Internet Computer Protocol: democratic evolution of a web3 platform

DEBS Keynote, June 29, 2023

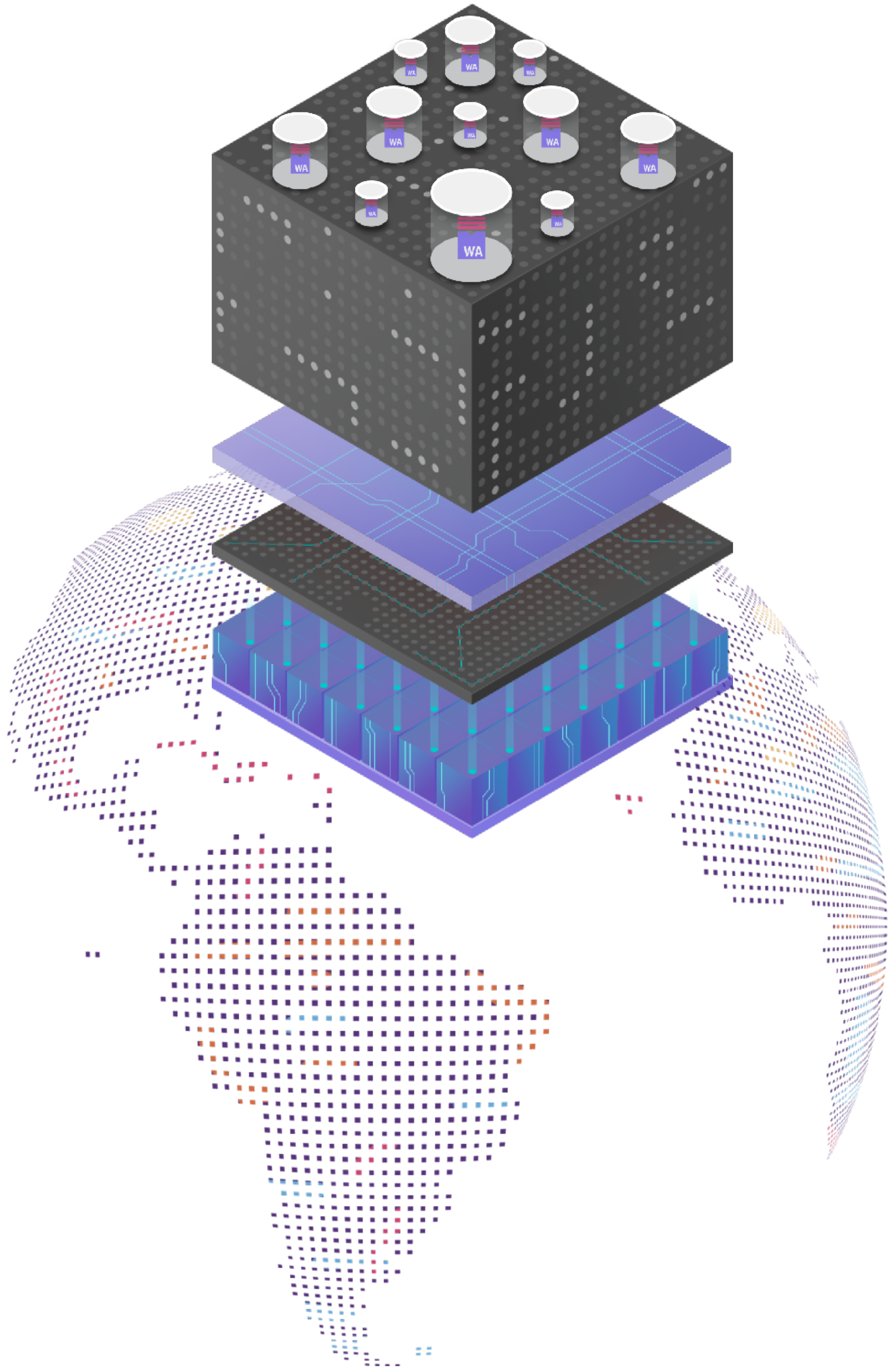
Yvonne-Anne Pignolet
Director of Research, yvonneanne@dfinity.org

DFINITY

- Not-for-profit organisation developing for the Internet Computer
- Roots in early Ethereum community
- DFINITY Foundation established in 2016
- Headquarter: Zurich, Switzerland
- RnD enters in Zurich and San Francisco
- Staff: +250

Outline

- What is the Internet Computer?
- Apps, numbers and stats
- How does the IC evolve?



What is the vision of the Internet Computer?

4

The Internet Computer does to computation what the Internet does to communication

5

Autonomous public cloud to run **general purpose applications** using blockchain technology for decentralisation and security

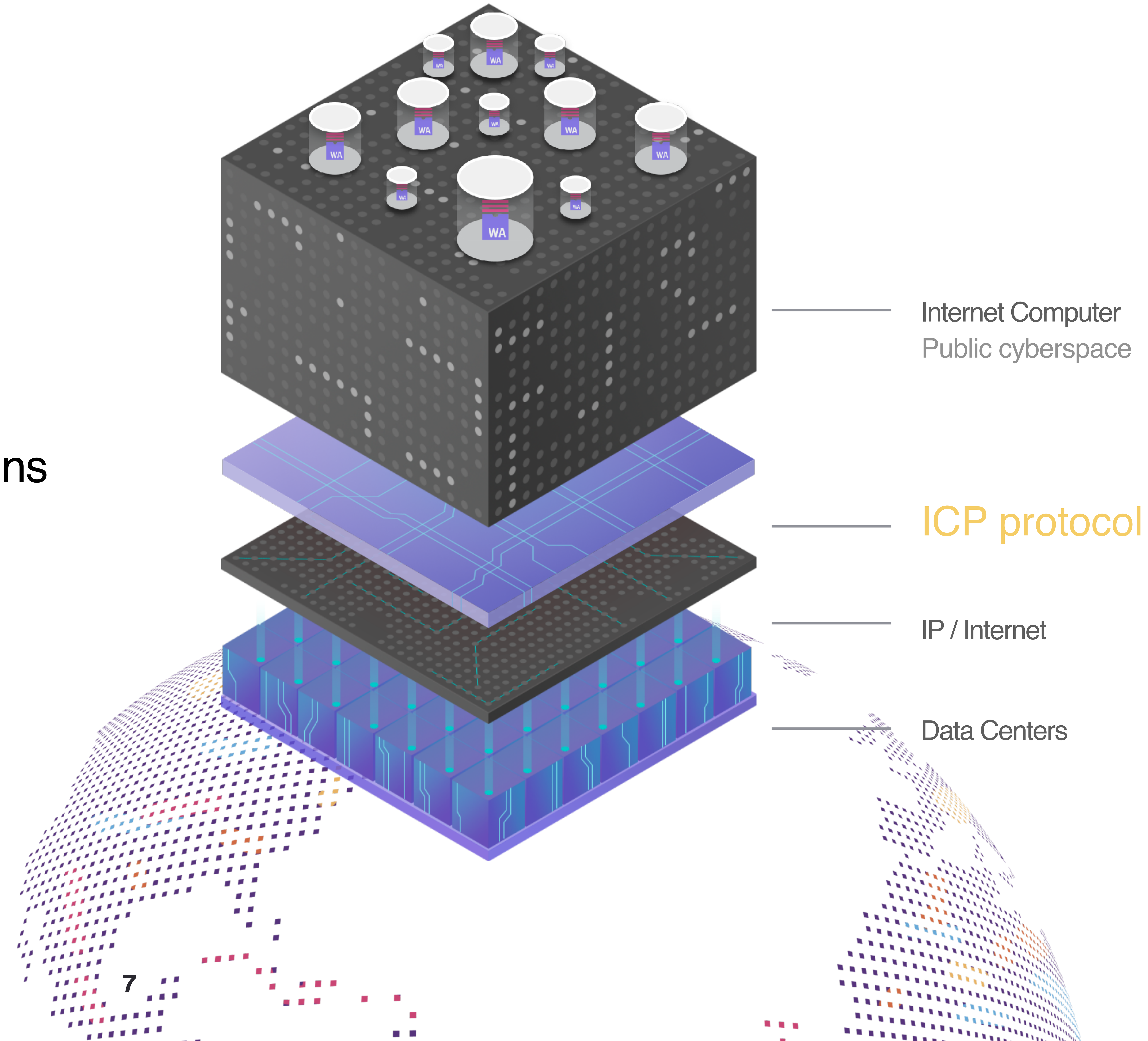
6

Internet Computer Protocol (ICP)

Coordination of nodes in independent data centers, jointly performing any computation for anyone

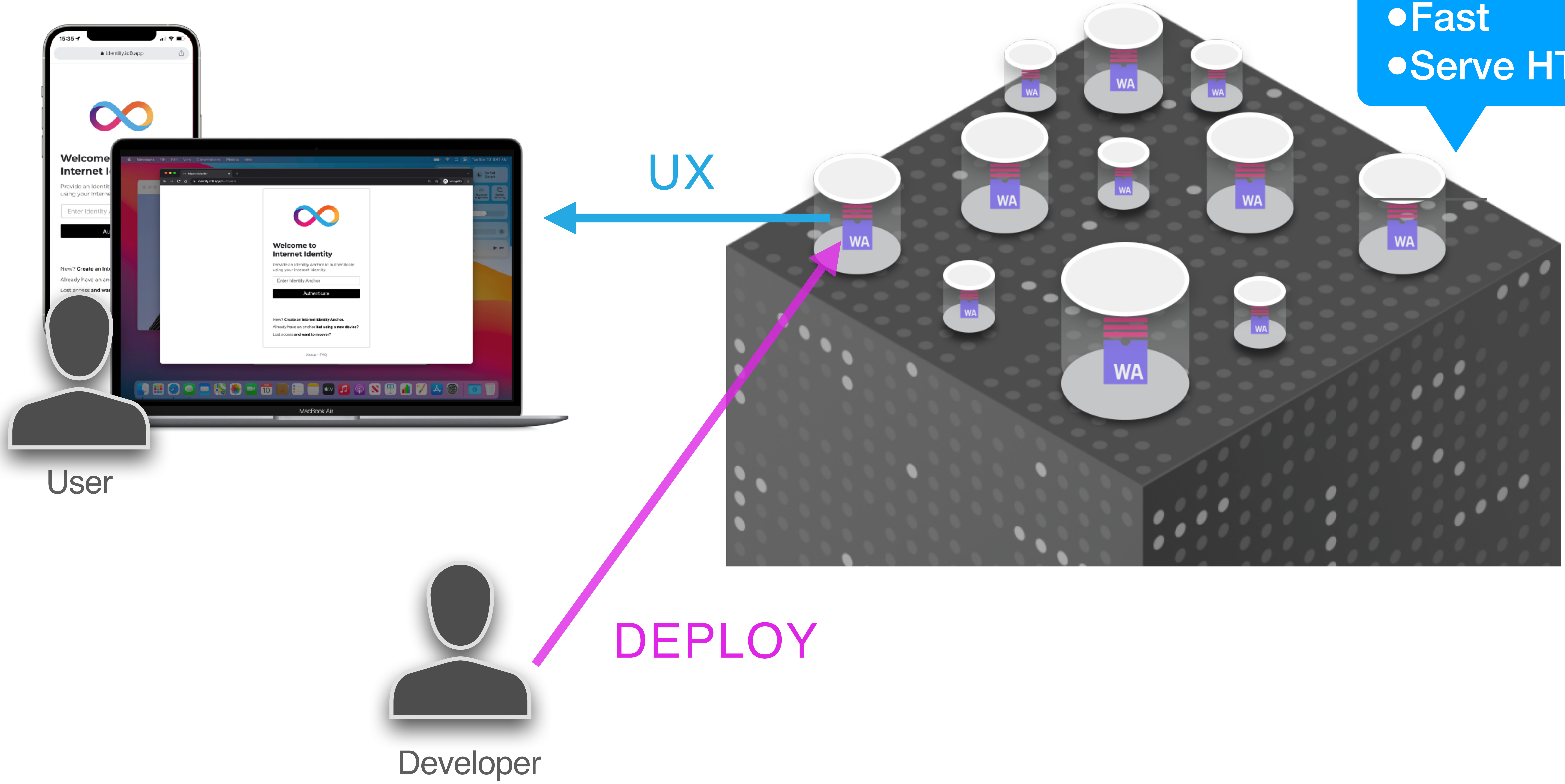
ICP creates the Internet Computer blockchains

