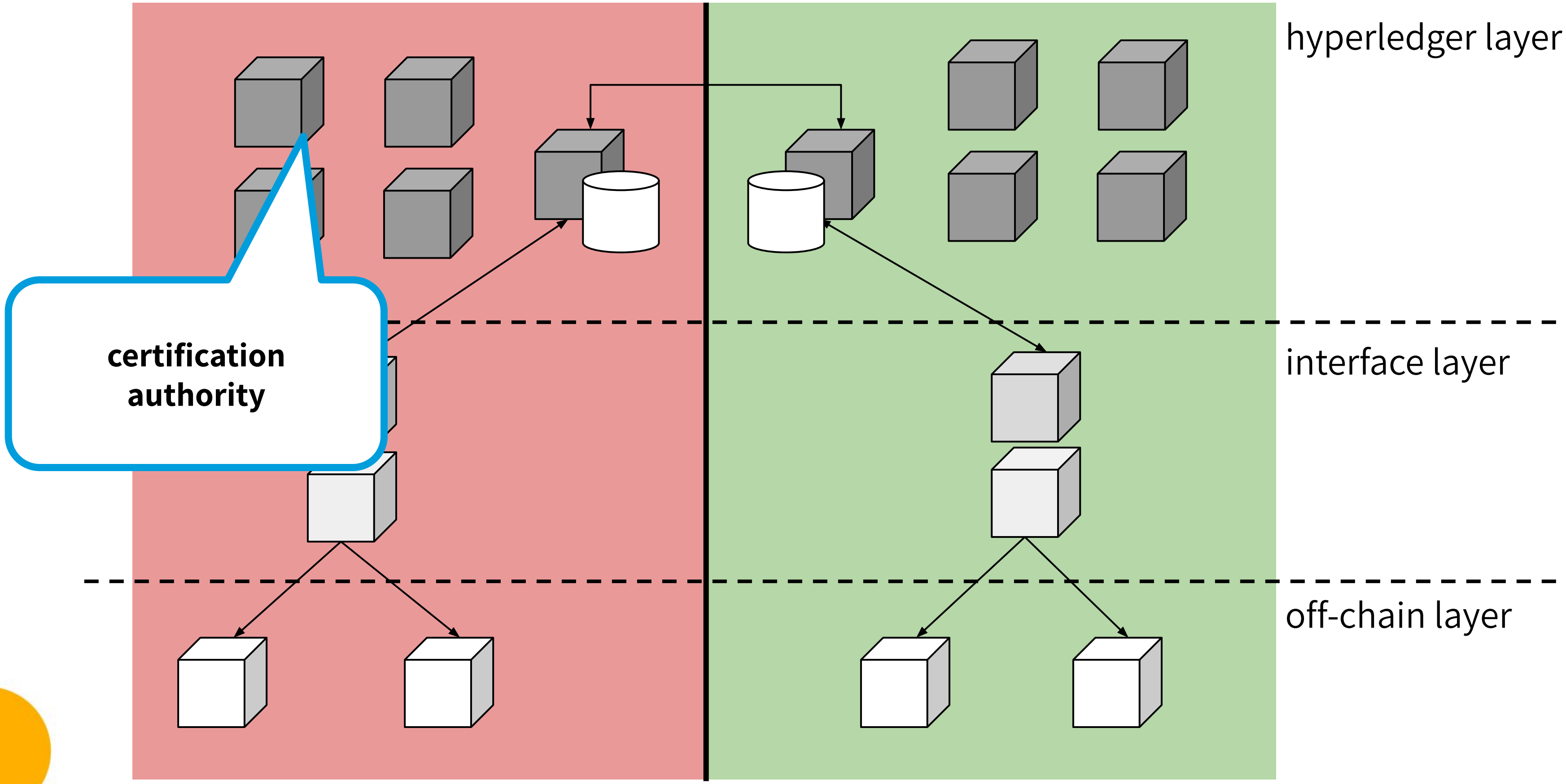
Peer cross validation

