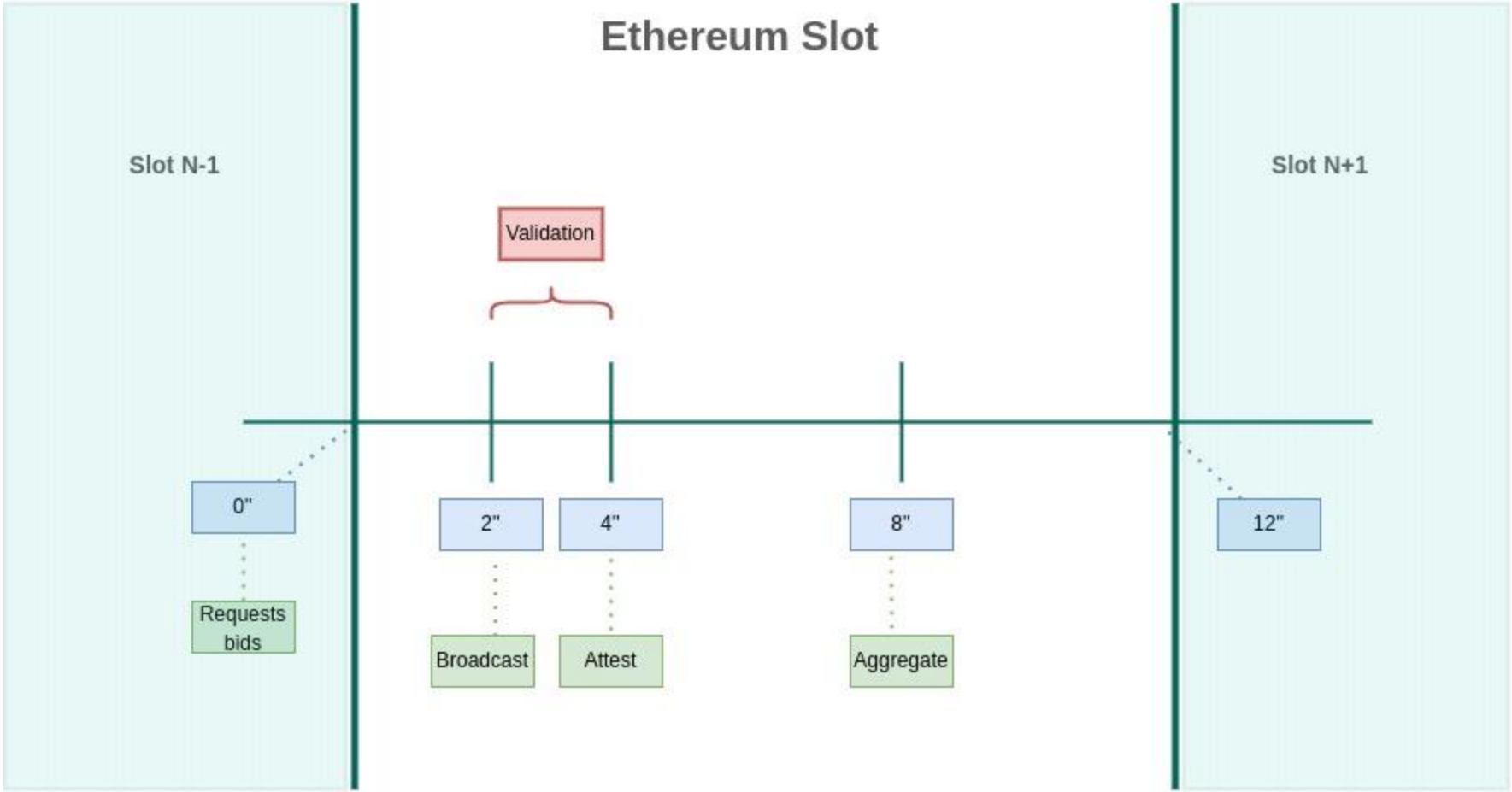
ePBS

EPF - AMA

PRYSM

A Product of Offchain Labs

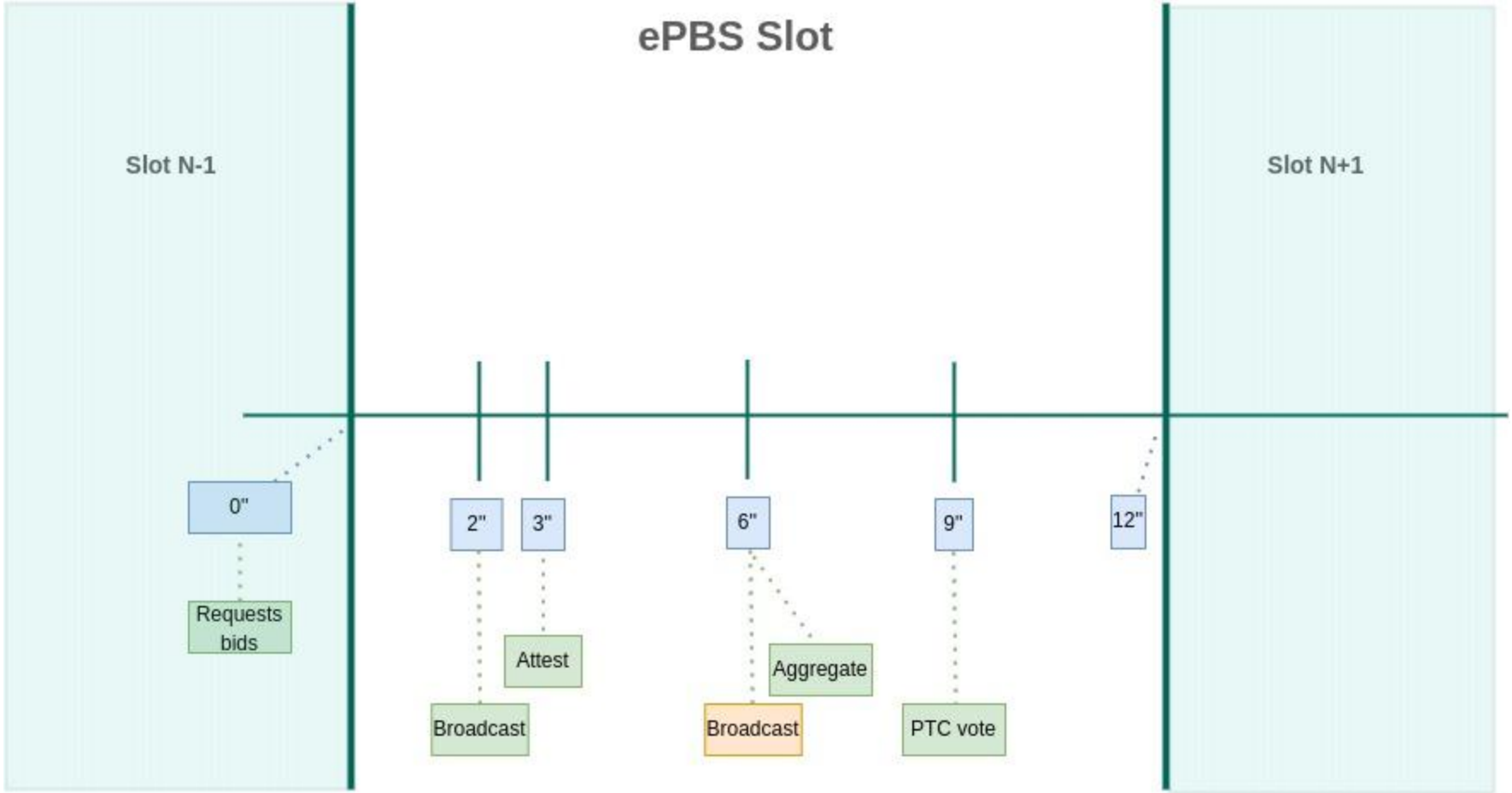
potuz@prysmaticlabs.com

