Operations Smart Contract (OpsSC) for Hyperledger Fabric v2.x: Smart contract-based system operations for blockchain-based systems

Financial Innovation Lab, R&D Division, Hitachi America, Ltd.
Tatsuya Sato and Taku Shimosawa

Summary

- *Operations Smart Contract (OpsSC)*
  - Goal: Establishing decentralized system operations across multiple organizations for blockchain-based systems
  - Idea: Define a system operational workflow as a smart contract, each organization (admin / agent program) operates their own nodes according to the smart contract
  - Value: inter-organizational operations can be performed
    (1) without relying on decisions by a specific organization
    (2) with uniform procedure / configuration parameters
    (3) efficiently
- We have developed OpsSC for Hyperledger Fabric v2.x
  - This helps make typical end-to-end operational workflows more efficient
    - Currently, for typical chaincode ops (deploying etc.) and channel ops (adding orgs etc.)
  - This is available on https://github.com/satota2/fabric-opssc
  - We would like to start this as a "hyperledger-labs" project

Concept of OpsSC (for blockchain-based system in general)

[1] Smart-Contract Based System Operations for Permissioned Blockchain, BSC 2018, p.6

[2] Design and Evaluation of Smart-Contract-based System Operations for Permissioned Blockchain-based Systems, arXiv:1901.11249, p.11, 2019

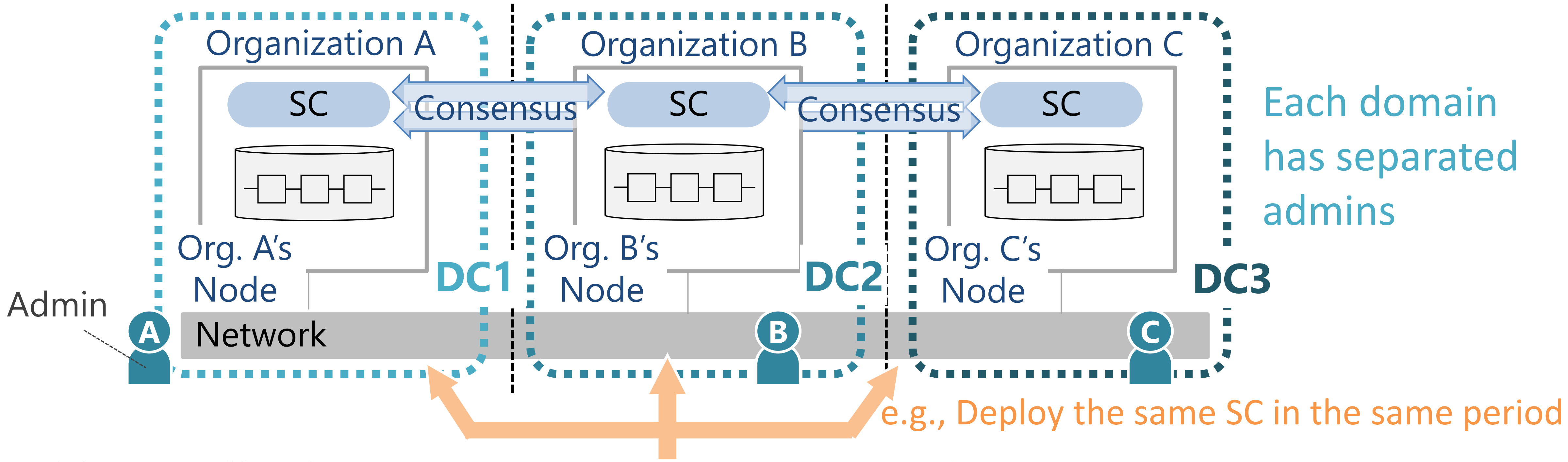
(*) [1] https://ieeexplore.ieee.org/abstract/document/8328745

[2] https://arxiv.org/abs/1901.11249

Background

(*) BC: Blockchain, SC: Smart Contract, DC: Data Center

- Toward production uses, **system operations** become more important
  - e.g., Upgrading a SC and the applications, taking snapshot of ledger data
- Target: Blockchain-based system built across **multiple management domains**



- Problem: Difficult to execute **inter-organizational system operations**
  : need to collaborate with other organizations

