Revisiting Ethereum's Exit Queue

Mallesh Pai (SMG) & Mike Neuder (EF)

(Good) Friday – April 18^{th}, 2025

Beam Call #4

Contents

Preliminaries

Why do we rate limit exits?

Definition (Neu, Tas, and Tse 2023)

Accountable safety means that in any case of inconsistency, a certain fraction of validators can be identified to have provably violated the protocol.

- **Why:** Foundation of the underlying *economic security* of a Proof-of-Stake protocol.
- **Core tension:** Many consensus models assume static validator sets, but real-world blockchains need to allow people to enter and **exit**.
- When you allow a validator to exit, they may cause a safety violation and exit the protocol before detection: in this case, their stake cannot be held accountable.

Model

- We model the Accountable Safety requirement as a collection of validator set consistency constraints:

$$\mathcal{C} = \{(\delta_1, T_1), ..., (\delta_k, T_k)\}$$

- "No more than δ_1 of the stake can exit within T_1 days" and "no more than δ_2 of the stake can exit within T_2 days."
- e.g., "No more than 10% of the stake can exit within 7 days" and "no more than 2% of the stake can exit within 0.5 days."

Model

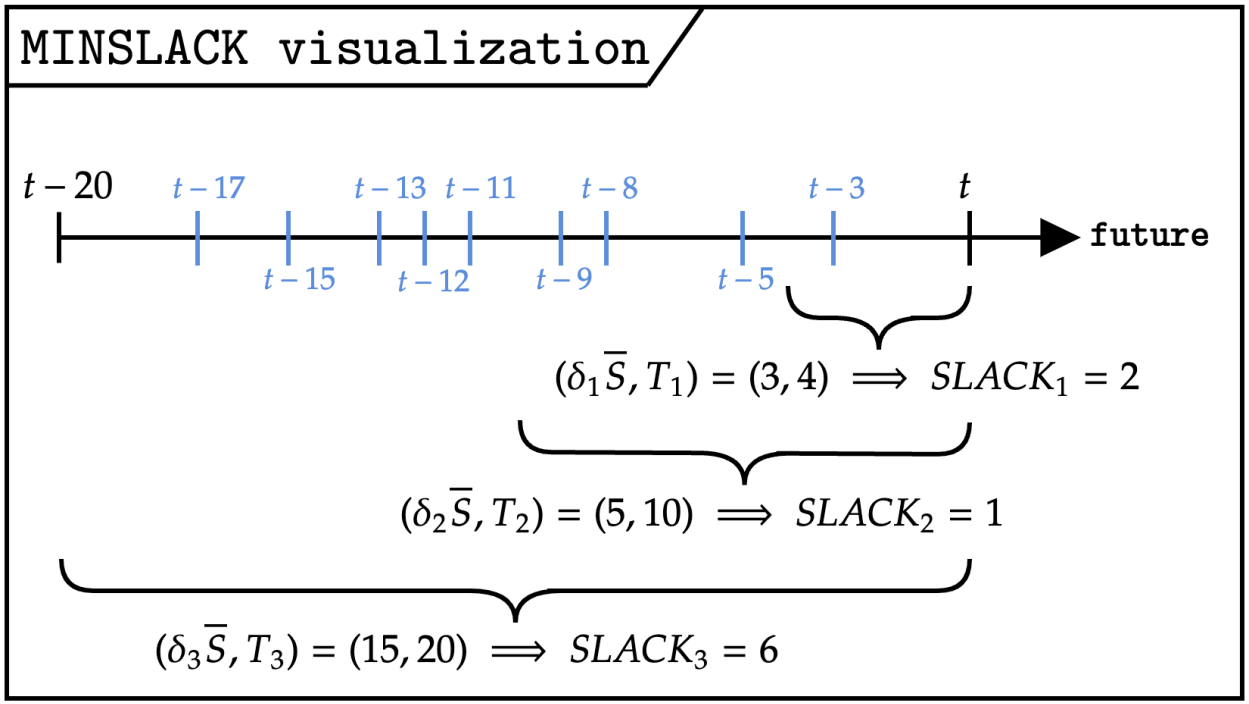
- Time is discrete and represents when withdrawals can be processed (e.g., epoch boundaries in Ethereum).
- Each agent has a cost of waiting, $c_i > 0$. Thus, their overall disutility is their cost times the amount of time they spend in the queue:

$$U_i = -c_i \Delta_i.$$

- We want to minimize the expected disutility of the validators under some arrival process for new withdrawal requests *under the protocol constraints*.