Guarantees safety and liveness of smart contract execution despite Byzantine participants



Deploying and Using Canister Smart Contracts

- Multi-chain
- Scalable
- Energy-efficient
- Low-cost
- Fast
- Serve HTTP directly



Tokens

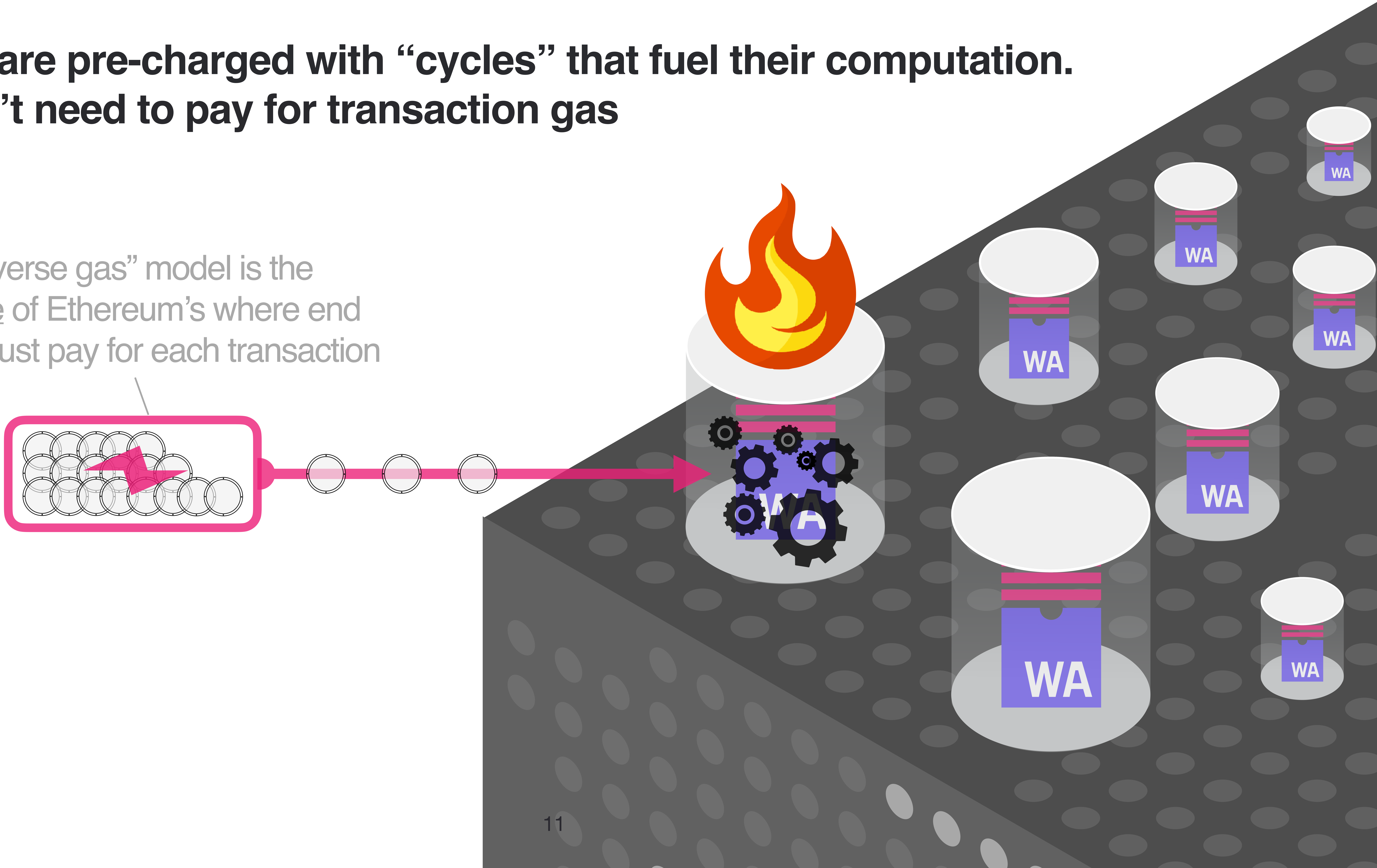
ICP Tokens are used...

- To facilitate **participation** in network **governance**.
- To **reward participants** that participate in governance and operate the node machines.
- To produce the **cycles**, i.e., the fuel used to power computation.

besides ICP, the other native token on the IC

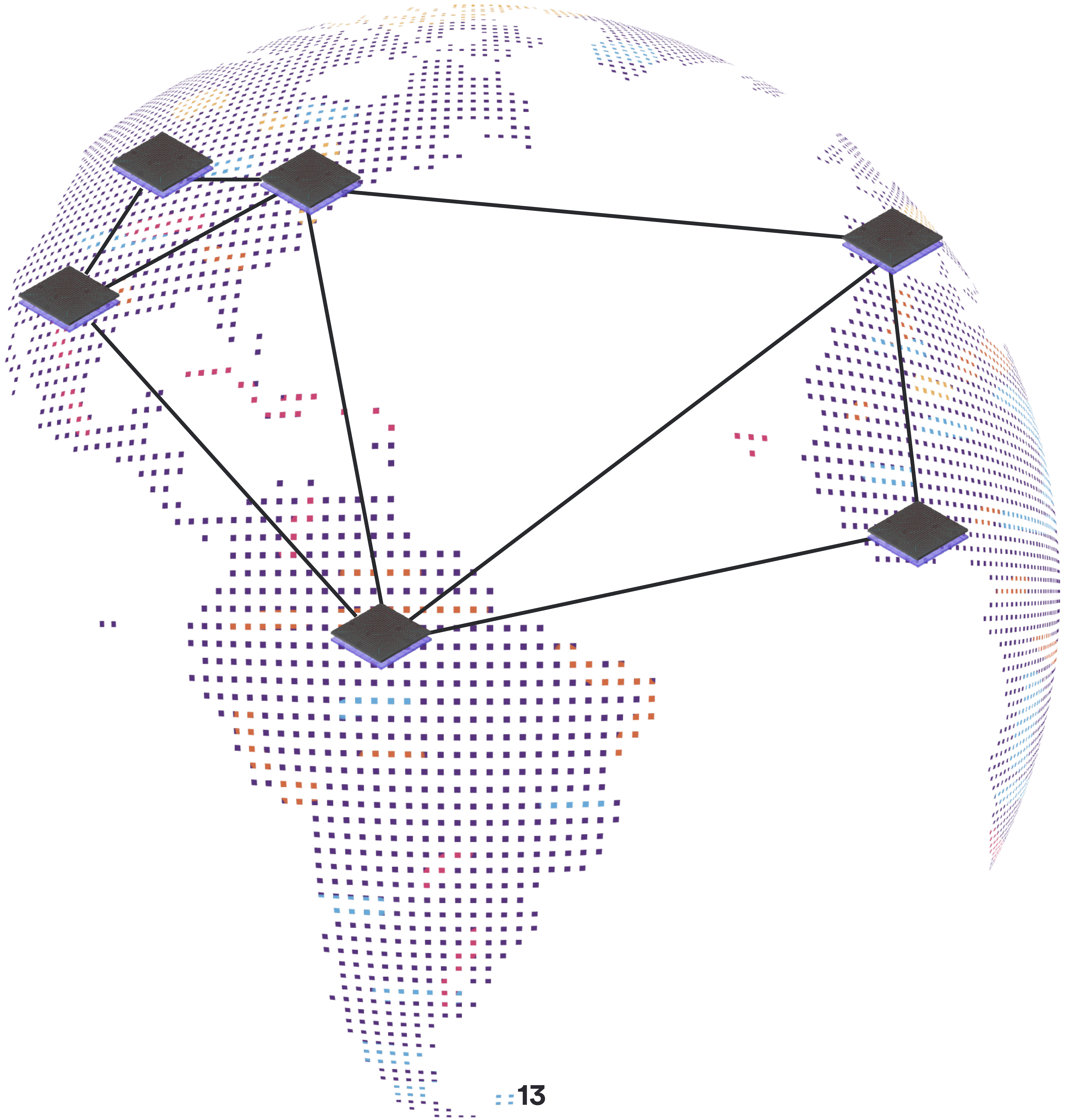
Canisters are pre-charged with “cycles” that fuel their computation. Users don’t need to pay for transaction gas

This “reverse gas” model is the opposite of Ethereum’s where end users must pay for each transaction



Architecture and Governance

Nodes in Independent Data Centers

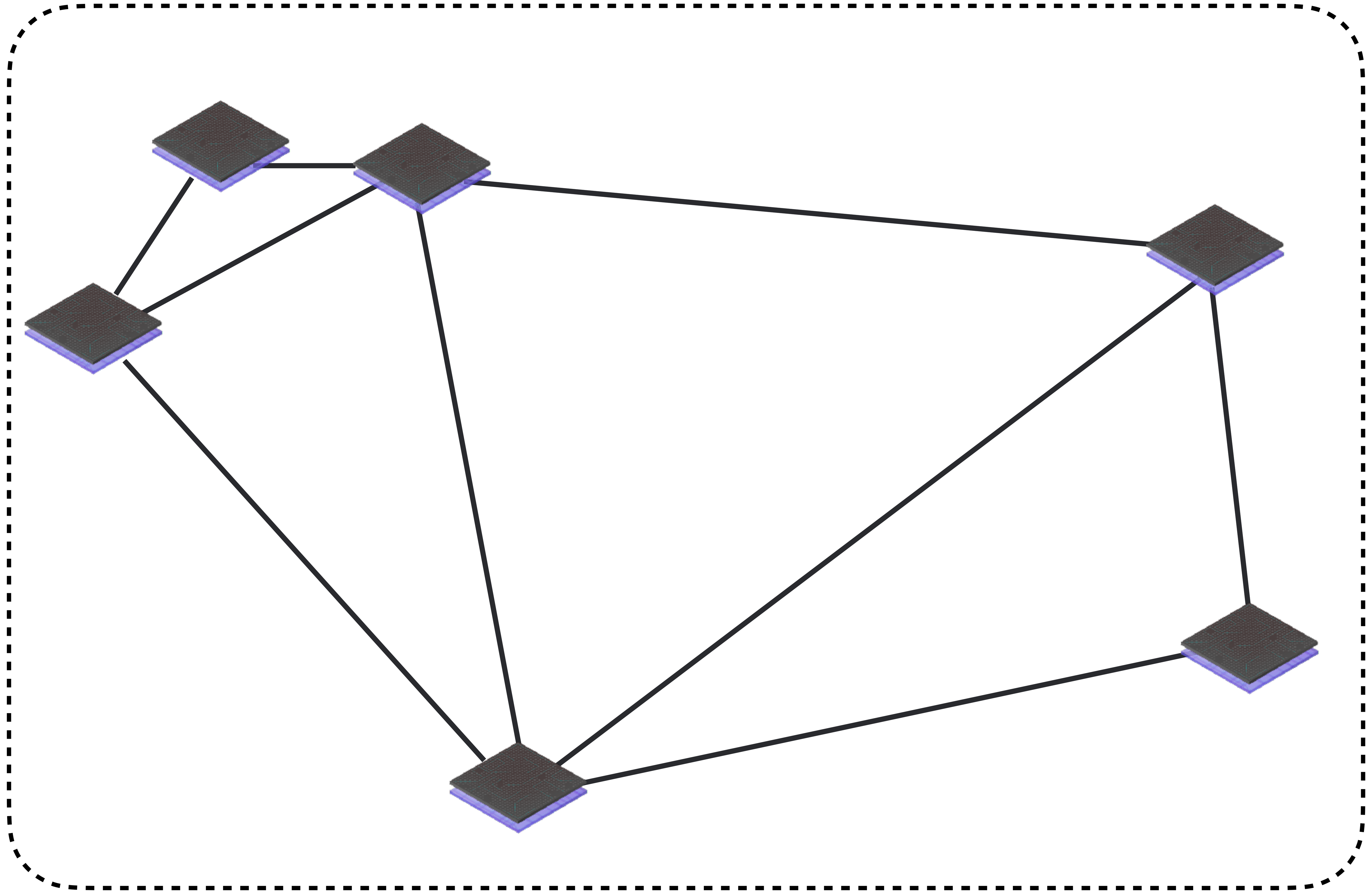


Internet Computer Consensus

Assumption: **n > 3f**

Guarantees **agreement** even under asynchrony

Guarantees **termination** under partial synchrony



Internet Computer Consensus

Jan Camenisch, Manu Drijvers, Timo Hanke,
Yvonne-Anne Pignolet, Victor Shoup, Dominic Williams

