HITACHI
Inspire the Next

Hyperledger Global Forum 2021

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

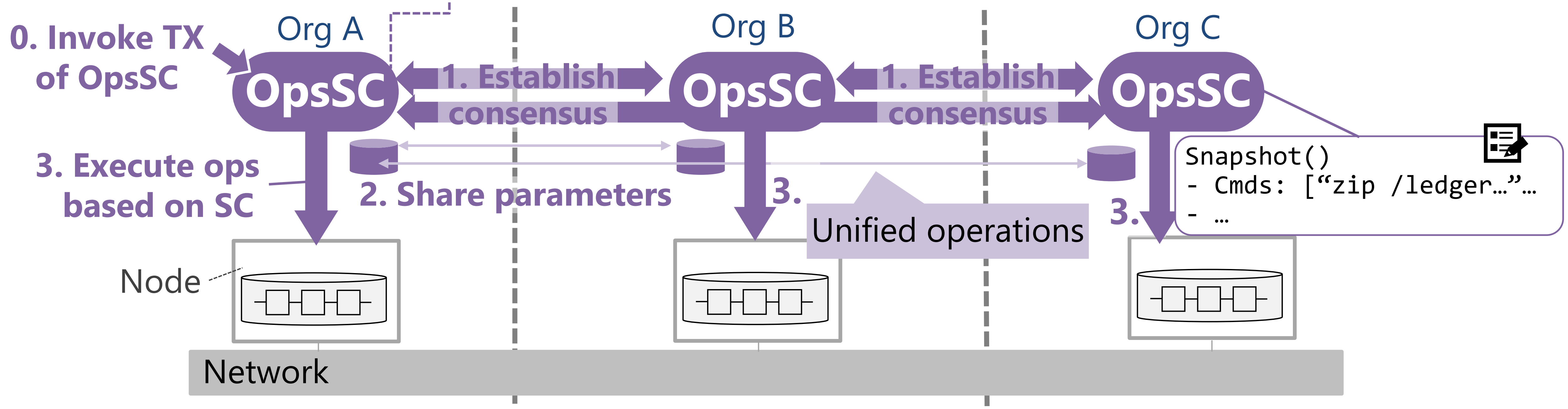
https://github.com/hyperledger-labs/fabric-opssc

Research and Development Group, Hitachi, Ltd.
○Tatsuya Sato and Taku Shimosawa

What is *Operations Smart Contract (OpsSC)*?

HITACHI
Inspire the Next

- Background: Blockchain-based system built across multiple organizations (with separated admins)
- **Goal**: Establishing decentralized system operations across multiple organizations
- **Idea**: **Define a system operational workflow as a smart contract**, each organization (admin / agent program) operates their own nodes according to the smart contract



```
Snapshot()
- Cmds: ["zip /ledger…"…
- …
```

- **Value:** Inter-organizational operations can be performed (1) without relying on decisions by a specific organization (2) with uniform procedure/configuration parameters (3) efficiently