MINSLACK

- "At each period, greedily process as many withdrawals as possible given the constraints."
- Simple example: $\mathcal{C} \implies \{(3,4),(5,10),(15,20)\}$



- $\texttt{MINSLACK} = 1 \implies$ "process one withdrawal in time t."

Introducing heterogeneity

- `MINSLACK` is optimal under common values...
- **Intuition**: People might have **extremely** different time preferences (e.g., you need to top up an AAVE borrow position that's borderline, I just want to rotate my keys).
- With heterogeneous values, `MINSLACK` is no longer optimal (with respect to the welfare loss given each validator's "cost", c_i) because it doesn't reserve any capacity.

Optimal under heterogeneity & PRIO-MINSLACK

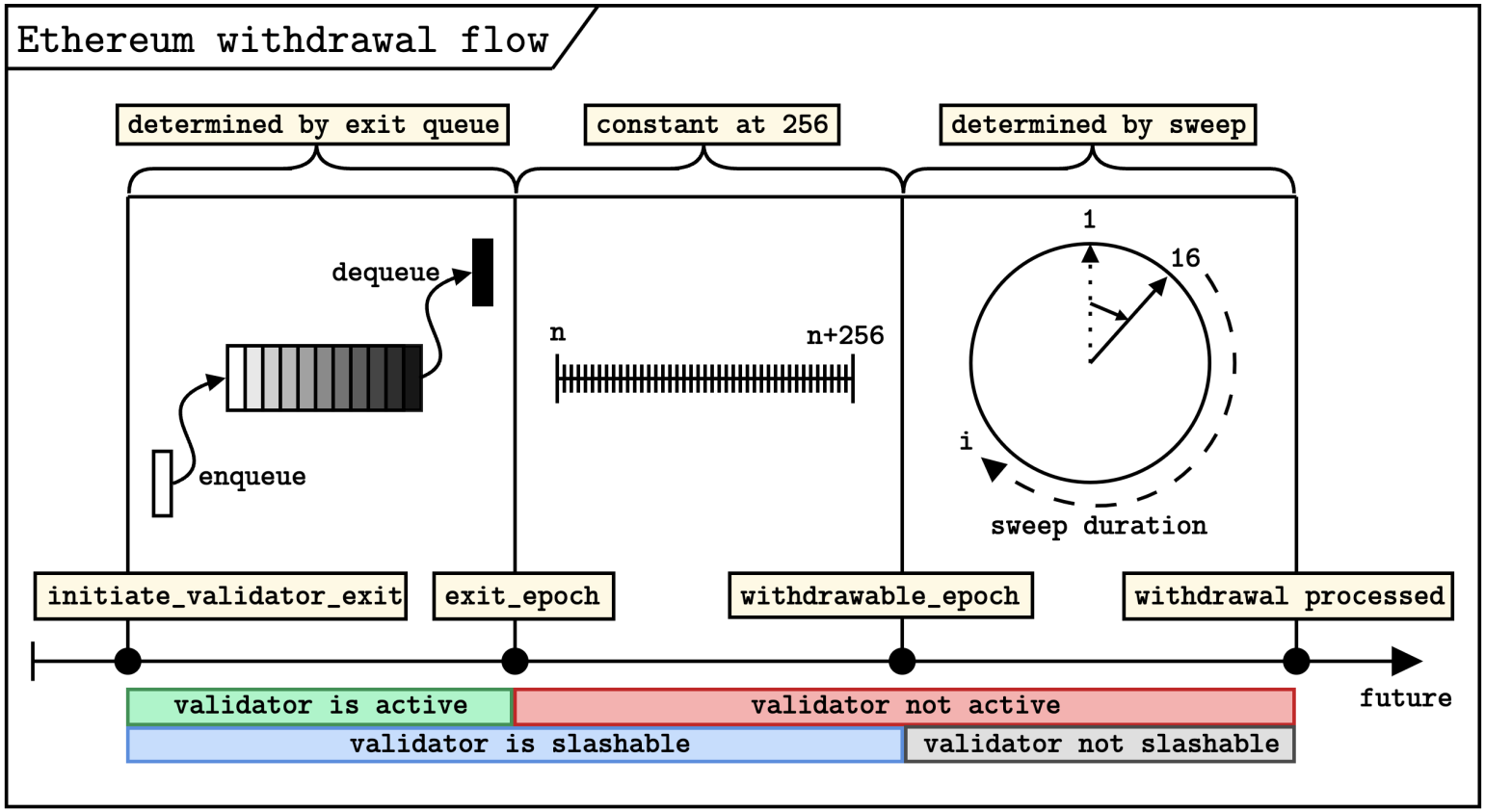
- Given the (i) history, (ii) pending withdrawals, and (iii) distribution of future arrivals, we need to determine how much capacity to consume and how much to reserve for future "high disutility" arrivals.
- $\uparrow$ is solvable with an MDP but super messy — what if we just did a priority queue instead of FCFS: PRIO-MINSLACK.

