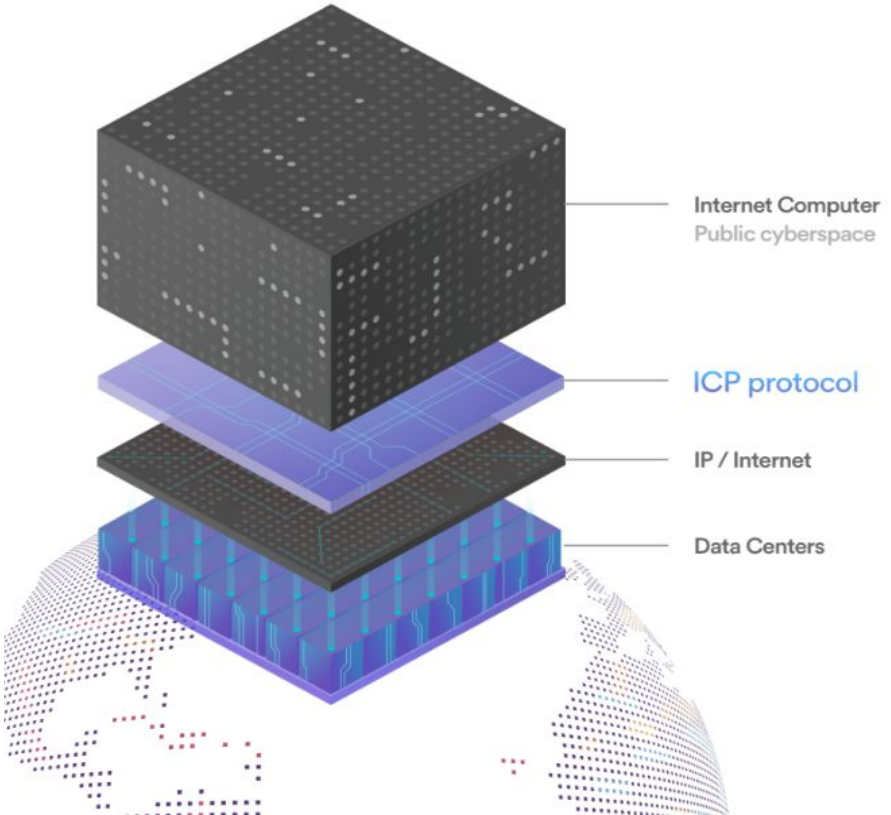
Dfinity: Internet computer: under partial synchrony

Yu Tang, Shuhan Chen

Background: Dfinity; Internet Computer Blockchain

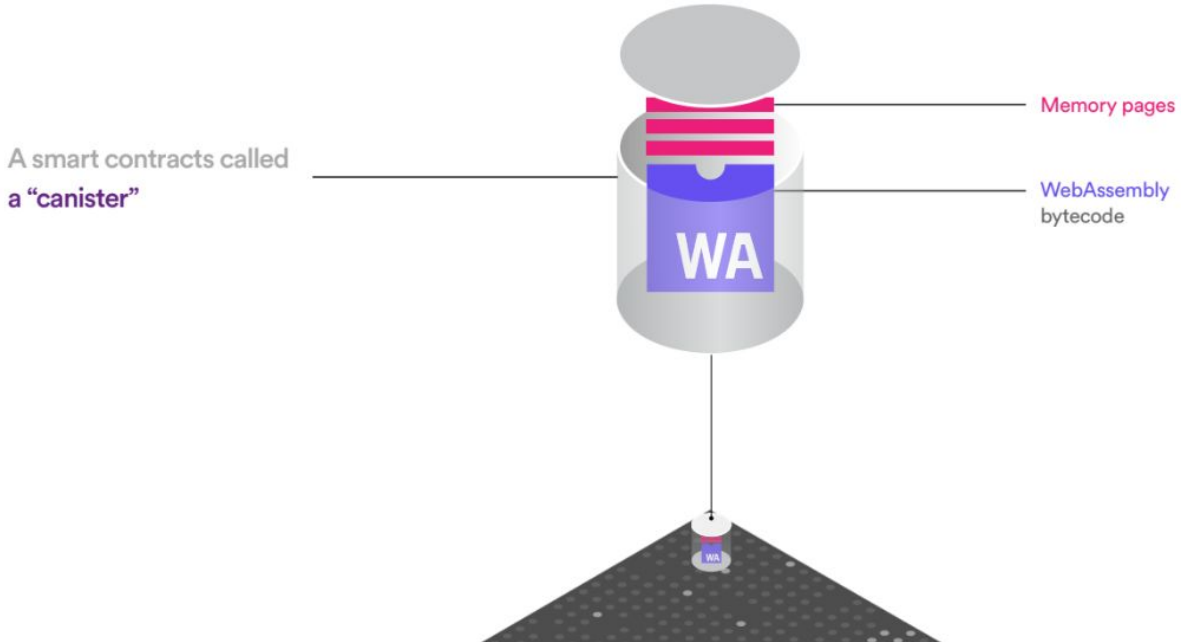
Internet Computer Blockchain:

