Outline

• What is the Internet Computer?
• Consensus on the Internet Computer
• The Internet Computer Today
What is the Internet Computer?
What is the Internet Computer?

Platform to run any computation, using blockchain technology for decentralisation and security.
ICP creates the Internet Computer blockchains. Guarantees safety and liveness of smart contract execution despite Byzantine participants.

Coordination of nodes in independent datacenters, jointly performing any computation for anyone.

Internet Computer Protocol (ICP)
Canister Smart Contracts: Combination of Data and Code

- Data: Memory pages
- Code: WebAssembly bytecode
Developers and users interact directly with Canisters on the IC
Developers and users interact directly with Canisters on the IC
Nodes are partitioned into **subnets**

Canister smart contracts are assigned to different subnets
Nodes are partitioned into subnets

Canister smart contracts are assigned to different subnets

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
- …
Chain Key Technology

• Public key of NNS never changes, nodes in NNS share private key

• NNS generates key of subnets and certifies them

• Node in subnets use these keys to secure communication
Each Subnet is a Replicated State Machine

State:
• canisters and their queues

Inputs:
• new canisters to be installed,
• messages from users and other canisters

Outputs:
• responses to users and other canisters

Transition function:
• message routing and scheduling
• canister code
The Layers of the Internet Computer Protocol

Execution Environment

Message Routing

Consensus

Networking

Deterministic computation

Message acquisition and ordering
Consensus on the Internet Computer
Consensus Orders Messages
Replicas may receive input messages in different orders, but must process them in the same order, for example: 1 2 3 4 5 6
Consensus Properties

Messages are placed in **blocks**. We reach agreement using a blockchain.

The following properties must hold even if up to $f < n/3$ nodes misbehave:

- **Safety**: For any $i$, if two (honest) replicas think that the $i$-th block is agreed upon, they must have the same block.
- **Liveness**: For any $i$, at some point every (honest) replica will think that the $i$-th block is agreed upon.
- **Validity**: All agreed upon blocks are valid.

We use $n = 4$, $f = 1$ in examples.
A block maker selects available messages and combines them into a block and broadcasts it.

Note: We need more than one block maker in each round, otherwise the IC would not be fault tolerant!
Notarization

The notarization process ensures that a valid block proposal is published for every round.

**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**
Replica 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.
Replicas may notary-sign multiple blocks to ensure that at least one block becomes fully notarized.

**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 another height 30 block, which is also valid, and it broadcasts another notarization share.

**Step 4**
Both height 30 blocks get enough support to become notarized.
Multiple notarized blocks may exist at the same height
Random Beacon

At every height, there is a Random Beacon, an unpredictable random value shared by the replicas.

**Step 1**
Replica 1 has Random Beacon 29 and wants to help constructing Random Beacon 30

**Step 2**
Replica 1 signs RB29 using a threshold signature scheme, yielding a share of random beacon 30

**Step 3**
Replicas 1 sees that replica 2 also published a share of Random Beacon 30

**Step 4**
2 random beacon shares are sufficient to reconstruct a full threshold signature, which is Random Beacon 30

Unique (BLS) signature out of f+1 shares!
The Random Beacon is used to rank block makers.

<table>
<thead>
<tr>
<th>Rank 0</th>
<th>Replica 1</th>
<th>Replica 4</th>
<th>Replica 2</th>
<th>Replica 3</th>
<th>Replica 4</th>
<th>Replica 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank 1</td>
<td>Replica 4</td>
<td>Replica 3</td>
<td>Replica 3</td>
<td>Replica 1</td>
<td>Replica 2</td>
<td>Replica 1</td>
</tr>
<tr>
<td>Rank 2</td>
<td>Replica 3</td>
<td>Replica 1</td>
<td>Replica 4</td>
<td>Replica 4</td>
<td>Replica 1</td>
<td>Replica 3</td>
</tr>
<tr>
<td>Rank 3</td>
<td>Replica 2</td>
<td>Replica 2</td>
<td>Replica 1</td>
<td>Replica 2</td>
<td>Replica 3</td>
<td>Replica 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round 24</th>
<th>Round 25</th>
<th>Round 26</th>
<th>Round 27</th>
<th>Round 28</th>
<th>Round 29</th>
</tr>
</thead>
</table>