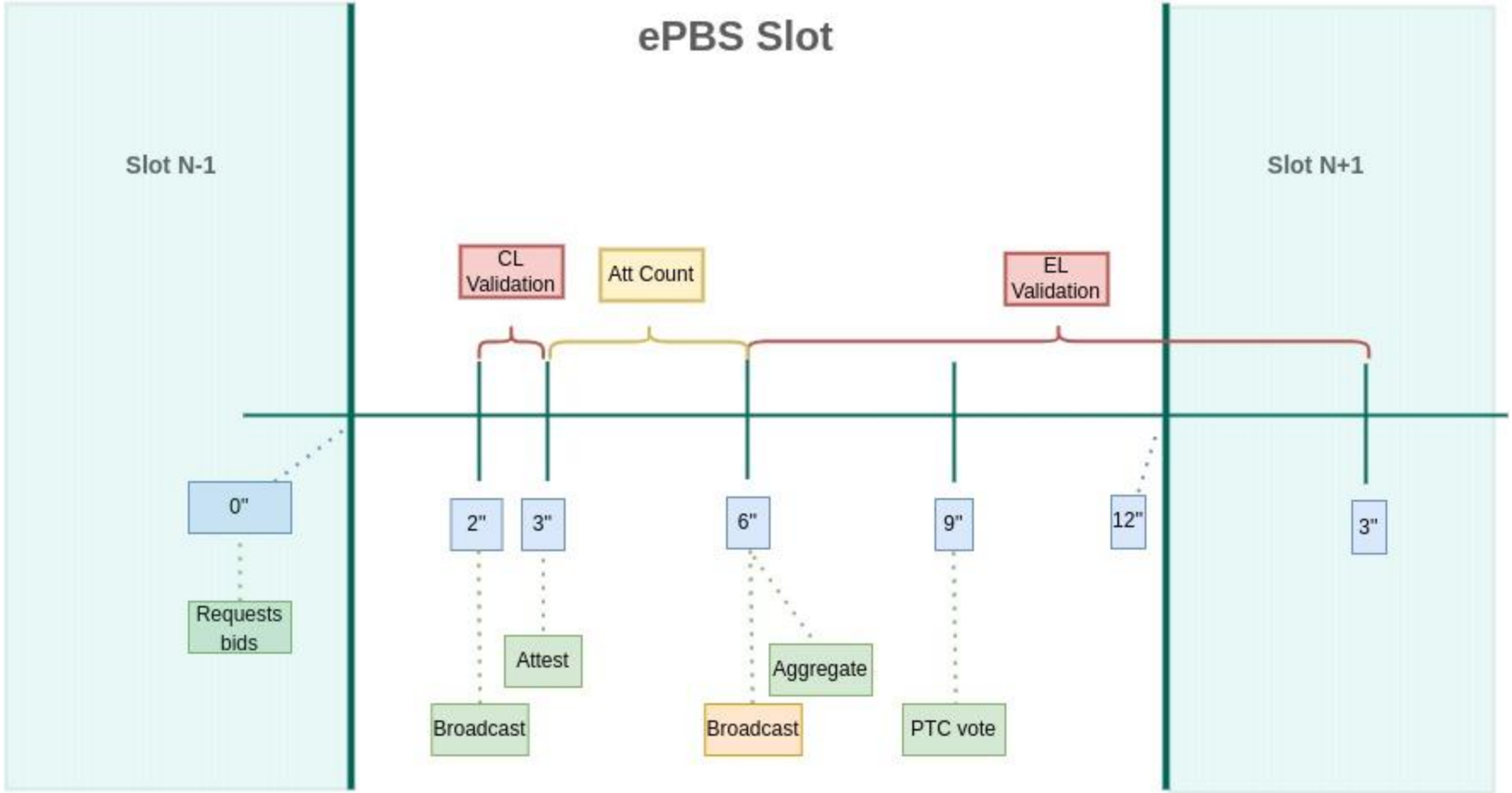

Ethereum Slot
Slot N-1
Slot N+1
Validation
0"
2"
4"
8"
12"
Requests bids
Broadcast
Attest
Aggregate

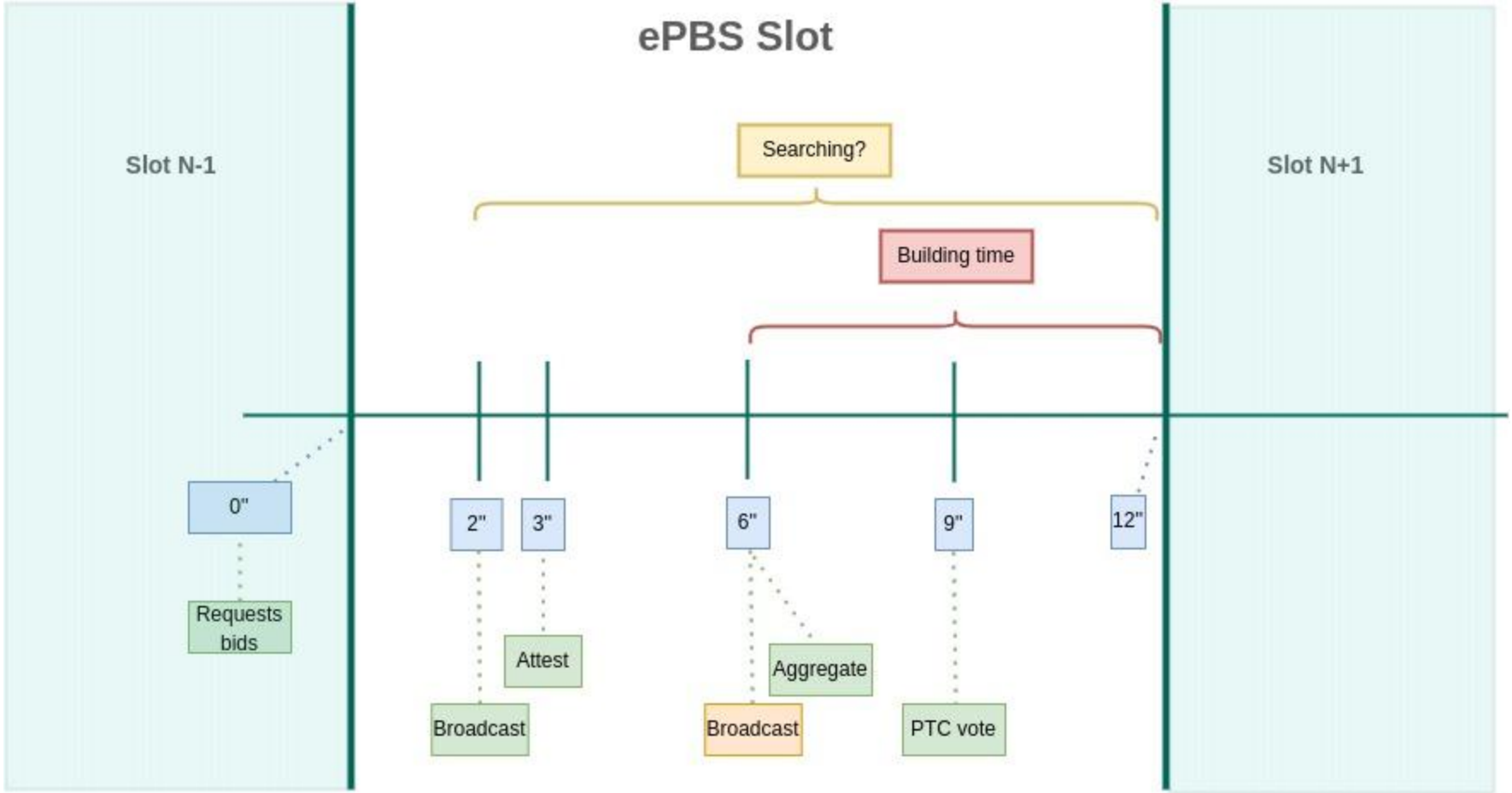
Current Ethereum Slot

- 2 seconds for payload validation.
- 4 seconds for attestation propagation (aggregation).
- 4 seconds for aggregation propagation (compute head).