DFINITY Foundation

May 13, 2021

Abstract

We present the Internet Computer Consensus (ICC) family of protocols for atomic broadcast (a.k.a., consensus), which underpin the Byzantine fault-tolerant replicated state machines of the Internet Computer. The ICC protocols are leader-based protocols that assume partial synchrony, and that are fully integrated with a blockchain. The leader changes probabilistically in every round. These protocols are extremely simple and robust: in any round where the leader is corrupt (which itself happens with probability less than 1/3), each ICC protocol will effectively allow another party to take over as leader for that round, with very little fuss, to move the protocol forward to the next round in a timely fashion. Unlike in many other protocols, there are no complicated subprotocols (such as "view change" in PBFT) or unspecified subprotocols (such as "pacemaker" in HotStuff). Moreover, unlike in many other protocols (such as PBFT and HotStuff), the task of reliably disseminating the blocks to all parties is an integral part the protocol, and not left to some other unspecified subprotocol. An additional property enjoyed by the ICC protocols (just like PBFT and HotStuff, and unlike others, such as Tendermint) is *optimistic responsiveness*, which means that when the leader is honest, the protocol will proceed at the pace of the actual network delay, rather than some upper bound on the network delay. We present three different protocols (along with various minor variations on each). One of these protocols (ICC1) is designed to be integrated with a peer-to-peer gossip sub-layer, which reduces the bottleneck created at the leader for disseminating large blocks, a problem that all leader-based protocols, like PBFT and HotStuff, must address, but typically do not. Our Protocol ICC2 addresses the same problem by substituting a low-communication reliable broadcast subprotocol (which may be of independent interest) for the gossip sub-layer.

1 Introduction

Byzantine fault tolerance (BFT) is the ability of a computing system to endure arbitrary (i.e., Byzantine) failures of some of its components while still functioning properly as a whole. One approach to achieving BFT is via *state machine replication* [Sch90]: the logic of the system is replicated across a number of machines, each of which maintains state, and updates its state is by executing a sequence of *commands*. In order to ensure that the non-faulty machines end up in the same state, they must each deterministically execute the same sequence of commands. This is achieved by using a protocol for *atomic broadcast*.

1

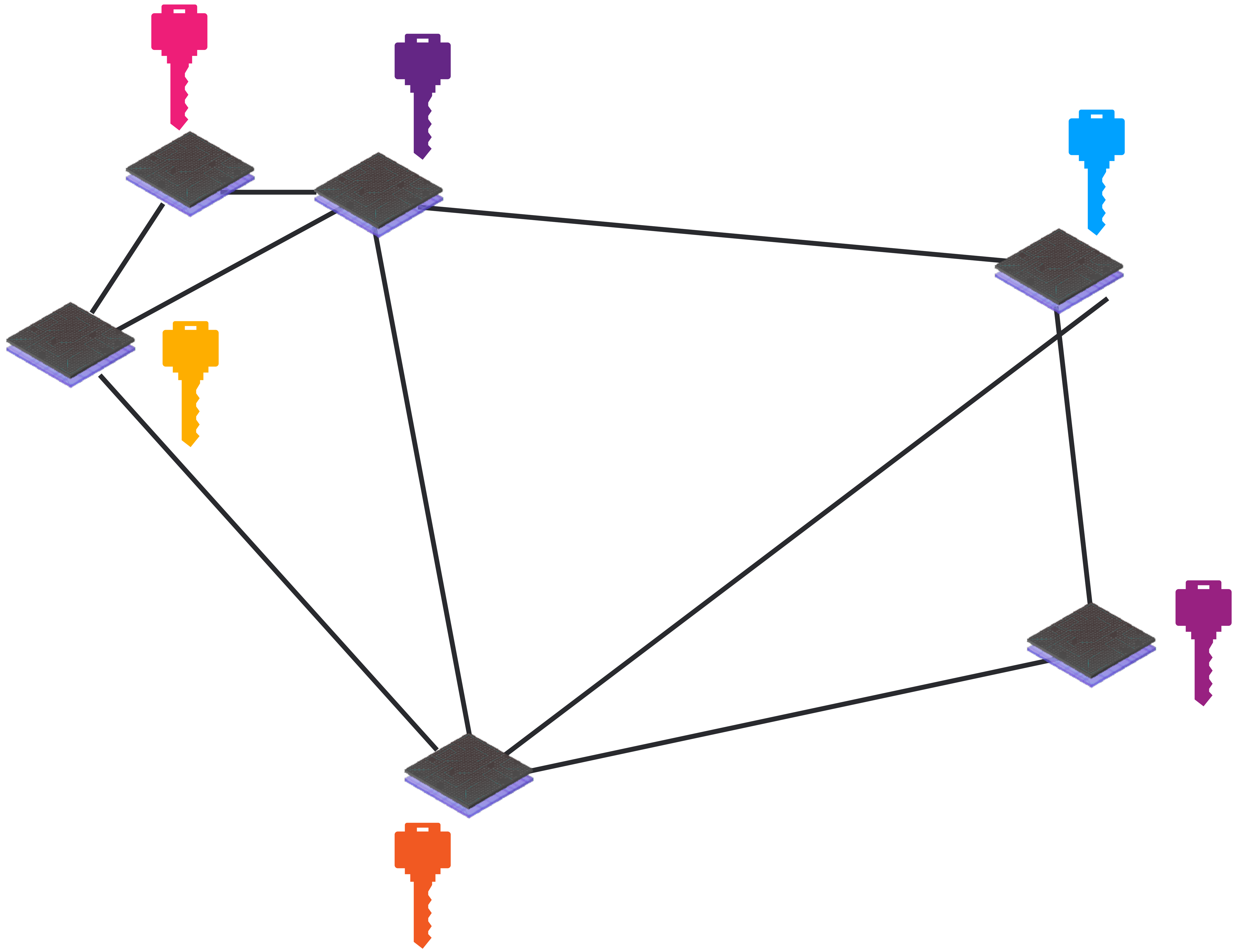
PODC'22

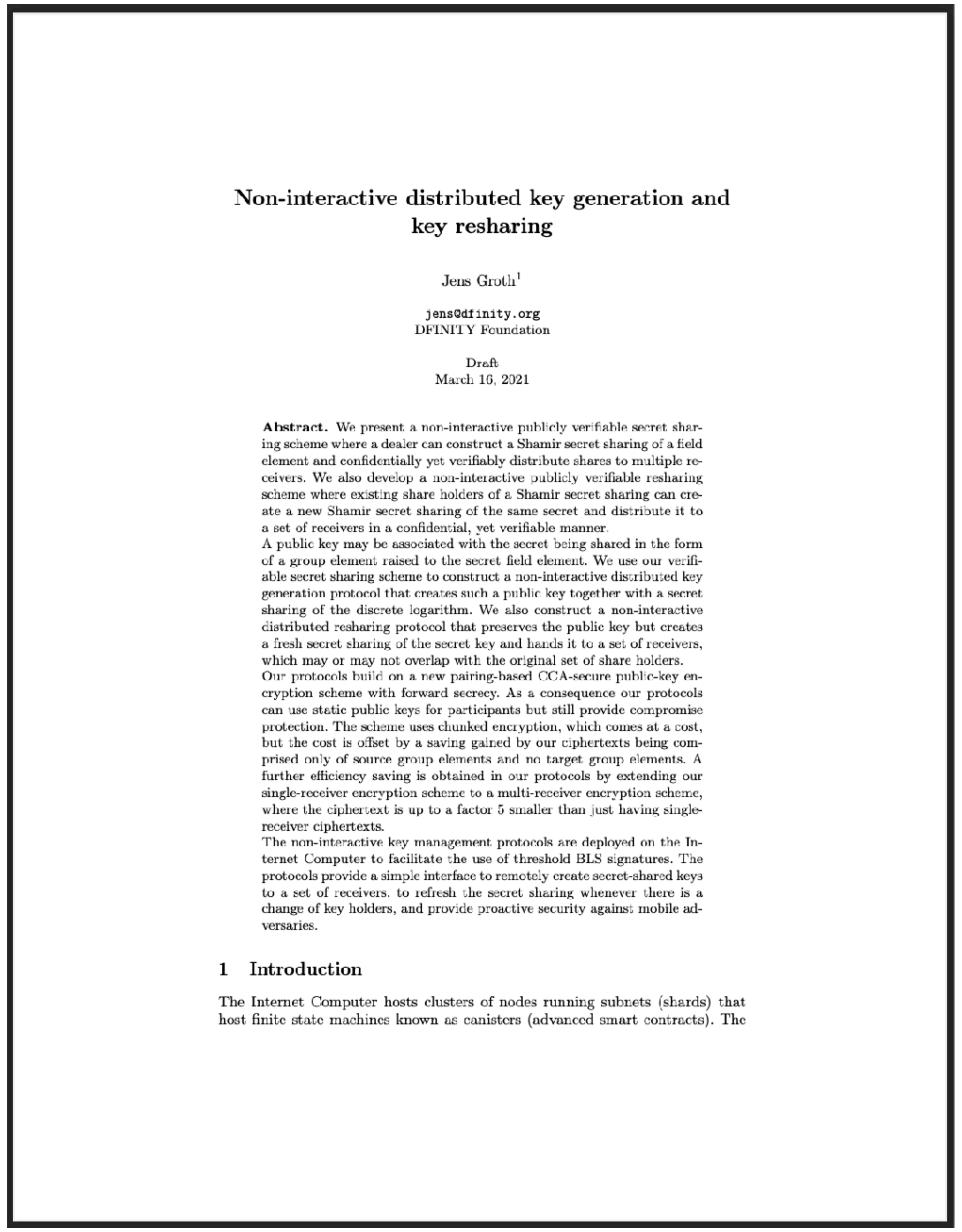
https://internetcomputer.org/how-it-works/consensus

Chain Key Cryptography

Single 48-byte public key

for a secret-shared private key





Non-interactive distributed key generation and key resharing

Jens Groth[1]

jens@dfinity.org
DFINITY Foundation

Draft
March 16, 2021

Abstract. We present a non-interactive publicly verifiable secret sharing scheme where a dealer can construct a Shamir secret sharing of a field element and confidentially yet verifiably distribute shares to multiple receivers. We also develop a non-interactive publicly verifiable resharing scheme where existing share holders of a Shamir secret sharing can create a new Shamir secret sharing of the same secret and distribute it to a set of receivers in a confidential, yet verifiable manner.
A public key may be associated with the secret being shared in the form of a group element raised to the secret field element. We use our verifiable secret sharing scheme to construct a non-interactive distributed key generation protocol that creates such a public key together with a secret sharing of the discrete logarithm. We also construct a non-interactive distributed resharing protocol that preserves the public key but creates a fresh secret sharing of the secret key and hands it to a set of receivers, which may or may not overlap with the original set of share holders.
Our protocols build on a new pairing-based CCA-secure public-key encryption scheme with forward secrecy. As a consequence our protocols can use static public keys for participants but still provide compromise protection. The scheme uses chunked encryption, which comes at a cost, but the cost is offset by a saving gained by our ciphertexts being comprised only of source group elements and no target group elements. A further efficiency saving is obtained in our protocols by extending our single-receiver encryption scheme to a multi-receiver encryption scheme, where the ciphertext is up to a factor 5 smaller than just having single-receiver ciphertexts.
The non-interactive key management protocols are deployed on the Internet Computer to facilitate the use of threshold BLS signatures. The protocols provide a simple interface to remotely create secret-shared keys to a set of receivers, to refresh the secret sharing whenever there is a change of key holders, and provide proactive security against mobile adversaries.