“the world’s first web-speed, web-serving public blockchain network that can scale”

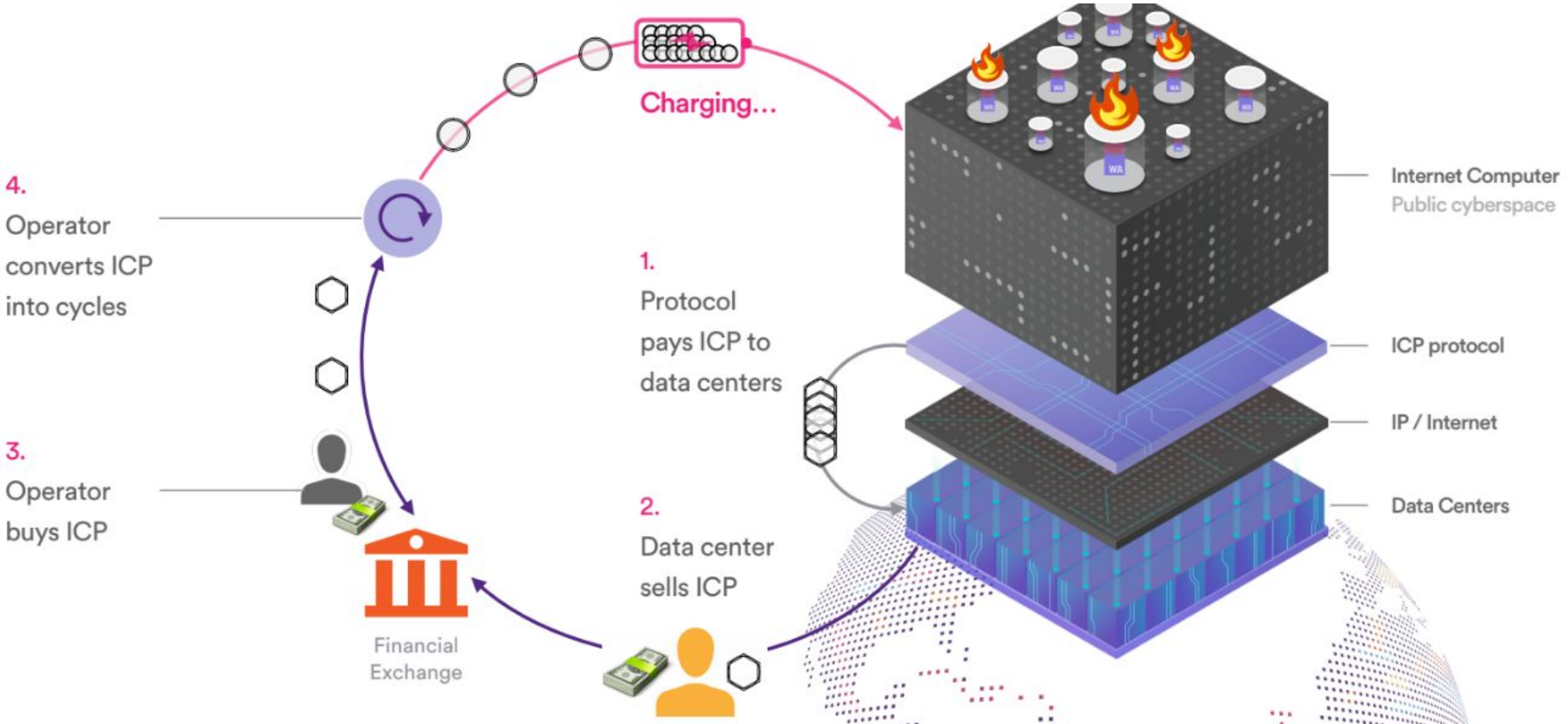


Background: Dfinity; Internet Computer Blockchain

The Internet Computer hosts special smart contracts, called “canisters”



