






# Midterm-talk-notes

 Date	
 Slides	
  Courses	<u>IDP</u>
 Lecture video link	

## Introduction

Consensus algorithm

NNS - Network Nervous System

Subnets

Replicas

Canisters

## Project Vision

## Experiments/Workloads

## Stages

### Stage 1

Local deployment

Ledger canister

Rosetta

Findings

# Introduction

The Internet Computer (IC) works very differently from other blockchains and is powered by advanced new cryptography. Internally, the network is able to strictly limit the replication of data and computation, while still providing the liveness and security guarantees expected of a blockchain. Hence it is important to first understand the inner workings of the

## Consensus algorithm

IC is formed by a sovereign network of standardized “node machine” hardware operated by independent parties. To participate, nodes must produce the same number of blocks as others in their cohort without deviation. In a scheme of Proof-of-Useful-Work (PoUW), replicated smart contract computation is their work, driving optimal efficiency. Nodes form into subnet blockchains and then into a unified limitler, when we moved to the neess blockchain, using game-changing Chain Key Crypto.

More info about their consensus algorithm.

Proof-of-useful-work (PoUW) involves a blockchain being produced by dedicated hardware called "node machines" that are of very similar, standardized specifications. These run highly sophisticated consensus protocols that lean into the power of advanced cryptography, often referred to as Chain Key Crypto. PoUW is concerned with membership in the network.

Naturally, as per PoW, the purchase, hosting and operation of node machine hardware acts as the stake. However, these machines don't do hashing, and simply produce and process blocks of transactions that represent smart contract computations. The reason that combined node machines must be built to the same standardized specification, is that rather than compete to perform hashing, they must try not to "statistically deviate" by producing more or less blocks. In essence, rather than trying to perform more computation, they try to perform the same amount of computation, and can be punished for deviating from the group.

## NNS - Network Nervous System

A key ingredient of the scheme is the Network Nervous System (or NNS), a sophisticated permissionless DAO that is integrated with the Internet Computer's protocols. This fully controls the network, configuring it, and upgrading the software run by node machines. Among its responsibilities, it combines node machines to create "subnet blockchains," which themselves are combined into a single blockchain using Chain Key Crypto.

This achieves two important things.

1. Firstly, expense aside, it is not possible for an adversary simply to add nodes to a subnet blockchain, since the NNS carefully selects nodes by looking at the node provider, the data center the node is installed within, and its geography and jurisdiction, in a scheme of "deterministic decentralization."
2. Secondly, the NNS can remove (or "slash") nodes that statistically deviate.