netherlands eScience center

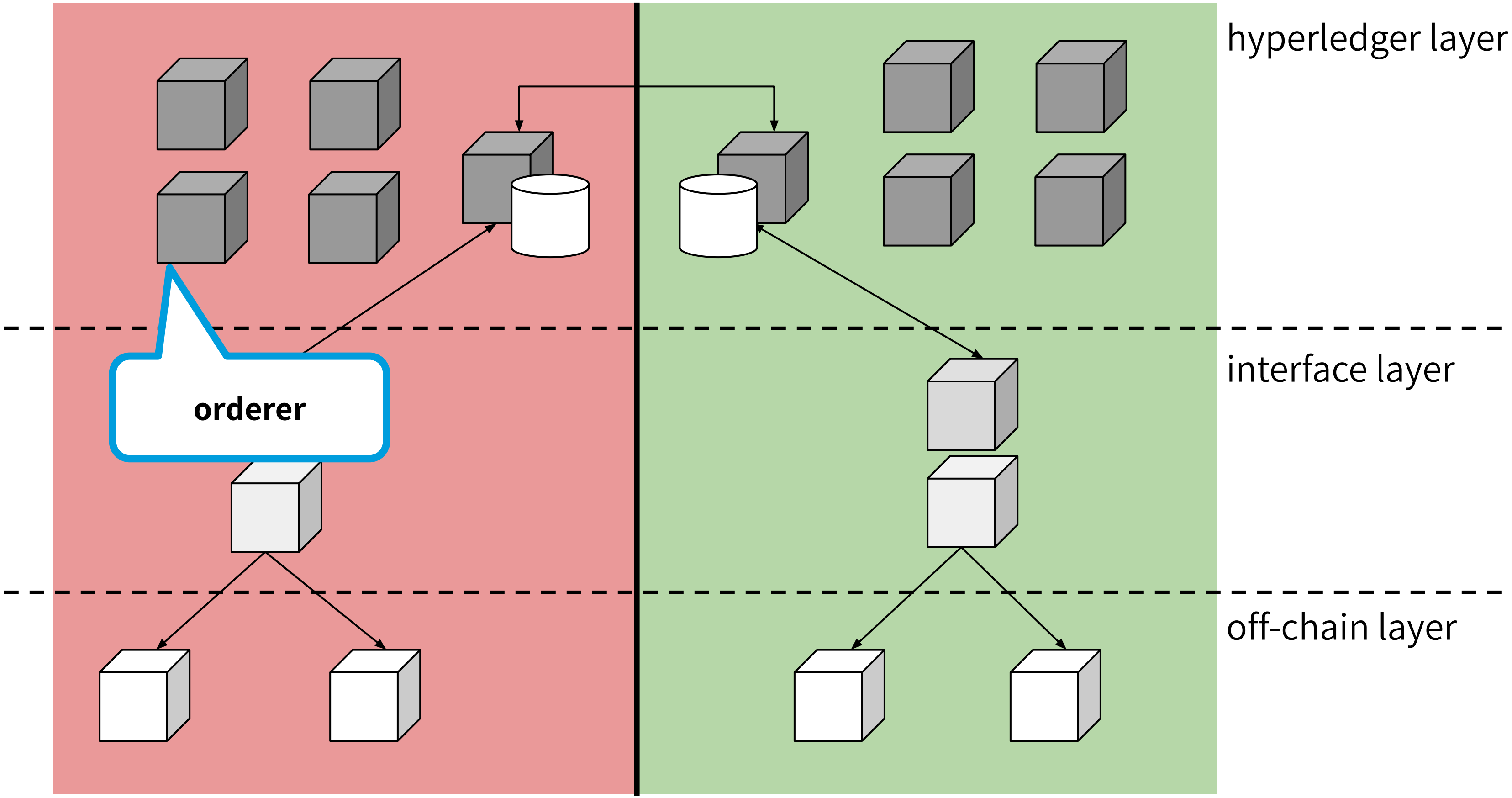