1 Introduction

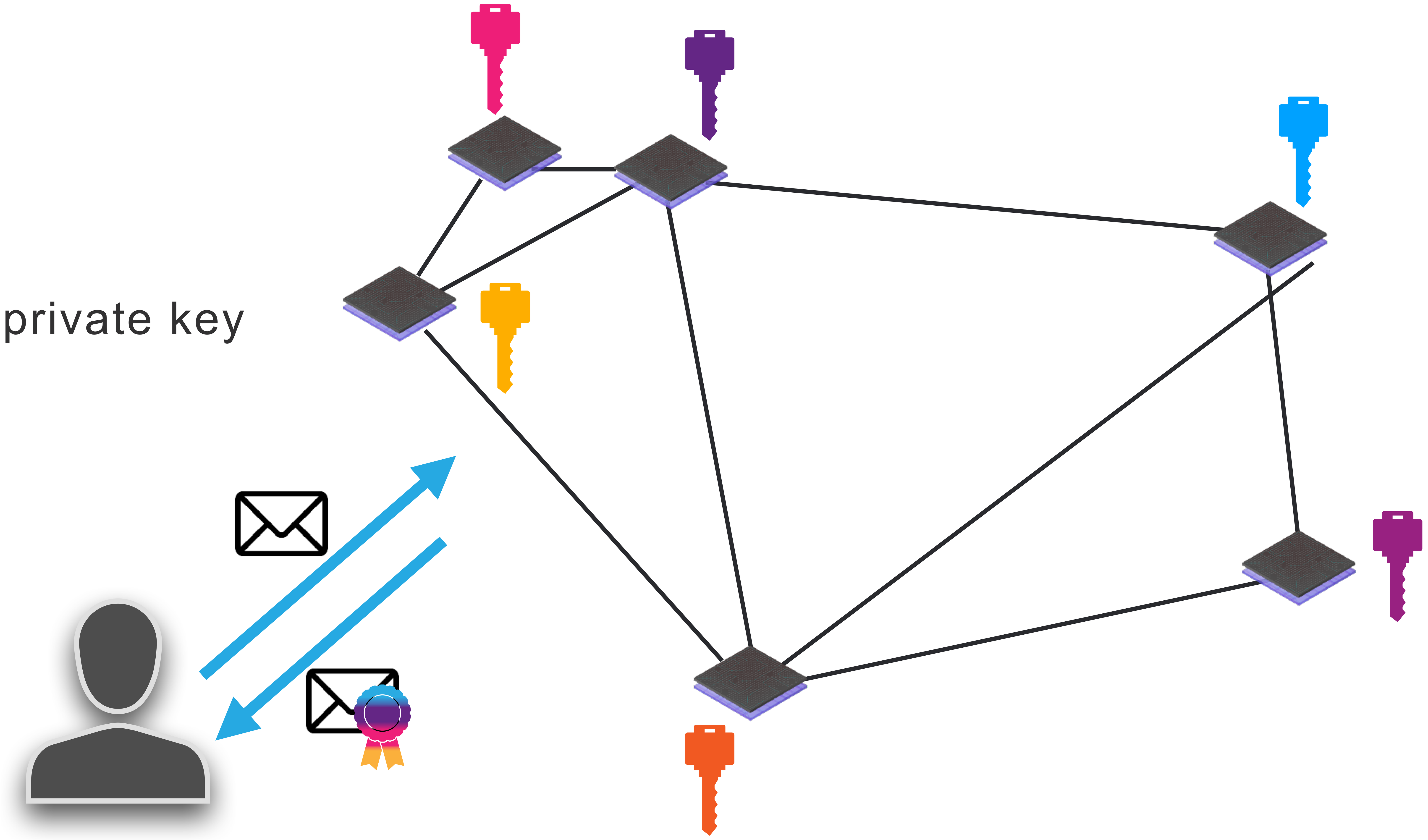
The Internet Computer hosts clusters of nodes running subnets (shards) that host finite state machines known as canisters (advanced smart contracts). The

https://internetcomputer.org/how-it-works/chain-key-technology

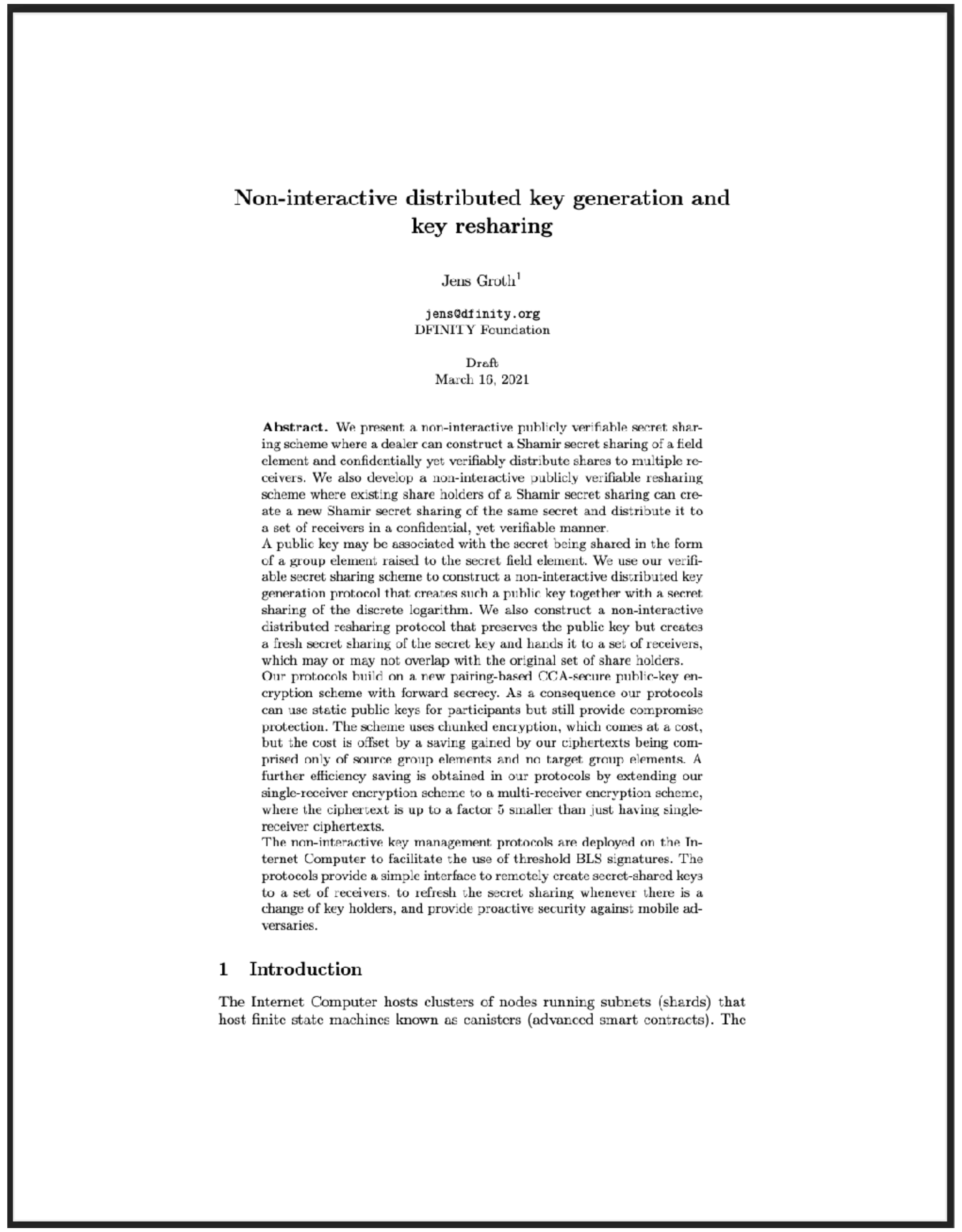
Chain Key Cryptography

Single 48-byte public key

for a secret-shared private key



verify()



Non-interactive distributed key generation and key resharing

Jens Groth[1]

jens@dfinity.org
DFINITY Foundation

Draft
March 16, 2021

Abstract. We present a non-interactive publicly verifiable secret sharing scheme where a dealer can construct a Shamir secret sharing of a field element and confidentially yet verifiably distribute shares to multiple receivers. We also develop a non-interactive publicly verifiable resharing scheme where existing share holders of a Shamir secret sharing can create a new Shamir secret sharing of the same secret and distribute it to a set of receivers in a confidential, yet verifiable manner.
A public key may be associated with the secret being shared in the form of a group element raised to the secret field element. We use our verifiable secret sharing scheme to construct a non-interactive distributed key generation protocol that creates such a public key together with a secret sharing of the discrete logarithm. We also construct a non-interactive distributed resharing protocol that preserves the public key but creates a fresh secret sharing of the secret key and hands it to a set of receivers, which may or may not overlap with the original set of share holders.
Our protocols build on a new pairing-based CCA-secure public-key encryption scheme with forward secrecy. As a consequence our protocols can use static public keys for participants but still provide compromise protection. The scheme uses chunked encryption, which comes at a cost, but the cost is offset by a saving gained by our ciphertexts being comprised only of source group elements and no target group elements. A further efficiency saving is obtained in our protocols by extending our single-receiver encryption scheme to a multi-receiver encryption scheme, where the ciphertext is up to a factor 5 smaller than just having single-receiver ciphertexts.
The non-interactive key management protocols are deployed on the Internet Computer to facilitate the use of threshold BLS signatures. The protocols provide a simple interface to remotely create secret-shared keys to a set of receivers, to refresh the secret sharing whenever there is a change of key holders, and provide proactive security against mobile adversaries.

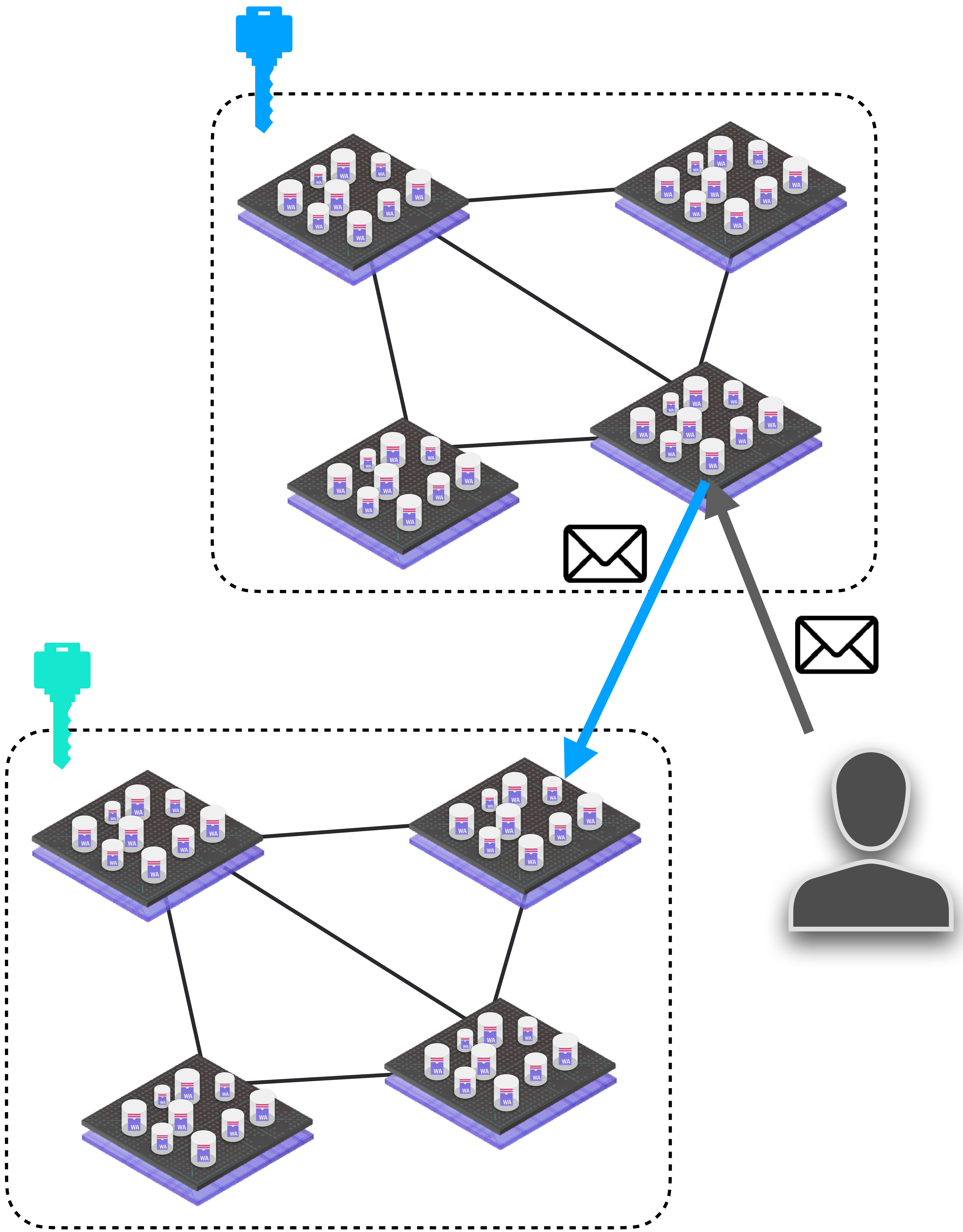
1 Introduction

The Internet Computer hosts clusters of nodes running subnets (shards) that host finite state machines known as canisters (advanced smart contracts). The

