DVT in Ethereum's PoS: Gains and Losses

Mikel Cortes-Goicoechea[1*], Tarun Mohandas-Daryanani[1], Jose Luis Muñoz-Tapia[2], Leonardo Bautista-Gomez[3]

[1*]Barcelona Supercomputing Center, Barcelona, Spain.
[2]Universidad Politécnica de Catalunya, Barcelona, Spain.
[3]MigaLabs, Barcelona, Spain.

*Corresponding author(s). E-mail(s): mikel.cortes@bsc.es;
Contributing authors: tarun.mohandas@bsc.es;
jose.luis.munoz@upc.edu; leobago@protonmail.com;

Abstract

With Ethereum's transition to Proof-of-Stake, the emphasis on network resilience has become paramount, necessitating mechanisms that ensure the network's robustness against all types of failures. Existing approaches have focused on achieving resilience at the consensus, software, and network-layer levels. However, these measures do not directly address validator or deployment-level resilience, where the penalty for conducting concurrent duties using the same key, known as "slashing," poses a significant risk for the staked capital. This paper explores the integration of Distributed Validator Technology within Ethereum's Proof of Stake framework to enhance validator resilience and fault tolerance. Thus potentially mitigating the single point of failure risk associated with traditional validator setups. The paper articulates DVT's pivotal role in advancing the staking landscape of Ethereum, providing a comprehensive empirical analysis of Distributed Validator Technology within Ethereum's staking ecosystem and offering a critical assessment of its performance, reliability, and profitability compared to traditional validator setups.

Keywords: Ethereum Staking, DVT, Staking Resilience, Ethereum Rewards

1 Introduction

Resilience is vital for any application that prevents service outrages from failure in critical infrastructure [31]. With multiple available methods to achieve it [14], resilience

has been present in the modern era since humans were aware of the system's likelihood of failure [33], offering a backup plan to overcome the disruption.

As systems tend to grow in size and complexity, so does the error and risk rate [25]. The concept of "resilience" also reached the software engineering field, which greatly benefited from it as single-core hardware and programs evolved to bigger and more complex multi-threaded or even distributed clusters of nodes at High-Performance Computing (HPC) [15]. Then, the idea of "fault tolerance" was introduced, aiming to provide solvency to systems, preventing them from critical failures [30] or at least define the recovery process to do so.

Following the same philosophy, Ethereum's Proof of Stake (PoS) includes several resilience techniques that ensure the protocol can tolerate certain failure rates or recover from a vital disruption. The consensus is designed to finalize beacon states[1] as long as the network achieves a 66% of honest participation on the consensus by attesting the corresponding blocks [5].

On a separate level of resilience, at the software level, the Ethereum community supported five teams to develop and maintain five open-source clients to participate in the network. This variety of software reduces the chances of a "single point of failure" at consensus from a software level. At the network level, based on the distributed nature of the project, each node relies on a set of nodes per GossipbSub topic [34] to receive chain notifications such as blocks, attestations, or sync committees. A rotation of these connections, with a fast node discovery and the aggregation of more advanced techniques like message gossiping, considerably reduces network attack vectors like Eclipse or Sybil attacks.

We've seen how the network considers several layers of resilience at the consensus level, at the software level, and at the network layer. However, it doesn't provide any resilience that could be applied at the validator or deployment level. In fact, the consensus somehow prevents such techniques, as sharing different duties at the same slot signed with the same key leads to an imperative penalty, a "slashing" [19]. This consensus design showcases the thin action line that validator agents work in, where their retribution for participating in consensus might stop if anything happens to the machine hosting the software. Some projects like Vouch[2] have emerged, allowing the reduction of this single client failure. In this case, Vouch is a multiplexer client placed as a middleware between the validator and the beacon node. Based on different strategies, it decides which node is chosen to participate in the network, either the fastest replier node or the node that produces the highest reward. This schema guarantees that as long as one of the nodes in the cluster stays up, the validator will still be able to submit its duties. However, as the penalty reports a significant risk for complex developments that ensure a 100% uptime, there is still a huge gap for a validator multiplexer with high-security guarantees.

It is at this step that new technologies like Distributed Validator Technology (DVT) have emerged to fill the missing chapters in distributed staking software. Based on the idea of signature aggregations using private BLS key shares [7] from a Distributed Key Generation (DKG) protocol/ceremony [18], this technology provides a specific degree

[1] Beacon State refers to the global state of the entire network, including the status and balances of all validators, while providing a snapshot of the consensus among validators.
[2] https://github.com/attestantio/vouch

of fault tolerance based on the number of participants in the cluster. The bigger the cluster, the higher the fault-tolerance degree in terms of nodes that can still produce the validator duties.

Despite the idea sounding good on paper, there is a side effect problem with this DVT technology: it adds an extra step before validators broadcast their duties to the network, which is the aggregation of partial signatures. Gasper's fork choice [29] contemplates the idea of asynchronously arriving commits. However, to prevent some attack vectors, the PoS protocol of Ethereum sets economic incentives to prevent them. The protocol sets limited time ranges to compose and broadcast each of the duties, ensuring that the duties need to be pseudo-synchronous for an optimal reward.

In other words, to simplify and ease the chain's fork choice and consensus, Ethereum's PoS reduces the consensus rewards as validators take longer to accomplish and broadcast their duties. While DVT offers a novel, more resilient way of staking, it is unclear whether it can match the performance of classic (non-distributed) validators. This opens a legitimate question of how this tight time sensitivity on the protocol side can interfere with the profitability of a DVT validator cluster. In particular, for setups that include clusters of nodes distributed worldwide and from different cloud providers, it is necessary to demonstrate that despite their latency, DVT can still provide the same level of performance as classic validators. This is the objective of this study.

In this paper, we present a thorough analysis of the performance and profitability of DVT validator clusters (the approach of Obol Labs, to be precise) compared to the classical validator approach. We present a clear comparison of the rewards obtained using both approaches, showcasing the latency restrictions that could affect them and detailed insights into the internal metrics of a DVT cluster.

The remainder of this paper is organized as follows. Section 2 presents Ethereum's PoS from a validator's perspective. Section 3 explains the internals and functioning of Obol's approach to implementing the DVT. Section 5 presents the different tools, indexes, and setups used to perform the analysis. Section 6 introduces the main study's results after running a DVT validator cluster compared to its counterpart, the standard validator software. Section 7 presents the internal performance indicators of the DVT cluster while operating. Finally, Section 8 concludes this work and presents some guidelines for setting up a DVT cluster.

2 Background on Ethereum Consensus

Since "The Merge" [9], Ethereum's consensus has been achieved through PoS and the interaction between validators. The network remains split into two separate layers that need each other, as Figure 1 shows: the Execution Layer (EL) and the Consensus Layer (CL). The first one still tracks, validates, and executes the transactions that users share to the network, including the processing of the *Smart Contracts* [36] through the Ethereum Virtual Machine (EVM) [20]. On the other hand, the second one, the CL, defines, captures, and processes the interaction between validators in their attempt to reach consensus through Gasper FFG for finalization [8], LMD GHOST [27] as a fork-choice rule and RANDAO [2] as RNG to assign the following duties. As Figure

1, Ethereum now needs several software to run: i) an Execution Client that operates in the Execution Layer and executes block payloads (transactions, smart contracts, etc.), ii) a Beacon Client or Beacon Node that operates in the Consensus Layer and decides which is the canonical chain with the rest of the network, and iii) a Validator Client that only communicates with the beacon client to sign the needed duties with the validator's private key.

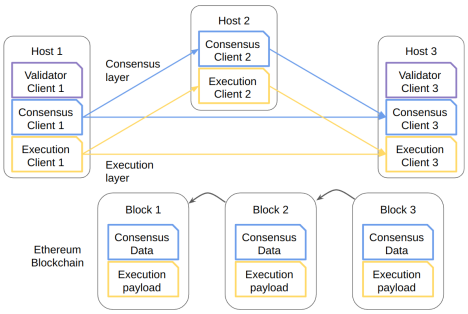

Fig. 1 Representation of the distinct software and layers involved in Ethereum's PoS network.

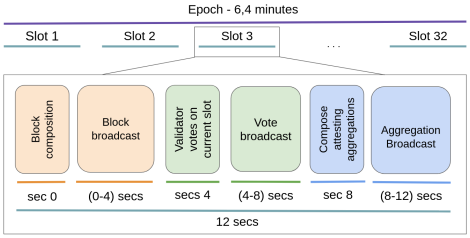

Fig. 2 Expected temporal division of a slot, including the time each duty should be sent and propagated over the network.