A few numerical results

Algorithm	Arrival dist.	Value dist.	Discount	Performance
OPTIMAL	$Y \sim [0,1,5]$ *w.p.* $[0.5,0.4,0.1]$	$X \sim [1,5]$ *w.p.* $[0.9,0.1]$	0.9	-2.428
PRIO-MINSLACK				-2.422
OPTIMAL		$X \sim [1,10]$ *w.p.* $[0.9,0.1]$		-2.959
PRIO-MINSLACK				-3.005
OPTIMAL		$X \sim [1,20]$ *w.p.* $[0.9,0.1]$		-3.902
PRIO-MINSLACK				-4.151
OPTIMAL	$Y \sim [0,1,2]$ *w.p.* $[0.4,0.4,0.2]$	$X \sim [1,10]$ *w.p.* $[0.9,0.1]$	0.9	-1.637
PRIO-MINSLACK				-1.638
OPTIMAL	$Y \sim [0,1,5]$ *w.p.* $[0.5,0.4,0.1]$			-2.925
PRIO-MINSLACK				-2.969
OPTIMAL	$Y \sim [0,1,10]$ *w.p.* $[0.6,0.35,0.05]$			-3.610
PRIO-MINSLACK				-3.620

- Depends a lot on your arrival and value distributions!
- But for most common distributions, `PRIO-MINSLACK` is pretty close to optimal, so just do that.

Ethereum exit flow today



- A rate-limited FIFO queue. Currently 16 validators per epoch.
- A fixed-length delay of 27.5 hours.
- A variable-length sweep by validator index. Currently 9.2 days.

What Prague/Electra changes

- **tl;dr;** *not too much.*
- Highlighting three features of EIP-7251 relevant for beam chain:
  1. Exit queue itself is denominated in ETH instead of # of validators.
  2. Validators can choose to compound their stake and trigger partial withdrawals from their withdrawal address.
     $\rightarrow$ an opt-out of the sweep!
  3. Validators can trigger full exits from their withdrawal address.
     $\rightarrow$ validators have a CL key-pair *and* a withdrawal credential!

Our proposal: single-constraint MINSLACK

Per-epoch constraints

1. $16/1,065,882 = 0.0015\%$ in one epoch,
2. $32/1,065,882 = 0.0030\%$ in two epochs,