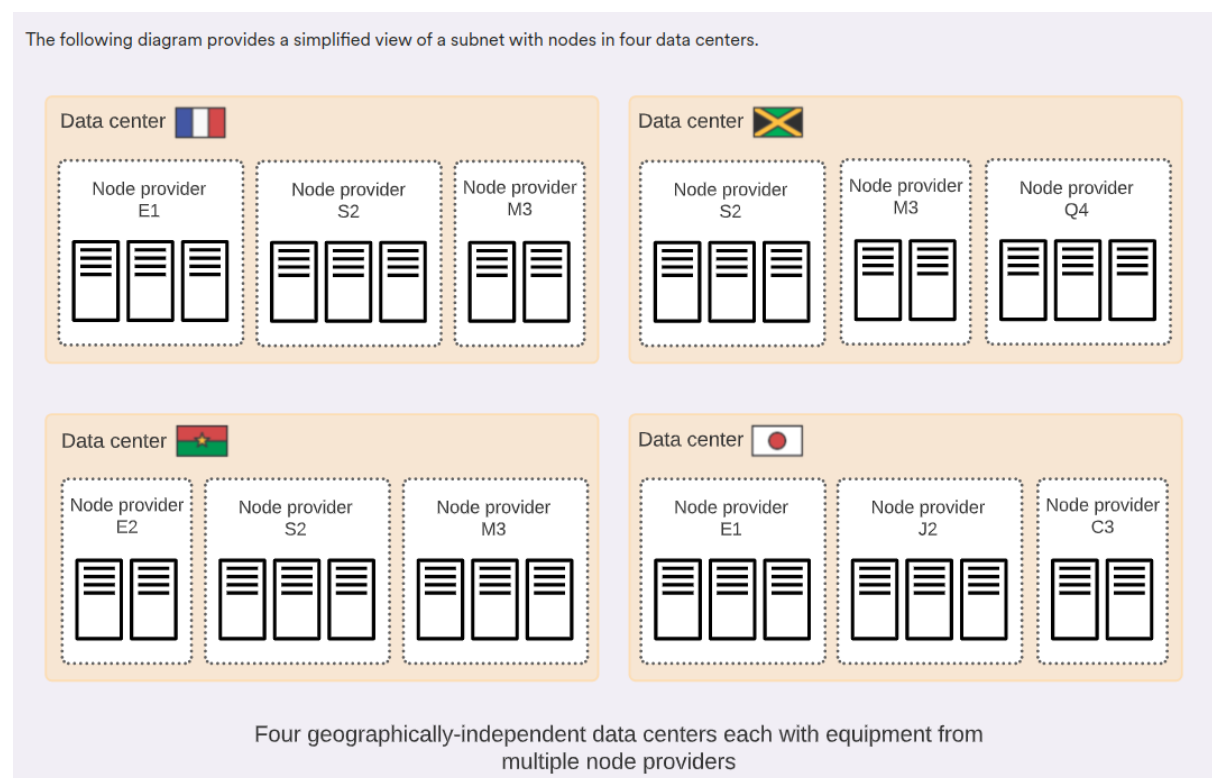
By applying **deterministic decentralization**, the NNS creates a highly secure scheme in which the Internet Computer runs on a sovereign network of dedicated hardware formed from node machines, which machinery can be tightly held to correct behavior in order to continue its participation in block production (through

which its owners, the node providers, earn rewards). In PoUW, the repetitive hashing work of PoW, whose purpose relates primarily to network operation, has been replaced by useful smart contract computation. Since this is work that must be performed anyway, a supremely efficient network is produced.

This brings us to the concept of Subnets and replica sets.

## Subnets

A so-called subnet is a collection of replicas that run a separate instance of the consensus mechanism to create their own blockchain on which a set of canisters can run. Each subnet can communicate with other subnets and is controlled by the root subnet, which uses chain key cryptography to delegate its authority to the various subnets.



## Replicas

There can be horizontal scaling due to the introduction of subnets. Each subnet contains multiple replica set. The core components of a replica are organized into the following logical layers:

1. A peer-to-peer (P2P) networking layer that collects and advertises messages from users, from other nodes in its subnet blockchain, and from other subnet

blockchains. Messages received by the peer-to-peer layer are replicated to all of the nodes in the subnet to ensure the security, reliability, and resiliency.

2. A consensus layer that selects and sequences messages received from users and from different subnets to create blockchain blocks that can be notarized and finalized by Byzantine Fault Tolerant Consensus forming the evolving blockchain. These finalized blocks are delivered to the message routing layer.
3. A message routing layer that routes user- and system-generated messages between subnets, manages the input and output queues for dapps, and schedules messages for execution.
4. An execution environment that calculates the deterministic computation involved in executing a smart contract by processes the messages it receives from the message routing layer.

## Canisters

When you write source code for a dapp that runs on the Internet Computer, you compile the source code into a WebAssembly module. When you deploy the WebAssembly module that contains your program on the Internet Computer blockchain, the program is executed inside a conceptual computational unit called a canister, or canister in short. Canisters can be developed in various programming languages. Besides Motoko, a programming language purposefully designed for the Internet Computer, you can also use existing programming languages like C, Rust, JavaScript/TypeScript, AssemblyScript, and Python.