Background: Dfinity; Internet Computer Blockchain



Background: Related Work

- PBFT
  - Optimistic responsive
- HotStuff
  - Eliminates all-to-all communication steps of PBFT
  - Does not rely on view-change; rely on a “pacemaker” subprotocol
  - Optimistic responsive
- Tendermint / Algorand
  - Rely on a peer-to-peer gossip sub-layer for communication
  - Not optimistically responsive
- MirBFT
  - Instances of PBFT run concurrently

Background

- The Internet Computer Consensus (ICC) protocols
  - Dynamic collection of intercommunicating replicated state machines
  - ICC0, ICC1, ICC2
  - Features:
    - Fully specified: no unspecified protocols
    - Extremely simple: no complicated subprotocols
    - Robust consensus: peak performance is partially sacrificed to ensure reasonable performance
    - ICC1: peer-to-peer gossip sub-layer; ICC2: subprotocol for reliable broadcast
    - Optimistically responsive

Assumptions

- n parties, at most t < n/3 corrupt parties
- Static corruption
- Partial synchronous
- Blockchain based
- Proceed in rounds
  - A random beacon to assign to each party a rank; lowest rank is the leader of the round
- Local computation is negligible relative to network latency
- The only type of communication: broadcast
- Each party has a pool which holds the set of all messages received from all parties

Protocol ICC0: Primitives

Threshold signature scheme:

(t, h, n)-threshold signature scheme:

- n parties are initialized with a public-key/secret-key pair
- public keys for all n parties
- a global public key

    - Signing algorithm
    - Signature share verification algorithm
    - Signature share combining algorithm
    - Signature verification algorithm

Blocks

For k>=1, a round-k block B: (block, k, α, phash, payload)

- Authentic
- Valid
- Notarized
- Finalized

Root serves as its own authenticator, notarization, and finalization.

Protocol Components

- A collision resistant hash function, H
- A signature scheme S_auth
- An instance S_notary of a (t, n − t, n)-threshold signature scheme
- An instance S_final of a (t, n−t, n)-threshold signature scheme
- An instance S_beacon of a (t, t + 1, n)-threshold signature scheme, used to implement a random beacon

High level description of the protocol

- Under cryptographic assumptions, it is guaranteed that in each round k >=1:

  P1 (deadlock freeness): at least one notarized block of depth k will be added to the block-tree.

  P2 (safety): if a notarized block of depth k is finalized, then there is no other notarized block of depth k.

  P3 (liveness): if the network is synchronous over a short interval of time beginning at the point in time where any honest party first enters round k, and the leader in round k is honest, then the block proposed by the leader in round k will be finalized.

Invariants in Sync settings

I An honest leader's block will be uniquely certified in an iteration.
II In each iteration, at least one block, but possibly many, will be certified.
III At the end of iteration k, if all iteration-$(k-1)$ blocks extend the same B_{k-2}, then B_{k-2} is uniquely extendable. i.e., no other iteration-$(k-2)$ block can be extended from then on.

Protocol details

- Authentic
- Valid /consensus/src/consensus/validator.rs
- Notarized /consensus/src/consensus/notary.rs
- Finalized /consensus/src/consensus/finalizer.rs

- Tree Building Subprotocol
- Finalization Subprotocol

Protocol details:
Tree Building Subprotocol

broadcast a share of the round-1 random beacon
For each round $k = 1, 2, 3 \ldots$:

wait for $t + 1$ shares of the round-k random beacon
compute the round-k random beacon (which defines the permutation π for round k)
broadcast a share of the random beacon for round $k + 1$

let r_{me} be the rank of P_α according to the permutation π
$\mathcal{N} \leftarrow \emptyset$ *// the set of blocks for which notarization shares have been broadcast by* P_α
$\mathcal{D} \leftarrow \emptyset$ *// the set of ranks disqualified by* P_α

done $\leftarrow$ false
proposed $\leftarrow$ false
$t_0 \leftarrow$ clock()

repeat
wait for either:
(a) a notarized round-k block B,
or a full set of notarization shares for some valid
but non-notarized round-k block B:
// Finish the round
combine the notarization shares into a notarization for B, if necessary
broadcast the notarization for B
done $\leftarrow$ true
if $\mathcal{N} \subseteq \{B\}$ then **broadcast** a finalization share for B