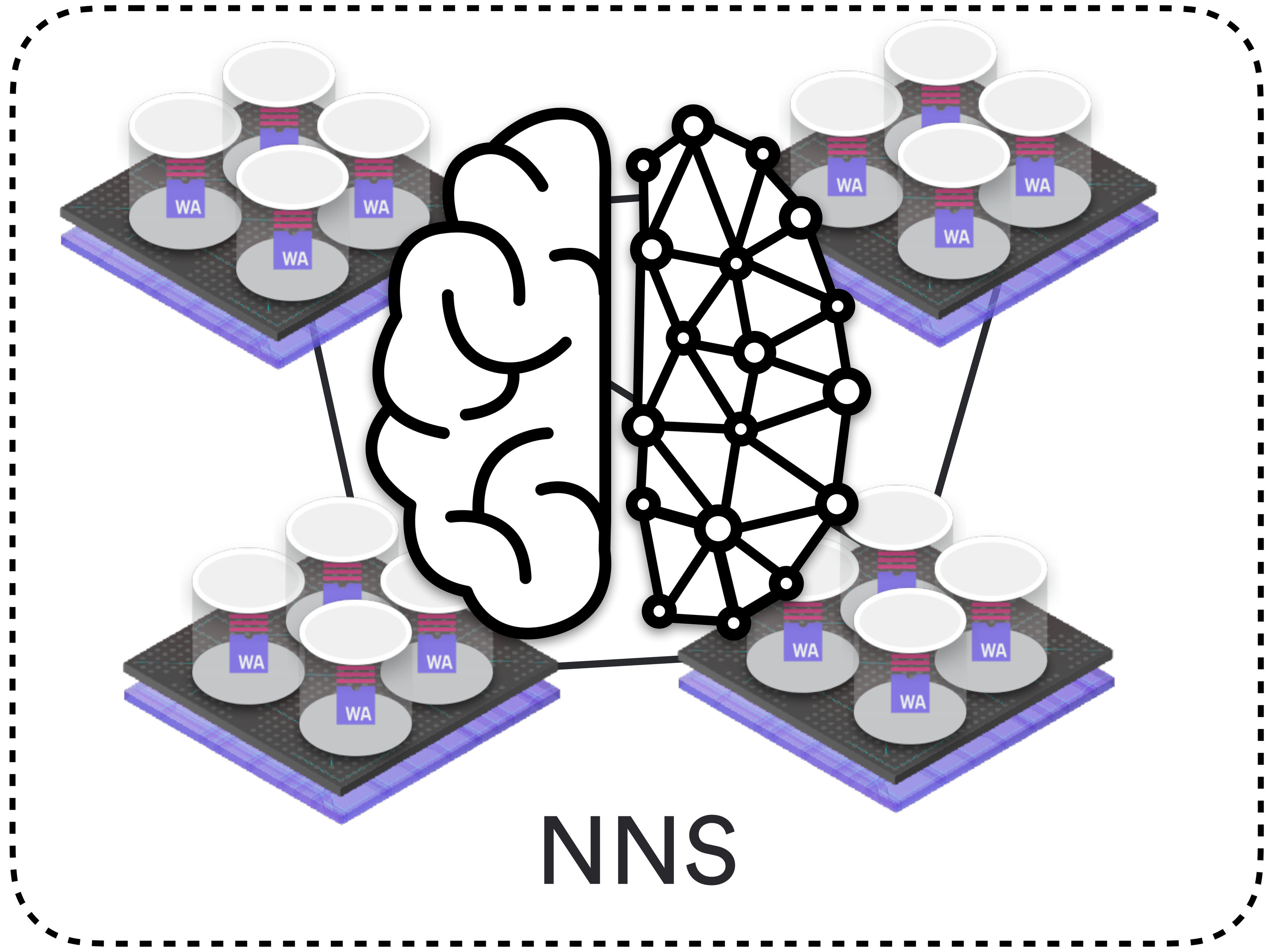
https://internetcomputer.org/how-it-works/chain-key-technology

Subnets for Scalability

- Each canister is assigned to one subnet
- Each subnet is a **replicated state machine**
- A canister can call canisters on other subnets
- Subnets make the Internet Computer **scalable**!

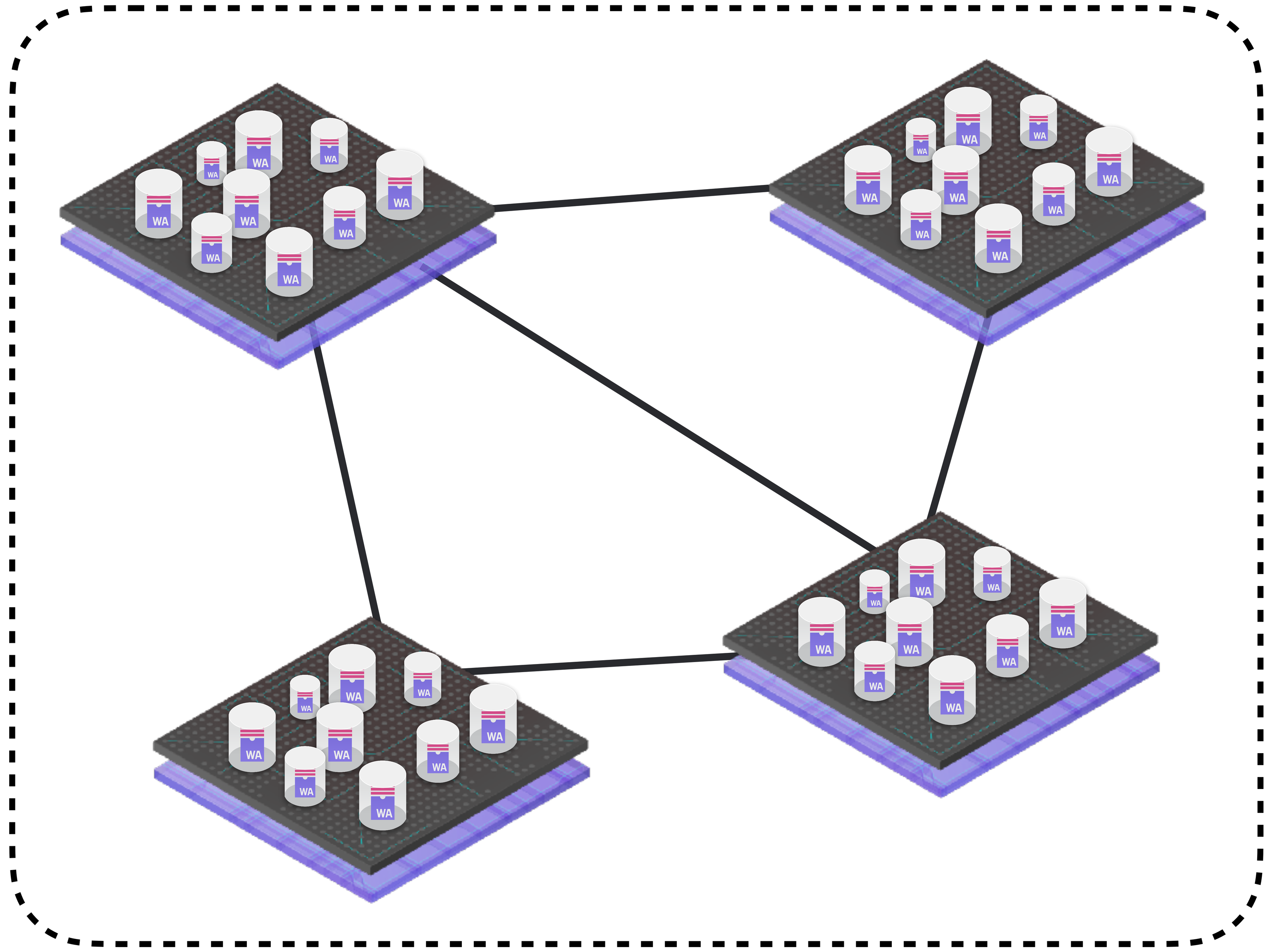


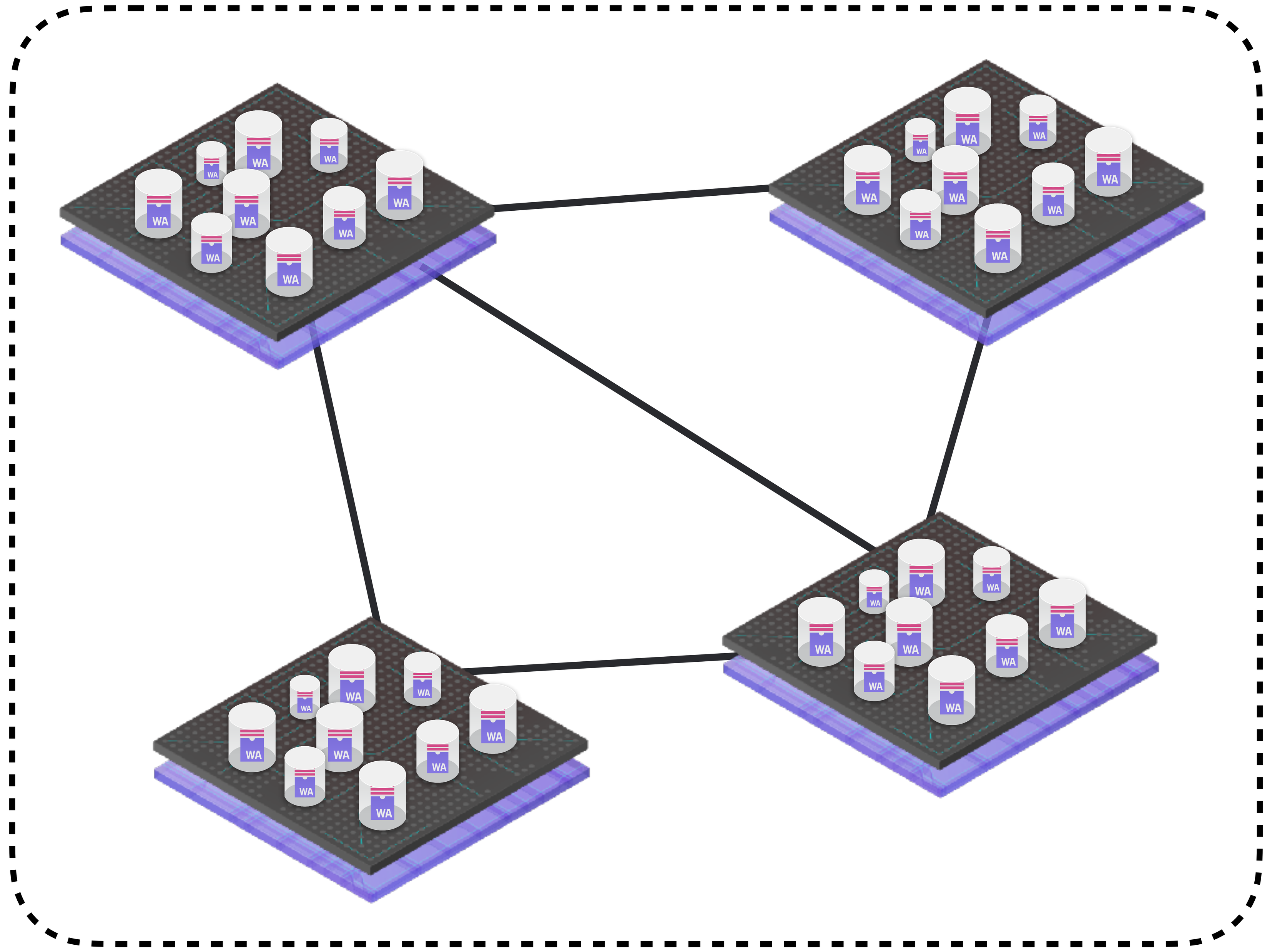
Governance: Network Nervous System



One subnet is special: it host the **Network Nervous System (NNS)** canisters which govern the IC