- **Can we make validation longer?**
  No: timing games minimize validation inherently, proposers delay broadcast as much as possible. There are constructions with changes to the EL consensus
- **Bad utilization of CPU time**
  CPU is used only in the 2 seconds of validation and in the last milliseconds for head computing.


ePBS Slot
Slot N-1
Slot N+1
0"
2"
3"
6"
9"
12"
Requests bids
Attest
Broadcast
Aggregate
Broadcast
PTC vote


ePBS Slot
Slot N-1
Slot N+1
CL Validation
Att Count
EL Validation
0"
2"
3"
6"
9"
12"
3"
Requests bids
Attest
Broadcast
Aggregate
Broadcast
PTC vote


ePBS Slot
Slot N-1
Slot N+1
Searching?
Building time
0"
2"
3"
6"
9"
12"
Requests bids
Broadcast
Attest
Broadcast
Aggregate
PTC vote

ePBS Slot

- 9" payload validation / 1"-2" CL validation (vs. 2" for both). Higher gas limit.
- 6" payload validation for proposer (vs 10")
- 6" building time (vs 10")
- CPU usage is distributed throughout the slot.
- Highly connected builders can broadcast early.
- Builders can broadcast blobs as soon as the block is seen.

Unconditional Payment

- Builders are staked.
- Bids are signed by the builder.
- Bids are included in the CL block.
- Payment is fulfilled in the CL when processing the block (in the first 3 seconds of the slot).

```
class ExecutionPayloadHeader(Container):
    parent_block_hash: Hash32
    parent_block_root: Root
    block_hash: Hash32
    builder_index: ValidatorIndex
    slot: Slot
    value: Gwei
    blob_kzg_commitments_root: Root
```

```
class SignedExecutionPayloadHeader(Container):
    message: ExecutionPayloadHeader
    signature: BLSSignature
```

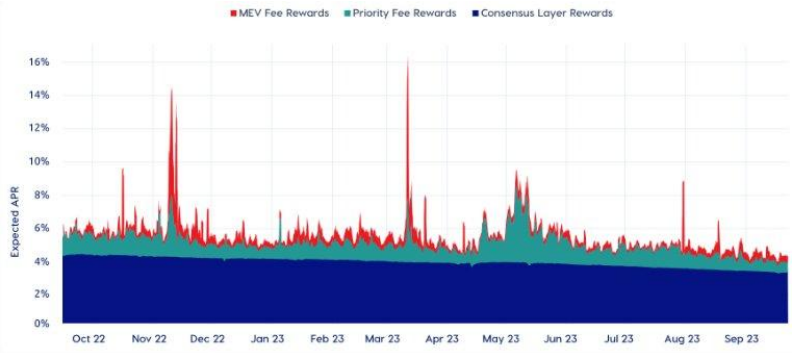
If the block is canonical, the builder is deducted the bid immediately.