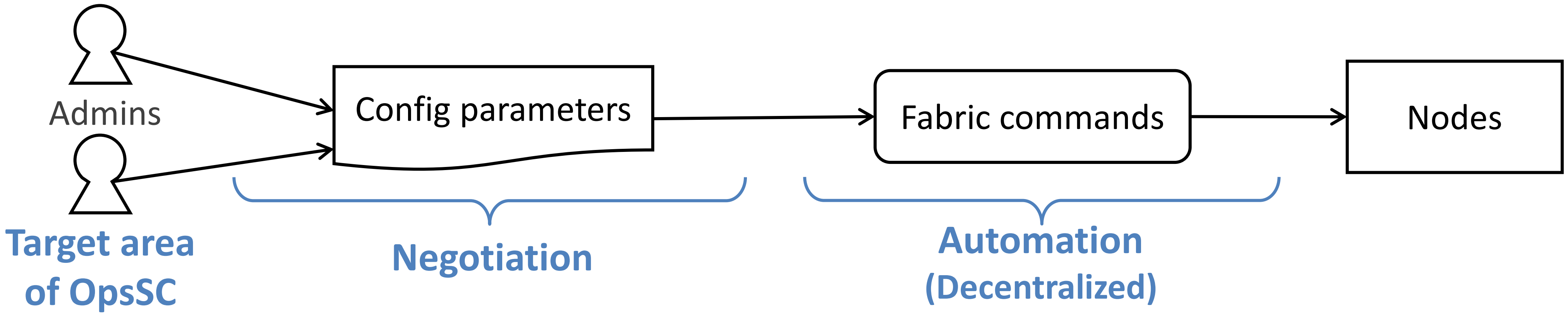
For Hyperledger Fabric v2.x

(*) 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

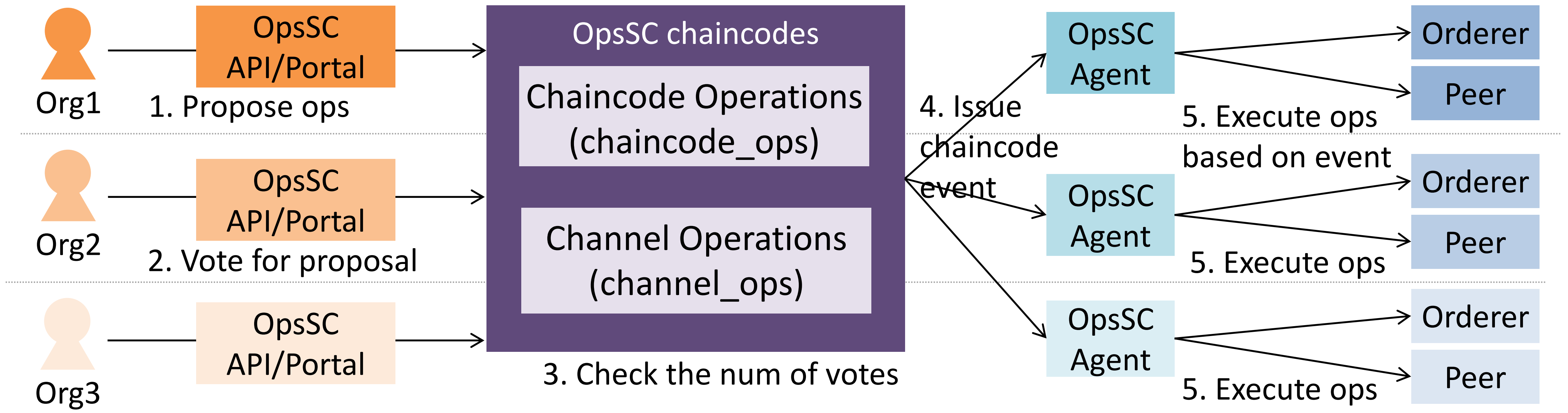
e.g., Chaincode deployment:
- Each organization must approve the chaincode definition with the same parameters as the other organizations
- Organizations need to share and coordinate the source code and parameters on the chaincode offline with other organizations (in typical cases)

