Subnets

Architecture Meeting
12/10/21

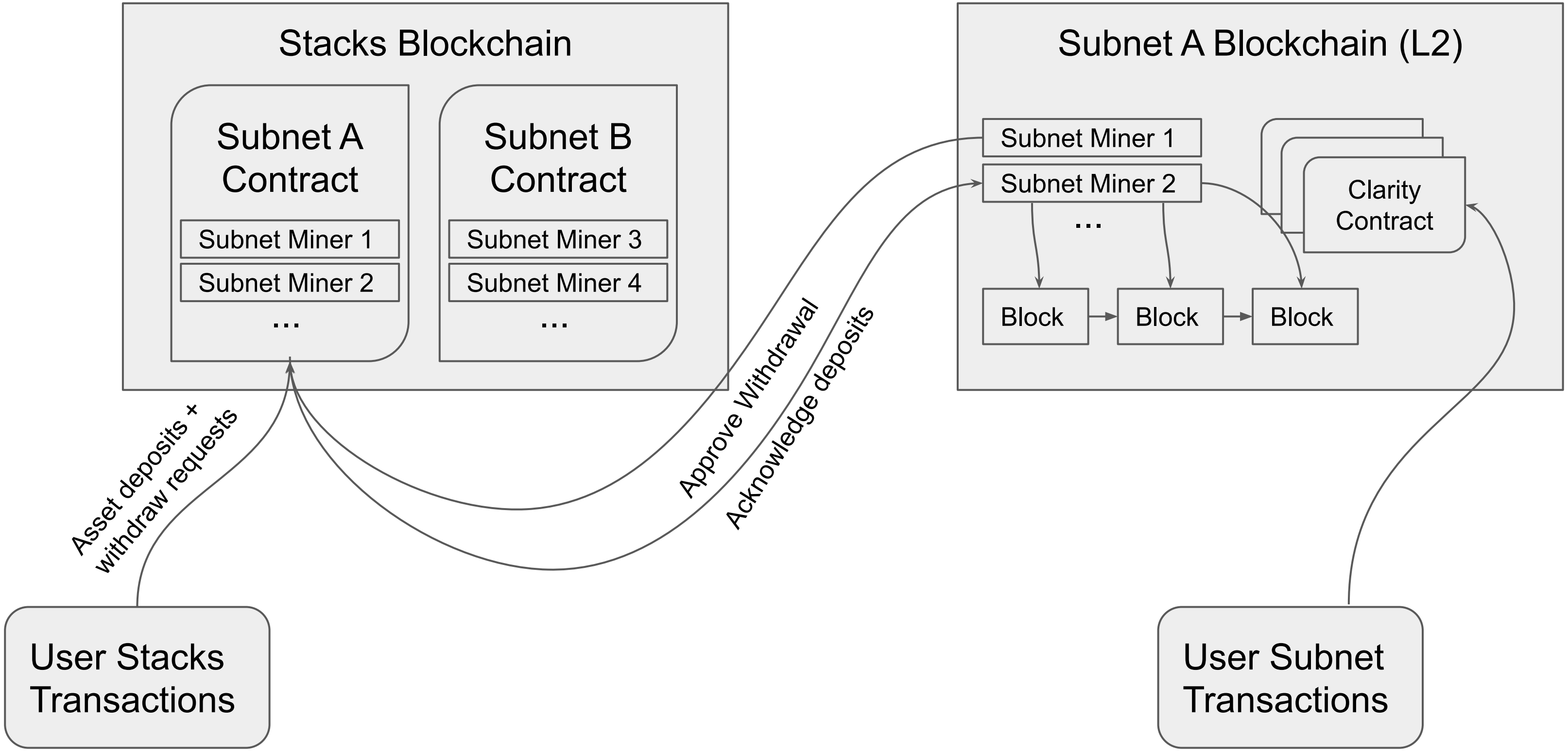
What are subnets?

