Building the Internet Computer:

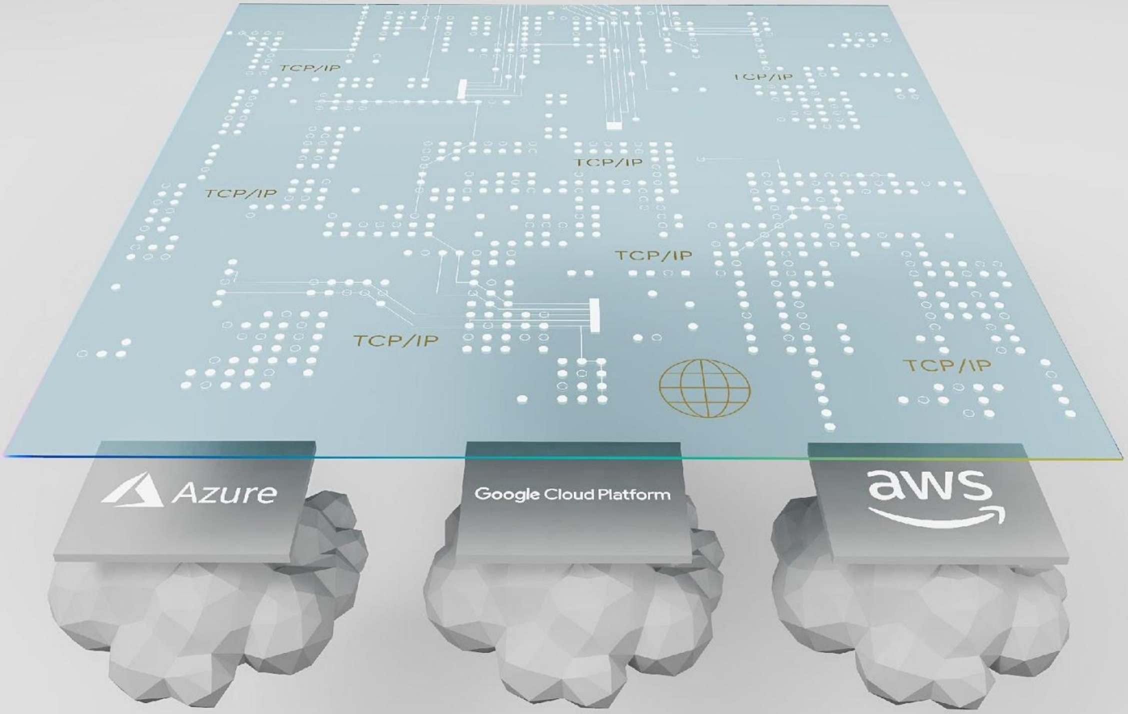
A glimpse into an ambitious adventure

Yvonne-Anne Pignolet
yvonneanne@dfinity.org

November, 2020

Today we have a thin Internet that only provides connectivity

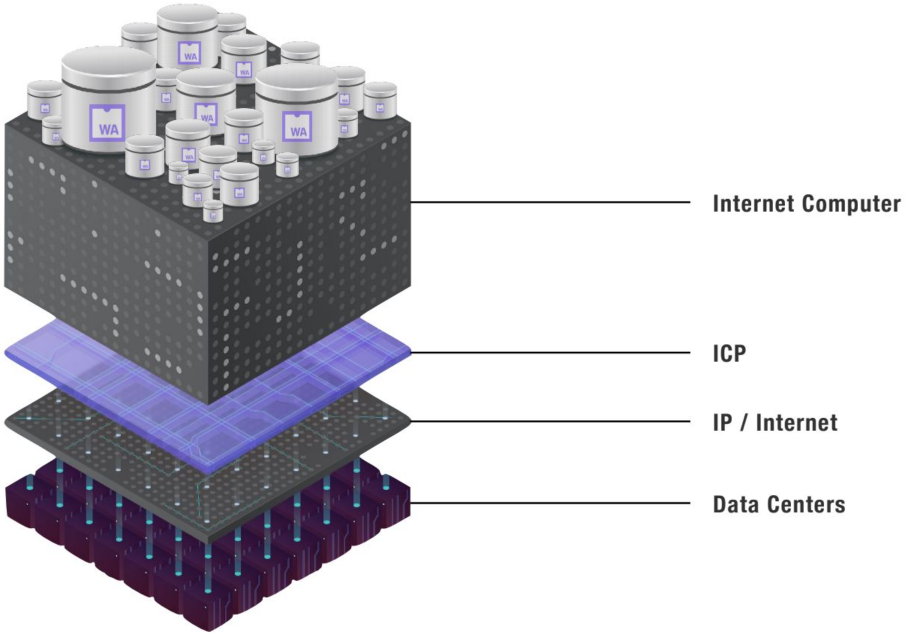
Software systems have to run on proprietary infrastructures… your own servers, or servers packaged by big tech