Staked Builders

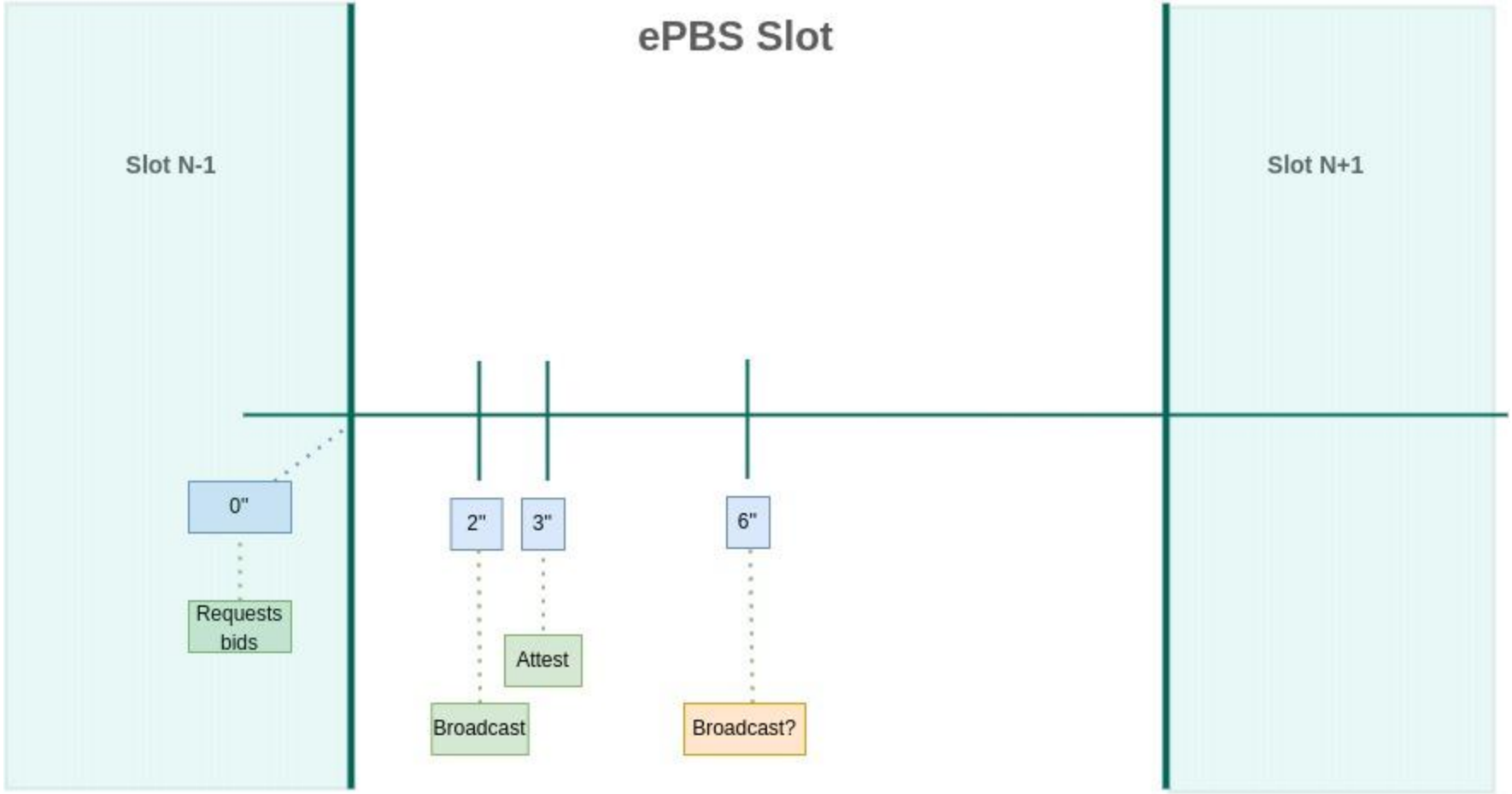
- Collateral needs to be available before the block either way as we only know how to process payment in the EL top of block (unknown ZK witchcraft is required otherwise).
- There's capital flow from the CL to the EL, is the cost of the refunding transaction critical?
- The 99.99th percentile block has ~30ETH MEV reward.
- Will builders risk not getting these blocks when they fall in vanilla proposers?