(b) not *proposed* and clock() $\geq t_0 + \Delta_{\text{prop}}(r_{\text{me}})$:
// Propose a block
choose a notarized round-$(k-1)$ block B_{p}
payload $\leftarrow$ *getPayload*(B_{p})
create a new round-k block $B = (\text{block}, k, \alpha, H(B_{\text{p}}), \textit{payload})$
broadcast B, B's authenticator, and the notarization for B's parent
proposed $\leftarrow$ true

(c) a valid round-k block B of rank r such that
$B \notin \mathcal{N}$, $r \notin \mathcal{D}$, clock() $\geq t_0 + \Delta_{\text{ntry}}(r)$, and
there is no valid round-k block B^* of rank $r^* \in [r] \setminus \mathcal{D}$:
// Echo block B
// and either broadcast a notarization share for it or disqualify its rank
if $r \neq r_{\text{me}}$ then
broadcast B, B's authenticator, and the notarization for B's parent
if some block in $\mathcal{N}$ has rank r
then $\mathcal{D} \leftarrow \mathcal{D} \cup \{r\}$
else $\mathcal{N} \leftarrow \mathcal{N} \cup \{B\}$, **broadcast** a notarization share for B

until *done*

Protocol details: Tree Building Subprotocol

Delay functions: $[n] \rightarrow R \geq 0$,based on the rank of the proposer-

- Δprop: delay proposing a block
- Δntry: delay generating a notarization share on a block

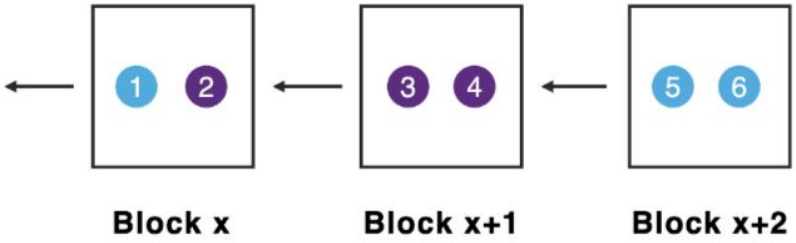
Liveness: $2\delta + \Delta\_prop(0) \leq \Delta\_ntry(1)$, δ: network delay

Protocol details: Finalization Subprotocol

$k_{\max} \leftarrow 0$ *// max round finalized by* P_α
repeat
    wait for:
        (i) a finalized round-k block B with $k > k_{\max}$, or
        (ii) a complete set of finalization shares for some valid but non-finalized round-k block B with $k > k_{\max}$:
        // Commit to the last $k - k_{\max}$ *blocks in the chain ending at* B
        combine the finalization shares into a finalization for B, if necessary
        broadcast the finalization for B
        output the payloads of the last $k - k_{\max}$ blocks in the chain ending at B
        $k_{\max} \leftarrow k$
forever

Consensus Properties

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



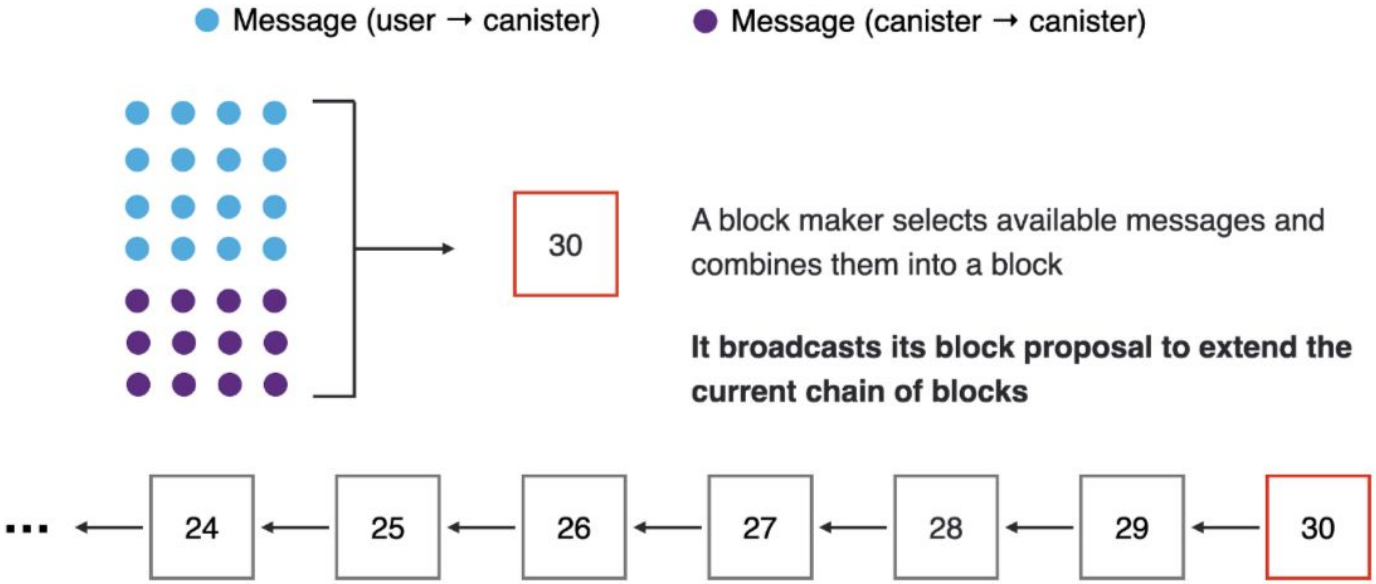
Suppose we have *n* replicas running a subnet together.