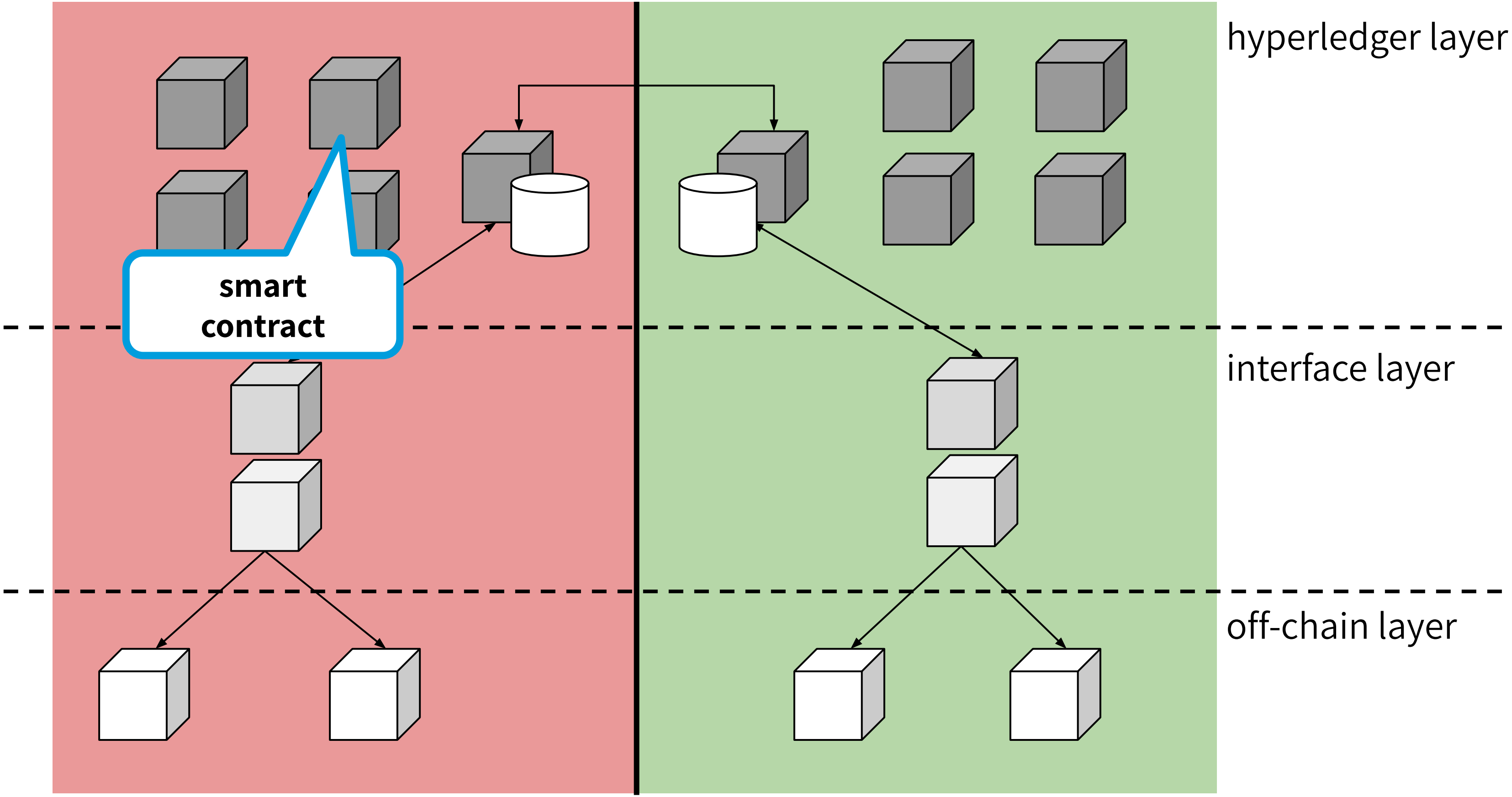
netherlands eScience center