Due to our goal of analyzing the profitability and reliability of DVT validators, the paper will focus entirely on the interaction between the validators and the CL.

The consensus layer of Ethereum is organized in epochs. Each epoch contains 32 time windows of 12 seconds called *slots* where a single validator elected from the RANDAO Reveal algorithm has the chance to aggregate a new block to the beacon chain. Since the rest of the existing active validators must reach a consensus over each proposed block, splitting the epoch in 32 slots helps reduce the computational load of processing the duties of 900.000+ (at the time of writing this paper) active validators[3]. Thus, the whole list of validators is divided into the 32 slots and then into a maximum of 64 committees. This way, each added block serves as the main unit of time where new historical data is added to the beacon chain. Therefore, each block includes validators' attestations through aggregations, which, due to the low frequency at which validators need to produce them, are one of the main performance and steadiness indicators for analyzing and comparing the performance of a validator. This leaves us three main duties describing a validator's participation and profitability in Ethereum's PoS.

Attestations

Each validator in the CL is assigned to a specific slot and committee per epoch. Furthermore, it must contribute with votes that later on are collectively aggregated before their inclusion in subsequent proposed blocks. This aggregation process significantly decreases block size by efficiently tracking validator duties and streamlines the process for future block proposers, who only need to consider the most recent aggregations

[3] https://ethseer.io/

from each slot. As these contributions or votes create the immutable links between the blocks, the votes have three primary factors, which also determine the 'quality' of their attestation and, thus, the reward of the validator:

- **Source:** This is the hash of the justified checkpoint when the proposed block was proposed. In the context of the CL, checkpoints represent the Beacon State root at the start of an epoch, encompassing the state transition outcomes from the preceding epoch.
- **Target:** This refers to the hash of the epoch's initial block.
- **Beacon Block Root**: This is the block's hash being attested to.

Validators have 32 slots within which to produce these attestations. A second crucial parameter influencing the quality of an attestation is the 'inclusion delay'. This delay denotes the number of slots that pass before an attestation is included in a block after the one attested to. Optimal validator performance is achieved when a vote is cast with all three flags correctly identified and included in the next block, resulting in an inclusion delay of just one slot.

Block proposals

A designated active validator creates and submits a beacon block in each slot. At this juncture, the validator integrates the necessary metadata and incorporates the maximum number of aggregated attestations, subject to a cap of 128 per beacon block. The compensatory reward for the proposer is contingent on both the quantity and quality of these attestations. Specifically, for each attestation flag not previously included, a portion of the generated reward is allocated to the proposer of the block containing it. Consequently, incorporating more novel attestations directly enhances the proposer's reward.

Similarly, the proposer benefits from including sync committee duties, receiving a share of the overall reward they produce. This arrangement motivates the block proposer to include the maximum number of sync committee duties, maximizing potential rewards.

Sync committees

Sync committees represent the consensus of a small group of validators over the "Root" of the head Beacon State. This helps light clients validate blocks without fully downloading and processing the beacon chain. Each sync committee comprises 512 randomly selected validators who sign new block headers every slot and rotate every 256 epochs (8192 slots). However, we won't consider them as performance indicators due to the low frequency at which a validator participates in a sync committee.

2.1 Slot time ranges

We've already delved into the temporal segmentation of Ethereum CL's blockchain. However, as Figure 2 shows, the protocol still defines smaller time subdivisions inside each slot. These numbers are just guidelines. However, following them is vital for an optimal network operation. The figure summarizes the tasks that need to be performed in the following order:

1. First, the elected block proposers must create and broadcast a new block at the beginning of the slot (second zero). This provides 4 entire seconds for the message to propagate over the participants in the network. This means that the block proposer had 4 seconds before the start of the slot to receive and group aggregated attestations from the previous 32 slots.
2. After the first 4 seconds of the slots, validators assigned to attest to it are expected to generate and broadcast their votes with their perception of the chain (attesting to the *source*, *target*, and *head* they see). They share this vote with the corresponding beacon committee aggregators, and the spec assigns the same time range of 4 seconds to broadcast the message.
3. Finally, committee aggregators must collect votes between seconds 4 to 8, producing the aggregated attestations at the 8th second of the slot. In all committees, 16 validators are randomly selected to aggregate and broadcast the attestations. After that 8th second, the network disposes of 4 extra seconds so that the next block proposer has enough time to receive all the aggregations.

Keeping the correct timing between these tasks is crucial to avoid confusion in the network. For example, if a block proposer extends the creation of its block for 10 seconds, the block could be received later than 12 seconds since the start of the slot, risking being voted as a missed block. If a validator waits too long to generate and send the attestation, the aggregators might not include that vote in the same slot, increasing the inclusion delay and reducing the final reward.

3 Distributed Validator Technology: how does it work?

Distributed validator technology, or DVT, is a critical security primitive that allows a single Ethereum validator to be run on a cluster of nodes working together as a distributed validator. The concept of "splitting" the validator into a cluster removes the single point of failure for validators, creating an active-active redundancy with a failure threshold. However, the network is agnostic to this sub-division and perceives the cluster as a single validator. DVT provides some safe margins where if one or several cluster nodes fail to send their partial signatures, the distributed validator can keep performing its duties as long as enough nodes (over the threshold) submit their partial duties.

In this paper, we will closely examine the DVT approach of Obol Labs[4], monitoring and analyzing the performance of a few DVT validator clusters using Charon. Charon[1] is a middleware client that sits between the beacon client and the validator client of each node within a distributed validator cluster. It provides the logic for communicating with the beacon node and the cluster, scheduling and deciding what to sign and when to sign.

Obol's approach to DVT starts from a Distributed Key Generation (DKG) [17, 21, 24, 32] ceremony, where the cluster participants generate their private BLS [6] key shares and then sign the validator deposit. Once the deposit is processed, the cluster

[4] https://obol.tech/

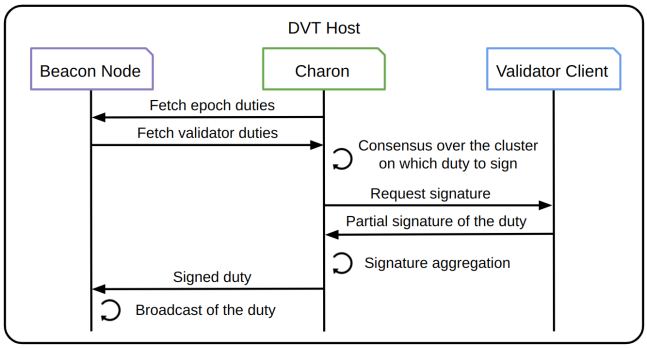

Fig. 3 Internal communication between the beacon node, Charon, and the validator client when performing a consensus duty of a DVT validator.

starts contributing to Ethereum's consensus in a collaborative effort. As each DVT cluster participant runs its Charon client, check Figure 3 as a reference, which, in collaboration with the beacon node, tracks and schedules the duties that the given activated validator or validators need to accomplish. However, the cluster must still sync and decide what to sing before aggregating their partial signatures.

Obol relies on a private network between the users participating in the cluster. The network, besides being used to propagate the partial signatures, creates the space to play a Quorum Byzantine Fault Tolerance (QBFT) [4] consensus game to help the cluster agree on which duty the sign as a single validator. Thus, when a new duty is planned for a validator, the validator client retrieves the duty to be signed and sends it to the Charon client. The Charon client now waits for enough partial signatures from the rest of the nodes. After receiving enough partial signatures to meet the threshold, Charon broadcasts the signed duty to the beacon node, which broadcasts it to the network. The threshold of each cluster (minimum number of partial signatures to perform duties) depends on the number of nodes in the cluster. A cluster can stay active as long as more than 66% of its participants send their partial signatures. For a more in-depth explanation of the DVT, please refer here.

4 Related Work

There has been a huge effort to map the topology of Ethereum's peer-to-peer network. Previous works have shown a range of participating routable nodes in Ethereum's network oscillates between the 11.000 for the PoW network [16] and 9.000 for the younger beacon chain [10]. Although authors report a healthy network behaviour, some previous work like [22] have demonstrated that the node's location directly affects the networking performance and, inevitably, the performance of PoW-based application's nodes.