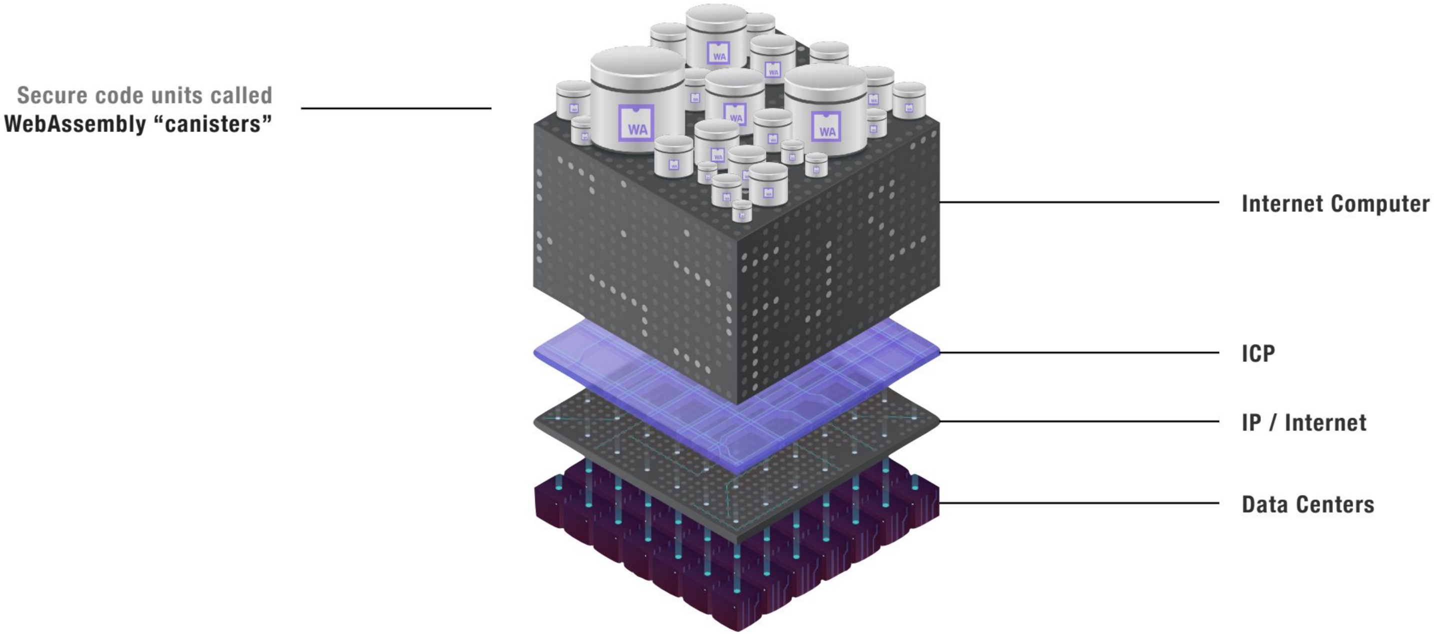
The ICP protocol will create a thick Internet, that's also a serverless cloud

The Internet will become a distributed OS that also hosts and runs software and services

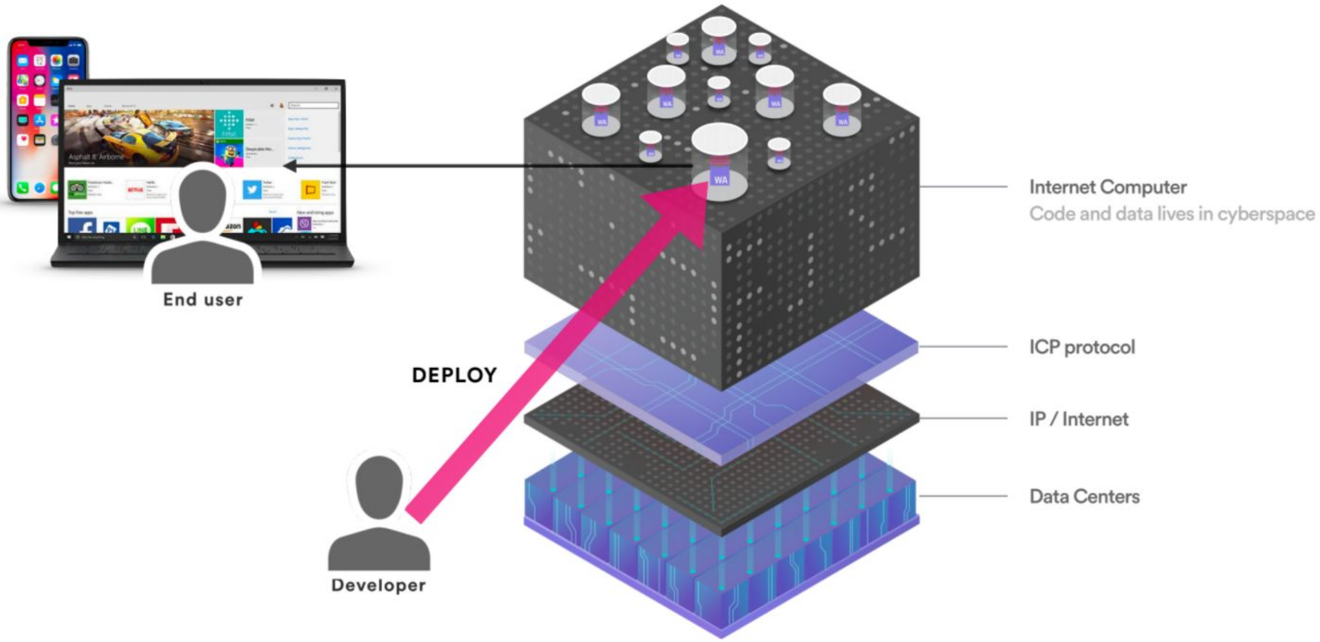
The “Internet Computer” is created by the ICP protocol



