CliqueSensus: Ephemeral Overlays for Efficient Attestation Dissemination in Ethereum 2.0

Alexandros Antonov
aantonov@aueb.gr
Department of Informatics
Athens University of Economics and Business
Athens, Greece

Evangelos Kolyvas
ekolyvas@aueb.gr
Department of Informatics
Athens University of Economics and Business
Athens, Greece

Spyros Voulgaris
voulgaris@aueb.gr
Department of Informatics
Athens University of Economics and Business
Athens, Greece

Abstract

Reaching consensus in Proof-of-Stake (PoS) based consensus protocols, requires supermajority agreement among participating validator nodes. Such protocols need significant network resources due to the concurrent voting of a large number of consensus nodes. As a solution, these nodes are divided into committees, with each committee voting individually at a dedicated time slot. In this paper, we introduce CliqueSensus, a protocol that, given a distribution of consensus nodes into committees, lets them self-organize into small, ephemeral clusters structured in clique topologies, to accelerate the voting process, while using only a small fraction of the network resources required by conventional message dissemination methods. Our evaluation demonstrates that our protocol exhibits rapid convergence and operates with minimal network overhead. We focus on the PoS consensus algorithm adopted by Ethereum 2.0. In addition to our protocol, we also analyze and simulate the clustering approach that Ethereum has adopted, showcasing that our protocol can reduce validation message dissemination time by 23% to 70%, while requiring about 190 times fewer validation message forwards.

Keywords: Blockchains, Proof-of-Stake, Peer-to-Peer, Gossiping, Ethereum

1 Introduction

Distributed voting has found usage in a range of decentralized applications, notably in replicated log management and database replication [15, 27, 29], cloud computing coordination [7], and Decentralized Autonomous Organization (DAO) governance [17, 22]. This has been emphatically stressed in recent years through the widespread adoption of sophisticated consensus protocols in the domain of blockchains, particularly those behind Proof-of-Stake (PoS) mechanisms.

Blockchains that adopt such protocols typically utilize wealth distribution to engage a moderated number of nodes in a consensus voting process, to establish properties such as liveness and finality [11, 12, 19, 24], These properties are crucial for the seamless operation and usability of these blockchains.

In pure Peer-to-Peer (P2P) terms, distributed voting entails each consensus participant disseminating a single message (its vote) to all other participants. Unfortunately, this problem is far from trivial, as the number of nodes participating in consensus can grow to very high figures. At the time of writing, 800K consensus nodes are being reported[1] for the Ethereum blockchain. This causes major scalability issues. Involving this number of nodes into concurrent dissemination through a conventional method, such as flooding, would certainly result in the depletion of network resources.

In order to address this, the entire set of consensus nodes is usually divided into committees. Each committee votes at a specific point in time, influencing the determination of the blockchain state as it has progressed until that particular moment. Such committees are formed in an ephemeral fashion. Once the voting procedure has concluded, committees have already fulfilled their objectives and are subsequently disbanded.

Our work concentrates on designing and developing a fully decentralized, self-organizing, dynamic network overlay that facilitates the efficient dissemination of messages within ephemeral, short-lived groups of nodes. We specifically focus on the Ethereum blockchain [33], presenting the problem within its context and adopting its terminology.

The contributions of this paper are twofold:

1. We propose CliqueSensus, a protocol that clusters nodes into small, ephemeral overlays, with each overlay corresponding to a distinct committee. The constructed overlays are structured as cliques, offering desirable properties in terms of dissemination speed and reduced message overhead.

Middleware '25, December 15–19, 2025, Nashville, TN, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

[1] History of daily active validators: https://beaconcha.in/charts/validators

2. We analyze and simulate the existing approach adopted by Ethereum, replicating the message dissemination procedure and its current configuration for network sizes up to 2^{20} nodes. These simulations allow for a comprehensive comparison with our proposed protocol.

Blockchains with validator sets as large as Ethereum's are uncommon. This may be due to factors such as stake delegation or higher staking requirements, as seen in Solana with 1,400 active validators[2], Avalanche with 1,571[3], and Harmony with 292[4] at the time of writing. Adopting a similar strategy to Ethereum's large-scale validator participation could potentially enhance decentralization, reduce the impact of individual node failures, and limit collusion risks. However, this paper does not evaluate such design choices. Instead, we focus solely on Ethereum's protocol due to its extensive validator network and the need for parallel voting at scale.

The rest of this paper is structured as follows. Section 2 lays down the necessary background, outlining the Ethereum 2.0 components relevant to our research, explaining its dissemination approach, and pointing out the networking concerns stemming from it. Section 3 presents our system model. In Section 4 we elaborate on our protocol's design, describing its operations in detail and providing the rationale behind our design decisions. An evaluation of our protocol is provided in Section 5. Section 6 surveys the related work, and Section 7 concludes our work.

2 Background

Our work focuses on the PoS consensus protocol of Ethereum 2.0. Ethereum initiated its transition to PoS with the launch of the Beacon Chain component on December 1st, 2020, running alongside its original Proof-of-Work mechanism. The definite switch to the new consensus mechanism took place on September 15th, 2022, on an upgrade event coined as *The Merge*. In the context of this work, when we mention Ethereum, we are specifically referring to the post-merge Ethereum 2.0.

2.1 Ethereum Consensus

Consensus in Ethereum is collaboratively carried out by a large number of decentralized nodes, known as *validators*. Anyone can start a validator by staking 32 ETH. The initial stake value opts to achieve Sybil resistance, while preserving decentralization, considering consensus speed and message overhead.