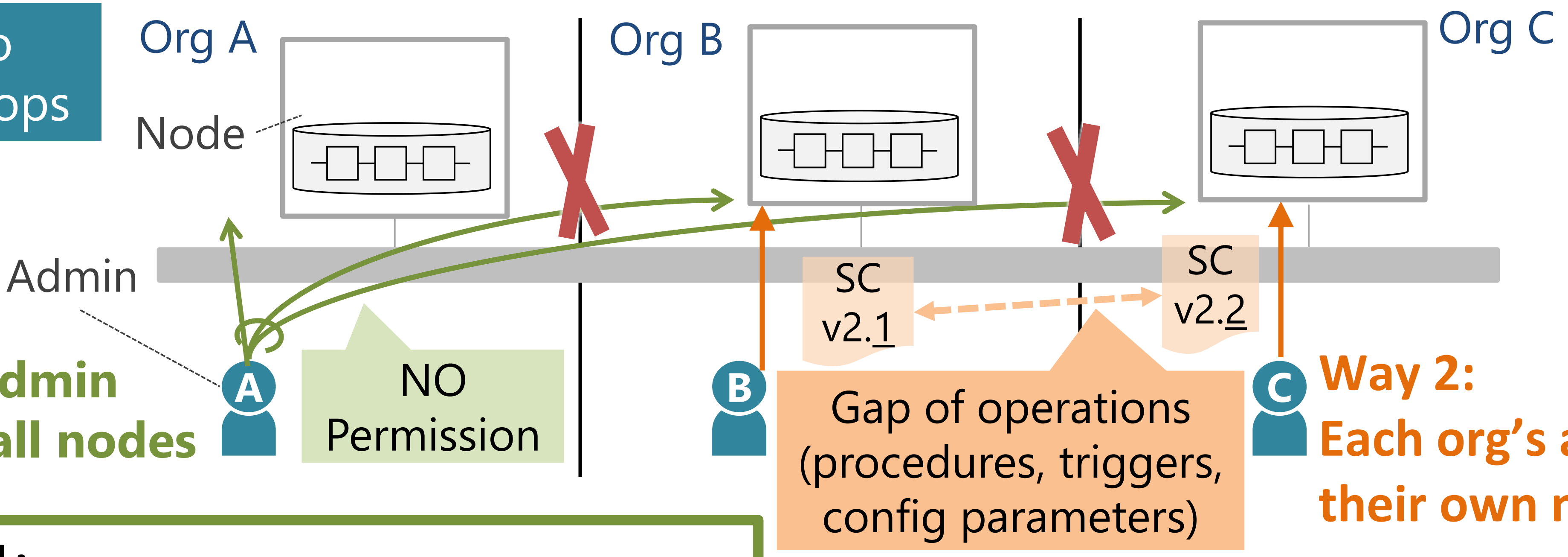
Problems about the system operations for BC-based systems

Conventional operation management tools (e.g., Job mgmt. servers, IaC tools):
- Enable admins to do general (= single-organizational) operations efficiently