The Internet Computer can host unlimited units of secure code called WebAssembly "canisters" (advanced smart contracts)



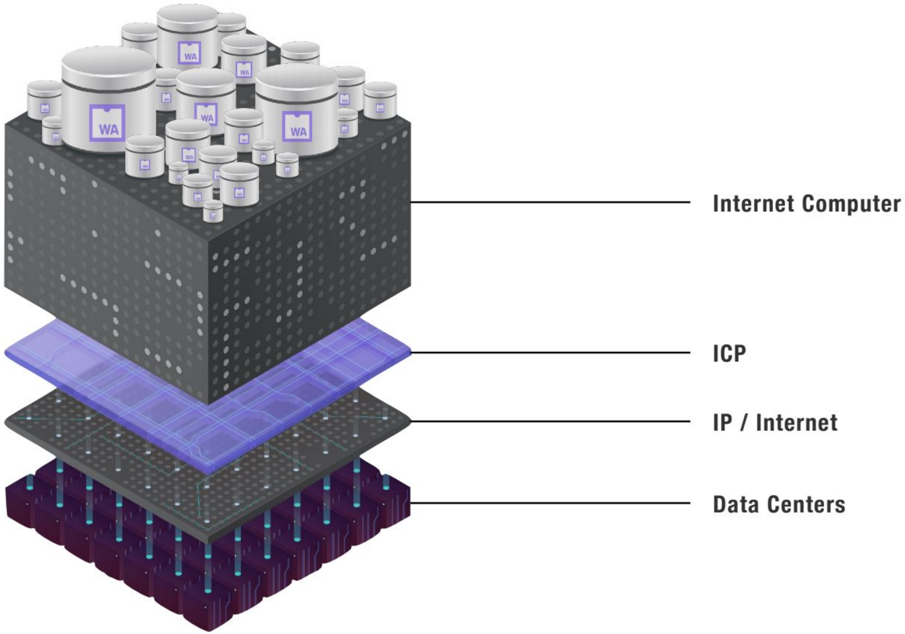
Developers build simply by writing their code into cyberspace



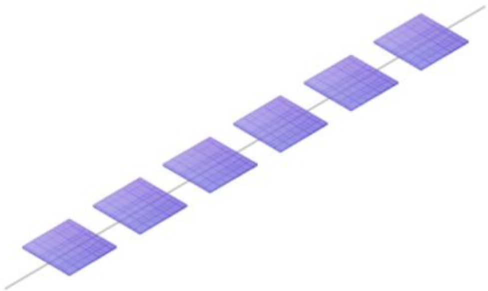
DFINITY

Build anything !

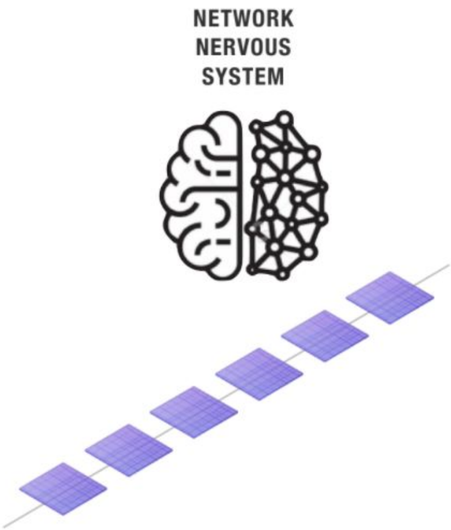
- Websites
- Open Internet services
- Enterprise systems
- Pan industry platforms
- DeFi applications