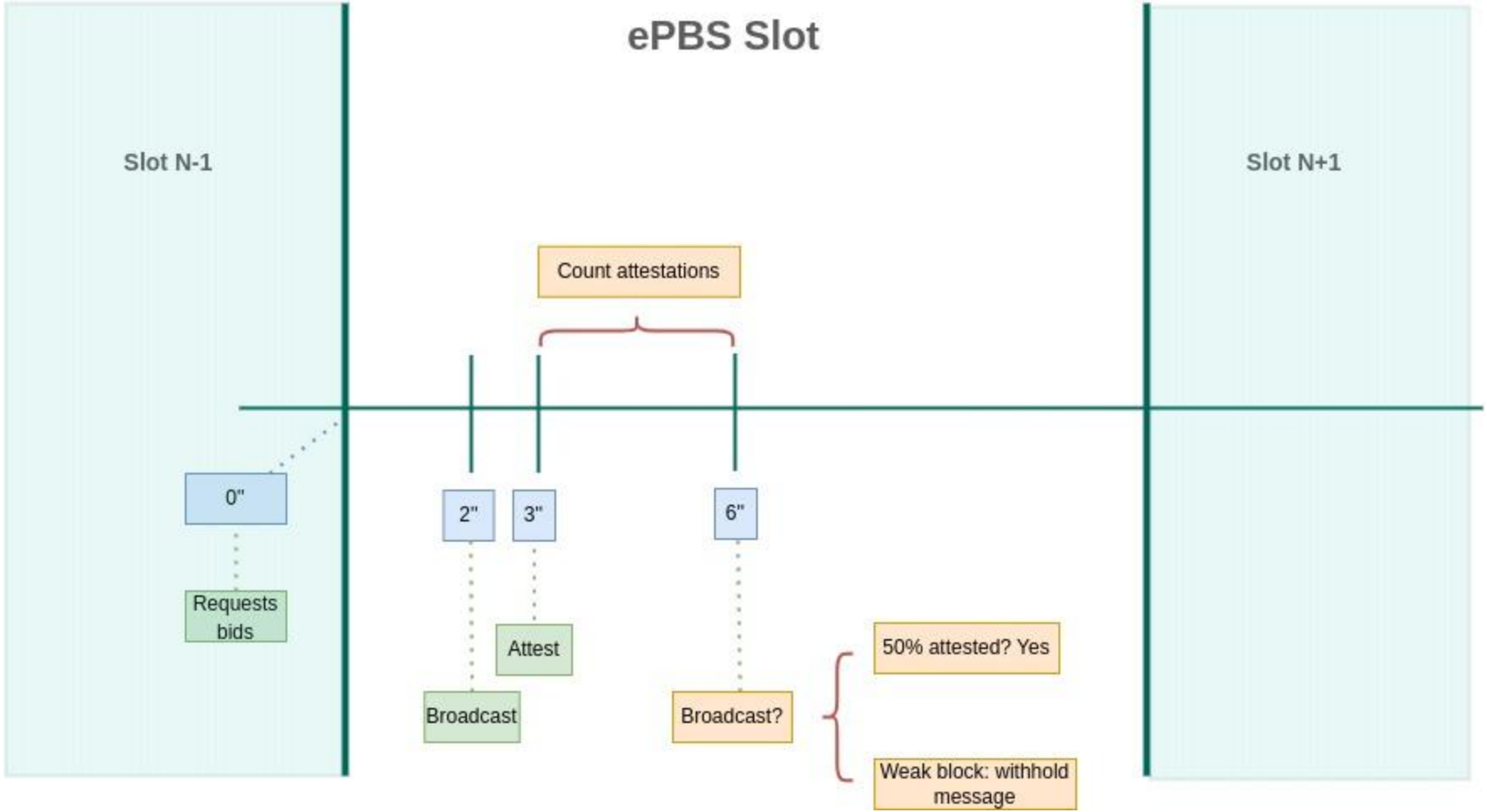
Credit: BlockScholes

	Consensus Layer*	Priority Fees	MEV	Total APR
Median**	3.836%	0.567%	0.0%	4.461%
Mean**	3.809%	1.151%	0.539%	5.499%
Std Dev**	6.758%	3.601%	7.838%	10.958%



Builder Safety Concerns


ePBS Slot
Slot N-1
Slot N+1
0"
2"
3"
6"
Requests bids
Attest
Broadcast
Broadcast?


ePBS Slot
Slot N-1
Slot N+1
Count attestations
0"
2"
3"
6"
Requests bids
Attest
Broadcast
Broadcast?
50% attested? Yes
Weak block: withhold message

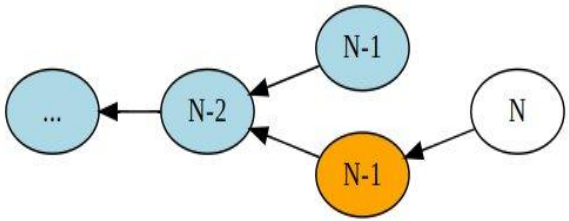
Builder's safety

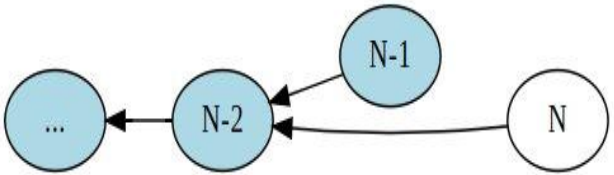
- Builders can broadcast a payload or a "withhold message".
- Builders count attestations until the 6" mark to make a decision. Do clients need to implement this? (Beacon API)
- PTC votes at 9", if there's consensus the builder receives a "boost" of 40%.
- Boost works both ways, if the payload was broadcast, honest validators will give 40% more weight to the current block as head. If a *withhold message* was present, honest validators will give 40% more weight to **reorg the current block.**

Unbundling safety

Proposers of N-1 and N collude against builder of N-1. Proposers are assumed to control network topology.

- Unbundling for current slot is **impossible**
- Reorging a payload requires 40% of stake
  - Builder loses the bid
- Reorging a block requires 20% of stake
  - Builder doesn't lose collateral
  - Builder's transactions can be slot-bound
- All attacks require preparation at N-1.

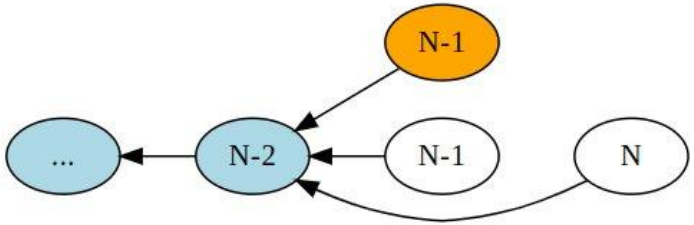


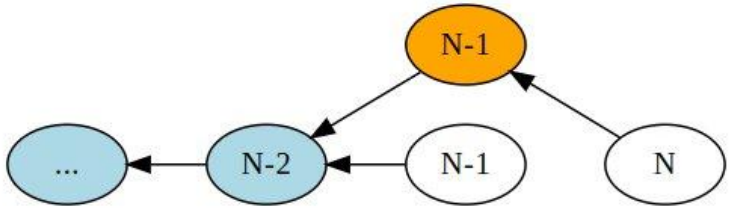


Withholding safety

Proposers of N-1 and N collude against builder of N-1. Proposers are assumed to control network topology.

- Grieving the builder requires 20%





Time constraints

- PTC votes on payload presence at 9". **Only presence, not validity.**
- As soon as 50% of the committee is seen, builder's can broadcast safely their payloads. In the normal case this should give up to *5" to propagate the payload.*
- Head is only known at 6":
  - Builder of N+1 has only 6" to build the next block.
  - Builder of N has extra 4"~6" to build it. Can this be exploited?
- Blobs can be sent 4" before the payload is broadcast.
- How does the searcher market change?

The Auction

Builders as vertically integrated relays

Builders can open HTTP endpoints to serve bids.

Builders lists can be kept open on-chain.

Only serve a bid when requested (if deemed appropriate).

Serving multiple bids leads to builder complexity.

P2P propagated bids will be heavily rated and are unlikely to win the auction (they set the floor price).