ICP token holders vote on
- Creation of new subnets
- Upgrades to new protocol version
- Replacement of nodes
- …



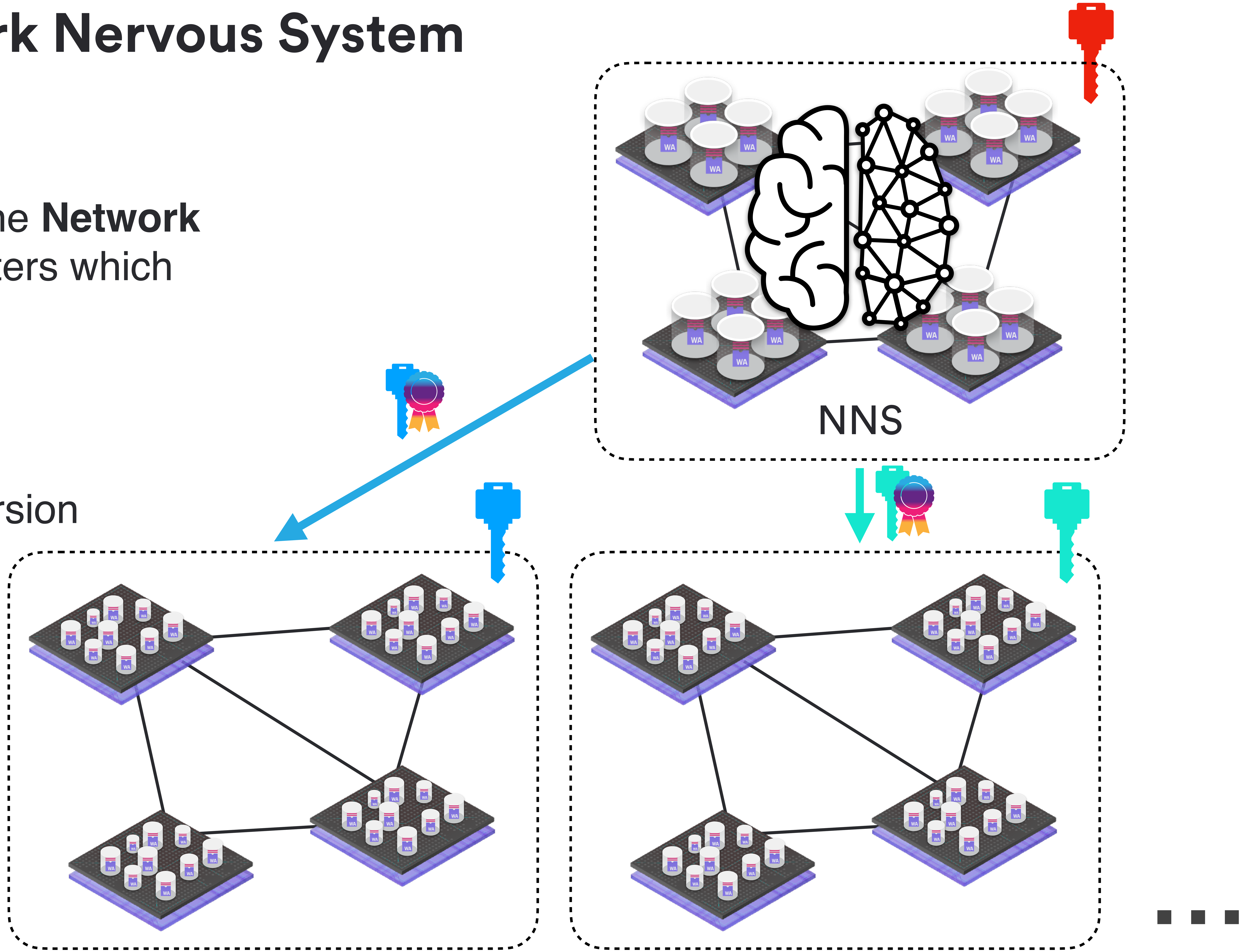


https://internetcomputer.org/nns

Governance: Network Nervous System

One subnet is special: it host the **Network Nervous System (NNS)** canisters which govern the IC

ICP token holders vote on
- Creation of new subnets
- Upgrades to new protocol version
- Replacement of nodes
- …

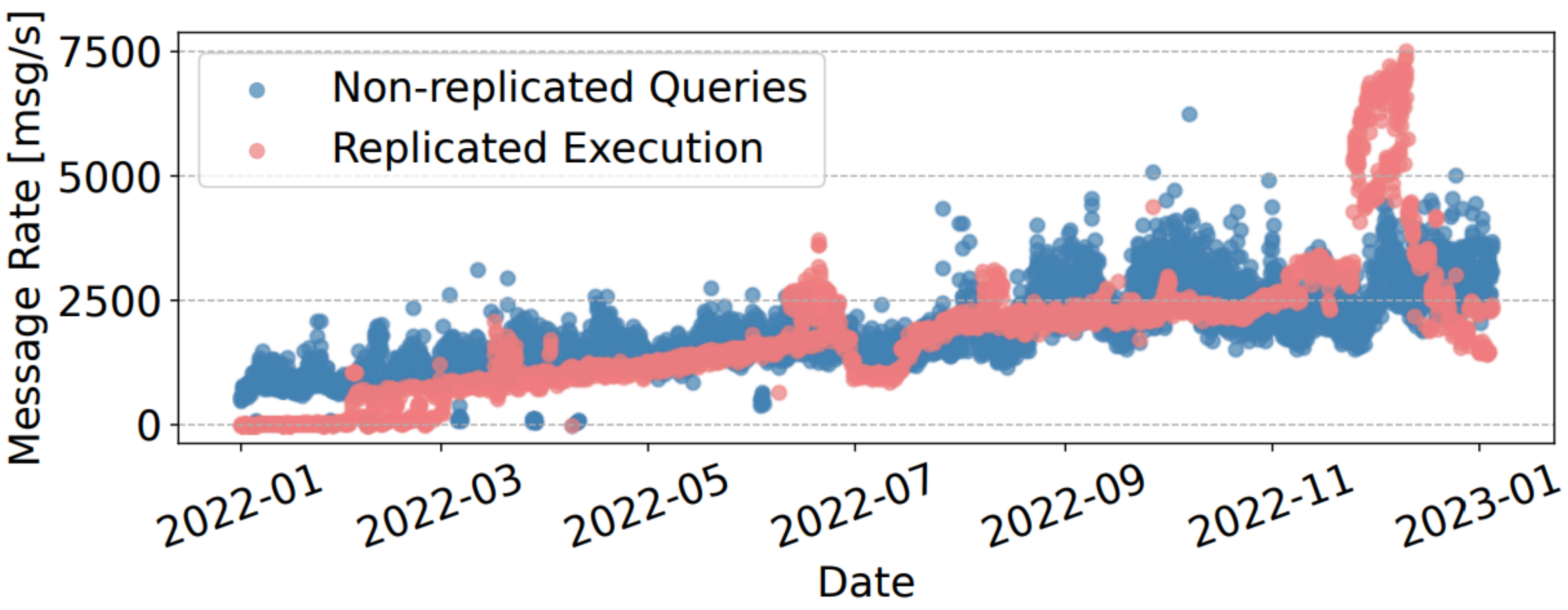


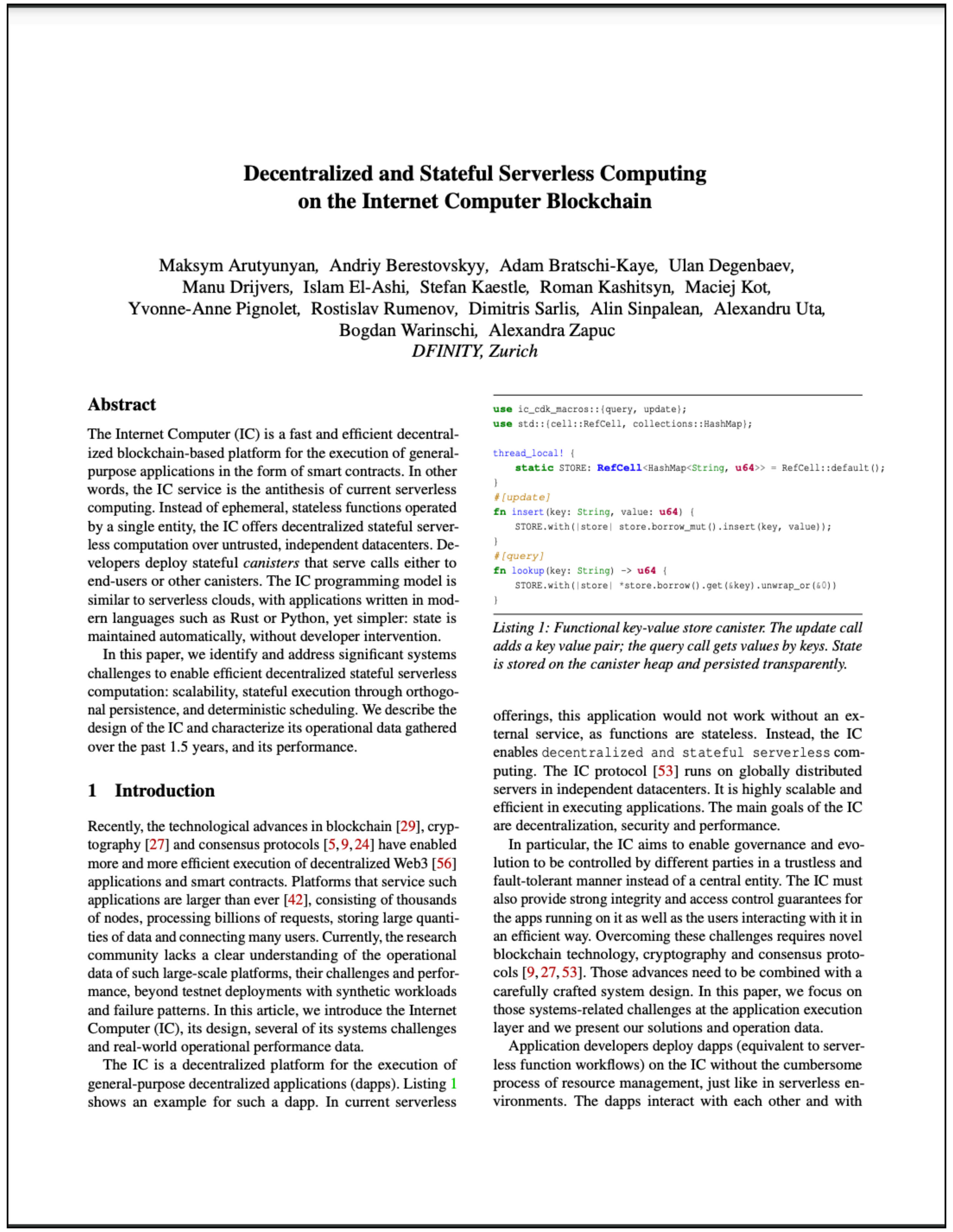
https://internetcomputer.org/nns

Execution layer

Systems challenges:
- statefulness (orthogonal persistence)
- scalability (caching, time slicing)
- determinism (scheduling)
- security (sandboxing, accounting)

=> Alternative and improvement upon current *centralized & stateless* serverless paradigm





Decentralized and Stateful Serverless Computing on the Internet Computer Blockchain

Maksym Arutyunyan, Andriy Berestovskyy, Adam Bratschi-Kaye, Ulan Degenbaev, Manu Drijvers, Islam El-Ashi, Stefan Kaestle, Roman Kashitsyn, Maciej Kot, Yvonne-Anne Pignolet, Rostislav Rumenov, Dimitris Sarlis, Alin Sinpalean, Alexandru Uta, Bogdan Warinschi, Alexandra Zapuc
DFINITY, Zurich

Abstract

The Internet Computer (IC) is a fast and efficient decentralized blockchain-based platform for the execution of general-purpose applications in the form of smart contracts. In other words, the IC service is the antithesis of current serverless computing. Instead of ephemeral, stateless functions operated by a single entity, the IC offers decentralized stateful serverless computation over untrusted, independent datacenters. Developers deploy stateful *canisters* that serve calls either to end-users or other canisters. The IC programming model is similar to serverless clouds, with applications written in modern languages such as Rust or Python, yet simpler: state is maintained automatically, without developer intervention.