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

We want this even if up to *f* of the *n* nodes misbehave, with *n = 3f+1*

In examples on these slides, we use ***n* = 4, *f* = 1**

Block Maker



Note: We need sufficiently many replicas to take the role of block maker each round. If we were to choose only one, we would not be fault tolerant.

Random Beacon

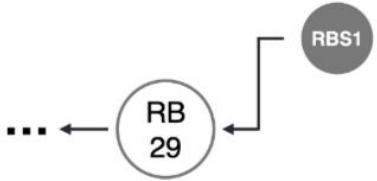
At every height, we have a Random Beacon, which is an unpredictable random value shared by the replicas

Step 1

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

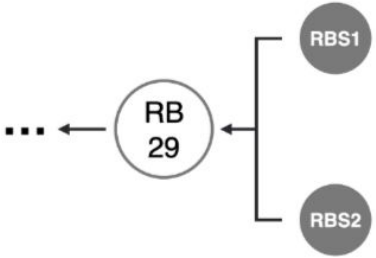
Step 2

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

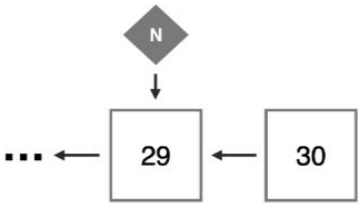
2 random beacon shares are sufficient to reconstruct a full threshold signature, which is Random Beacon 30

Notarization

Block Proposals may be invalid. Every block in the chain must be *notarized*.
The notarization process ensures that every round a *valid* block proposal is published.

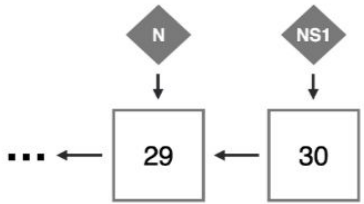
Step 1

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



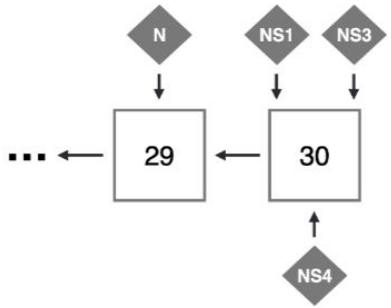
Step 2

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



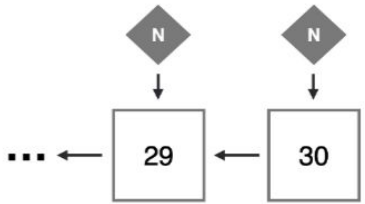
Step 3

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

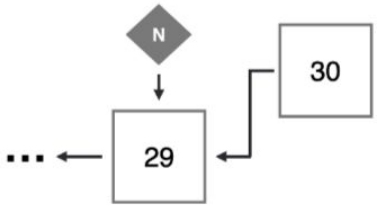


A Notary May Notary-Sign Multiple Blocks

To ensure that at least one block becomes fully notarized, notaries may notary-sign multiple blocks