Bid cancellation is possible within the builder's endpoint, but not once it's served and signed.

Sealed nature

- We can seal bids in-protocol: proposer signs the bid request on the HTTP endpoint and encrypts it to the builder.
- The auction seems to be sealed regardless: builders bidding on a service-providing relay, cannot trust others aren't bidding on their own endpoints or even off-protocol.
- Are there proposer strategies to request and publicize builder's bids?
- Should the protocol enforce the minimum bid to be specified by P2P? (is there any censorship resistance gained from this?)

Relays post ePBS

In https://ethresear.ch/t/relays-in-a-post-epbs-world/16278 it is highlighted that relays offer an advantage over ePBS for:

- ~~Cancellation support.~~
- Payment flexibility
  - Allows for bottom of block payment verification.
  - Requires off-protocol software (or the relay is a defacto-builder).
  - Requires either staked builders or loosing vanilla-assigned blocks.
- Will proposers and builders use this?
- Do we even care?

Relays post ePBS

- Either the relay needs to be staked and sign the payloads or, as today, the proposer signs as if it were self-building
- Both cases require JIT full validation of the payload (vs the 8"--11" in-protocol).
- Both cases require a few extra networking roundtrips: Builder <-> Relay, Proposer <-> Relay, all before broadcasting the CL block in time to be attested.

Future/Alternative designs/addons

Slot auctions

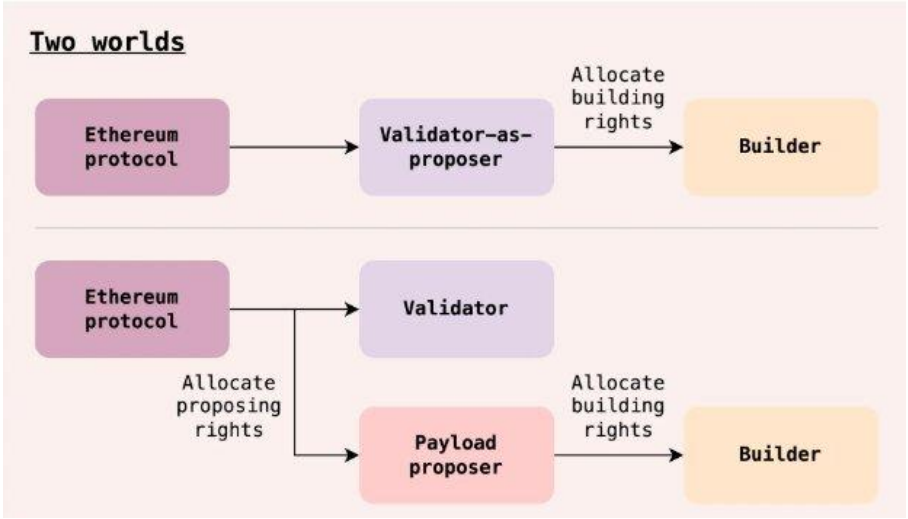
- Slot auctions are a simple change from this version of ePBS: not commit to a payload hash.
- Builders get 12" instead of 6" to build a block (no advantages to ticket holder)
- Builders can run MEV-Boost off-protocol and resell.
- Do we have to deal with equivocations?
- Do we require a new voting round?
- Do we have to deal with slashing conditions?
    - There are no FFG problems as the payload commits to the same target as the CL block.
    - There are no LMD problems as the next proposer resolves the split view.

Inclusion Lists

- The design solves some of the fundamental problems of forced inclusion lists (EIP 7547) because of (block, slot) voting.
- It requires an EL fork.
- Inclusion lists are not mandatory if self-building is possible.

Execution tickets

- Execution tickets can be implemented as an evolution of slot auctions.
- The protocol carries the auction and not the proposer.
- The same forkchoice concerns as in slot auctions are now valid since the payload is not bound to a given consensus block.