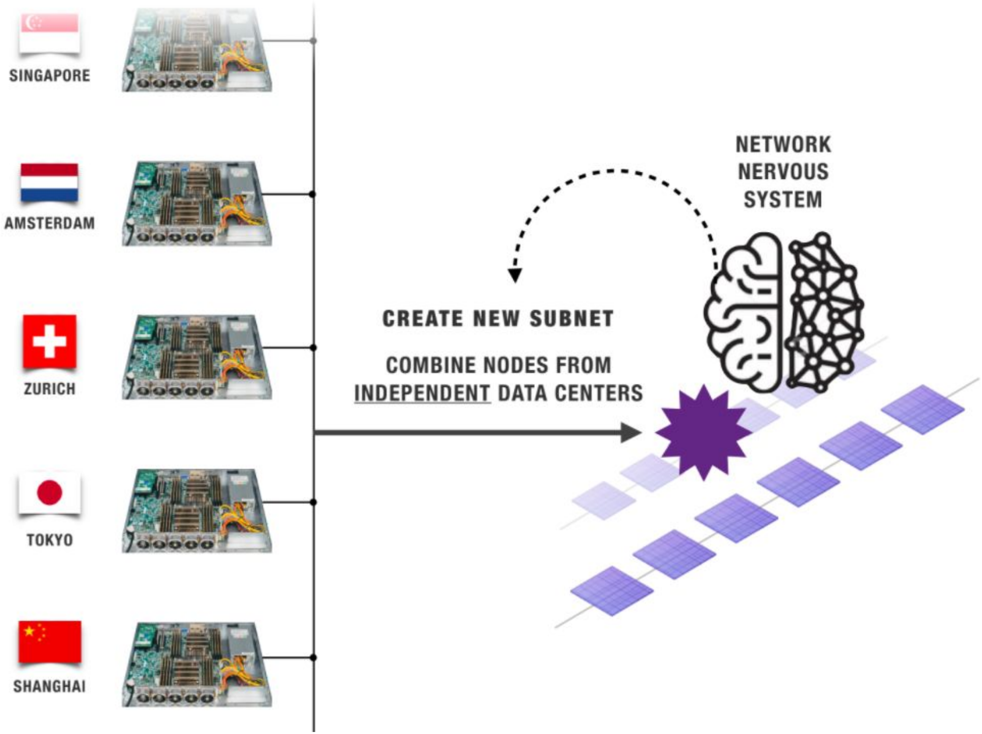
ICP is an advanced blockchain protocol...



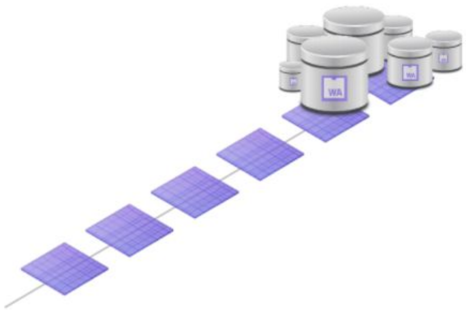
A “master” blockchain hosts the Network Nervous System (NNS)

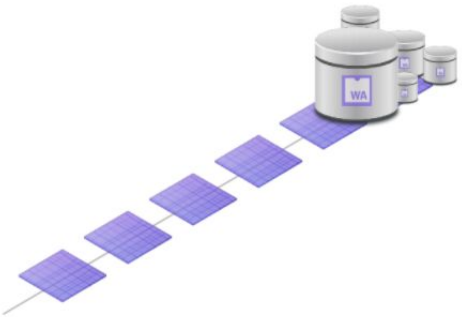


The NNS increases capacity by creating new blockchains ("subnets")

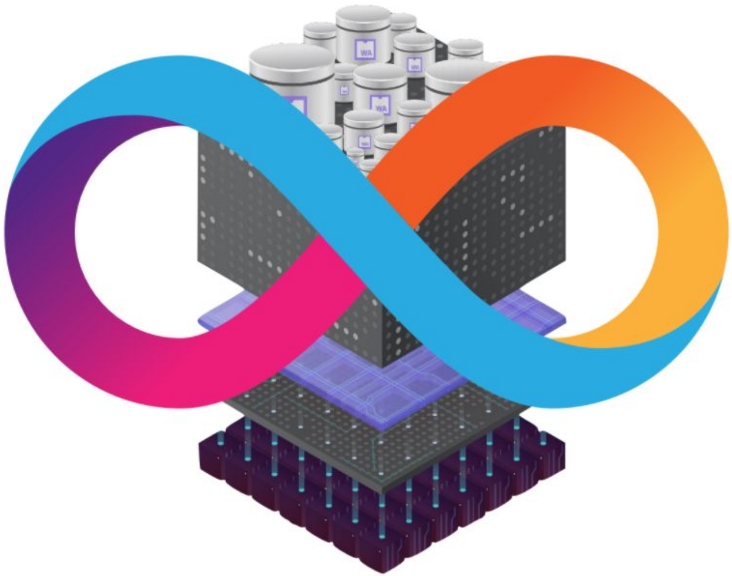


Each additional blockchain (“subnet”) hosts more canisters



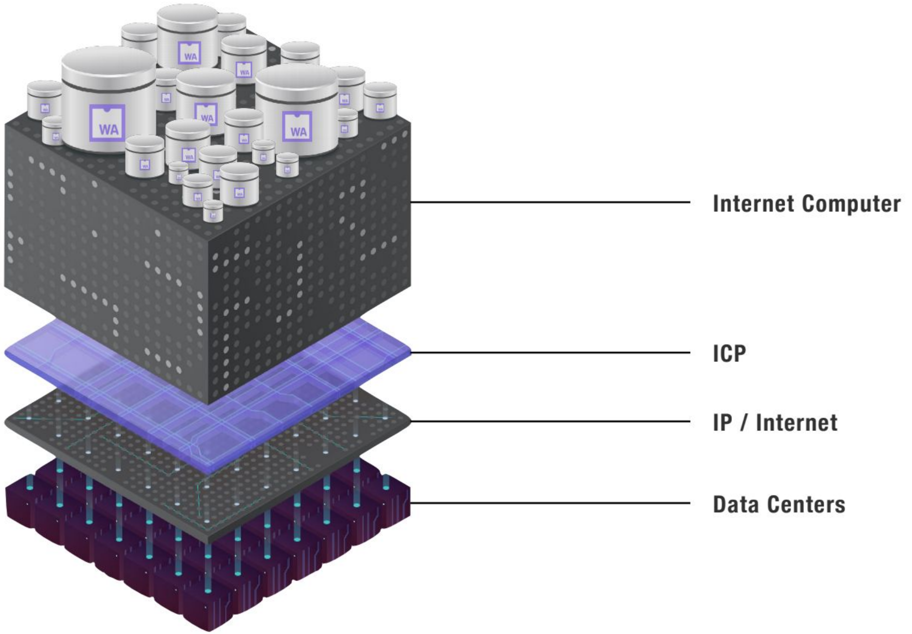


Unbounded scaling of computation and data storage...