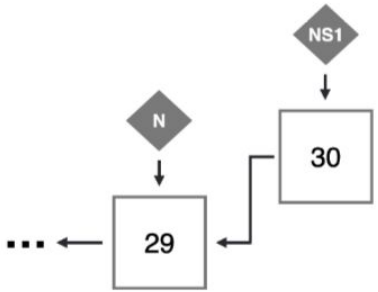
Step 1

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



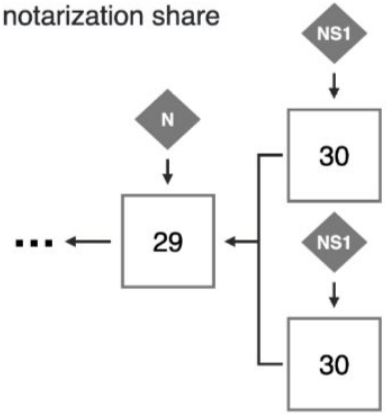
Step 2

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



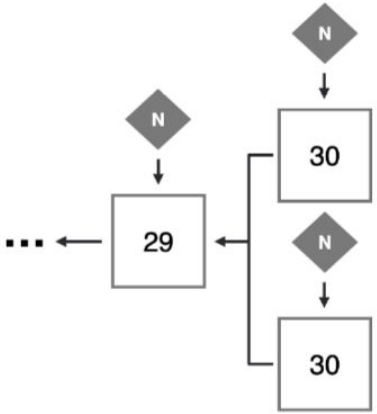
Step 3

Replicas 1 sees another height 30 block, which is also valid, and it broadcasts another notarization share



Step 4

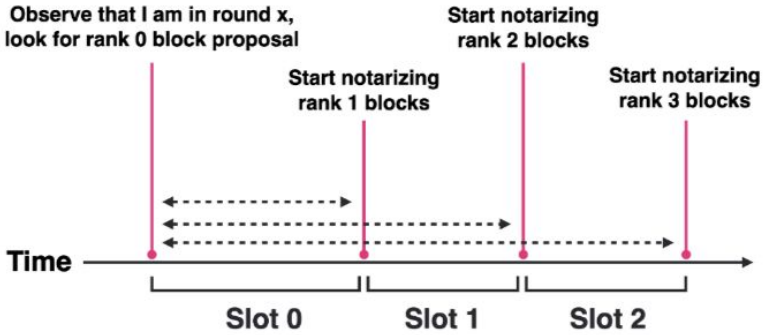
Both height 30 blocks get enough support to become notarized



Notarization with Block Maker Ranking

We can reduce the number of notarized blocks per round by letting notaries use the block maker ranking in the notarization process.

- The time following a round start is divided into time slots
- Initially, in time slot 0, a notary only notary-signs a block proposal from the rank-0 block maker
- After some timeout, if they still haven't seen a valid rank 0 block proposal, slot 1 starts, and the notary is also willing to notary-sign a rank 1 block proposal. After more timeout, slot 2 starts, and notaries would accept rank 2 blocks if no better block proposal was observed, etc.
- If the network works well and the rank 0 block maker sends a valid block that reaches notaries before they are willing to notary-sign other blocks, this will be the only notarized block at that height

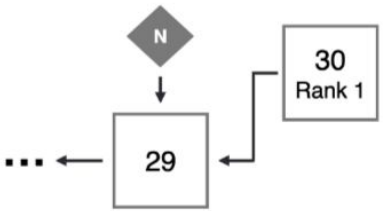


Notarization with Block Rank

By taking block rank into consideration, we can reduce the amount of notarized blocks

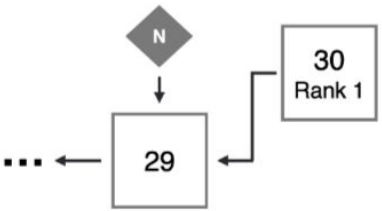
Step 1

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



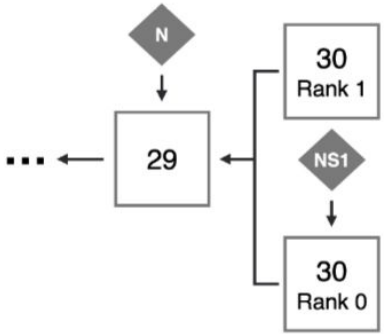
Step 2

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



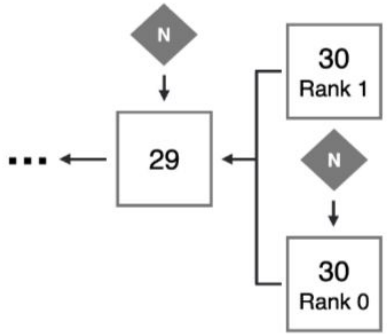
Step 3

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

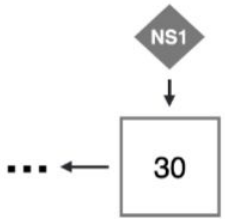


Finalization

Notaries create “finalization shares” on a block if they did not notary-sign any other block at that height.