- Layer 2 networks: blockchains which can confirm transactions, produce blocks independent of the Stacks chain
- Can interact with Stacks assets by depositing *into* the subnet, and withdrawing *out of*.
  - These are the only interactions with the subnet that are “visible” on the Stacks chain
- There may be many subnets on the Stacks chain, and in most cases, each subnet would be application-specific
- Subnet blocks are *lossy* with respect to the Stacks chain
  - Subnet blocks should be able to hold much more data, mutations, than Stacks blocks
  - Operations in the subnet should not have 1-1 correspondence to operations on the Stacks chain

Interacting with subnets

- Stacks subnets run Clarity VMs, and process and validate transactions exactly like the main Stacks chain:
  - Transaction wire format is the same
  - Event emitters are the same (so subnets can support explorers, APIs)
  - Subnet nodes expose a strict subset of the RPC interface
- Stacks addresses in the subnet use the same version bytes as mainnet, but transactions will use different ChainID bytes
- Wallets, to support interactions on a subnet would need to:
  - Change API backend to subnet-specific API
  - Change ChainID in stacks.js transaction construction (will need to confirm that hardware wallets will allow alternate ChainIDs)

General Subnets Architecture



Trusted, Trustless, and Incentive Schemes

- Participation in a subnet is *optional*, but when participating the amount of trust in the subnet miners can vary depending on the scheme
- Fully-trusted: miners are responsible for issuing subnet blocks, users can validate, but withdrawals are issued *arbitrarily* by a subnet miner
  - Trust can be federated with a BFT protocol of miners for block issuance
  - Federation: require majority of miners to approve withdrawals
- Fully-trustless: miners are responsible for issuing subnet blocks, users can validate, and withdrawals are issuable *only if* they correspond to a correct state in the most recent valid subnet block.
  - This is the ideal subnet, but requires that subnet blocks be validated on the Stacks chain: this either breaks the “lossy” goal, or requires novel cryptographic techniques (PCPs?)
- Incentivized trust: miners may issue arbitrary withdrawals, but can be *punished* for doing so.

Proposal for Hiro's Iterations on Subnets

- Emphasizing the *lossiness* goal of subnets
  - Approaches like peer swaps require operations on the Stacks chain linear with the number of subnet transfers
  - Jude's vector-clocking merkle tree proposal *doesn't* require linear operations, but it does require fixed membership sets
- Start with fully-trusted approach, federated miners with BFT.
- Iteration 1: incentive scheme for processing user->user asset transfers
  - This can be achieved with merkle tree proofs, periodic Stacks chain block commits, and *proof challenges* to deal with non-responsiveness

Far Future Iterations on Subnets

- Iteration 2: incentive scheme for processing *contract* asset transfers
  - This requires validating *contract execution* and is similar to the solution posed by Arbitrum
  - Clarity likely could not support this kind of validation yet
- Iteration 3: PCP Magic?
  - Probabilistically Checkable Proofs provide a theoretical framework for a Stacks smart contract to act as a “verifier” to the layer 2 “prover” -- this is similar to the promise of “ZK rollups”, and would require similar amount of refinement and research to discover it was workable

Fully-Trusted Approach with BFT

- Stacks smart contract governs the subnet
  - Specifies who the miners are
  - Receives deposits
  - Processes Withdrawals
- Subnet miners run a “stacks-subnet-node”
  - Accepts transactions
  - Exposes normal RPC interface, emits events
  - Monitors Stacks chains for withdrawal requests
  - Implements a BFT protocol with the other miners to build and issue blocks
- Subnet users do not need to run nodes
- Subnet APIs / explorers can run follower nodes