Message distribution is essential in PoS systems such as Ethereum. Even if each slot gives a time window of 12 seconds to propose a block, commit the attestations, and aggregate them, receiving half a second sooner or later the block can directly impact the attestation of validators, as their attestation could shift from "the block is valid" to "there wasn't a block at all".

Authors in [13] presented how placing Ethereum nodes on a poorer connected region impacted the latency towards the network, placing the client in a tougher situation than nodes closer to the core of the same one. With these measurements, the authors demonstrated that the network had been incentivised to keep concentrating validators in the same geographical locations due to their lower latency towards the network.

Since a latency increase puts the performance of a validator and the economic stimulus attached to it at risk, any extra step that adds delays to the accomplishment of duties could potentially compromise, to some degree, the overall performance of the network. To showcase the importance of following the defined strict timeline inside an Ethereum slot, authors in [28] demonstrated that, in an attempt to maximize the execution payload's reward through the usage of Maximum Extractable Value (MEV) relays bids [35] [23], the extra time delay from the interaction between the validators, the relay, and the bidder, highly expose validators to a missed block or to a chain re-organization.

In such a time-demanding scenario, this paper studies how DVT validators perform compared to the canonical validator software. In the study, we empirically analyze Obol's DVT approach's steadiness, profitability, and internals in the Goerli testnet. We analyze the differences in the quality of the accomplished validator duties from the different available DVT cluster sizes and compare them with the most mature, reliable counterparts.

5 Methodology

Comparing the consensus performance between DVT and canonical validators required multiple hardware and software resources. This section will summarize all the used requirements, providing a detailed description in case the experiment wants to be replicated with further versions as the DVT technology keeps polishing implementation details.

5.1 Software services

Participating on Ethereum's network in a post-merge scenario requires a wide set of software as introduced before with Figure 1. Making the whole software setup a bit more complex, participating in an Obol DVT cluster implies relying on a fourth extra software, Charon. The whole client setup includes the following layers:

Execution Layer nodes

Since "The Merge" happened in September 2022, to follow up with the canonical state of the chain, the beacon node needs to validate the EVM interaction from users to accept or propose a new block. Thus, it needs to be paired one to one with an execution layer client. There are quite a few options to choose from; however, we relied on Nethermind[5] on its versions $v1.17.1$ and $v1.17.3$ for all our cluster deployments to support the client diversity in the network.

[5] https://github.com/NethermindEth/nethermind

Beacon Nodes and Validator clients

On the consensus layer, the ecosystem has five main software [11] to keep track of the chain's head and track the fork choices of the consensus. However, as described in Section 3, not all beacon nodes support the necessary endpoints at the REST API when writing this paper. Obol's DVT client needs some specific endpoints:

- $POST/eth/v1/validator/beacon\_committee\_selections$
- $POST/eth/v1/validator/sync\_committee\_selections$

For this reason, since only two out of the five available clients had a reliable API, we relied on those two available options for the study: Lighthouse [6] on its versions $v3.5.1$ and $v4.0.1$, and Teku [7] on its versions $v23.3.0$ and $v23.3.1$.

To maintain consistency between the validator clients and the beacon nodes and ensure optimal interoperability, we decided to pair them with their respective validator clients on identical versions.

Charon

As previously mentioned, deploying a cluster of DVT validators requires some extra complexity between the beacon chain's client and the duty signer validator client. In this study, we will be taking a closer look at Obol's implementation, measuring their DVT software, Charon, on its versions $v0.14.0$ and $v0.15.0$.

Setup replication

To ease the deployment of all these clients, ensuring the same configuration in all of them, we relied on a *docker-compose* setup [8] which is based on the official configuration provided by the Obol team [9].

This setup includes:

- Nethermind configuration for Teku and Lighthouse.
- Teku configuration (beacon node and validator client).
- Lighthouse configuration (beacon node and validator client).
- Charon as DVT client.
- Prometheus for metrics gathering.

5.2 Performance Indexes

Comparing the steadiness and profitability of a validator in Ethereum's consensus is not such an easy task; it involves many parameters that are hard to quantify and compare across multiple validators. To do so, we used a set of pre-defined and tested indexes that aggregate distinct metrics to help understand the performance of a node or a validator.

---

[6] https://github.com/sigp/lighthouse
[7] https://github.com/ConsenSys/teku
[8] https://github.com/tdahar/charon-distributed-validator-cluster
[9] https://github.com/ObolNetwork/charon-distributed-validator-cluster

Maximum Extractable Reward (MER)

We calculated both validator sets' rewards at a given epoch using the reward models proposed by [12], the Maximum Extractable Reward (MER). The MER indicator assessed the maximum attestation and sync committee rewards a validator could get at each epoch if all the duties were performed correctly and within the optimal time window. The concept of MER allows for calculating the ratio of achieved rewards out of the MER. This allowed us to compare the profitability impact of running a distributed validator versus its canonical option.

Node Score

To compare the distributed validator technology with its counterpart, a standard validator, we must quantify how much overhead each node of the DVT cluster adds to the overall performance of the cluster. Measuring a single client's uptime and steadiness is generally easy, as we could only track the signature aggregation time. However, this metric is not that easy to track on a distributed cluster of validators that need to share messages across multiple nodes. In this case, we adopted Obol's Beacon Node Score to measure the performance of each cluster node. This score represents the individual score of a single node from the rest of the cluster's perspective.

The score represents a synthetic parameter sensible to the error rate of the beacon node's REST API and to the latency between the rest of the nodes participating in the DVT cluster, which is summarized in the following equation:

$$Score = 0.5 * (1 - (10 * \frac{APIerrors}{APIlatency})) + 0.5 * (1 - max(perc99(P2pDVTlatency) \quad (1)$$

5.3 Data Collection Tools

Generating a comparison of this magnitude requires a reliable generation and collection of all the involved metrics. With this purpose, we've selected a small variety of software that helps us identify the performance of the validators and monitor internal DVT operational metrics.

GotEth

To index each of the controlled validator's duties accomplishment and their respective MER over the study time range, we employed the *GotEth* software [26]. This open-source tool indexes essential information from a trusted beacon node, including:

- Validator-specific duties.
- The quality of these duties, such as whether validators missed a block proposal and the number of flags successfully voted.
- The maximum attestation and sync committee reward that each validator could have earned, provided they were part of a sync committee during that epoch.

Provider	CPU	Memory	Storage	I/O Speed
OVH	Intel Xeon E-2274G 8x	32GB	900GB	R: 422MiB/s W: 141MiB/s
Digital Ocean	Intel Xeon Plat. 8358 4x	32GB	600GB	R: 234 MiB/s W: 78.3 MiB/s
Hertzner	AMD Ryzen 9 5950X 32x	128GB	10.5TB	R: 512MiB/s W: 511MiB/s

Table 1 Hardware specifications of different cloud service providers.

Provider	Location	Services (Beacon Node + Validator)	Node Name
OVH	Frankfurt	Lighthouse + Lighthouse	happy-body
OVH	Strasbourg	Teku + Teku	precious-food
OVH	Warsaw	Teku + Teku	expensive-mountain
Digital Ocean	London	Lighthouse + Lighthouse	mindful-movie

Table 2 Cloud provider hosting the four-node DVT cluster.

Prometheus

Most software nowadays produces internal metrics for debugging and monitoring purposes. Following the industry's tendency, the Ethereum ecosystem relies on such techniques to provide real-time information using top-leading tools such as Prometheus[10]. Prometheus is a time-based database service that captures all the metrics each tracked endpoint offers on a specific *HTTP* path. It allows the configuration of multiple targets, scraping their metrics at the desired frequency, which can be accessed later through evaluation and ruling expressions.

5.4 Server hardware configuration

5.5 Node cluster deployment

The experiment aimed to analyse the DVT performance over different cluster sizes, geolocations and two cloud providers to give the experiment more robustness. Furthermore, this wide variety of hardware providers and software combinations helps stress-test the DVT cluster aggregation times for extreme cases. We decided to test three of the most relevant DVT clusters, the ones that add one extra node of threshold to the signature aggregations:

1. A **DVT cluster of four nodes** has a signature aggregation threshold of three nodes. The node deployment is displayed in Table 2.
2. A **DVT cluster of seven nodes** has a signature aggregation threshold of five nodes. The node deployment is displayed in Table 3.
3. A **DVT cluster of ten nodes** has a signature aggregation threshold of seven nodes. The node deployment is displayed in Table 4.