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



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*

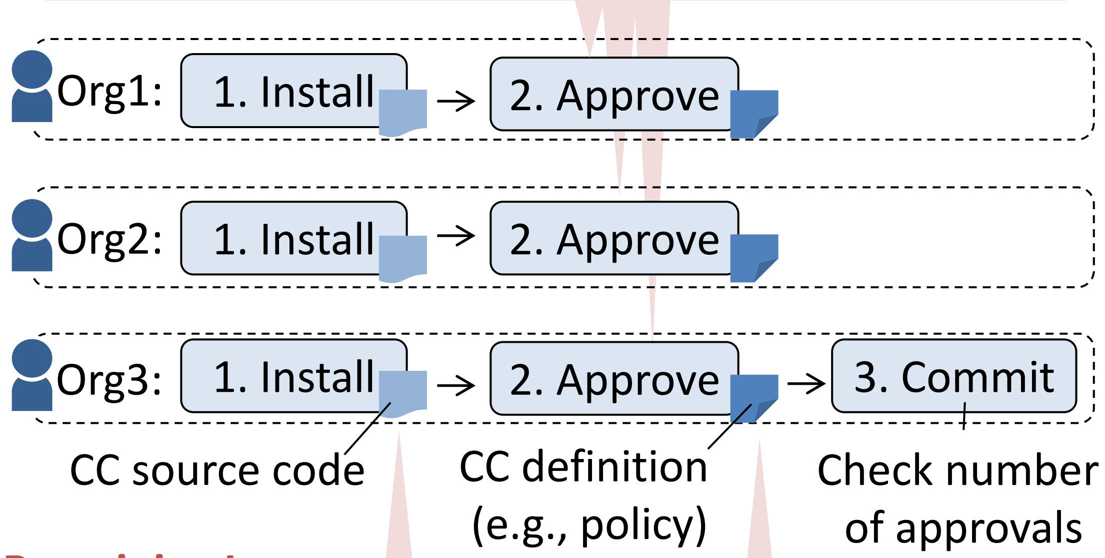
(*) CC: Chaincode

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



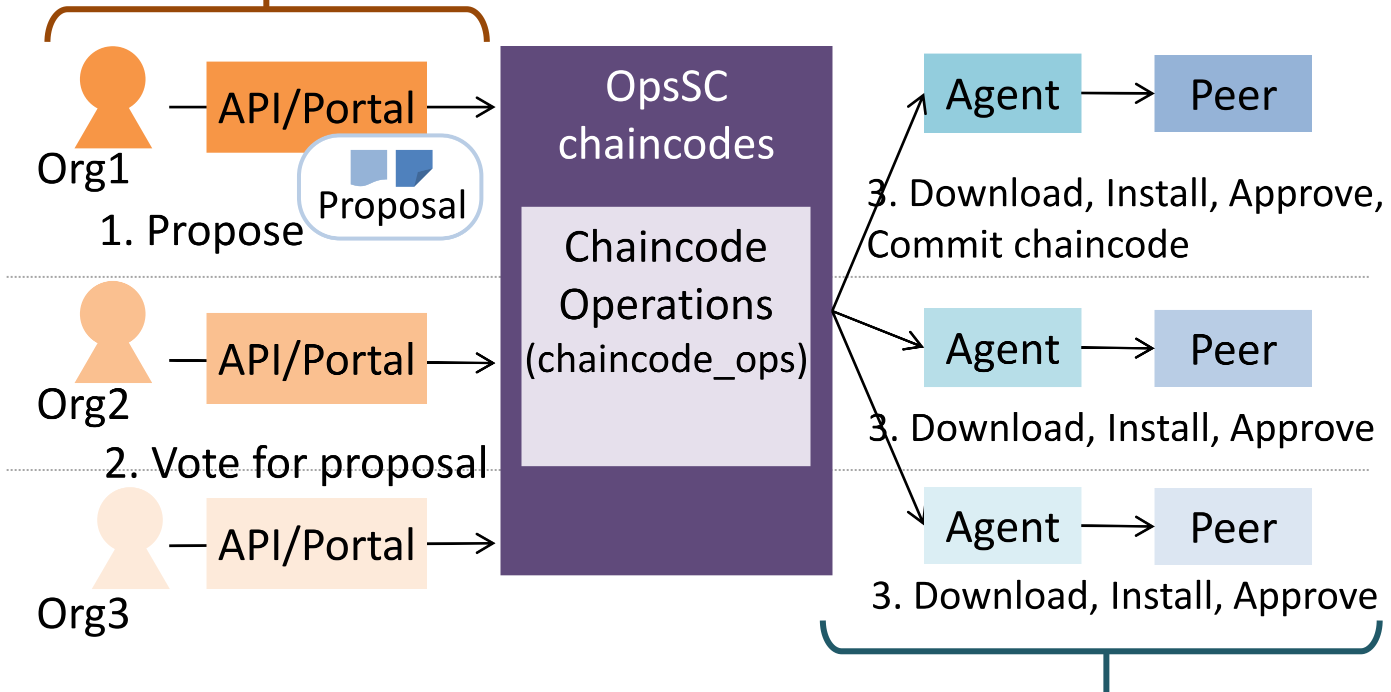
Remaining Issue:

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

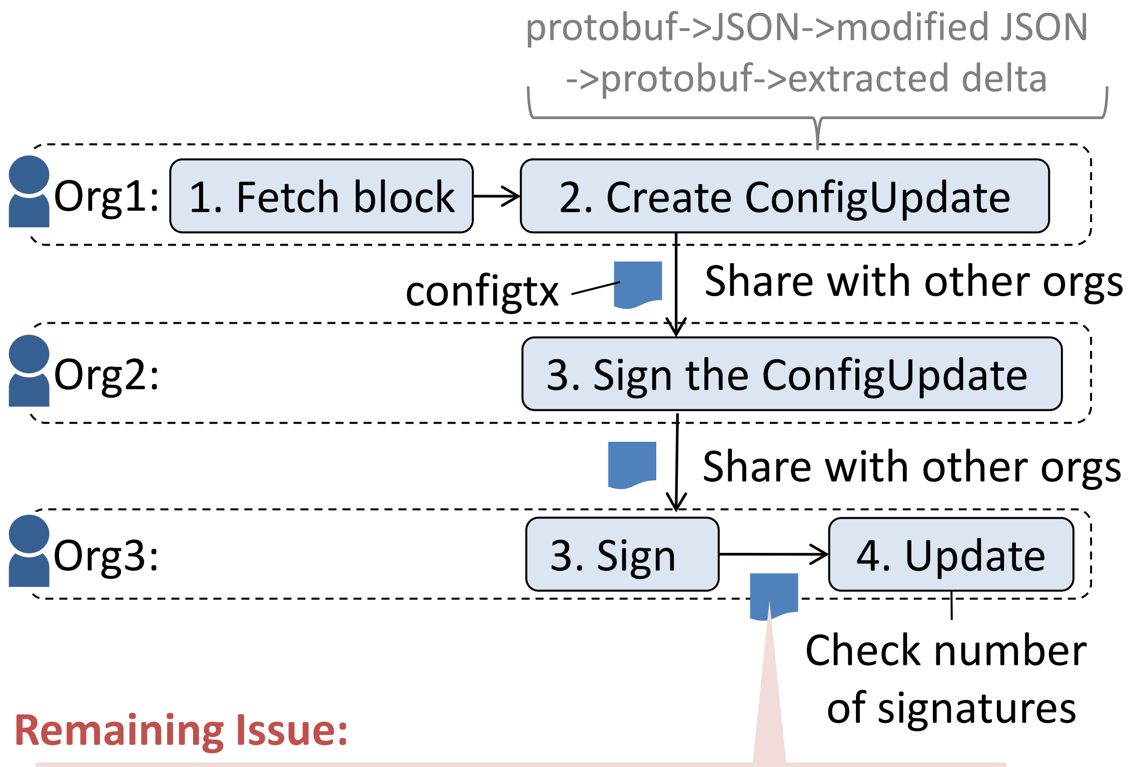

Org1
API/Portal
Proposal
1. Propose
Org2
API/Portal
2. Vote for proposal
Org3
API/Portal
OpsSC chaincodes
Chaincode Operations (chaincode_ops)
Agent
Peer
3. Download, Install, Approve, Commit chaincode
Agent
Peer
3. Download, Install, Approve
Agent
Peer
3. Download, Install, Approve

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., Add an organization / orderer
- Process: create configtx, collect signatures from each org and send the configtx to nodes



Remaining Issue:

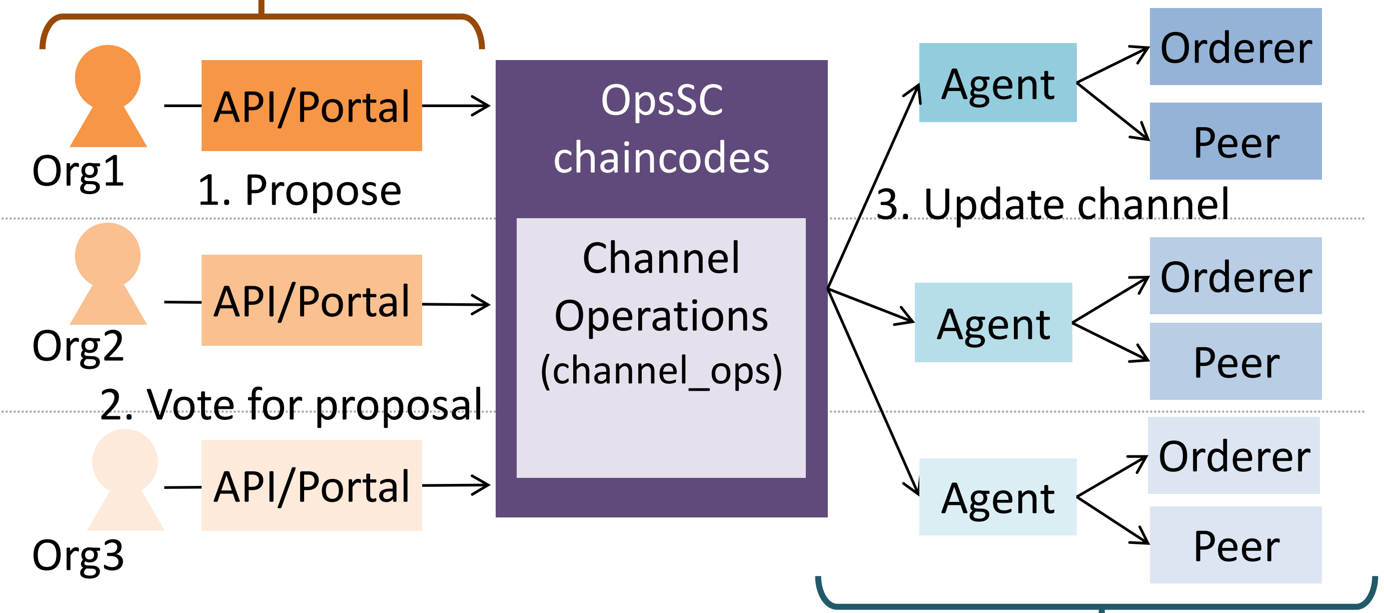
Need to share configtx with the other orgs

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

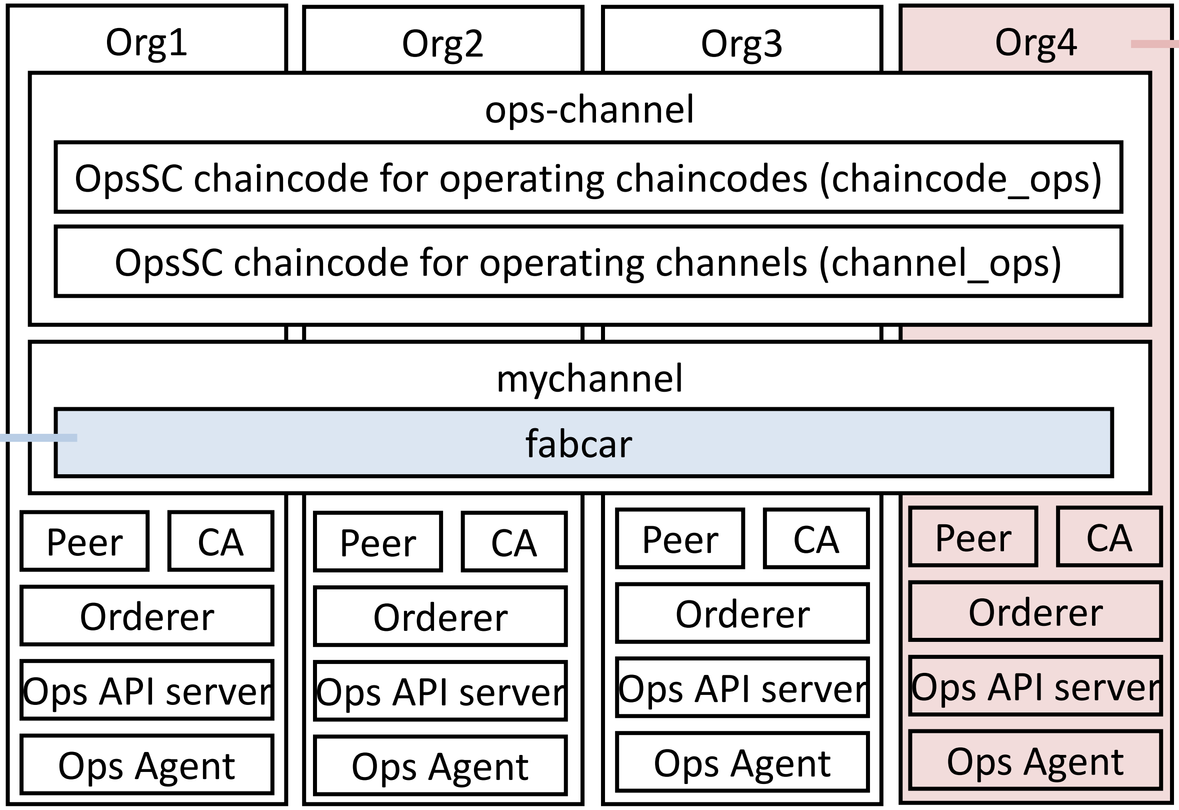
[Scenario 1. Add a new CC]