In this paper, we identify and address significant systems challenges to enable efficient decentralized stateful serverless computation: scalability, stateful execution through orthogonal persistence, and deterministic scheduling. We describe the design of the IC and characterize its operational data gathered over the past 1.5 years, and its performance.

1 Introduction

Recently, the technological advances in blockchain [29], cryptography [27] and consensus protocols [5, 9, 24] have enabled more and more efficient execution of decentralized Web3 [56] applications and smart contracts. Platforms that service such applications are larger than ever [42], consisting of thousands of nodes, processing billions of requests, storing large quantities of data and connecting many users. Currently, the research community lacks a clear understanding of the operational data of such large-scale platforms, their challenges and performance, beyond testnet deployments with synthetic workloads and failure patterns. In this article, we introduce the Internet Computer (IC), its design, several of its systems challenges and real-world operational performance data.

The IC is a decentralized platform for the execution of general-purpose decentralized applications (dapps). Listing 1 shows an example for such a dapp. In current serverless

```
use ic_cdk_macros::{query, update};
use std::{cell::RefCell, collections::HashMap};

thread_local! {
    static STORE: RefCell<HashMap<String, u64>> = RefCell::default();
}
#[update]
fn insert(key: String, value: u64) {
    STORE.with(|store| store.borrow_mut().insert(key, value));
}
#[query]
fn lookup(key: String) -> u64 {
    STORE.with(|store| *store.borrow().get(&key).unwrap_or(&0))
}
```

Listing 1: Functional key-value store canister. The update call adds a key value pair; the query call gets values by keys. State is stored on the canister heap and persisted transparently.

offerings, this application would not work without an external service, as functions are stateless. Instead, the IC enables `decentralized and stateful serverless` computing. The IC protocol [53] runs on globally distributed servers in independent datacenters. It is highly scalable and efficient in executing applications. The main goals of the IC are decentralization, security and performance.

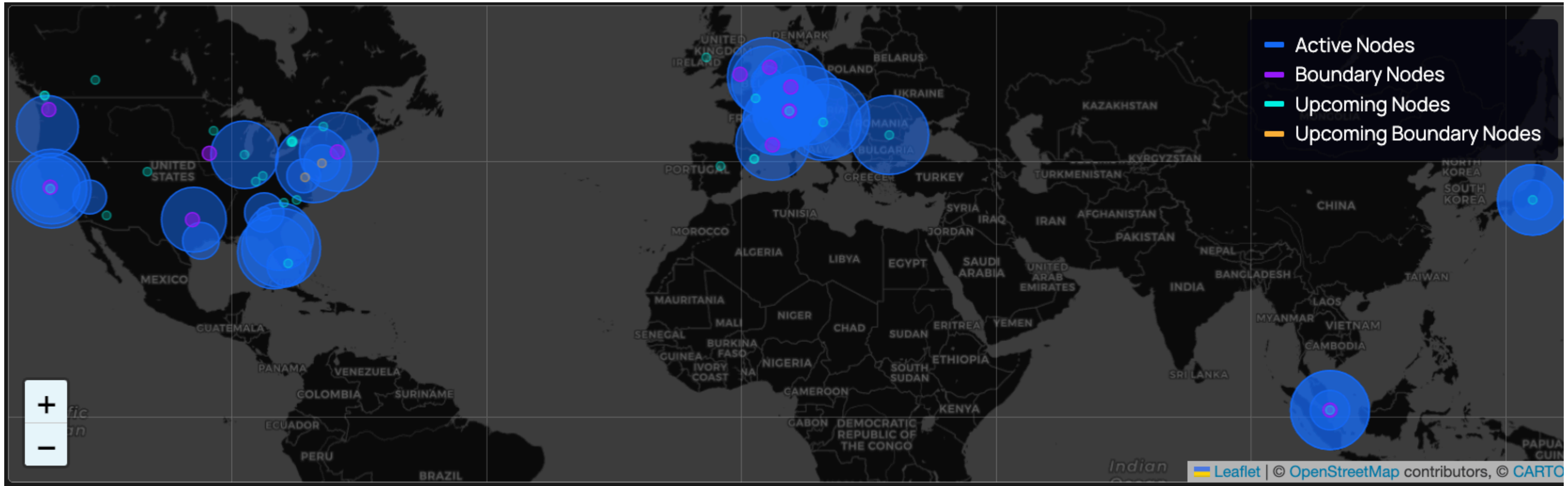
In particular, the IC aims to enable governance and evolution to be controlled by different parties in a trustless and fault-tolerant manner instead of a central entity. The IC must also provide strong integrity and access control guarantees for the apps running on it as well as the users interacting with it in an efficient way. Overcoming these challenges requires novel blockchain technology, cryptography and consensus protocols [9, 27, 53]. Those advances need to be combined with a carefully crafted system design. In this paper, we focus on those systems-related challenges at the application execution layer and we present our solutions and operation data.

Application developers deploy dapps (equivalent to serverless function workflows) on the IC without the cumbersome process of resource management, just like in serverless environments. The dapps interact with each other and with

Usenix ATC'23

The Internet Computer Today

Live Since May 2021!



Total Canister State	3.28 TB
Node Providers	76
Subnets	36
Boundary Nodes	17
Total Nodes	1’235
Nodes in Subnets	549
Internet Identity Anchors	2’151’186

Total Supply	495’958’980 ICP
Circulating Supply	291’335’534 ICP
ICP Transactions	5’690’496
Burn ICP Txns	100’004 ICP
Burn Fees	506 ICP
NNS Proposals	110’377
Community Fund Neurons’ Stake	20’527’909 ICP

https://dashboard.internetcomputer.org

Comparison with other Blockchain Systems

	Ethereum	Cardano	Solana	Avalanche	Algorand	Internet Computer
Avg # TX/s	14.4	2.95	381	49.52	15.5	5000
Avg finality	15min	-	5-12.8s	2.3s	3.5s	1.4s
Wh / TX	-	51.59	0.166	4.76	2.7	0.008
1GB storage	15M$	17-113k$	48k$	200k$	off-chain storage	5$

https://newsbtc.com/all/assessing-the-top-performing-layer-1-blockchain-protocols/,

see also https://wiki.internetcomputer.org/wiki/L1_comparison

Internet Computer Ecosystem

Internetcomputer Ecosystem

See dapps running on ICP

Evolution


IC DASHBOARD
Search ⌘/
Home
Governance
Neurons
Network
Releases
Canisters
ICP
SNS
Bitcoin
Releases
142
Mar Apr May Jun Jul Aug Sept Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri
Subnet Upgrades
2’710
Mar Apr May Jun Jul Aug Sept Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri
Less
More


IC DASHBOARD
Search ⌘/
Home
Governance
Neurons
Network
Releases
Canisters
ICP
SNS
Bitcoin
Releases
142
Mar Apr May Jun Jul Nov Dec Jan Feb Mar
Mon
Wed
Fri
Subnet Upgrades
2’710
Mar Apr May Jun Jul Nov Dec Jan Feb Mar
Mon
Wed
Fri
Less
More
WA
HTTP
HTTPS
IoT


IC DASHBOARD
Search ⌘/
Home
Governance
Neurons
Network
Releases
Canisters
ICP
SNS
Bitcoin
Releases
142
Mar Apr May Jun Jul Aug Sept Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri
Subnet Upgrades
2’710
Mar Apr May
Feb Mar
Mon
Wed
Fri
Less More

Decentralised Upgrade Challenges

How to

- Select version to upgrade to?
- Ensure all nodes in a subnet know about new version?
- And switch to the new version at the same time?
- And minimise time without processing messages?
- And minimise compatibility risks?

https://nns.ic0.app/proposals/
Incognito
NETWORK NERVOUS SYSTEM
My Tokens
My Neuron Staking
Vote on Proposals
Launch Pad
My Canisters
$1'382'072'000
Total ICP Value Locked
Vote on Proposals
Topics (6/13)
Reward Status (4/4)
Proposal Status (2/5)
112617
Type Add or Remove Node Provider
Topic Participant Management
Proposer 62985...00523
" Add node provider: niw4y-easue-l3qvz-sozsi-tfkvb-cxcx6-pzslg-5dqld-ooudp-hsuui-xae
Executed 1 day, 3 hours remaining
112386
Type NNS Canister Upgrade
Topic System Canister Management
Proposer 59
" Upgrade Nns Canister: qoctq-giaaa-aaaaa-aaaea-cai to wasm with hash: 87743bc2e1ed4c1739bd2073fcb54674ed2db2fe1022e3e7a945fb803bfaf72f
Executed
111932
Type Bless Replica Version
Topic Replica Version Management
Proposer 46
" Elect new replica binary revision (commit 8487a2be2a0a1d05843d03f07079d97ea782d440)
Executed
111901
Type NNS Canister Upgrade
Topic System Canister Management
Proposer 70
" Upgrade Nns Canister: mqygn-kiaaa-aaaar-qaadq-cai to wasm with hash: 9525c491b534a854d31624ac36d155befd52acc607f5ae5316715027c70a351a
Executed
111724
Type Bless Replica Version
Topic Replica Version Management
Proposer 40
" Elect new replica binary revision (commit 9fde647b04e9994c11207a6529148d5f9d5ae895)
Executed
111717
Type NNS Canister Upgrade
Topic System Canister Management
Proposer 73
" Upgrade Nns Canister: rdmx6-jaaaa-aaaaa-aaadq-cai to wasm with hash: 38b54cb8b8cc6e7ee3cf0c028461f5f351f80fad23dd143b605c036f46ba2a01
Executed

NNS Canister Upgrade

Type	NNS Canister Upgrade
Topic	System Canister Management
Status	Executed
Reward Status	Ready to Settle
Created	Mar 13, 2023 11:19 AM
Decided	Mar 13, 2023 11:29 AM
Executed	Mar 13, 2023 11:29 AM
Proposer	59

Voting Results

Adopt 437'303'219.45

Reject 601.93

Sign in to vote

Proposal Summary

Upgrade Nns Canister: qoctq-giaaa-aaaaa-aaaea-cai to wasm with hash: 87743bc2e1ed4c1739bd2073fcb54674ed2db2fe1022e3e7a945fb803bfaf72f

---

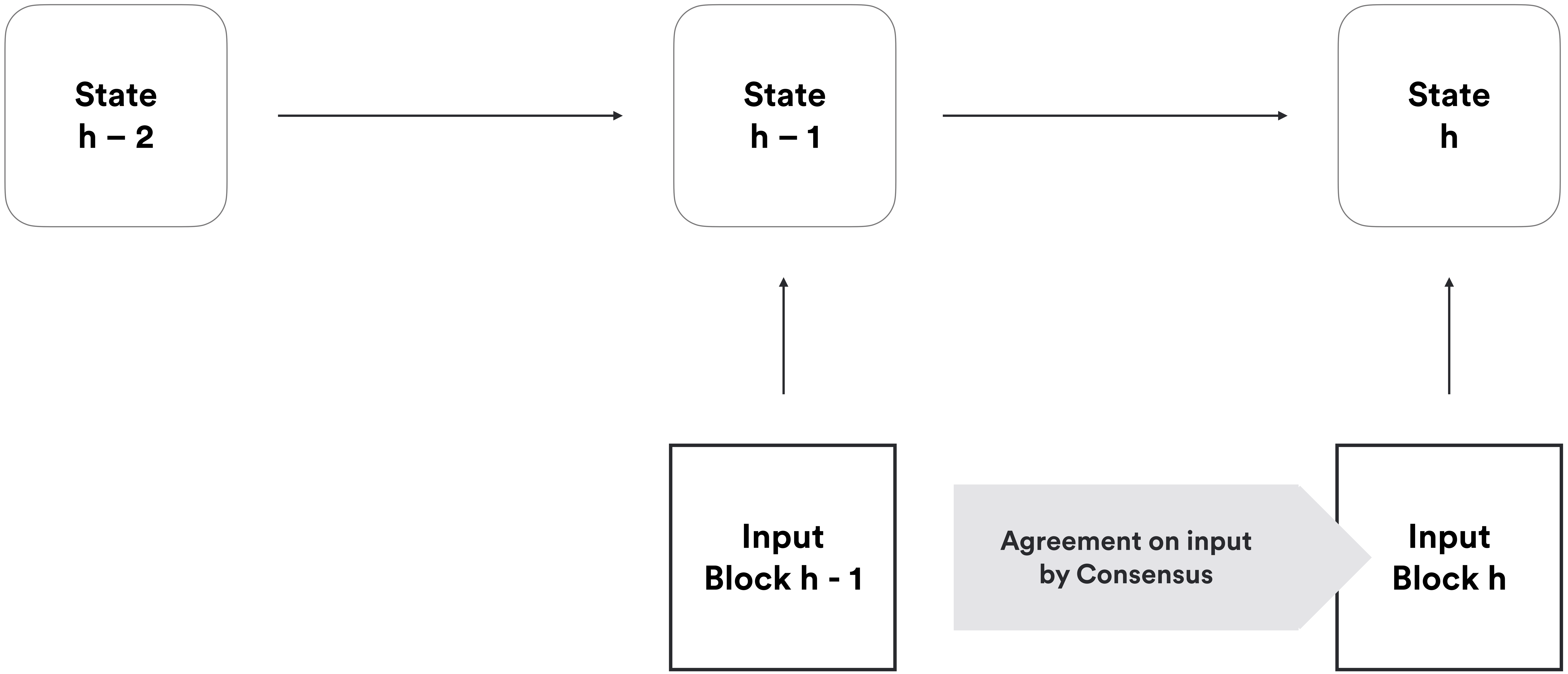
Upgrade frontend NNS Dapp canister to commit `0733e33fc64001e8904497a388c40516e57c1304`