Decentralized infinity

How does the IC decide what to compute next?



State Machine Replication

Internet Computer is State Machine

- Goal: Store and execute over data in a machine that is safe and live
- Single machine is not trusted. Replicate on multiple machines
- Machines are connected to each other via p2p network layer

State Machine Replication

- Each machine has its own state and perform execution
- All machines agree on the order of inputs they wish to consume
- Execution is deterministic
- All have the same state and produce the same output

Internet Computer is a State Machine

- A distinguished start State (S_0).
- A set of States $S_0, S_1, \ldots, S_t$
- A set of Inputs $I_0, I_1, \ldots, I_x$
- A set of Outputs $O_0, O_1, \ldots, O_y$
- A transition function (Input × State → State)

$$S_{i+1} = f_s(S_i, I_i)$$

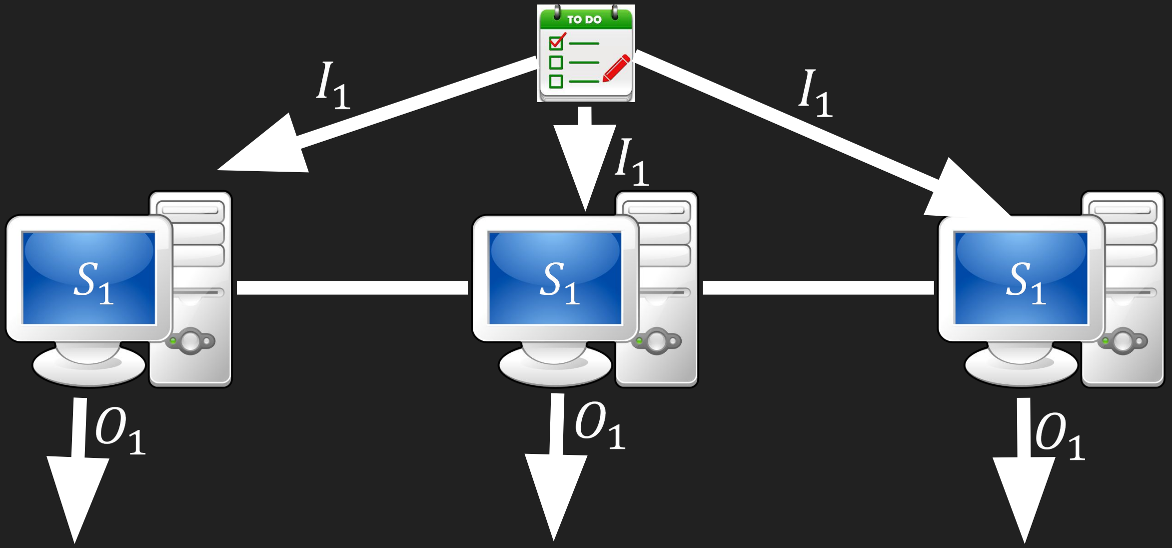
- An output function (Input × State → Output)

$$O_{i+1} = f_0(S_i, I_i)$$

State Machine Replication in two steps

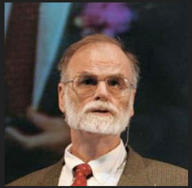
1. All the nodes agree on the order of inputs (consensus)

2. All the nodes update their state and provide output deterministically (execution).



Agreement is what we need!

Byzantine Generals Problem (1978)



- *n* nodes, each starts with an input value
- *t* of them are dishonest, controlled by an adversary
- **Agreement**: All honest nodes output same bit *b*
- **Validity**: *b* is an input bit of an honest node
- **Termination**: All honest parties eventually decide on *b*.

GOAL

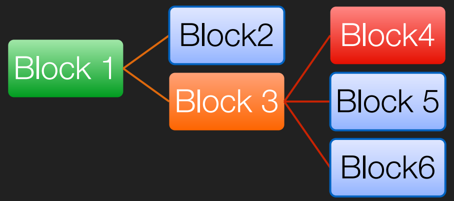
A consensus algorithm which is

- Provably Secure
    - Safety (nothing bad happens)
    - Liveness (something good will happen)
- Performant (fast, scalable)

Leader-Based Protocol (e.g., PBFT)

- Choose a leader
- Leader proposes a message
- All check for agreement
- Terminate if agreed
- Repeat if not

What is a Blockchain?