➡ But do not cover with ***inter-organizational operations***

(*) IaC: Infrastructure as Code

How to do inter-org ops



Way 1: A single admin operates all nodes

Way 2: Each org's admin operates their own nodes

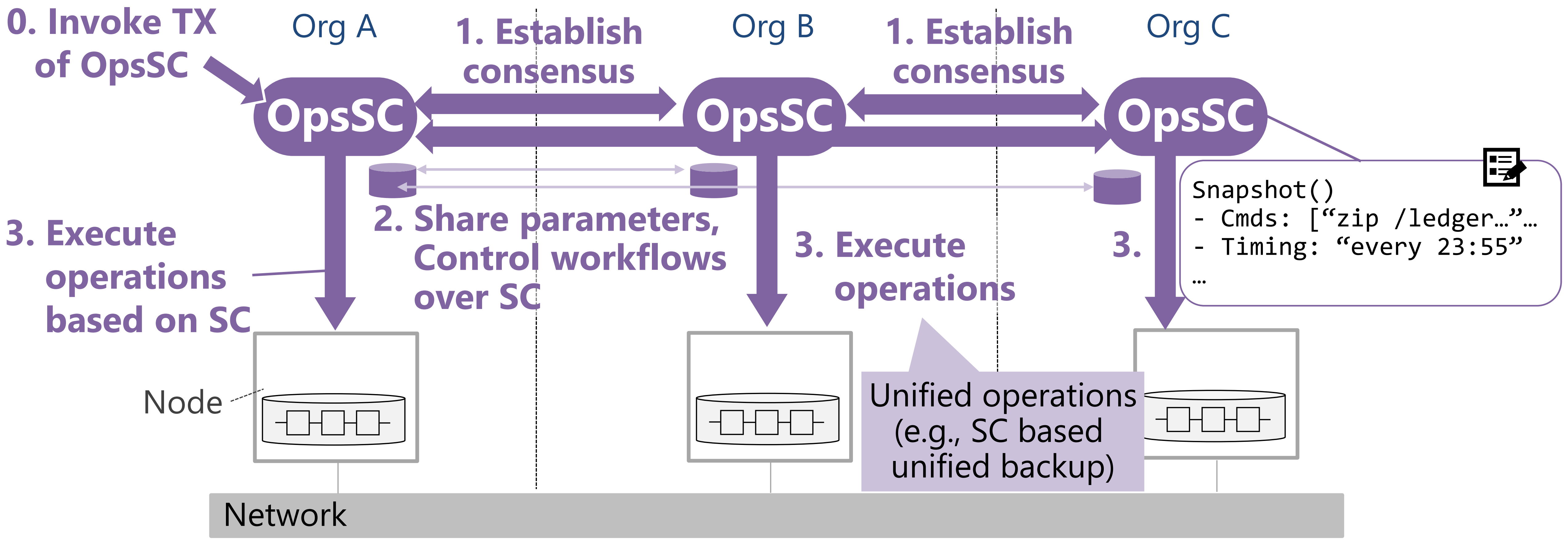
Problem 1:
- The admin is SPOT (Single Point of Trust)
- Cannot access to nodes owned by other orgs because of lack of permissions

Problem 2:
Different configs may prevent the system from working

Smart contract-based system operations method

(*) SC: Smart Contract, TX: Transaction,
SPOT: Single Point of Trust

To define ***system operational flow as a SC***
- Cross-domain operations w/o SPOT and sharing credentials by <u>BC consensus</u>
- Unified procedures with unified config parameters based on <u>SC</u>



```
Snapshot()
- Cmds: [“zip /ledger…”…
- Timing: “every 23:55”
…
```

Unified operations (e.g., SC based unified backup)

OpsSC for Hyperledger Fabric v2.x

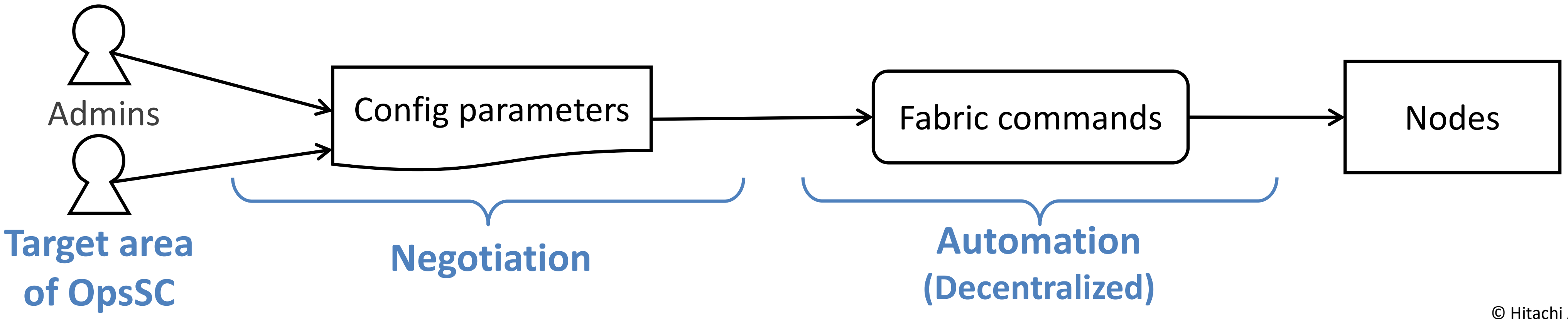
For Hyperledger Fabric v2.x

(*) SPOT: Single Point of Trust

- Current status of Hyperledger Fabric v2.x
  - Individual operational tasks (e.g., *peer* commands) has been refined, and SPOT is eliminated (e.g., introduced the new chaincode lifecycle from v2.0)
- **Remaining issue**: Efficient end-to-end operational workflows using the individual tasks
  - Increased tasks which are executed by each org and must use the same parameters

e.g., Chaincode deployment:
- Each organization must approve the chaincode definition with the same parameters as the other organizations
- Organizations need to share and coordinate the source code and parameters on the chaincode offline with other organizations (in typical cases)