There are only two types of calls:

1. non-committing **query calls** (any state change is discarded) and
2. committing **update calls** (state changes are persisted).

## Project Vision

In this project, we aim to benchmark Dfinity blockchain (Internet computer). When we talk about benchmarking, we want to understand the performance of the blockchain under controlled environment in order to understand the benefits and limitation of the whole system. In order to benchmark the blockchain, we need to first explore how it is working under the hood, and deploy the network on our testbed

(controlled environment) and then understand the performance implications based on the different experiments performed.

## Experiments/Workloads

Important properties that we need to understand:

1. **Orthogonal persistence (OP):** the IC uses orthogonal persistence as a programming abstraction to simplify memory management for canister smart contract developers.
2. **User-level networking:** traditionally, a syscall is needed to send data across the network, since the OS needs to make sure concurrent and safe access to shared resources is provided to all processes running on the OS.

TODO: figure out a way to deploy a subnet to their IC deployment on mainnet and testnet if possible.

## Stages

We divided this project into multiple stages based on the progress and the roadblocks encountered during the experiments.

### Stage 1

**Goal:** Deploy dfinity on the local testbeds and have a multi-node setup for benchmarking using diablo.

In order to deploy it locally, the first method that was tested was using the dfinity sdk called `dfx`. The idea was to investigate the possibility of checking whether it was possible to deploy the replicas using `dfx` on multiple nodes and try to set up communication between them.

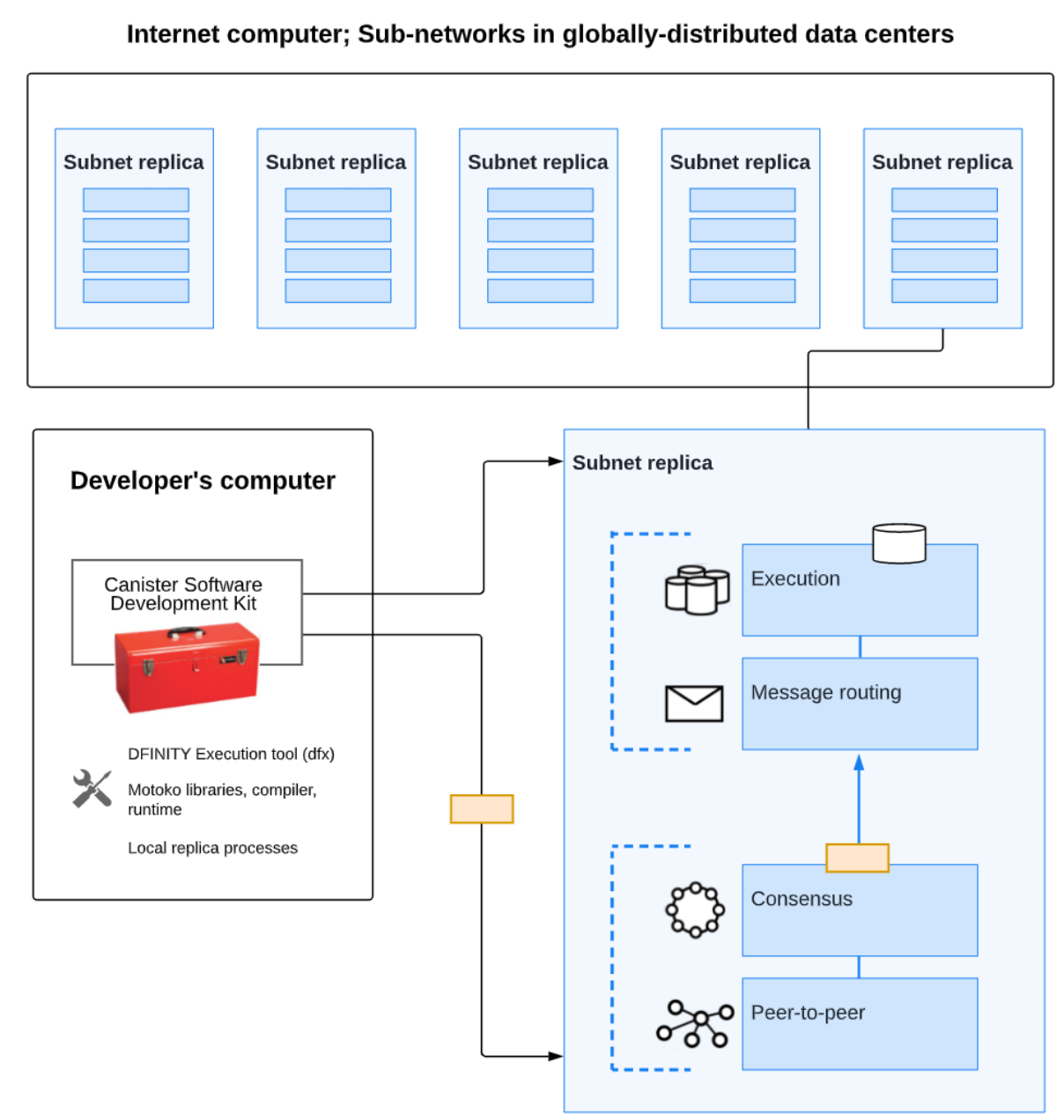
Hence, we started the local replica using `dfx` and then tried to set up the ledger canister on the local replica in order to interact with the blockchain.