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:

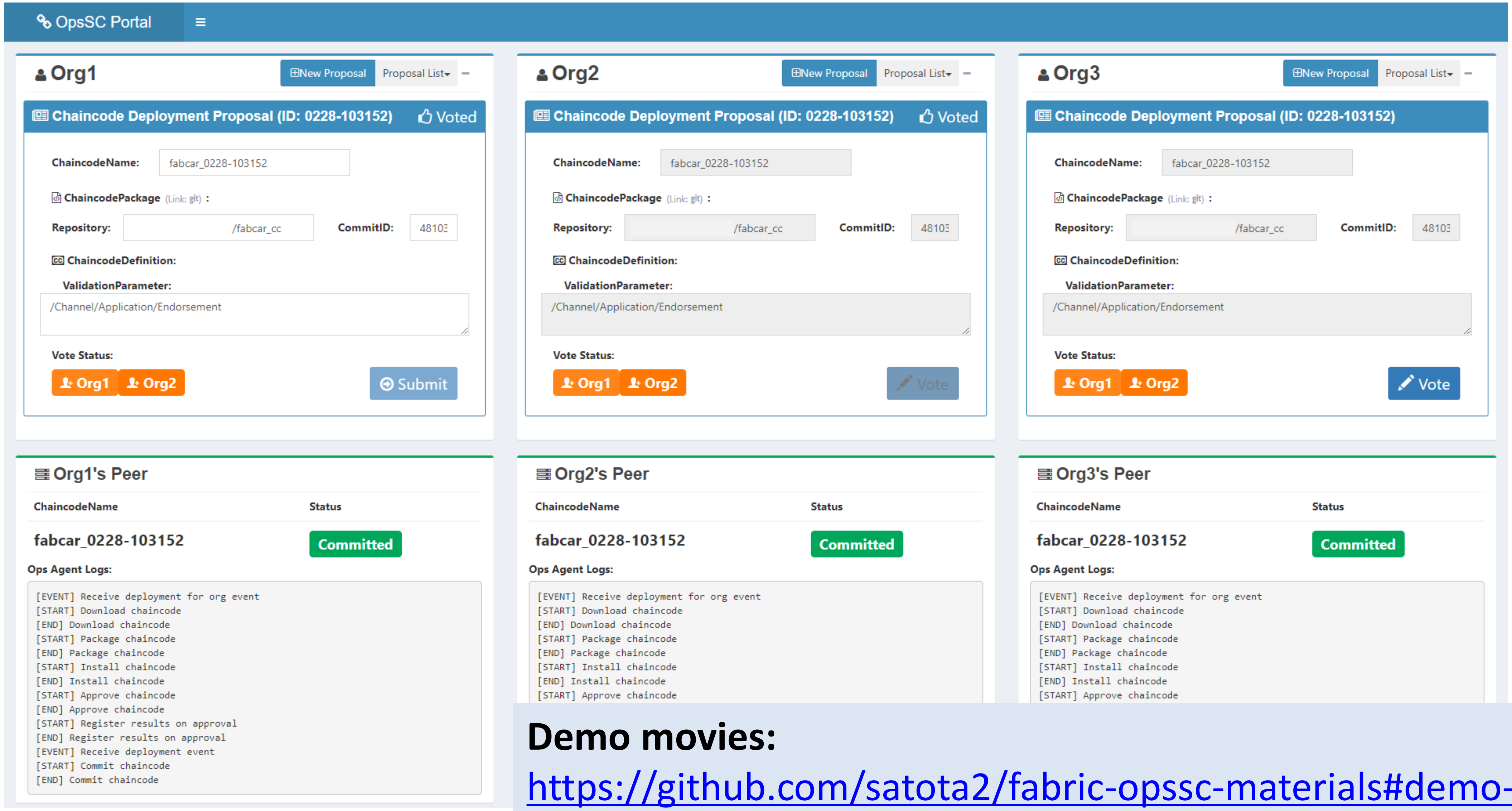
Org4 is added to all channels

- OpsSC and fabcar are deployed

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

HITACHI Inspire the Next

Portal Screen for OpsSC



Chaincode Deployment Proposal (ID: 0228-103152)