[10] https://github.com/prometheus/prometheus

Provider	Location	Services (Beacon Node + Validator)	Node Name
OVH	Frankfurt	Teku + Teku	unusual-state
OVH	Strasbourg	Teku + Teku	selfish-rule
OVH	Warsaw	Teku + Teku	determined-party
Digital Ocean	Bangalore	Lighthouse + Lighthouse	enthusiastic-area
Digital Ocean	Frankfurt	Lighthouse + Lighthouse	affectionate-day
Digital Ocean	London	Lighthouse + Lighthouse	jolly-life
Digital Ocean	Singapore	Lighthouse + Lighthouse	cloudy-flowers

Table 3 Cloud provider hosting the seven-node DVT cluster.

Provider	Location	Services (Beacon Node + Validator)	Node Name
Digital Ocean	Bangalore	Lighthouse + Lighthouse	dangerous-mobile
Digital Ocean	Toronto	Lighthouse + Lighthouse	clear-fish
Digital Ocean	Frankfurt	Teku + Teku	beautiful-word
Digital Ocean	London	Lighthouse + Lighthouse	amazing-tea
Digital Ocean	Singapore	Lighthouse + Lighthouse	plain-news
Digital Ocean	New York	Lighthouse + Lighthouse	tough-city
OVH	Beauharnois	Teku + Teku	fine-adult
OVH	Frankfurt	Teku + Teku	alert-waterfall
OVH	Strasbourg	Teku + Teku	delightful-dates
OVH	Warsaw	Teku + Teku	powerful-road

Table 4 Cloud provider hosting the ten-node DVT cluster.

For a fair comparison across the different node deployments, we've activated a total of 6.000 validators in Ethereum's Goerli testnet[11]. 3.000 of those validators were split evenly, first in sets of 1.000 validators, one for each of the described DVT clusters, and second, each validator was split in the respective private key-shares for the cluster size.

5.6 Ethereum validators and testing networks

Protocols like Ethereum are complex in many aspects. They are hard to consolidate at the technical and human levels; they have too many components and contributors. From that basis, organising the possible upgrades of the protocols is even more complex, proposing changes that need to be done, reviewed and tested. Like many other projects in the web3 ecosystem, Ethereum adopted a basic methodology of splitting all the ideas into features or Ethereum Improvement Proposals[12] (EIPs). When a group of proposals or EIPs are considered mature enough and align with the public roadmap of the protocol, they are tested on development environments such as testing networks or, as we refer to them, testnets.

Since Ethereum's launch in 2015, it has had many EIPs and testnets. This testing environment represents an isolated environment where a few specific changes, events, interactions, or transitions can be stress-tested. Although the testnet aims to represent

[11] Entire list of validators is available at https://migalabs.io/reports/MigaLabs-Obol-Annexe.pdf
[12] https://eips.ethereum.org/

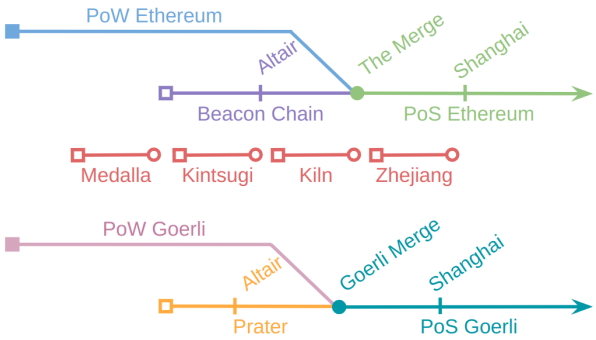

Fig. 4 Ethereum network's hard-forks organization, public testnets in red, Goerli testnet at the bottom. Note that hardforks are tested on testnets before they occur on mainnet.

the reality of a main network as closely as possible, some aspects or events, like a sudden peak in user interaction, can't be entirely replicated in these environments. However, because these testnets justify their presence by testing the limitations of the changes, it doesn't mean that their results are negligible. They are stress-tested to find the weak points of the protocol or the implementation. Figure 4 shows the road map of the main public testnets around the consensus layer's hard-forks timeline, showcasing how each major upgrade on Ethereum was previously tested on a testnet and its prior test in the Goerli/Prater testnet. It is necessary to notice that, since public testnets such as Goerli[13], participation is not monetarily incentivised, participation tends to be lower than on the main networks. This might not sound that critical if it stays above the 66%. However, this can slightly affect the steadiness and profitability results of validators participating in the network. Therefore, we've set a set of control nodes running concurrently with the DVT validator set, representing the fairest comparison with the canonical way of running them.

6 DVT cluster vs canonical validator set up comparison

To evaluate Obol DVT, we performed a long multi-phased experiment that lasted for 10.000 epochs, about a month and a half of continuous execution. The purpose of running for such an extended period was to provide robust statistical analysis, as a shorter time range could subject the measurements to network perturbations, non-finality periods, or bias the profitability of a set of validators based on being "lucky" more often and provide an unusual number of blocks. The experiment was performed on Ethereum's Goerli/Prater network, and it was run between the epochs 163000 (Mar-18-2023 00:40:00 UTC) and 173000 (May-01-2023 11:20:00 UTC). The Ethereum Prater network is a testnet, and as such, it may have dangling validators that may not be running. This means there are more missed blocks than in the Ethereum Mainnet network, resulting in delayed attestation inclusion and more chain reorgs. These conditions are harder than Mainnet, which stresses the software even

[13] https://goerli.net/

further. The experiment ran three clusters described in Tables 2 3 and 4 with 1.000 validators attached to each. Therefore, the experiment involved running a total of 3.000 distributed validators. The following sections present the results of comparing the performance of the DVT validators with the counterpart validators.

6.1 Steadyness of the validator

In such complex protocols, software performance depends on multiple conditions: hardware resources, network connectivity, software optimizations, etc. However, in Ethereum's case, specifically on its PoS version, this performance can be reflected in two main aspects: the accomplishment of validators' duties and the timing and correctness. The following sections will analyse these aspects individually, comparing the results of the different instances deployed worldwide.

The coming up analysis was performed between epochs 163000 (Mar-18-2023 00:40:00 UTC) and 173000 ((May-01-2023 11:20:00 UTC)) on Ethereum's Goerli/Prater network. As significant upgrades were performed on Charon's software, we upgraded the versions of all software (EL client, Beacon node, Charon, Validator client) at the middle of the run in epoch 168511 (April-11-2023 14:30:24 UTC). This system setup upgrade enabled the benchmarking of the performance progression of the software, but more specifically, Charon's DVT implementation.

6.1.1 Missed attestations

At every epoch, validators are assigned to attest to one of the 32 slots, having the obligation of always submitting one attestation per epoch. These attestations, as Section 2 introduced, create the link across blocks, enabling the chain to finalize previous states. To do so, each validator needs to provide three flags: one that links to the last epoch (source), one for the current epoch (target), and a third one that references the last available block (head).

A wide range of things could go wrong with the attestation's flags. Due to the sensitivity of the head slot, it requires having the same perspective as the majority honest part of the validators, and it would only be counted as correct if the attestation with the correct head flag has an inclusion delay of 1 slot (it must be included on the right next proposed block). This means that there could be three major reasons to miss the flag: i) the beacon node had a wrong vision of the chain, i.e., the block on that slot somehow arrived late to the beacon node, causing a fork from the canonical chain, ii) the there was a missed block right after our attesting slot; thus, the attestation couldn't be registered to the beacon chain on time, and iii) the attestation was broadcasted late to the network, and it wasn't included in the next block. The many reasons to miss the head flag make it the most missed flag in an attestation. Especially in testing networks like Goerli/Prater, which generally have lower participation.