   ...
3. $3600/1,065,882 = 0.34\%$ in 225 epochs (one day),

   ...
4. $50,400/1,065,882 = 4.7\%$ in 3,150 epochs (two weeks),

   ...
5. $108,000/1,065,882 = 10.1\%$ in 6,750 epochs (30 days),

   ...
6. $324,000/1,065,882 = 30.4\%$ in 20,250 epochs (90 days),

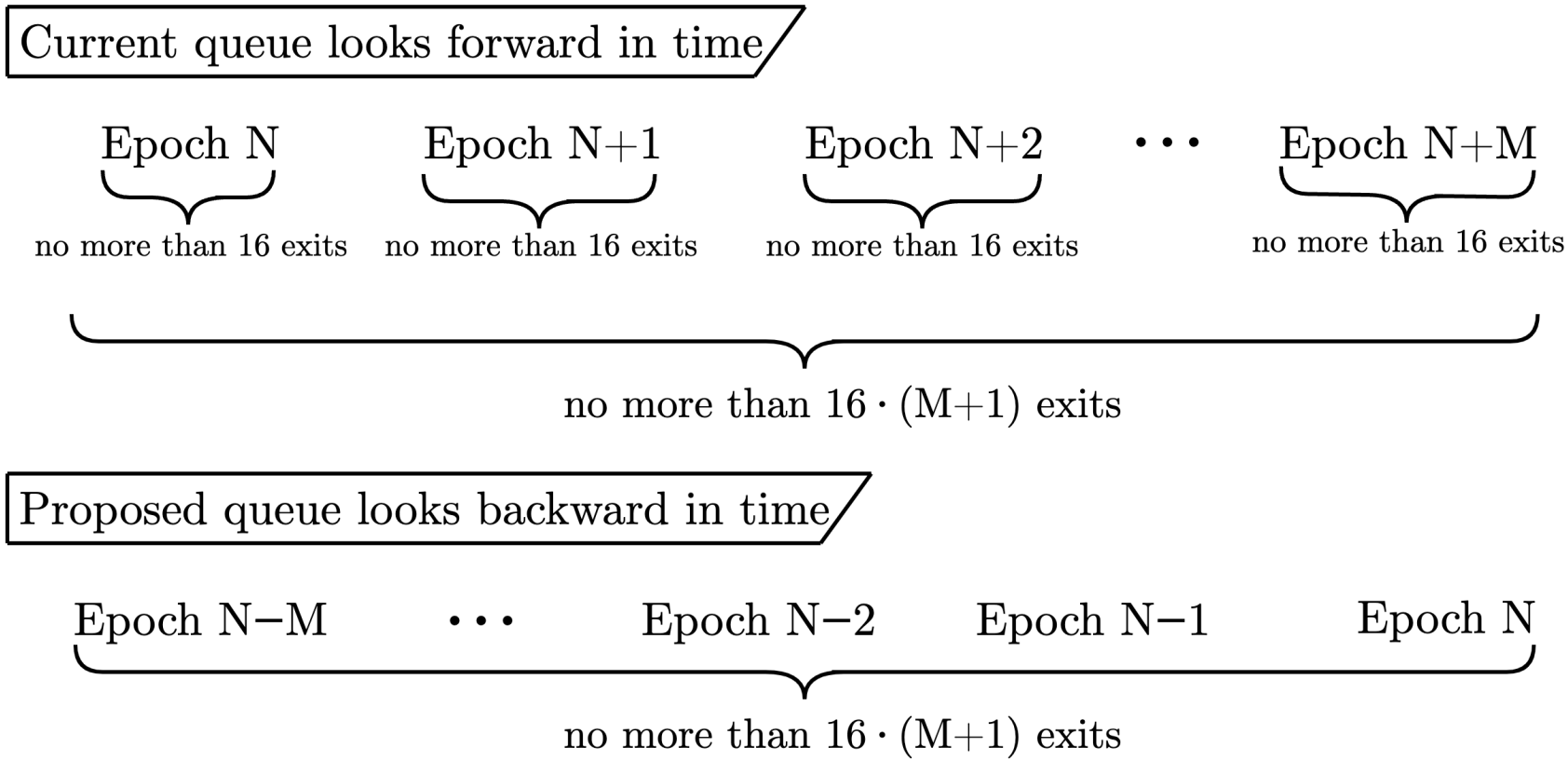
- We are imposing a new constraint *per epoch*.
- **In reality, this is probably too many constraints.** No one needs per-epoch granularity on the rate of degradation of their finality guarantee.

Our proposal: single-constraint MINSLACK

Flexible

Our proposal: single-constraint MINSLACK

Retrospection



- **Looking back in time is all we need.**

Our proposal: single-constraint MINSLACK

Spec highlights

Preset

Name	Value	Comment
`EPOCHS_PER_CHURN_GENERATION`	`uint64(2**8)` (= 256)	~27 hours
`GENERATIONS_PER_EXIT_CHURN_VECTOR`	`uint64(2**4)` (= 16)	~18 days
`GENERATIONS_PER_EXIT_CHURN_LOOKAHEAD`	`uint(2**1)` (= 2)	
`EXIT_CHURN_SLACK_MULTIPLIER`	`uint(2**3)` (= 8)	

New State Variables

Add the following to the state:

```
exit_churn_vector: Vector[uint64, GENERATIONS_PER_EXIT_CHURN_VECTOR]  # total exited balance per generation for GENERATIO
```

- We use "generations" to simplify the accounting.

```
for generation in range(oldest_generation, current_generation):
    generation_index = generation % GENERATIONS_PER_EXIT_CHURN_VECTOR
    churn_usage = state.exit_churn_vector[generation_index]
    if churn_usage < per_generation_exit_churn:
        total_unused_exit_churn += per_generation_exit_churn - churn_usage
```

- Some tech debt (e.g., explicitly setting `exit_epoch`).

Further proposals

- **Get rid of the partial withdrawal sweep.**
  - Either remove the sweep all together or separate partials out.
  - This saves an average of 4.6 days per full exit!
- **Priority fee ordering for withdrawals.**
  - Implementing `PRIO-MINSLACK` is a clear improvement.
  - *Especially* if different stakers have varying speed preferences (e.g., meeting a margin call versus rotating your keys).

Beam chain suggestions

1. Denominate in ETH.
2. No sweep.
3. Single constraint `MINSLACK`.
4. Pay for priority: `PRIO-MINSLACK`.
5. Native integration in the execution layer.

thanks :)

1. https://arxiv.org/pdf/2406.05124
2. https://hackmd.io/@mikeneuder/eli5-ethereum-validator-exits
3. https://eips.ethereum.org/EIPS/eip-7922
4. https://ethresear.ch/t/adding-flexibility-to-ethereums-exit-queue/22061