The OpsSC for Fabric v2.x: aims to enhance negotiation and automation capabilities



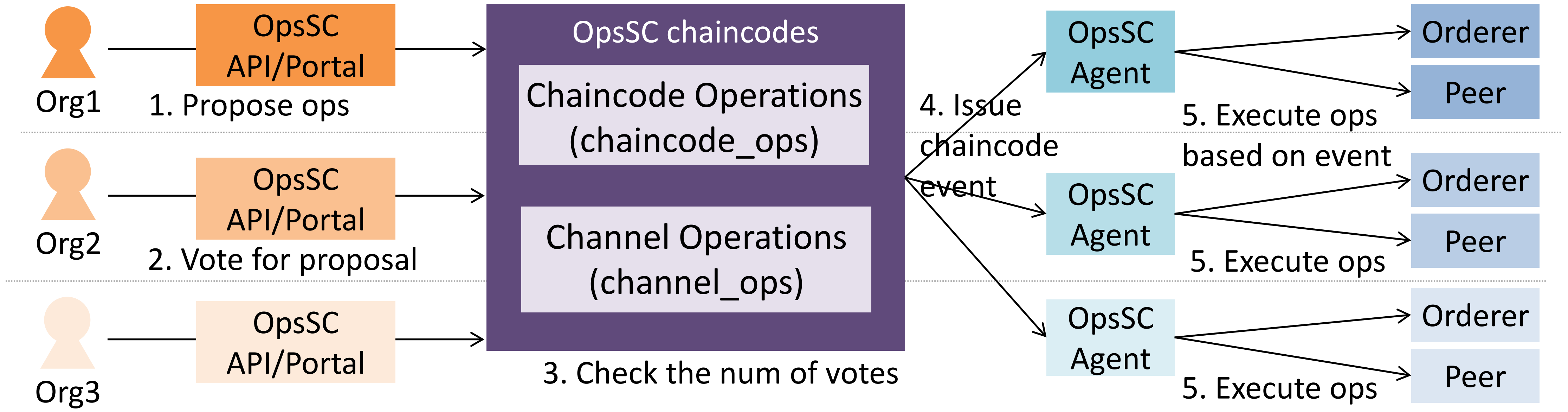
Related Activities

- **System chaincode** [1]
  - Special chaincode which runs within the peer process and it is currently used for internal processing and configuration-value sharing on the Fabric platform (e.g., *_lifecycle* to manage chaincode lifecycle, *CSCC* to handle changes to a channel config)
  - ➡ Our OpsSC internally uses system chaincodes to operate the Fabric network
- **Fabric Interop Working Group** [2]
  - Purpose: To promote the interoperability of Fabric network service
    - Focusing on a scenario that new organization joins a running Fabric network
  - Approach: Create artifacts for the join request (= *configtx*) with "Consortium Management Chaincode (CMCC)"
  - ➡ The concept is very similar with ours although the scope is slightly different
    - In fact, current OpsSC for channel ops. reuses part of the CMCC implementation
  - Our OpsSC could be positioned as a form or application of the CMCC

[1] https://hyperledger-fabric.readthedocs.io/en/release-2.2/smartcontract/smartcontract.html#system-chaincode
[2] https://wiki.hyperledger.org/display/fabric/Fabric+Interop+Working+Group

Implementation of OpsSC for Hyperledger Fabric v2.x

- Consist of 3 components: OpsSC chaincode, OpsSC API server and OpsSC Agent
  - *Chaincode* provides functions to manage operational workflows and issues chaincode events including the operational instructions
  - *API server* provides REST API for each org's admin to interact with the OpsSC chaincodes
  - *Agent* for each org executes operations based on the chaincode events to ALL nodes for the org



Ph.1: Provide a purpose-specific OpsSC which is essential for managing the Fabric network (for operating chaincodes and channels)