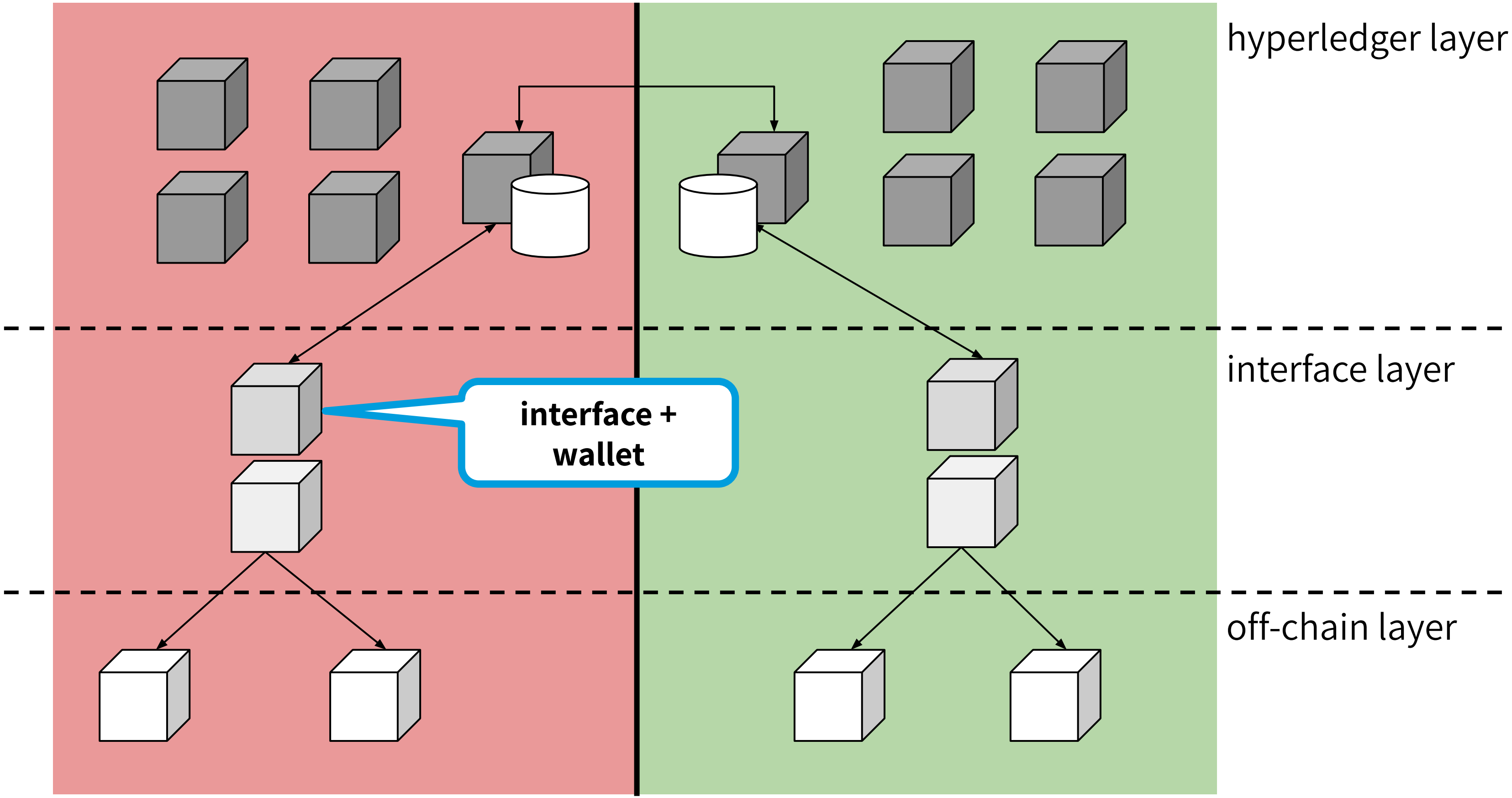


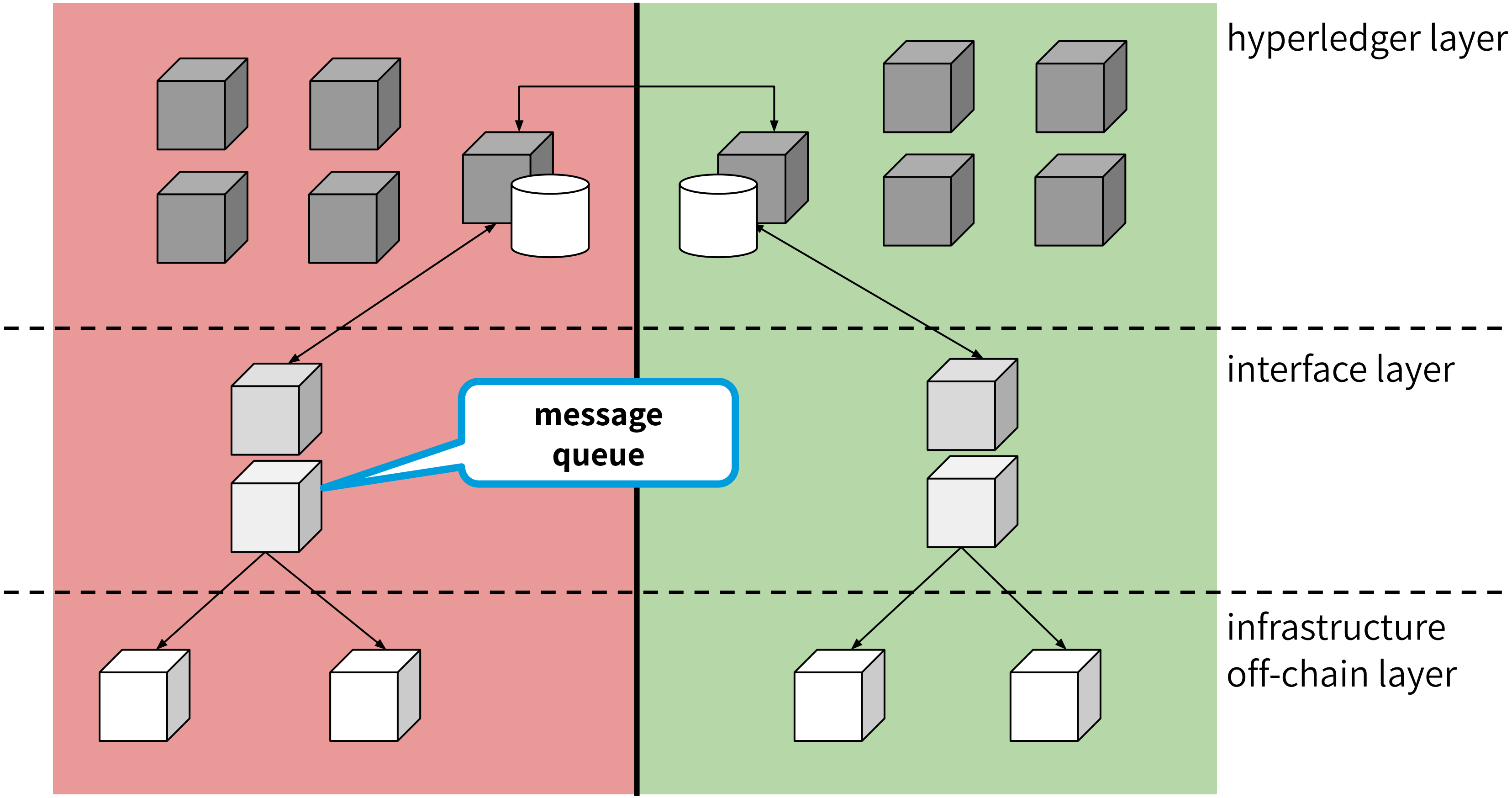
hyperledger layer
certification authority
interface layer
off-chain layer