- Append-only data structure
- Composed of blocks, each referencing previous block
- Created and maintained in a distributed setting
- Goal: reach agreement on one path with ordering

Leader-Based Consensus

- Choose a leader
- Leader proposes a message
- All check the agreement
- Terminate if agreed
- Repeat if not

Leader-Based Chain Agreement

- Choose a block maker (BM)
- BM proposes a block
- All update the chain
- No termination
- Repeat

Key Observation

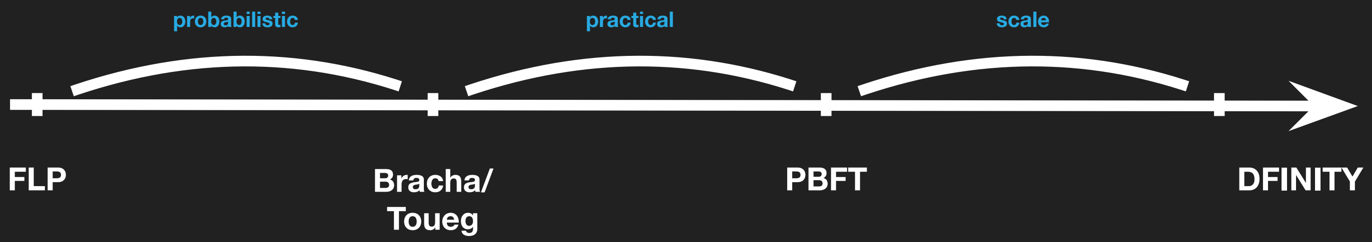
- Blockchain is a sequence of agreements
- We can amortize the cost and agree once in a while

Consensus algorithm landscape

BYZANTINE AGREEMENT IN AUTHENTICATED SETTING

(n nodes, f adversarial nodes, asynchrony)

$f = 0$	$n > 3f$	$n > 3f$	$n > 3f$
		+ synchrony (for liveness)	*+ synchrony (for liveness)*



Fundamental Steps in DFINITY Consensus

Fundamental Steps

1

Creating Randomness

CHALLENGES:
Randomness

Derive randomness from solving Proof of Work (PoW)?

PoW is expensive

Solve PoW in each step

→

Expensive protocol cannot scale

Fundamentally limited throughput

CHALLENGES:
Randomness

Derive randomness from a chain?

Chain is not random, manipulable

Assumes everybody agrees on the chain

Must not depend on chain content

Must not fork

CHALLENGES:

Randomness

“LAST ACTOR” BIAS

The “last actor” sees the randomness and aborts.

Any fallback mechanism introduces bias.

CHALLENGES:
Randomness

"LAST ACTOR" BIAS

The "last actor" sees the randomness and aborts.

Any fallback mechanism introduces bias.

EXAMPLES:

- Miner discards block
- Commit-reveal schemes

FUNDAMENTAL
OBSERVATIONS:

No “last actors”

k-of-n THRESHOLD GROUP

k out of **n** members

have to act

(necessary and sufficient)

EXAMPLES:

- Secret sharing, signatures, encryption

BLS Threshold Signatures

- **Pseudo-random**
  Not predictable
- **Distributed key generation:**
  No trusted dealer required
- **Unique:** For every message there exists only a single valid signature
- **Non-interactive:** Signature shares created independently

CHALLENGES:

Randomness

Pseudo - Random Function

A deterministic, pseudo-random sequence → “agree once” ↓ “agree always”

EXAMPLES:

- DFINITY: Use BLS
- Algorand: VRF + BA*

CHALLENGES:

Randomness

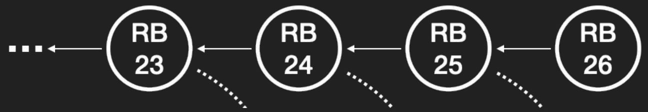
At each block generate new randomness

Last Actor Bias

Threshold crypto is expensive

Create by a threshold group

Reuse randomness in beacon chain



Fundamental Steps

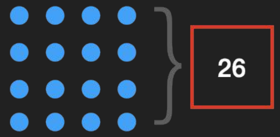
3

Reaching Agreement on the Input Order

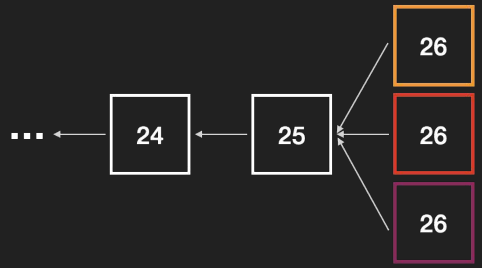
User messages

1) Users send messages for canisters to IC. 

2) P network layer broadcasts messages thus they will reach all honest nodes.Each node creates a list of candidate messages (block proposal).



3) Replicas need to agree on which block proposal to execute next.



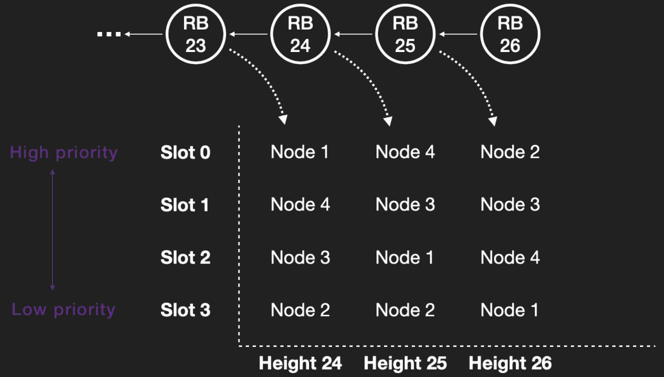
Block Making

1) The protocol proceed in rounds.