On the other hand, the source and the target flags are harder to miss. The wrong perception of the previous and the current epoch could only mean that the beacon node was unavailable or that the head of the chain was lost. For this reason, these two flags generally fail together and represent the steadiness of a validator. However, there is a small difference between them: the source flag has a maximum inclusion delay of

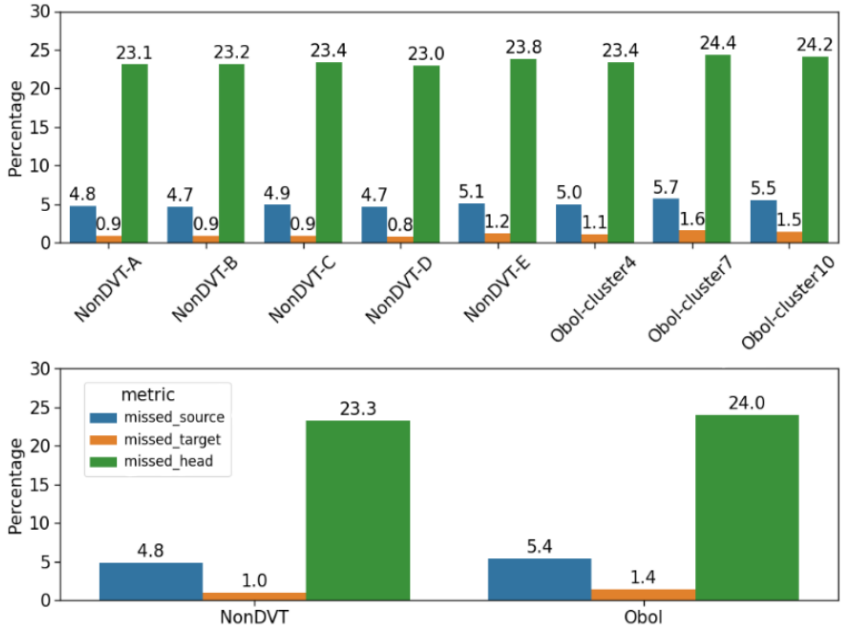

Fig. 5 Percentage of missed attestation flags for the first part of the experiment with Charon on its version $v0.14.0$. The upper figure shows the missed flags across the different validator sets. In the bottom one, the missed flags are aggregated by types of clusters (non-DVT vs DVT/Obol).

five slots, which is generally the reason why it is missed more than the target that has 32.

Figure 5 shows the ratio of missed attestation flags per cluster of validators (non-DVT vs DVT clusters), where we can appreciate that there is a clear miss pattern across the three flags: the head flag gets missed 4.66 more times than the source flag, which then is missed 4.54 times more than the target flag. The bottom part of the figure also shows that, although not that different from each other, the DVT validators failed 0.7% more of head flags, 0.6% of source flags, and 0.4% of target flags. These differences generally come from the largest clusters of seven and ten DVT nodes, which, as expected, could add a larger delay to the first consensus and then the signature aggregation of the duties.

On the other hand, Figure 6 shows the same missed attestations split by flags but for the later Charon version $v0.15.0$. The figure shows a similar pattern of missed flags as with $v0.14.0$. However, we do appreciate a slightly lower difference between the non-DVT validators and the DVT clusters. In this newer version, the differences get reduced to 0.5% of missed head flags, 0.6% missed source flags and 0.4% missed targets, which means no major or very few upgrades were applied to the cluster attestation duties during the upgrade.

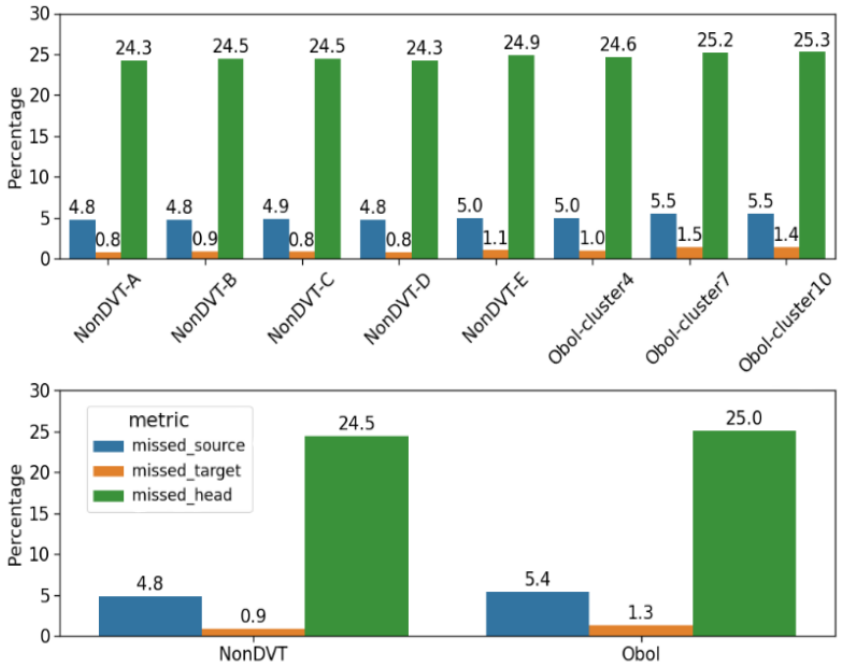

Fig. 6 Percentage of missed attestation flags for the first part of the experiment with Charon on its version $v0.15.0$. The upper figure shows the missed flags across the different validator sets. In the bottom one, the missed flags are aggregated by types of clusters (non-DVT vs DVT/Obol).

6.1.2 Inclusion delays

As previously mentioned, the inclusion delay clearly has a significant presence on the validator's performance, as it can draw the line on whether the flag is correct or not, regardless of the value itself. In that regard, Figure 7 shows the cumulative distribution function (CDF) of the inclusion delay of the attestations produced by the DVT validators (aggregated by version, as there was no sign of significant improvements over the version upgrade). The figure showcases that over 72% of the attestations had an inclusion delay of one slot, which is the ideal scenario. However, the figure also shows that there is a 20% of attestations that stay with an inclusion delay between two and five slots, and the resting 8% beyond the five slots mark. We can observe that the three clusters show the same behaviour and percentiles. Although the matching of the CDFs across DVT clusters is not appreciated in the figure, the three CDFs are displayed in the figure, showing an overall similar performance.

6.1.3 Block proposals

At every slot of the chain, a single random validator is chosen to propose a new block, which includes all the "non-tracked-yet" attestations and sync committees broadcasted over the previous slots. As shown in Figure 2, the block proposal is the first event that should occur inside a new slot. The protocol leaves a security window of 4 seconds to propagate the block over the network; thus, small variations on when the block

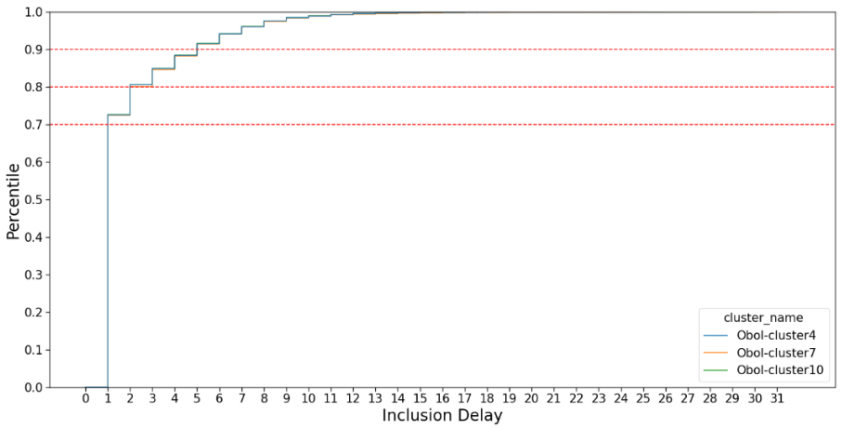

Fig. 7 Cumulative distribution function of the attestation inclusion delay per DVT cluster, with both versions aggregated.

was first broadcasted might compromise the actual block proposal. This means that a block is only accepted when the 1/3th of the network receives the block proposal within the expected 4 seconds. In the case of the DVT validator clusters, there is a risk where the signature aggregation could extend the initial block propagation to the rest of the validators, exposing the cluster to a missing block. It is, therefore, very important to ensure that DVT does not add any critical delay when submitting a new block if they are chosen as block proposers.

In Figure 8, we can observe that all validator sets had a successful block proposal rate above 92%. In fact, six of the validator sets stayed at the 98.5% or over it, except for the non-DVT client D and the Obol cluster7 that stayed at 96.5% and 92.6%, respectively. On the right part of the figure, the aggregation of the successful block proposals per type of validators show that the non-distributed validators still managed to get an extra 2% of proposed blocks. It is worth mentioning that the missing blocks are concentrated in the biggest clusters, cluster7 and cluster10, while the validators in cluster4 did not miss any block proposal. In particular, cluster7 struggled because two out of the seven nodes of the cluster had poorer connections to the rest of the cluster participants. However, this is something that we will further discuss in Section 7.1. Either way, in both sets of validators, non-DVT and the DVT, non-DVT client D and cluster7 behaved like outliers.