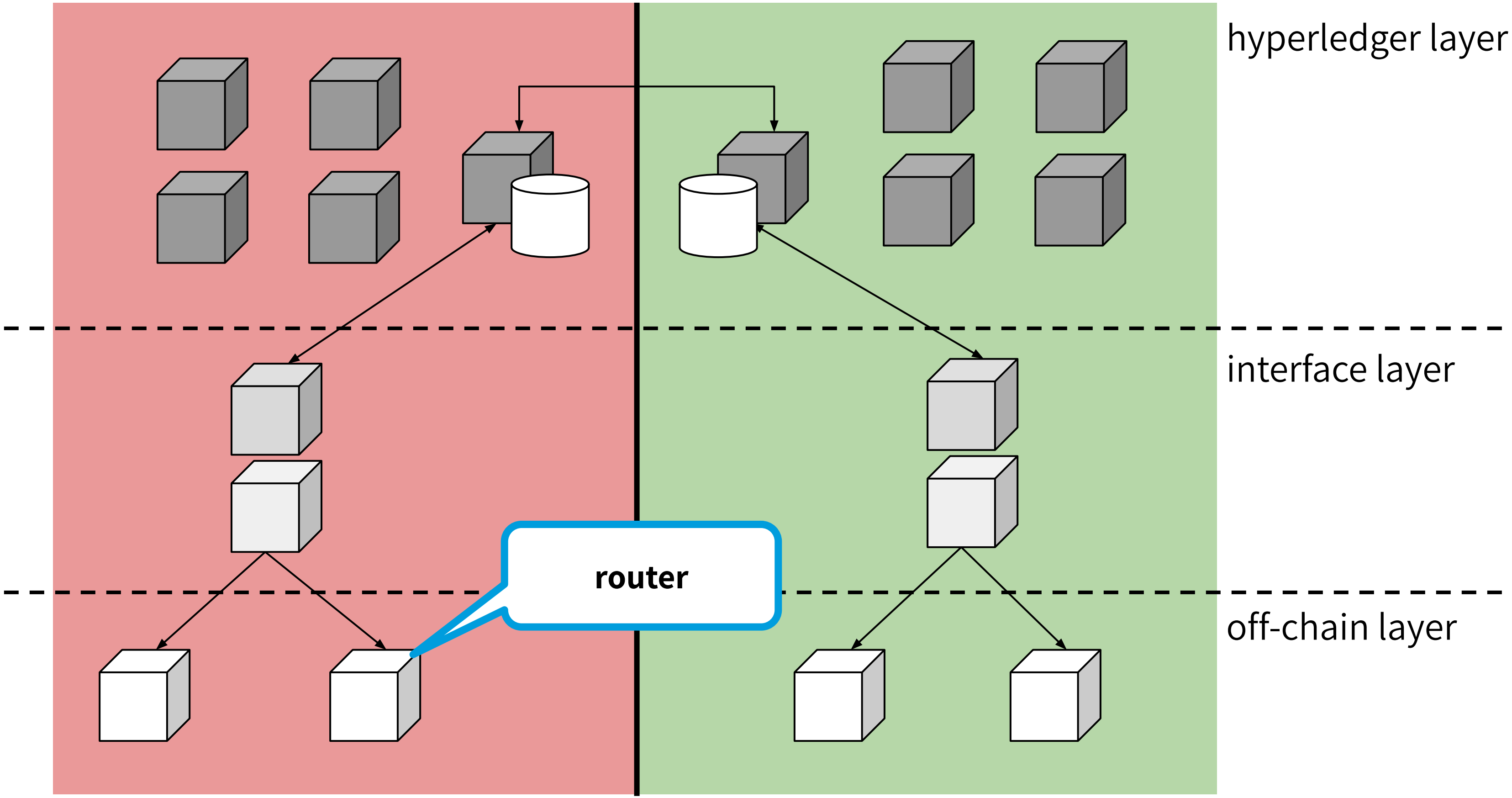

orderer
hyperledger layer
interface layer
off-chain layer


hyperledger layer
smart
contract
interface layer
off-chain layer

netherlands eScience center




hyperledger layer
message
queue
interface layer
infrastructure
off-chain layer


hyperledger layer
interface layer
off-chain layer
router

Remarks

- Decentralizing trust is complex
  - A simple use-case is already complicated
- Petri nets are not user friendly
  - Intermediate modeling
  - Translating other workflows such BPM to Petri nets
- Container functions need to be audited
  - Incentivization requires peers to validate off-chain functions
  - Per use-case validation functions
- Still not privacy can be improved
  - Transactions expose data to other organizations.

netherlands eScience center

Future privacy considerations

- Zero knowledge asset transfers
  - Adding **privacy** at the transaction level.
  - Not disclosing data to whom it is not meant.
  - Role of **auditor** as a participant.
  - Auditor assigned to organization **only** sees relevant transactions.

Conclusions

- Petri nets on blockchain provide an abstraction
  - Model contracts vs hard coding
  - Validate Petri net against higher level workflow e.g. BPM
- Chaincode programming is a different paradigm.
  - Logic is modelled as reads and writes to a ledger.
  - Data is replicated on all peers.
  - Execution is done multiple times
  - Execution only happens as a reaction to a user call.

netherlands eScience center

Reach out

Reggie R.Cushing@esciencecenter.nl

Xin X.Zhou@uva.nl

Github

github.com/dl4ld/petrinet

Dl4lD site

DL4LD.nl