Another essential thing to note about Internet Computer is that it doesn't ship with RPC endpoints for the blockchain. Hence once we deploy the local ledger, to create transactions or perform any kind of operations on the blockchain, you have to deploy

rosetta API on top of the replica set to interact with the blockchain in an RPC-like format.

## Local deployment

Install `dfx` by following the steps [here](#). Once we run `dfx start` it starts a local replica network on your computer, as shown in the picture below.



The thing to note here is that in the developers machine, only one single subnet replica is deployed.

## Ledger canister

The Internet Computer Protocol (ICP) implements management of its utility token (ticker "ICP") using a specialized canister, called the ledger canister. There is a single ledger canister which runs alongside other canisters on a special subnet of the Internet Computer - the NNS subnet. The ledger canister is a smart contract that holds accounts and transactions. These transactions either mint ICP tokens for accounts, transfer ICP tokens from one account to another, or burn ICP tokens, eliminating them from existence, e.g. while converting ICP tokens to cycles. The ledger canister maintains a traceable history of all transactions starting from its genesis state (initial state).

**Local ledger setup:** You can follow the steps shown [here](#).

In order to further understand how their ledger works, I run some of their test examples. One of the most important one is the [ledger-transfer example](#). It helps you create transactions, transfer ICP etc. The source code is added to `code/` folder. Once you run this example, you can run the rosetta client on top of this which will provide you with RPC-like endpoints.

To run the `ledger-canister example`, the easiest way is to ignore the steps in the readme provided in the repository and just follow the step-by-step procedure in the `code/ledger-transfer/demo.sh` bash script file. Once you have successfully deployed this, you can deploy the rosetta API to get the API endpoints to interact with the ledger.

## Rosetta

Rosetta is an open standard introduced by Coinbase to simplify the integration of blockchain-based tokens in exchanges, block explorers, and wallets. You can set up a Rosetta API-compliant node to interact with the Internet Computer and exchange Internet Computer Protocol (ICP) tokens.

In order to deploy your own rosetta client on top of the local ledger you can follow the steps [here](#)

## Findings

Using this process, we were able to successfully deploy the replica node on one of our testbed machines. However, when we moved to the next step, where we had to deploy it in multiple nodes and connect, we found that there was no functionality of this available in `dfx`. It deploys only a single replica set, and there is not possible

way to connect it with other local deployments. Hence we reached a dead end for this stage.