In contrast to the missed attestation performance, the upgrade of Charon to its version $v0.15.0$ did impact the number of successful block proposals. Figure 9 shows the same data as in Figure 8 but after upgrading the Charon client. Our measurements perceive an improvement of 2.7% in the number of proposed/missed blocks in distributed validators. While the non-DVT client D got stuck at 96.4% of block proposals, cluster7 achieved a solid 99.1% of the proposals, showing a big improvement in what we think is related to the internal cluster connectivity.

Due to the randomness of the block proposer election, a two-month study might not provide as many data points as we would desire, making the dataset susceptible to small variations or outlier scenarios. Table 5 shows the entirety of the missed blocks

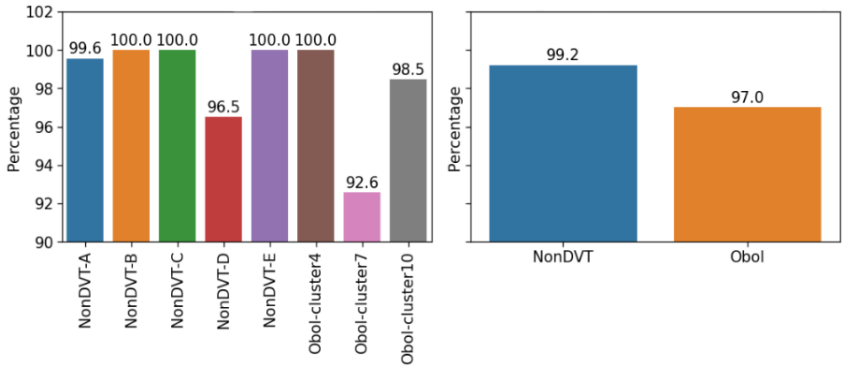

Fig. 8 Percentage of successful block proposals of Charon on its version $v0.14.0$. On the left, the successful block proposals are aggregated by cluster. On the right, they are aggregated by validating technology.

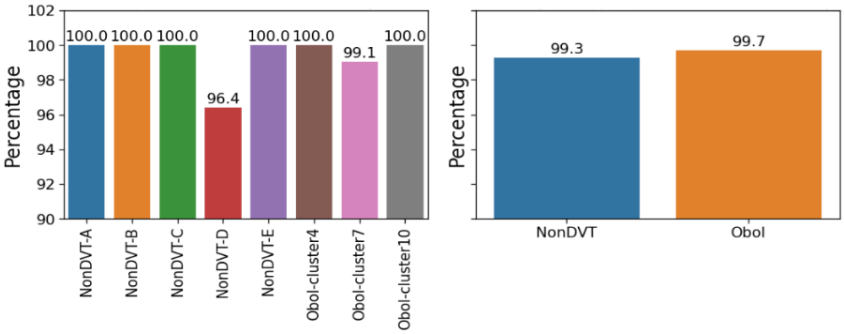

Fig. 9 Percentage of successful block proposals of Charon on its version $v0.15.0$. On the left, the successful block proposals are aggregated by cluster. On the right, they are aggregated by validating technology.

by cluster of vlaidators. In the table, as well as in the right side of Figure 9, we could observe that the DVT validators got a higher ratio than its canonical alternative non-distributed validators over these last 4.500 epochs of the experiment. Despite the difference, which might sound exciting, it is clear that the difference comes from one particular that performed poorly during that period, bringing the mean performance down for NonDVT in general. However, this showcases that block proposals indeed do not get affected by the extra latency on version $v0.15.0$.

6.2 MER ration Achievement

The performance of a validator in Ethereum's PoS is heavily dependent on the correctness of its duties [12]. In fact, the 61% of the rewards a validator produces are directly associated with the attestation rewards, while the resting 29% are related to more random variables such as block proposals, sync committees, and fees from the included transactions at the execution level.

		v0.14.0 (5500 epochs)		v0.15.0 (4500 epochs)	
Validator Set	**Validators**	**Proposed**	**Missed**	**Proposed**	**Missed**
NonDVT-A	600	225	1	176	0
NonDVT-B	600	235	0	208	0
NonDVT-c	600	240	0	191	0
NonDVT-D	600	222	8	188	7
NonDVT-E	600	216	0	203	0
obol-cluster4	1000	363	0	341	0
obol-cluster7	1000	361	29	313	3
obol-cluster10	1000	382	6	294	0

Table 5 Table summarizing all the exact block proposal duties per validator cluster, including the number of missed blocks over the whole study.

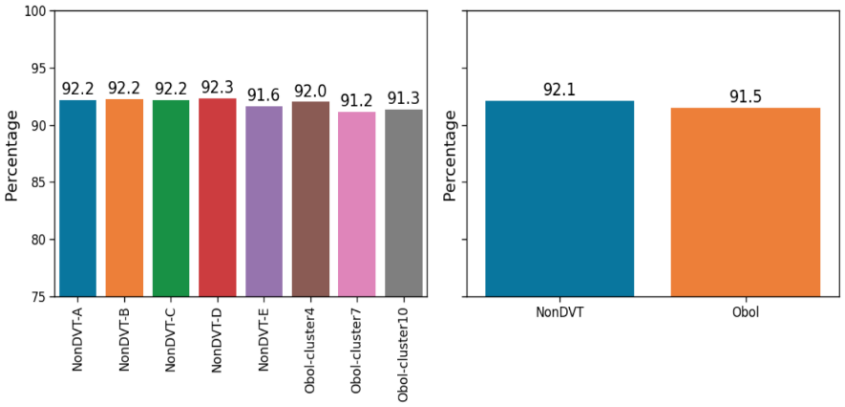

Fig. 10 Ratio of achieved rewards with the MER of the validators aggregated by cluster on the left and by validator technology on the right on Charon's $v0.14.0$ version.

To quantify the profitability impact of the previously presented points, we measured the reward of each validator and compared them with their respective MER 5.2. Figures 10 and 11 show that for all the validator clusters on the Charon's versions $v0.14.0$ and $v0.15.0$, respectively. The first one already shows that the profitability doesn't differ much from non-DVT validators to DVT ones. On Charon's $v0.14.0$ version, the difference only reached 0.6% in favour of non-DVT validators. On the other hand, the differences got narrower with the version upgrade, reducing the differences to a negligible 0.4% considering the extra resilience that DVT provides.

7 DVT isolated performance

The previous section 6 compared the performance of DVT validators versus its natural canonical competition. However, many relevant questions remain for the Ethereum staking community, i.e., in a staking ecosystem where over the 66% of the validators belong to staking entities or staking pools[14], how many validators could these large entities run on a single cluster?

[14] https://monitoreth.io/nodes#active-validators-entities

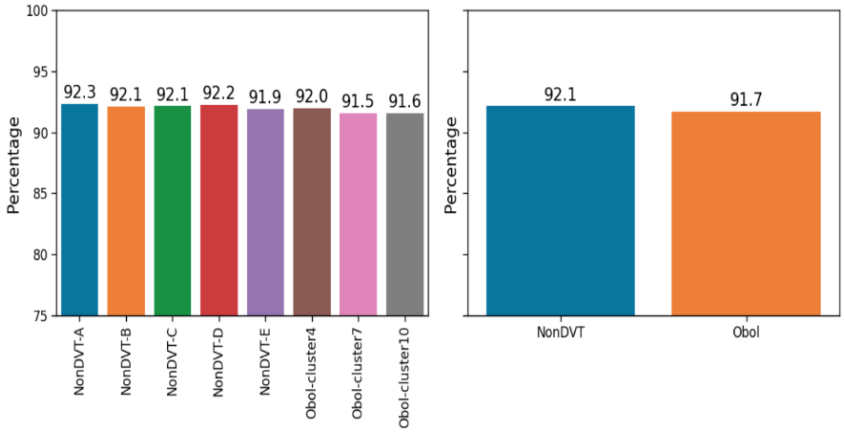

Fig. 11 Ratio of achieved rewards with the MER of the validators aggregated by cluster on the left and by validator technology on the right on Charon's $v0.15.0$ version.