OpsSC for operating *chaincodes*

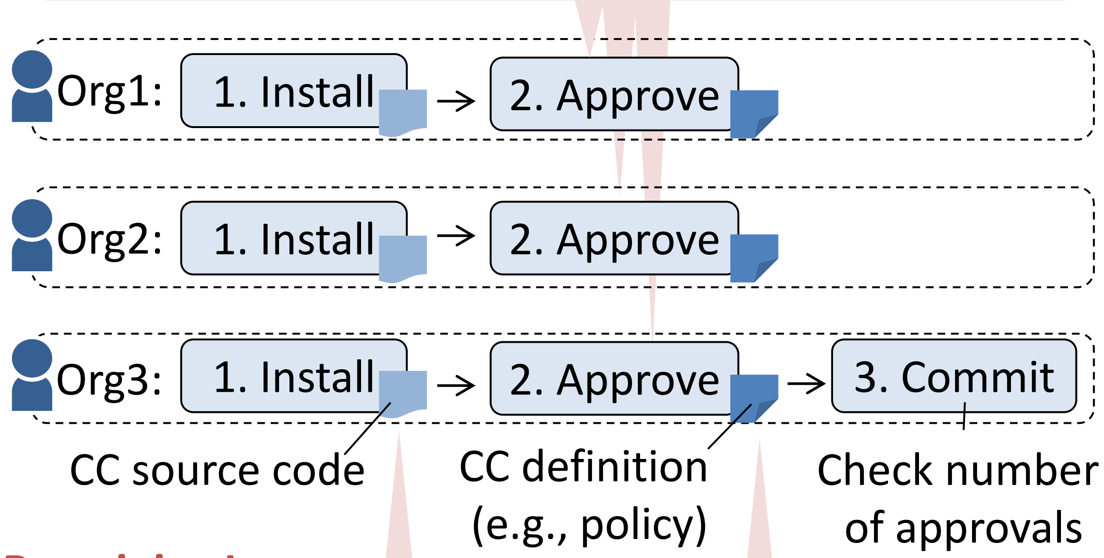
(*) CC: Chaincode

New Chaincode Lifecycle from v2.0

- Deploy in 3 phases: Install, Approve, Commit
  - Eliminated centralized process

Remaining Issue:

Increase operations which are executed by each org and must use the same parameters



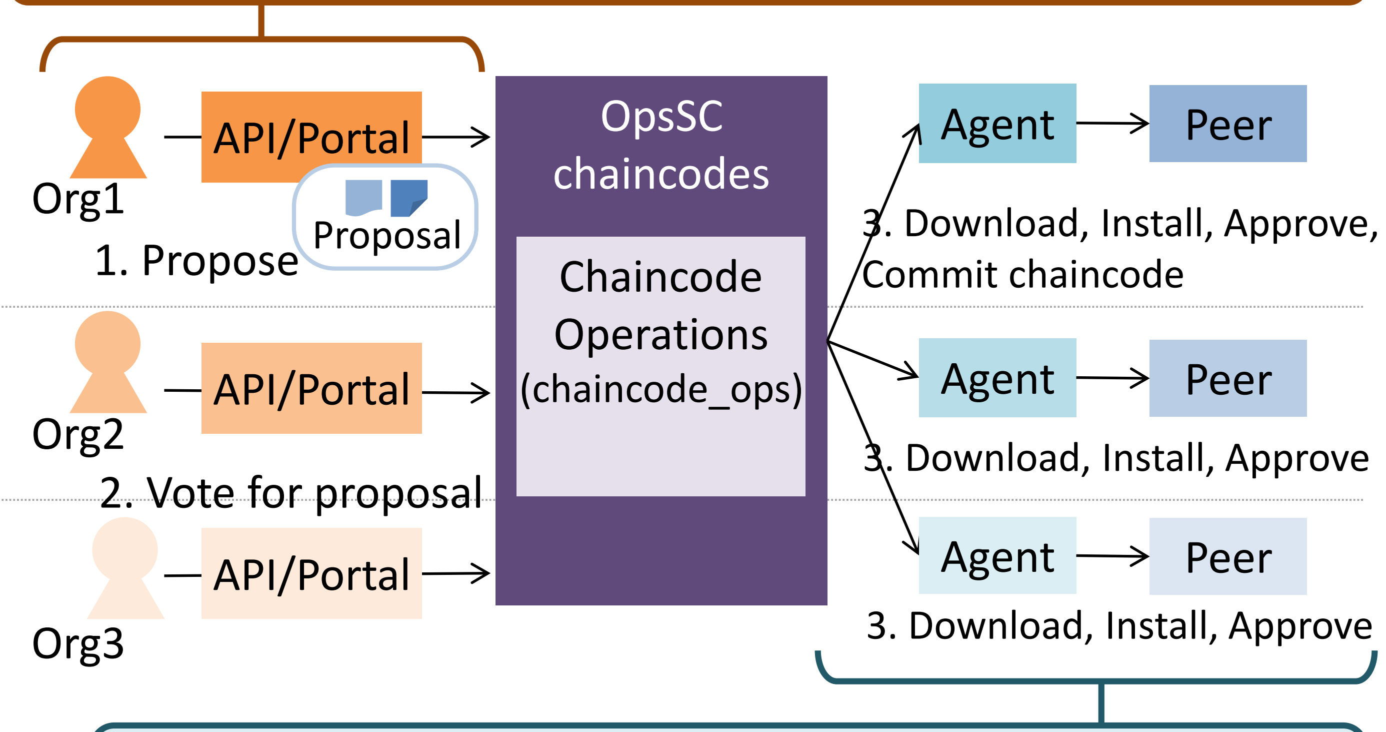
Remaining Issue:

Need to share and negotiate the source code and parameters with the other orgs (in typical case)

OpsSC for operating chaincodes

- Streamline end-to-end chaincode deployment

1. An org creates a proposal with CC source code and definition
2. Other orgs vote for the proposal shared on the OpsSC

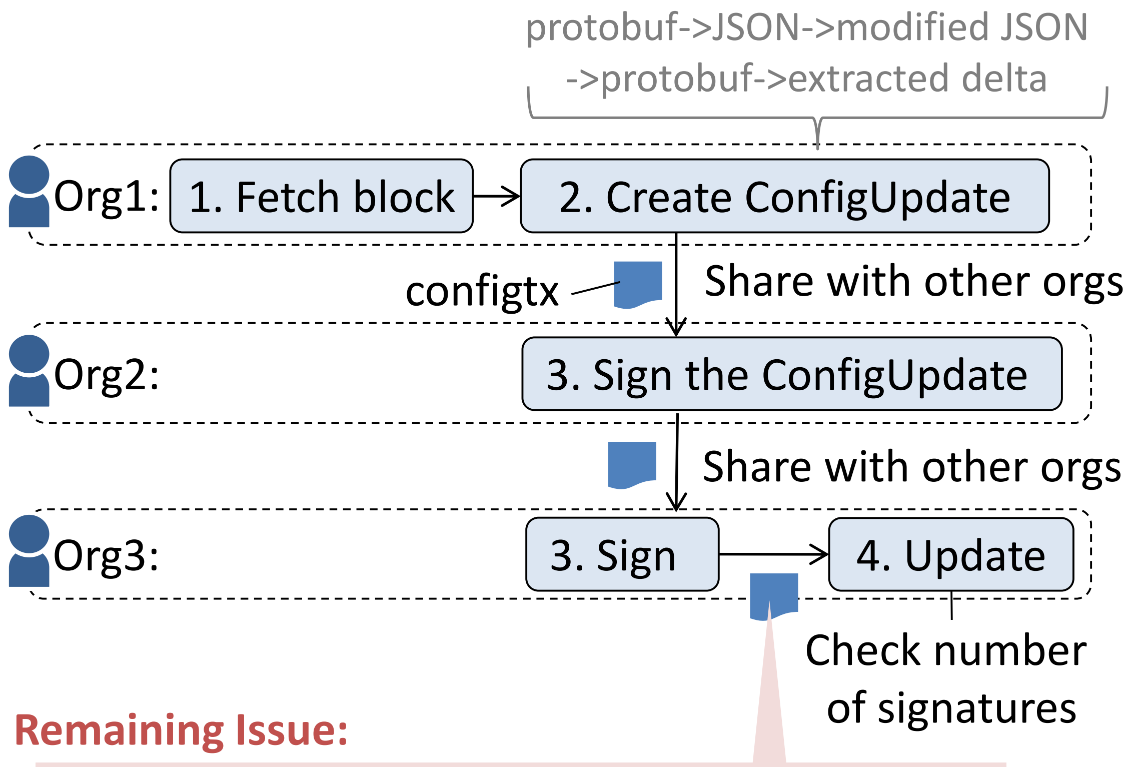

Org1
API/Portal
Proposal
1. Propose
Org2
API/Portal
2. Vote for proposal
Org3
API/Portal
OpsSC chaincodes
Chaincode Operations (chaincode_ops)
Agent
Peer
3. Download, Install, Approve, Commit chaincode
Agent
Peer
3. Download, Install, Approve
Agent
Peer
3. Download, Install, Approve