ChaincodeName: fabcar_0228-103152

ChaincodePackage (Link: git) :

Repository: /fabcar_cc CommitID: 48103

ChaincodeDefinition:

ValidationParameter: /Channel/Application/Endorsement

Vote Status: Org1 Org2

Org1's Peer / **Org2's Peer** / **Org3's Peer**

ChaincodeName	Status
fabcar_0228-103152	Committed

Ops Agent Logs:

```
[EVENT] Receive deployment for org event
[START] Download chaincode
[END] Download chaincode
[START] Package chaincode
[END] Package chaincode
[START] Install chaincode
[END] Install chaincode
[START] Approve chaincode
[END] Approve chaincode
[START] Register results on approval
[END] Register results on approval
[EVENT] Receive deployment event
[START] Commit chaincode
[END] Commit chaincode
```

Demo movies:
https://github.com/satota2/fabric-opssc-materials#demo-movies

OpsSC Roadmap (Plan)

- Improve the quality of existing features and develop new features for general purpose ops
- Aim to get the part of functionality of OpsSC merged in Hyperledger Fabric

	(2021 April-Sep)	**(2021 Oct-2022 Mar)**	**(2022 April-)**
Feature improvements	Continuous improvements		
New features	General operations support (Basic) Initial version ▲	General operations support (Advanced) Initial version ▲	
Proposal to Fabric		Chaincode API to get org list; Simple voting for admins	…

Feature improvements

Common parts

- Support for both Fabric v2.2 and v2.3
- Sample and integration tests on K8s env.
- Authentication for API server (e.g., JWT)
- Full support for workflow state transitions (e.g., Rejecting a proposal)
- RBAC/ABAC for OpsSC chaincodes
- Voting policy configuration

Chaincode ops

- Java chaincode support
- External chaincode support
- Chaincode initialization support
- Private collection configuration
- Support for minor chaincode ops (e.g., Disable chaincode)

Channel ops

- Import channel information to OpsSC chaincodes from genesis block
- Support for channel mgmt. without system channel (Feature from Fabric v2.3)
- Operation History management (improvement of the implementation)

New features

General operations support

- Basic (simple workflow)
- Sample for general operations
- Advanced (complex workflow)

Others

- Collaboration with ledger snapshot API
- …

Proposal to Fabric

- Chaincode API to get organization list
- Simple voting feature for administrators
- …

Others (Depending on situation)

- Porting the OpsSC API server and agent impl. from Node to Go SDK
- Utilizing Fabric Gateway

Summary

- *Operations Smart Contract (OpsSC)*
  - Goal: Establishing decentralized system operations across multiple organizations
  - Idea: Define a system operational workflow as a smart contract
- OpsSC for Hyperledger Fabric v2.x is now available:
  - https://github.com/hyperledger-labs/fabric-opssc
  - You can try some scenarios as same as today's demos by reading README
- Feedback and contribution welcome!
  - Contact
    - Hyperledger Rocket.Chat: https://chat.hyperledger.org/channel/fabric-opssc
    - e-mail: tatsuya.sato.so@hitachi.com
- Related presentation
  - "Extending the Operations Smart Contract for Hyperledger Fabric to Support Consortia Governance" - Todd Little, Oracle (Thursday June 10, 18:40- CEST)

Trademarks

- Linux Foundation, Hyperledger, Hyperledger Fabric and Kubernetes are registered trademarks or trademarks of The Linux Foundation.
- GitHub is a registered trademark or trademark of GitHub Inc.
- All other company names, product names, service names and other proper nouns are registered trademarks or trademarks of their respective companies.
- The TM and ® marks are not shown in the text and figures in this slide.

HITACHI
Inspire the Next

Appendix

Related Activities

HITACHI
Inspire the Next

- **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