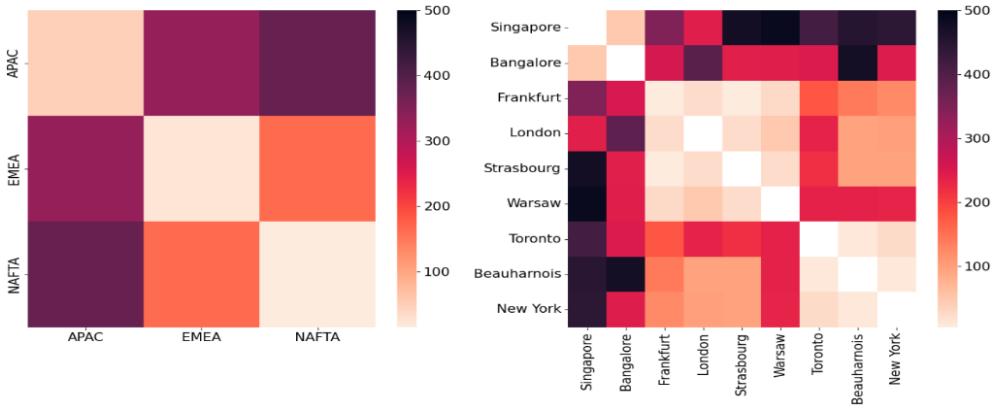

Fig. 12 Latency heatmap across the different cluster nodes using Charon on its version 0.14.0. On the left, aggregated by regions: APAC (Asia–Pacific), EMEA (Europe, the Middle East and Africa) and NAFTA (North American Free Trade Agreement). On the right, aggregated by city.

To answer this and another similar question about the limits of DVT clusters, we set up a second study to spot any existing limitations.

7.1 Cluster internal latency clustarization

We have previously introduced the extra delay that each DVT cluster adds to each duty's broadcasting. As expected, the performance of the DVT cluster heavily relies on the connectivity of the participating nodes. The lower the latency across nodes, the lower the delay, and the higher the number of participants, the higher the delay, as more signature aggregations need to be done.

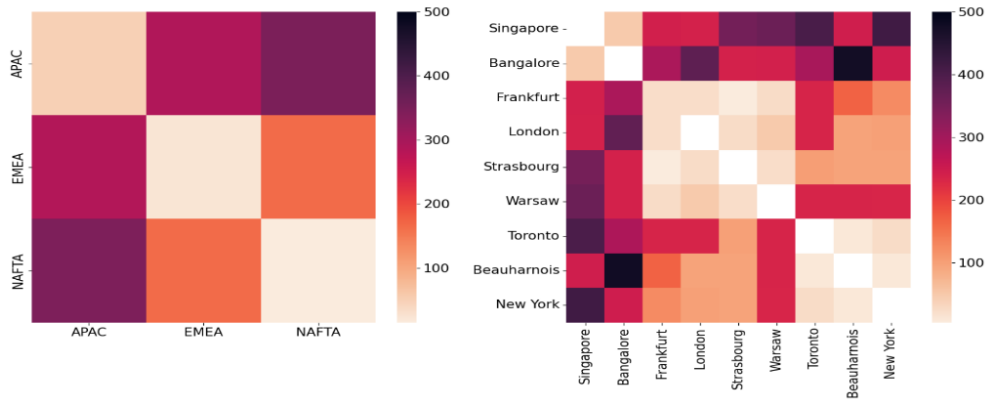

Fig. 13 Latency heatmap across the different cluster nodes using Charon on its version 0.15.0. On the left, aggregated by regions: APAC (Asia–Pacific), EMEA (Europe, the Middle East and Africa) and NAFTA (North American Free Trade Agreement). On the right, aggregated by city.

To measure the internal connectivity between the spawned DVT nodes, Figure 12 shows a heat map of the latency that each peer got with the rest aggregated by location and region. In this case, the figure shows the connectivity between Charon nodes using $v0.14.0$, where we can highly appreciate the clustering of low latency that peers get with others that are in the same region.

After upgrading to version $v0.15.0$, Figure 13 showcases that the latency in the worst-case scenarios (very far distance between peers) has improved substantially. For example, the latency between Strasbourg and Singapore has decreased from 470ms in $v0.14.0$ to 300ms in $v0.15.0$, representing a 37% improvement. Similarly, the latency between Beauharnois and Singapore has decreased from 448 in $v0.14.0$ to 245ms in $v0.15.0$, which represents a 46% improvement.

These figures explain why the DVT cluster7 got more missed attestations in section 6.1.1; adding two nodes on a different region to the rest of the cluster significantly improves the connectivity.

7.2 Nodes' performance

In addition to the previous setup, we performed a separate experiment to measure the validator hosting capabilities of Charon. This experiment consisted of scaling the number of validators that are run on a cluster, with the main goal of analyzing how the Charon client handles such a high load of duties to perform. This study combined the 3.000 validators into a single cluster (cluster4 setup). However, we performed it in three different rounds:

- four-node cluster with 1000 keys for 1000 epochs
- four-node cluster with 2000 keys for 1000 epochs.
- four-node cluster with 3000 keys for 1000 epochs.

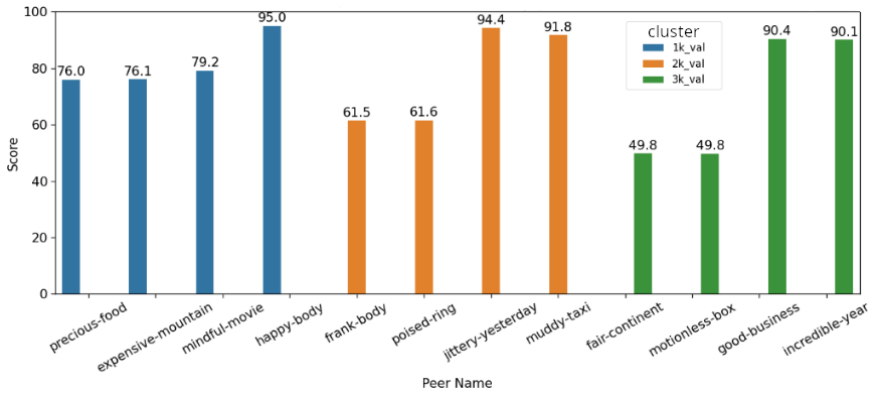

Fig. 14 Beacon score of each Charon client participating in the four-node cluster per step in the experiment.

The same machines were reused for all steps. However, we had to clean them up when changing the number of validators because adding key shares to an existing cluster is impossible. Instead, it is necessary to run the DKG process to generate all the key shares that will be used in the cluster.

7.3 Limit of DVT validators per cluster

A larger number of validators per cluster means more overhead to the hosting machine, as more duties must be performed. Let's remember that for every validator duty, the beacon node has to create the corresponding attention, which then needs to be signed by the validator client, which goes back to the beacon node to broadcast it. DVT adds extra steps for each of the duties as cluster participants first need to reach consensus through QBFT and then aggregate the signatures before sending the complete duty.

DVT cluster's beacon score

As the first performance indicator of the overhead of increasing the number of validators per cluster, Figure 14 presents the beacon node score of the peers that form the four-node cluster during the three phases of the experiment (each with a different number of validators). The figure shows that the more validators hosted in the cluster, the lower the beacon node score gets. On average, Obol's DVT cluster obtained a beacon node score of 81.5% with 1.000 validators, 77.3% with 2.000 validators (around 4% worse), and 69.9% with 3.000 validators (more than 11% worse compared to 1.000 validators).

Obtained rewards

While the beacon node score gives us some indication about the performance of the Obol DVT cluster, there is no better benchmarking than measuring the amount of rewards obtained during the three phases. As the number of validators increased with every step of the experiment, Table 6 summarized the achieved rewards by the whole cluster, normalized by the number of validators performing duties during that period.

Vals.	Total (ETH)	Per Validator
1000	9.424435	0.009424
2000	19.403201	0.009702
3000	30.437587	0.010146

Table 6 Rewards during scaling experiment with and without filtering block proposals and sync committee rewards.

	Total			Per Validator		
Vals.	**Missed Source**	**Missed Target**	**Missed Head**	**Missed Source**	**Missed Target**	**Missed Head**
1000	52320	12904	236131	52.320	12.904	236.131
2000	80770	18546	446948	40.385	9.273	223.474
3000	103889	22413	602982	34.630	7.471	200.994

Table 7 Number of missed attestation per different validator set.

The table shows an increase in the normalized achieved reward per validator as the number of validators increases.

Despite the unexpectedly decreasing beacon node score, rewards increase as the number of validators increases. On the one hand, this shows that Charon's internal beacon node score has a wide window for improvement. On the other hand, by analyzing the attestation flags (see Table 7), we noticed a significantly lower number of source and target attestation flags missed when the number of validators increased.