3. When the majority of votes is collected, each agent automatically deploys the chaincode based on the proposal

OpsSC for operating *channels*

Process for channel updates across orgs

- e.g., create a channel, add an org / orderer
- Process: create configtx, collect signatures from each org and send the configtx to nodes



Remaining Issue:

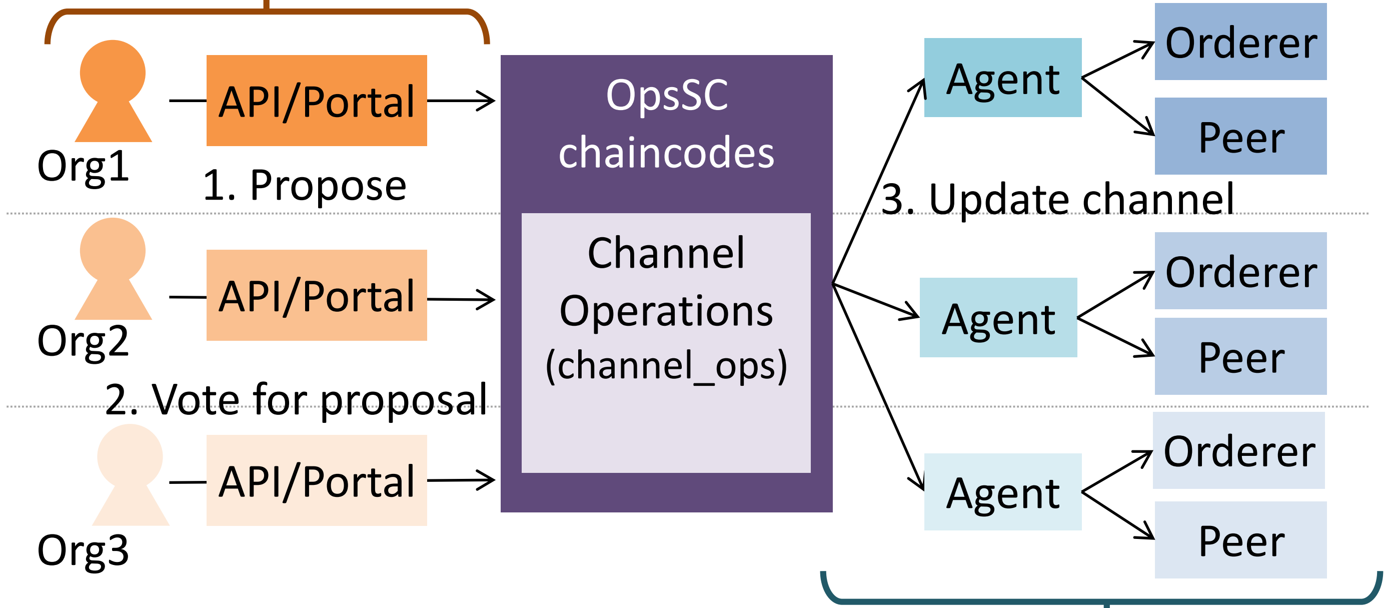
Need to share configtx with the other orgs

OpsSC for operating channels

- Streamline *only* channel updates across multiple orgs

1. An org creates a human-readable channel update proposal
2. Other orgs vote for the proposal shared on the OpsSC

(Internally convert to configtx with Config Transaction Library)



3. When the majority of votes are collected, one of the agents automatically updates the channel with the proposed configtx

Demo: Add a new chaincode, add a new organization using OpsSC

[Demo environment]

- Fabric version: v2.3.0
- Fabric network: test-network in fabric-samples (including some customizations)
  - Initial network: 3 orgs (all orgs have their CA, peer, orderer), and mychannel
  - OpsSC chaincodes has been deployed on ops-channel