High Priority

Low Priority
Notarization with Block Maker Ranking

Rounds are divided into time slots defining when block maker proposals are considered.
The block ranks can reduce the number of notarized blocks

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

Step 2
Replica 1 is still in time slot 0, so not willing to notary-sign a rank-1 block yet

Step 3
Replicas 1 sees a valid rank-0 height 30 block, and it broadcasts a notarization share

Step 4
Eventually, only the rank 0 block becomes notarized
Notarization with Block Maker Ranking

One notarized block $b$ at a height $h = \text{Agreement up to } h$

How can we detect this…?
Notarization with Block Maker Ranking

Synchronous communication $\rightarrow$ Forks can be removed
Notarization with Block Maker Ranking

Partially synchronous communication → Forks cannot be removed!
Finalization

Replicas create finalization shares if they did not sign any other block at that height

**Step 1**
Replica 1 notary-signs block $b$ at height 30

**Step 2**
Replica 1 observes that block $b$ is fully notarized and will no longer notary-sign blocks at height $\leq 30$

**Step 3**
Since replica 1 did not notary-sign any other block than block $b$, it signs block $b$, creating a finalization-share on $b$

**Step 4**
Replicas 2 and 4 also cast finalization shares on block $b$

**Step 5**
3 finalization-shares are sufficient approval: the shares are aggregated into a single full finalization

Replica 1 did not notary-sign any height 30 block other than $b$
Finalization

Finalization on block $b$ at height $h = \text{Proof that no other block is notarized at height } h$

The chain up to this block is final
Safety of Finalization

If block \( b \) at height \( h \) is finalized, then there is no finalized block \( b' \neq b \) at height \( h \).

Proof:
1. A full finalization on \( b \) requires \( n-f \) replicas to finality-sign (by construction)
2. At least \( n-2f \) of the \( n-f \) replicas that finality-signed \( b \) must be honest (by assumption that \( \leq f \) replicas are corrupt)
3. An honest replica that finality-signed \( b \) did not notary-sign any other block at height \( h \) (by construction)
4. At least \( n-2f \) replicas did not notary-sign any height \( h \) block other than \( b \) (by 2. & 3.)
5. A full notarization requires \( n-f \) notarization-shares (by construction)
6. The \( n-(n-2f) < n-f \) remaining replicas that may have notary-signed a block \( b' \) are not sufficient to reach the notarization threshold of \( n-f \) (by 4. & 5.)
The Internet Computer Today
Live Since May 2021!

Currently 375 machines by 53 node providers

Total Chain Data: 409.94 TB
Block Count: 325,971,476
NNS Proposals: 30,063

Chain CPUs: 21,720
Blocks / Second: 27.08
Canister Smart Contracts: 13,526

https://dashboard.internetcomputer.org/
Many distributed systems problems

- Disseminating messages among all nodes in the same subset
- Exchanging canister and control messages between subnets
- Scheduling and concurrent execution of canister messages
- Catching up after a node has been offline for a while
- Handling churn (adding and removing nodes)
- Guaranteeing consistency (different users need a consistent view of data and operations)
- Upgrading to next protocol version
- Creating new subnets
- Load balancing
- ...
## Internet Computer vs. …

<table>
<thead>
<tr>
<th></th>
<th>Bitcoin</th>
<th>Ethereum</th>
<th>Internet Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average block time:</strong></td>
<td>1 block / 10 minutes</td>
<td>1 block / 15 seconds</td>
<td>30 blocks / second</td>
</tr>
<tr>
<td><strong>Finality:</strong></td>
<td>1 hour</td>
<td>3 minutes</td>
<td>1-3 seconds</td>
</tr>
<tr>
<td><strong>TX per second:</strong></td>
<td>7</td>
<td>15</td>
<td>11,500 (write) / 250,000 (read)</td>
</tr>
<tr>
<td><strong>Validation data:</strong></td>
<td>380 GB</td>
<td>550 GB</td>
<td>48 bytes</td>
</tr>
</tbody>
</table>
More information

- Infographic: [here](#)

- Technical Library: [here](#) (videos of talks) and [here](#) (blogposts)

- 200,000,000 CHF Developer Grant Program [here](#)

- DFINITY SDK: [here](#)