This phenomenon is still not fully understood, and more experiments are necessary to reproduce and investigate these results. However, one fairly assumable cause of this unexpected behaviour is the impact of Goerli-related instabilities rather than a better cluster performance as the number of validators increased or a miss-performance of the client right after the new validator partial keys were added to the cluster.

8 Conclusion

This paper presents an exhaustive empirical study of the Distributed Validator Technology. The provided analysis has shown that there aren't significant differences in validators' steadiness and profitability while comparing a DVT validator cluster to a canonical validator setup.

The presented measurements have shown that the difference in missed attestations between running a DVT cluster and using a standard setup is lower than the 1% in favour of the standard validator software. However, in any case, the DVT validators achieved a steady ratio over the MER of over 91%, which results in a mean difference of 0.4% lower than its counterpart. As the experiment was performed on a testnet, which generally represents a harder network environment for validators, the presented results represent one of the hardest witness-able scenarios of a network like the mainnet beacon chain (real economic incentives make the network more steady).

We would like to remark that, even in a scenario where the profitability was slightly lower, the extra reliability that DVT validator clusters could bring to staking entities or node operators could be negligible if we compare it with existing slashing risks that

canonical staking can have. The technology has shown enough maturity to support up to 3.000 validators per cluster, which enables the possibility of running DVT clusters at the node-operator level. Furthermore, we must introduce an extra advantage point for the DVT, which helps lower the entry barrier for staking in Ethereum, as users could participate without having 32ETH.

However, it is worth remarking that, based on this study's profitability and latency results, we highly recommend keeping the DVT cluster in the same region. This would significantly improve the cluster's performance, achieving the full resilience that DVT aims to offer. The paper has also provided a clear overview of the evolution of DVT. One single version upgrade has meant a significant performance improvement in the block proposals and internal cluster connectivity. This means that technology still has a lot of room for improvement.

To summarize, this paper showcases the significant contribution of DVT to Ethereum's staking ecosystem, providing resilience to the validator deployment setups and achieving a remarkable performance compared to existing, more mature canonical validator software.

Acknowledgments. This work has been supported with a grant from Obol Labs, by the Spanish TCO-RISEBLOCK (PID2019-110224RB-I00) project, and the CHAISE: Blockchain skills for Europe, Erasmus+ projects. We want to thank the Obol developers and DevOps teams for their support in troubleshooting different encountered issues and their feedback on this work. Special thanks to Oisín Kyne and Luke Hackett for their direct implication in helping this study come out and for their constructive feedback.

Declarations

Data Records. The data used to generate the study was obtained from the Goerli/Prater[15] network's public blockchain. We indexed the data of the validators listed at [3] using the open-sourced tool *GotEth*[26]. The exact range of indexed epochs is from 163000 (Mar-18-2023 00:40:00 UTC) to 173000 (May-01-2023 11:20:00 UTC). This data can be extract from the blockchain using the "historical" indexing mode of the tool at anytime, as long as there is a synced node in the network.

References

[1] Charon. https://github.com/ObolNetwork/charon.

[2] Ethereum's rng randao. https://github.com/randao/randao.

[3] List of activated validators. https://migalabs.io/reports/MigaLabs-Obol-Annexe.pdf.

[15] https://goerli.net/

[4] Michael Abd-El-Malek, Gregory R Ganger, Garth R Goodson, Michael K Reiter, and Jay J Wylie. Fault-scalable byzantine fault-tolerant services. *ACM SIGOPS Operating Systems Review*, 39(5):59–74, 2005.

[5] Rameez Asif and Syed Raheel Hassan. Shaping the future of ethereum: Exploring energy consumption in proof-of-work and proof-of-stake consensus. *Frontiers in Blockchain*, 6, 2023.

[6] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrom, Ben Riva, Arnab Roy, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized bls multi-signature with key aggregation. *Cryptology ePrint Archive*, 2023.

[7] Dan Boneh, Manu Drijvers, and Gregory Neven. Bls multi-signatures with public-key aggregation. *URL: https://crypto. stanford. edu/˜ dabo/pubs/papers/BLSmultisig. html*, 2018.

[8] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.

[9] Franck Cassez, Joanne Fuller, and Aditya Asgaonkar. Formal verification of the ethereum 2.0 beacon chain. In *Tools and Algorithms for the Construction and Analysis of Systems: 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2–7, 2022, Proceedings, Part I*, pages 167–182. Springerg, 2022.

[10] Mikel Cortes-Goicoechea and Leonardo Bautista-Gomez. Discovering the ethereum2 p2p network. In *2021 Third International Conference on Blockchain Computing and Applications (BCCA)*, pages 81–88. IEEE, 2021.

[11] Mikel Cortes-Goicoechea, Luca Franceschini, and Leonardo Bautista-Gomez. Resource analysis of ethereum 2.0 clients. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–8. IEEE, 2021.

[12] Mikel Cortes-Goicoechea, Tarun Mohandas-Daryanani, Jose Luis Muñoz-Tapia, and Leonardo Bautista-Gomez. Autopsy of ethereum's post-merge reward system. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2023. doi: 10.1109/ICBC56567.2023.10174942.

[13] Mikel Cortes-Goicoechea, Tarun Mohandas-Daryanani, Jose Luis Muñoz-Tapia, and Leonardo Bautista-Gomez. Unveiling ethereum's hidden centralization incentives: Does connectivity impact performance? In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, pages 89–96, 2023. doi: 10.1109/BCCA58897.2023.10338892.

[14] Corinne Curt and Jean-Marc Tacnet. Resilience of critical infrastructures: Review and analysis of current approaches. *Risk Analysis*, 38(11):2441–2458, 2018.

[15] Jack Dongarra, Thomas Herault, and Yves Robert. *Fault tolerance techniques for high-performance computing*. Springer, 2015.

[16] Yue Gao, Jinqiao Shi, Xuebin Wang, Qingfeng Tan, Can Zhao, and Zelin Yin. Topology measurement and analysis on ethereum p2p network. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2019.

[17] Jens Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.

[18] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–176. Springer, 2021.

[19] Zhiguo He, Jiasun Li, and Zhengxun Wu. Don't trust, verify: The case of slashing from a popular ethereum explorer. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1078–1084, 2023.

[20] Yoichi Hirai. Defining the ethereum virtual machine for interactive theorem provers. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 520–535. Springer, 2017.

[21] Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. *Cryptology ePrint Archive*, 2012.

[22] Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. Under the hood of the ethereum gossip protocol. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 437–456. Springer, 2021.

[23] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Towards a theory of maximal extractable value i: Constant function market makers. *arXiv preprint arXiv:2207.11835*, 2022.

[24] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1837–1854, 2018.

[25] Azad M Madni and Scott Jackson. Towards a conceptual framework for resilience engineering. *IEEE Systems Journal*, 3(2):181–191, 2009.

[26] MigaLabs. Goteth. https://github.com/migalabs/goteth, 2022.

[27] Olivier Moindrot and Charles Bournhonesque. Proof of stake made simple with casper. *ICME, Stanford University*, 2017.

[28] Burak Öz, Benjamin Kraner, Nicolò Vallarano, Bingle Stegmann Kruger, Florian Matthes, and Claudio Juan Tessone. Time moves faster when there is nothing you anticipate: The role of time in mev rewards. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, pages 1–8, 2023.

[29] Giulia Pititto. *The Gasper Protocol: a Proof of Stake Era for Ethereum*. PhD thesis, Politecnico di Torino, 2022.

[30] Laura L Pullum. *Software fault tolerance techniques and implementation*. Artech House, 2001.

[31] David Rehak, Pavel Senovsky, and Simona Slivkova. Resilience of critical infrastructure elements and its main factors. *Systems*, 6(2):21, 2018.

[32] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. Distributed key generation with ethereum smart contracts. In *CIW'19: Cryptocurrency Implementers' Workshop*, 2019.

[33] R Fox Vernon. A brief history of resilience: From early beginnings to current constructions. In *Community planning to foster resilience in children*, pages 13–26. Springer, 2004.

[34] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2. 0 networks. *arXiv preprint arXiv:2007.02754*, 2020.

[35] Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. A flash (bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 458–471, 2022.

[36] Maximilian Wohrer and Uwe Zdun. Smart contracts: security patterns in the ethereum ecosystem and solidity. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 2–8. IEEE, 2018.