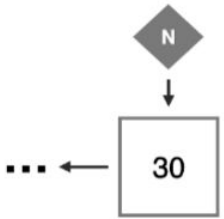
Step 1

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



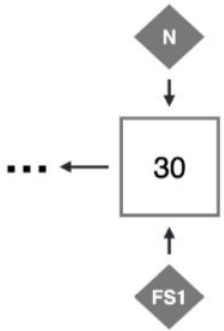
Step 2

Replica 1 observes that block *b* is fully notarized and will no longer notary-sign blocks at height ≤ 30



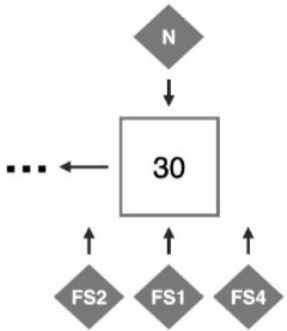
Step 3

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



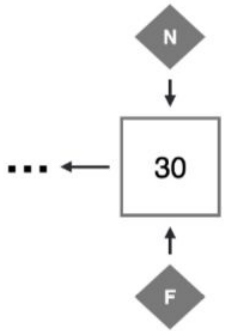
Step 4

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



n=4, f=1:

P1: at least one notarized block of depth k will be added to the block-tree.

P2: if a notarized block of depth k is finalized, then there is no other notarized block of depth k.

P3: if the network is synchronous over a short interval of time beginning at the point in time where any honest party first enters round k, and the leader in round k is honest, then the block proposed by the leader in round k will be finalized.

Analysis: Expected Message Complexity and Latency

- Partial synchrony (depend on value of h, the rank of the leading honest party in round k)
  - Message complexity in round k is: $O((h+1)n^2)=O(n^3)$
  - Latency:

(iv) the communication network is δ-synchronous at all times throughout the interval

$$[T, T + \Delta_0(h, \delta) + 2h\delta],$$

where

$$\Delta_0(h, \delta) := \max(2\delta + \Delta_{\text{prop}}(h), \delta + \Delta_{\text{ntry}}(h)).$$

Then the all honest parties finish round k by time T plus

$$\Delta_0(h, \delta) + (2h + 1)\delta.$$

Analysis: Expected Message Complexity and Latency

$$\Pr[h \geq i] = \frac{t}{n} \cdot \frac{t-1}{n-1} \cdot \cdots \cdot \frac{t-(i-1)}{n-(i-1)} < \frac{1}{3^i}.$$

$$\mathrm{E}[h] = \sum_{i \geq 1} \Pr[h \geq i] \leq \sum_{i \geq 1} \frac{1}{3^i} = \frac{1}{2}.$$

- With probabilistic analysis, the expected message complexity is $O(n^2)$.
- With probabilistic analysis, the expected latency is bounded by:

$$\Delta_{\mathrm{bnd}} + 3\delta + \max(\epsilon, \delta).$$

Analysis: Comparison

- Sync Dfinity (static adversary)
  - Message complexity:
    - basic design: unbounded
    - With equivocation check: $O(n^2)$
  - Latency: $O(\Delta)$
    - optimistic case ($c << \Delta$): 8Δ; pessimistic case ($c = \Delta$): 14Δ
- Partial sync Dfinity: ICC0
  - Message complexity:
    - $O(n^2)$ with overwhelming probability
  - Latency:

$$\Delta_{\text{bnd}} + 3\delta + \max(\epsilon, \delta).$$

Comparison to [HMW18, AMNR18]

- Proposing step
- only guaranteed safety in a synchronous setting
- not optimistically responsive
- potentially unbounded communication complexity.

Shortcomings

1. Difficulty on adaptive adversaries scenario (round k rank determined in round k-1)
2. Need to disseminate secret keys to parties