2) Each round starts by creating a new random beacon value (collecting BLS signature shares of previous beacon).

3) Block makers propose a new block.

4) All nodes use random beacon to rank the block proposals.



Notarization

1) Rank received block proposals based on random beacon

2) Sign highest ranked block proposal and broadcast the signature (*notarization share*).

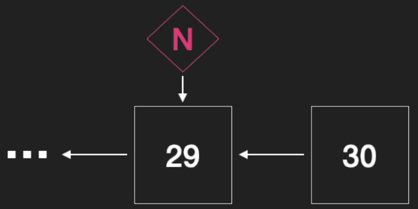
3) Gather signature shares on the block proposals

4) If a block proposal has *k* shares it is considered *notarized*

Notarization

Step 1

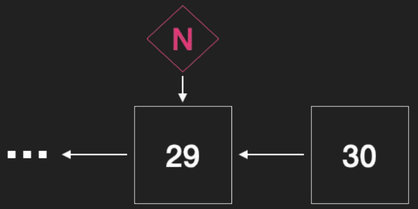
Replica 1 receives a block proposal for height 30, building on some notarized height 29 block



Notarization

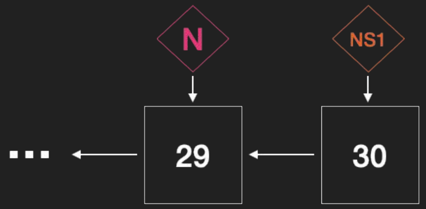
Step 1

Replica 1 receives a block proposal for height 30, building on some notarized height 29 block



Step 2

Replica 1 sees that the block is valid, signs it, and broadcasts its *notarization share*



Notarization

Step 1

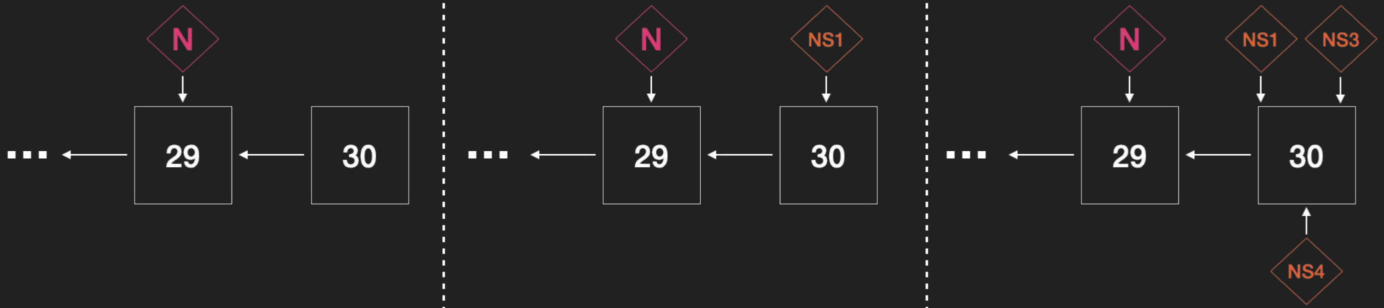
Replica 1 receives a block proposal for height 30, building on some notarized height 29 block

Step 2

Replica 1 sees that the block is valid, signs it, and broadcasts its *notarization share*

Step 3

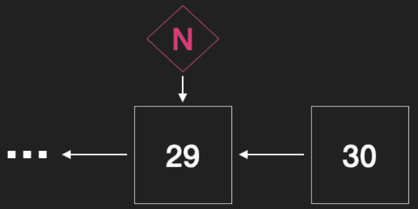
Replicas 1 sees that replicas 3 and 4 also published their notarization shares on the block



Notarization

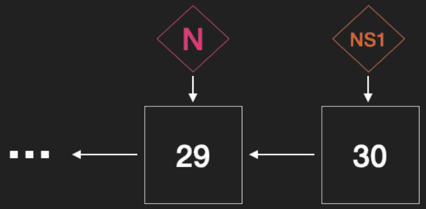
Step 1

Replica 1 receives a block proposal for height 30, building on some notarized height 29 block



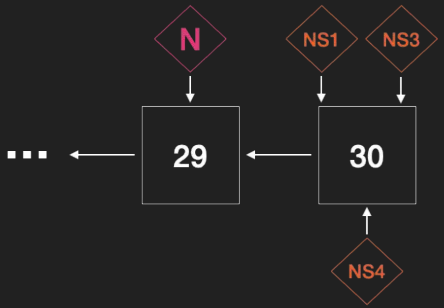
Step 2

Replica 1 sees that the block is valid, signs it, and broadcasts its *notarization share*



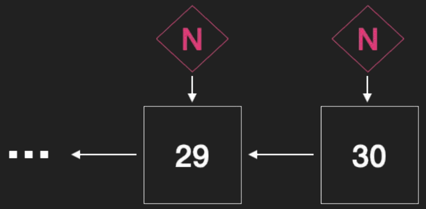
Step 3

Replicas 1 sees that replicas 3 and 4 also published their notarization shares on the block



Step 4

3 notarization shares are sufficient approval: the shares are aggregated into a single full notarization. Block 30 is now notarized, and notaries wait for height 31 blocks



Finalization

1. If only one block is notarized for this round, sign that notarized block and broadcast the signature share (finalization share).
2. Gather all the finalizing shares of the block.
3. A block is finalized if it has k or more finalizing shares
4. After finalizing a block, finalize its parent.