Fully-Trusted Approach with BFT: Contract Interface

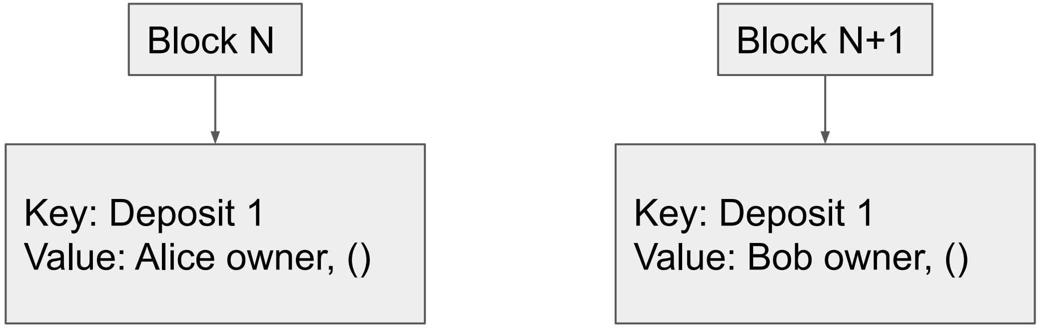
```
(define-read-only (get-miners))
 ;; returns (list principal) of the subnet’s miners

(define-public (deposit-ft (fungible-token <ft-trait>) (amount uint))
 ;; deposits a fungible token in the subnet

(define-public (deposit-nft (non-fungible-token <nft-trait>) (nft-id uint))
 ;; deposits a non-fungible token in the subnet

(define-public (request-ft-withdrawal (ft-contract principal) (amount uint))
 ;; initiates a ft-withdrawal request on behalf of the tx-sender, returns a withdrawal ID

(define-public (approve-ft-withdrawal (withdrawal-id uint))
 ;; invoked by a miner to approve a pending withdrawal

(define-public (execute-ft-withdrawal (fungible-token <ft-trait>) (withdrawal-id uint))
;; once a withdrawal has been approved, execute the withdrawal by invoking the ft contract
```

Semi-Trusted Approach with BFT

- Subnets miners stake assets in the subnet contract
- Subnet users deposit Stacks assets, received a *deposit identifier*
  - Subnet transactions operate on whole deposits
  - Deposits can be divided/split by a Stacks chain transaction with the subnet contract
- Periodic subnet blocks write a *commitment* to the Stacks contract
- The commitment is a merkle tree root reflecting the current state of deposit holders, and the signed-by-sender transforms since the last commitment:
  - Key: Deposit Identifier
  - Value: (Principal, Transformation list from Block n - 1)
- Withdrawals include the path in the current block (or are delayed to allow someone to challenge the withdrawal with a later block)

Validating and Challenging the Staked Approach

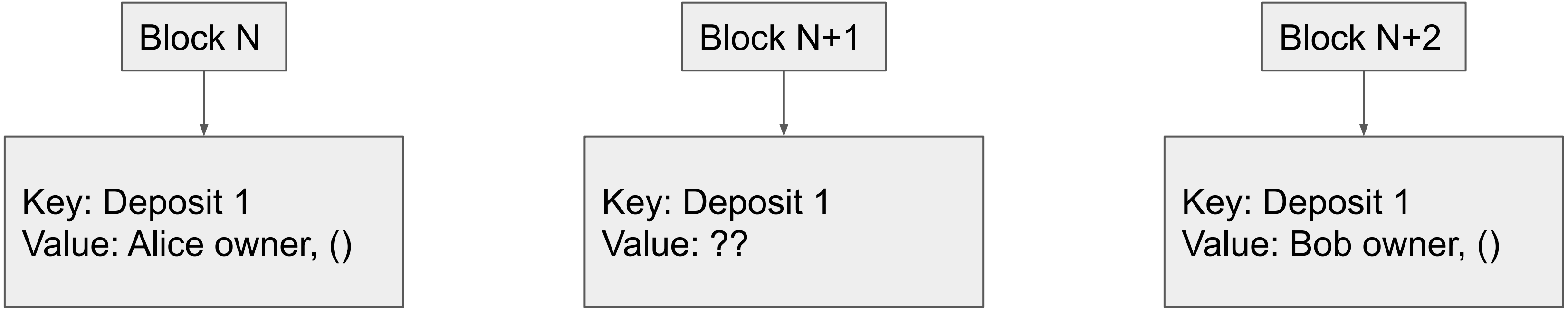
- Attack 1: Withdrawal with invented transforms
  - Miners produce a block commitment with an owner and empty transform set.



- Alice can provide the path in Block N to the smart contract as proof of misbehavior (requires Alice monitor deposit IDs she is interested in)

Validating and Challenging the Staked Approach

- Attack 2: Non-responsive mining
  - Miners just hide the contents of the block from the impacted users



- Alice can issue a “challenge transaction” on the smart contract: asking for the path and value of “deposit 1” in “block n+1”
- Non-response leads to loss of stake