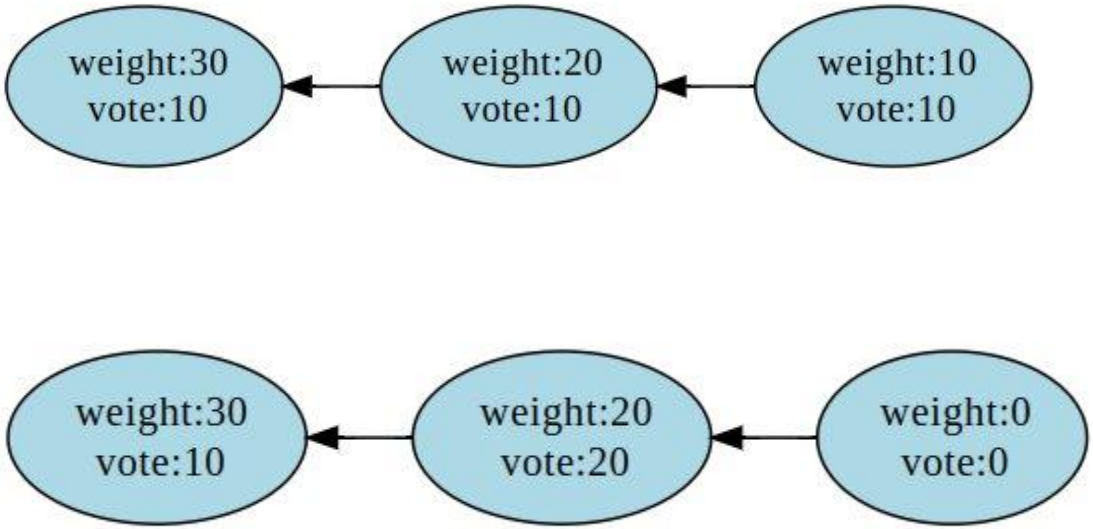
Credit: Barnabé Monot

Some implementation complexities

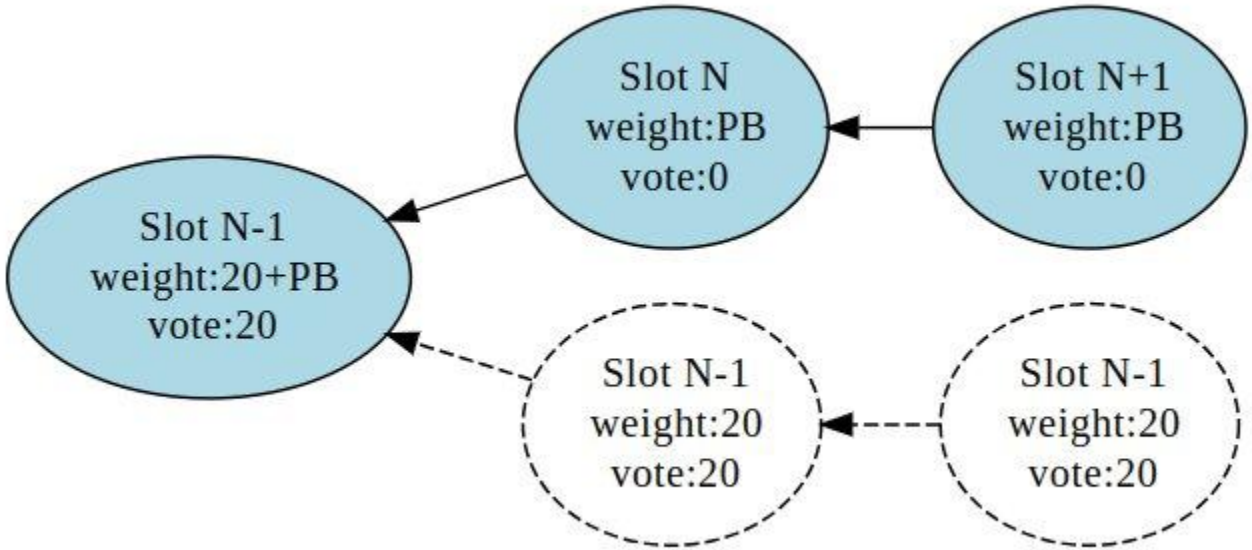
CL-EL locking: Withdrawals

- Withdrawals are determined by the beacon state but fulfilled in the EL.
- Proposer of N deducts the withdrawals on the CL, regardless of payload.
- The payload of N is not valid unless it fulfills these withdrawals.
- No payload is ever valid until these withdrawals are fulfilled, hence future CL blocks cannot process withdrawals until a valid payload has appeared.
- EL triggered withdrawals do not add much more complications.

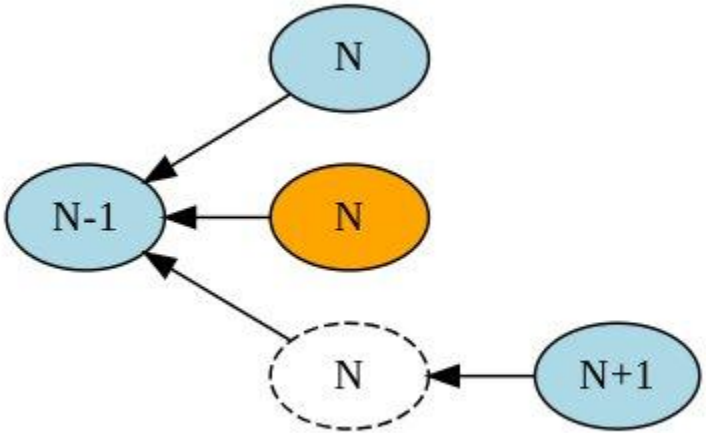
Forkchoice Complexity

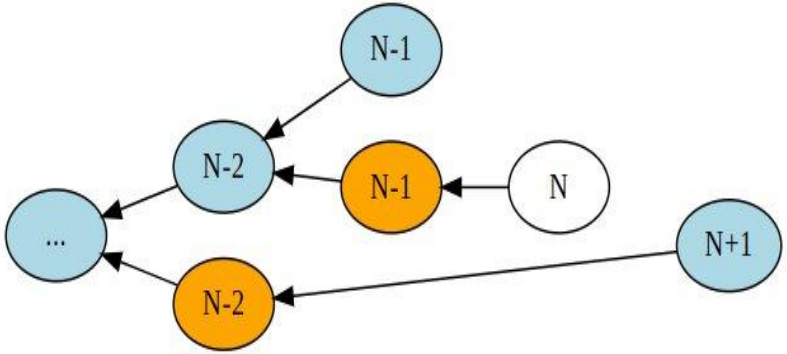


Forkchoice Complexity



Forkchoice Complexity





Thanks!