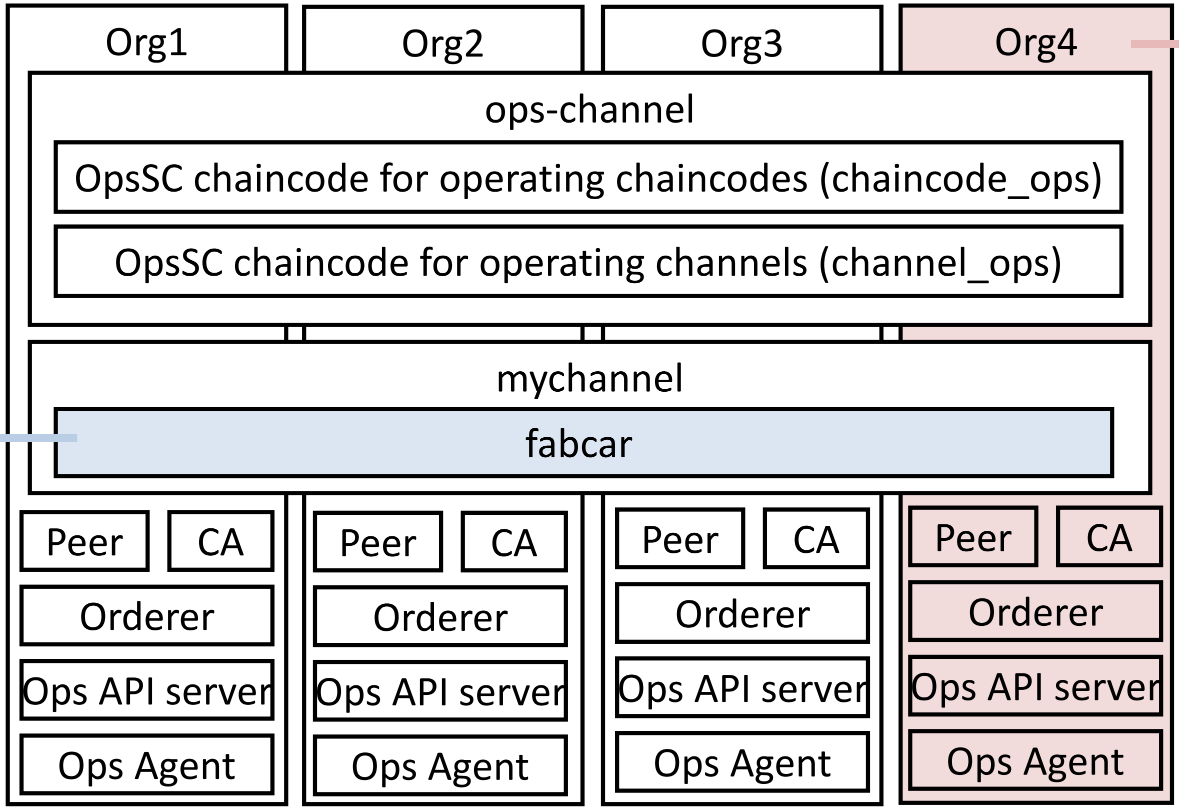
[Scenario 1. Add a new CC]

Steps:

1. Org1 proposes fabcar
2. Others votes for it

Result:

fabcar with the proposed parameters is deployed


Org1
Org2
Org3
Org4
ops-channel
OpsSC chaincode for operating chaincodes (chaincode_ops)
OpsSC chaincode for operating channels (channel_ops)
mychannel
fabcar
Peer
CA
Orderer
Ops API server
Ops Agent

[Scenario 2. Add a new org]

Steps:

1. Org4 prepares a CA and issues certs/keys for peers and orderers
2. Org1 proposes adding Org4 (with Org4's MSP)
3. Org2, 3 votes for it
(2, 3 are required for each channel)
4. Org4 launches other components (Need to get genesis from others)

Result:

Org4 is added to all channels

- OpsSC and fabcar are deployed

Plans

- Development
  - General operations support
    - Execute arbitrary command via OpsSC chaincode
  - v2.3.x new feature support
    - e.g., Channel participation without system channel
  - etc.

- Community contribution
  - We would like to start this as a "hyperledger-labs" project.
    - We are looking for a sponsor who could help us open a repository in the labs!!
  - In the future, I would like to make this a subproject of Fabric.
    (depends on demand and acceptance)

HITACHI
Inspire the Next