Wasm sha256 hash: `87743bc2e1ed4c1739bd2073fcb54674ed2db2fe1022e3e7a945fb803bfaf72f` (`https://github.com/dfinity/nns-dapp/pull/2076/checks`)

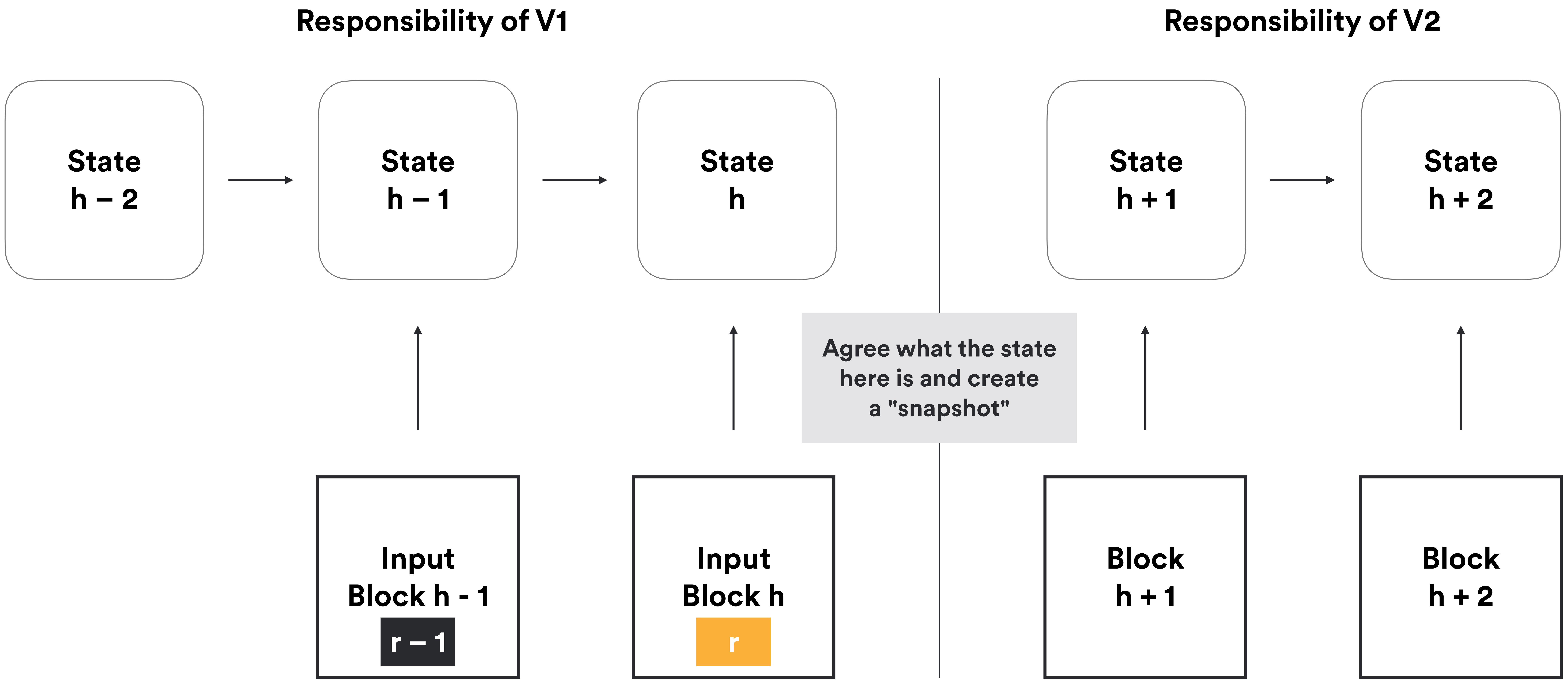
Change Log:

- Do not allow increasing stake for CF SNS neurons.
- Improve validations in address inputs.

Background: State Machine Replication



IC Versions Have Non-Overlapping Responsibilities



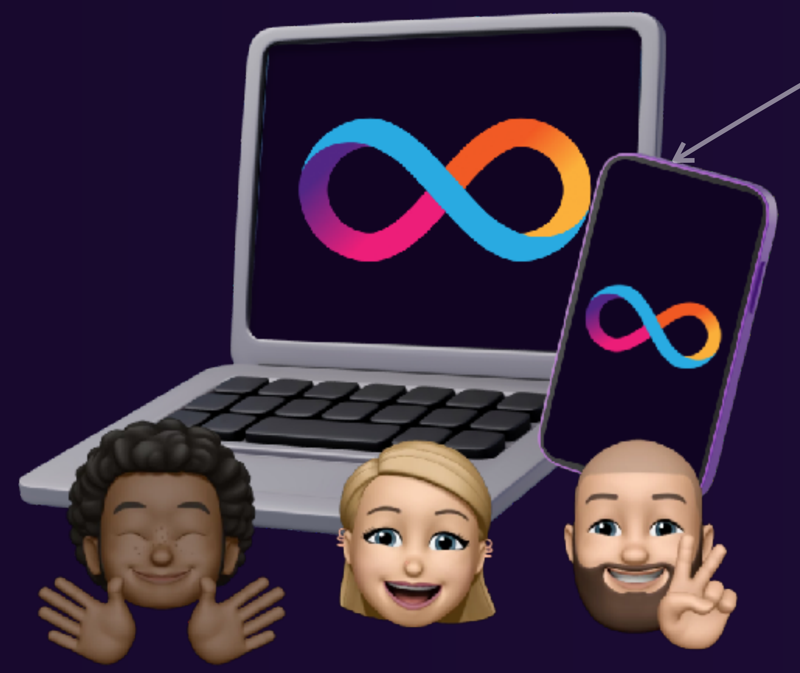
Decentralised Upgrade Challenges

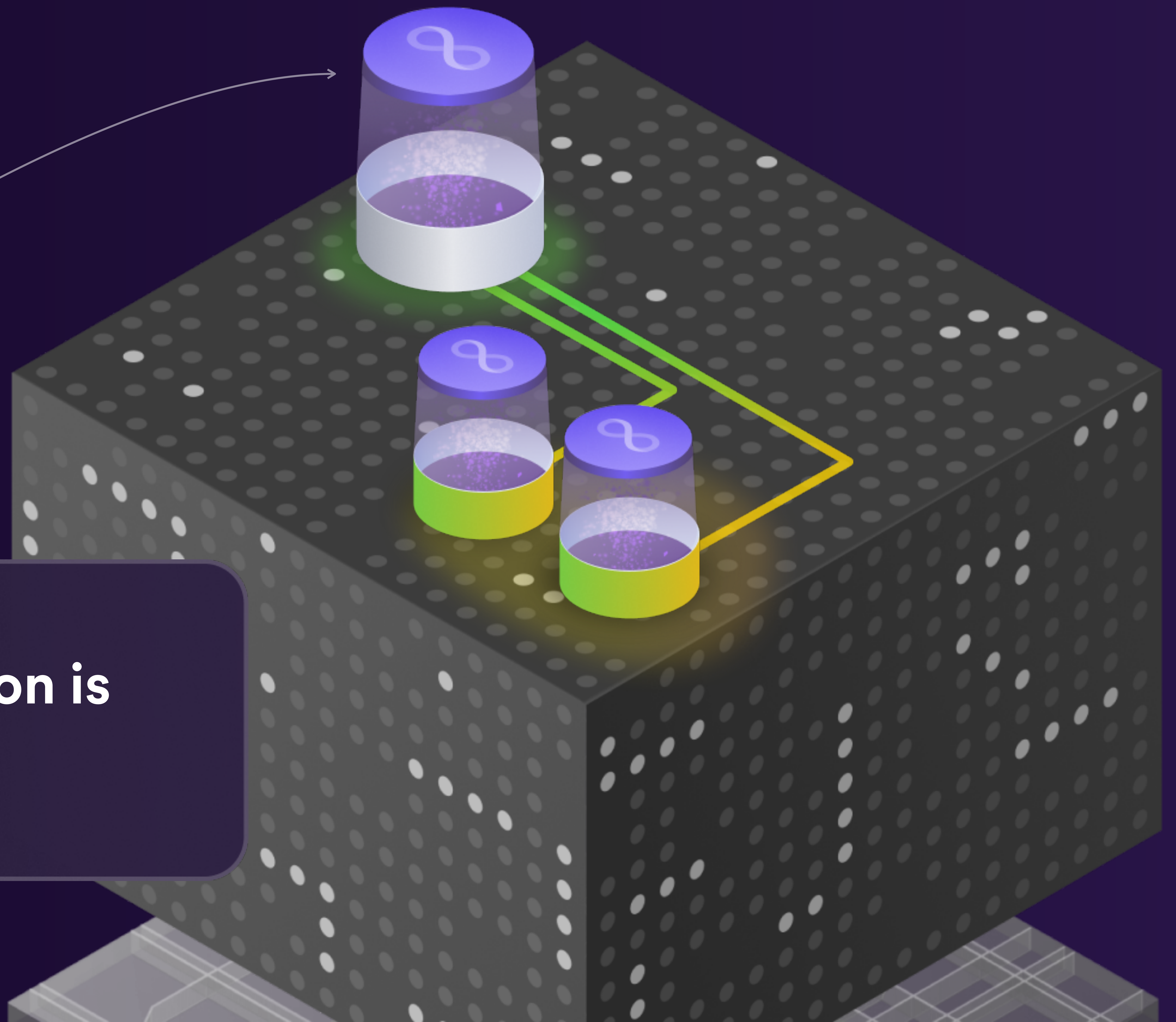
- Select version to upgrade to?
  - ✅ NNS-based community voting
- Ensure all nodes in a subnet know about new version?
  - ✅ Store version in NNS canister, nodes poll this canister
- And switch to the new version at the same time?
  - ✅ consensus on next version to use and at which height to switch
- And maximise time processing messages?
  - ✅ state snapshot on previous version,
    read-only until finalization of state from last block with old version
    A/B partition reboot, persist state
- And minimise compatibility risks?
  - ✅ simplicity > performance, extensive automated testing

DAOize your dapp

Game changer: DAO Factory

1. Proposal to turn dapp into DAO
2. ICP creates dapp token
3. ICP initiates decentralisation swap
4. Anyone who buys tokens becomes DAO participant
5. DAO fully controls dapp





Hot or Not DAO creation is ongoing right now

Take aways

The Internet Computer can

- Run rich canister smart contracts
- Serve requests at web speed
- Upgrade itself based on community votes
- DAOize apps

IC code: https://github.com/dfinity/ic
Dashboard: https://dashboard.internetcomputer.org/
Dataset API: https://ic-api.internetcomputer.org/api