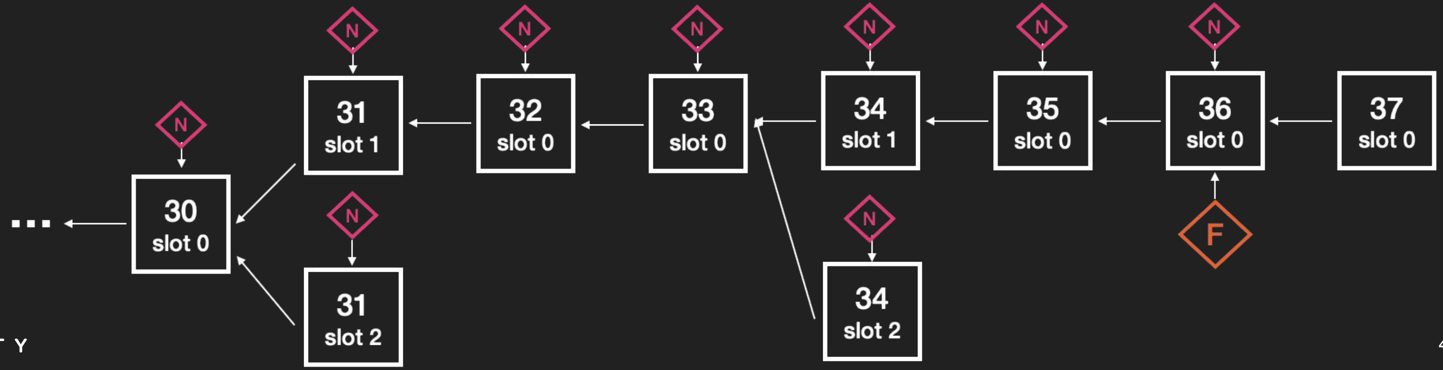
Finalization

1. If only one block is notarized for this round, sign that notarized block and broadcast the signature share (finalization share).

2. Gather all the finalizing shares of the block.

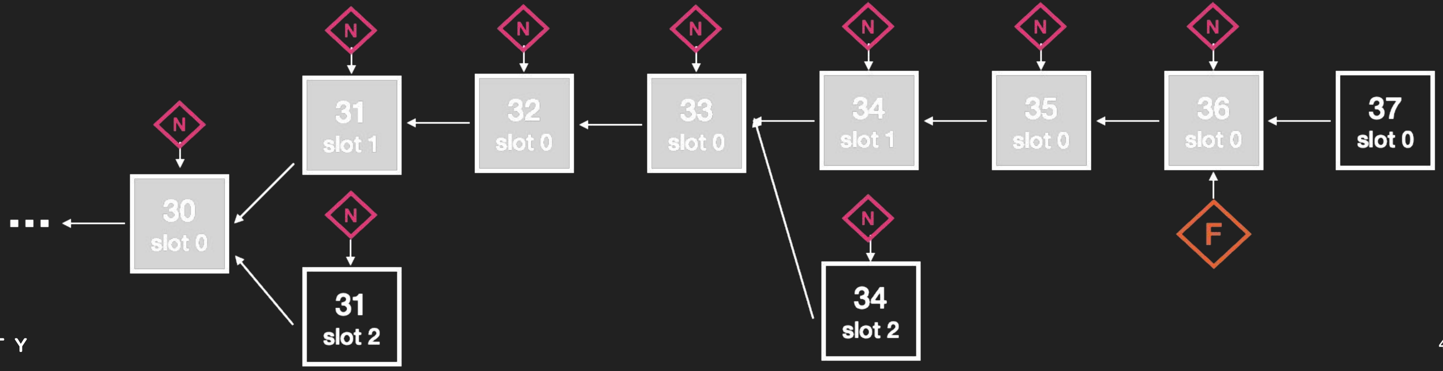
3. A block is finalized if it has k or more finalizing shares

4. After finalizing a block, finalize its parent.



Finalization

1. If only one block is notarized for this round, sign that notarized block and broadcast the signature share (finalization share).
2. Gather all the finalizing shares of the block.
3. A block is finalized if it has *k* or more finalizing shares
4. After finalizing a block, finalize its parent.



What we learned

Unique pseudo-random threshold signatures → **Randomness**

Notarization & Finalization → **Agreement**

Distributed Computing Problems @ DFINITY

… can be found on all layers

Today's focus

- Disseminate input among all nodes
- Reach agreement about what to execute next
- Sharding (operate on a state partition for scalability)
- Concurrent execution
- Guarantee consistency (user view of data and operations)
- Reconfiguration (add and remove canisters, data centers, shards, nodes)

Yvonne-Anne Pignolet
yvonneanne@dfinity.org

DFINITY