In Ethereum, time is divided into *epochs*, with each epoch further divided into *slots*. Each slot has a duration of 12 seconds, and each epoch consists of 32 slots (6.4 minutes). All validators contribute to consensus, being assigned specific roles in each epoch. By the end of an epoch, validators are reshuffled and their roles are pseudorandomly assigned anew.

More specifically, the following roles are being assigned. First, precisely one validator is appointed ***block proposer*** per slot, responsible for proposing a block in the respective slot. The validity of this block is subsequently *attested* (i.e., voted for) by other validators. Second, and in order to produce the aforementioned attestations, *all* validators are designated to serve as ***attesters*** exactly once per epoch. In fact they are equally distributed (or ±1, to accommodate uneven division) across all 32 slots of the epoch. All attesters assigned to a given slot are further split up into as many equal-sized ***committees*** as possible, of no less than 128 members each, with a maximum of 64 committees per slot. Finally, a number of validators are additionally selected as ***aggregators*** for an epoch, which entails aggregating duties alongside the voting process. Aggregator selection follows a probabilistic approach, aiming for an average of 16 aggregators per committee, per slot.

Given that there should be at least one committee per slot, and each committee should have at least 128 validators, a bare minimum of $128 \times 32 = 4096$ validators are required for Ethereum to operate. Currently, the count of active validators stands at 800K, overwhelmingly exceeding that bare minimum. That is, there are 25K active validators associated with each slot, corresponding to 64 committees per slot, with each committee comprising 390 validators. It should be pointed out that the number of committees per slot is already maxed out at 64, which we will consider fixed for the rest of this paper.

2.2 Shuffling Algorithm

Splitting up validators into equal-sized subsets to be appointed attesters in the respective slots, is achieved through an intricate shuffling algorithm, known as *swap-or-not shuffling* [20]. This algorithm constitutes a crucial element for the operation of Ethereum, as well as for our work.

Given a list of elements in some initial order, this algorithm shuffles the list into a pseudorandom, yet deterministic, permutation. Given that the list of validators is publicly known through the staking process (i.e., their public keys and staked amounts are listed on-chain), every validator executes this algorithm locally to figure out its assigned committee for the current epoch.

The interesting property of this algorithm is that it can also be run selectively for a single element, and in fact in both directions. That is, one may selectively compute the new index of a given validator, as well as selectively identify which validator will acquire a given index after shuffling, without having to compute the complete permutation for all 800K validators. Effectively, it allows each validator to *locally*

[2]Solana Active Validators: https://solanabeach.io/validators
[3]Avalanche Validators: https://subnets.avax.network/validators/dashboard
[4]Harmony Validators: https://staking.harmony.one/validators/mainnet

determine its own committee, as well as the committee of any given node as needed for each epoch.

2.3 Block Generation

Block generation is carried out in three phases, all taking place in a single slot:

Proposing: The block proposer generates a new block and proposes it to the entire network by disseminating it to *all* validators.

Attesting: Attesters of this slot convey their attestations to the aggregators of their respective committees. As the identity of aggregators is not publicly disclosed, attestations have to be disseminated to a much broader group of validators, effectively also reaching the target aggregators.

Aggregating: Aggregators of this slot disseminate a message with aggregated attestations to all validators. Again, the goal is to reach the next slot's block proposer, however as its identity is not disclosed, aggregations should reach the entire network of validators.

Of these, the disseminations during the first and last phases cannot be circumvented, as it is a consensus requirement that the entire population of validators be reached.

The disseminations, however, taking place during the second phase, can be substantially improved compared to the currently adopted approach, lowering network overhead by orders of magnitude and speeding up the dissemination of attestations, yet offering high reliability guarantees for message delivery. This is the scope of our work.

2.4 Dissemination of Attestations: Current Approach

Ethereum employs the GossipSub protocol [32] for disseminating messages. GossipSub is a distributed Publish/Subscribe (pub/sub) protocol facilitating the establishment of interest-specific channels (*topics*), ensuring message exchange gets conducted exclusively amongst nodes interested in those particular topics.

The operation of GossipSub can be synopsized in the following three main features:

1. Nodes maintain distinct bidirectional links (approximately eight per node) for each pub/sub topic, forming meshes that are utilized in rapid push-based dissemination.
2. Periodically, each node performs gossip with other participants in its subscribed topics to share missed messages.
3. Each node maintains individual scores for connected peers, evaluating their positive and negative behavior, and retains links only to well-behaved nodes.

Ethereum establishes 64 topics for the dissemination of attestations, forming the communications backbone of the corresponding 64 committees of each slot. Each validator randomly selects and joins two topics, establishing eight bidirectional links with arbitrary other validators in these topics. In order to keep topic membership fresh and balanced, validators leave their topics and randomly join new ones every 256 epochs.

A validator is informed regarding its upcoming role as an attester—and possibly also aggregator—at least one epoch in advance. This period allows it to prepare for vote dissemination. Attesters assigned to committee $C \in [0, 63]$ of *any slot* in the following epoch, simply need to ensure they have a link to at least one validator of topic C, which they will use as a gateway to publish their attestations to that topic. Aggregators of committee C of any slot of the following epoch need to also *join* that topic (in addition to the two they are randomly subscribed to), in order to receive the attestations that will be published in that topic.

2.5 Scalability Concerns

The current design employed for attestation dissemination in Ethereum raises some important scalability and incentivization concerns, discussed in this section.

First, the topics employed for vote dissemination lack optimization concerning node participation and overlay structure. Let the total number of validators be N. There is a fixed number of 64 topics, and each node participates in two of them. Therefore, there are $\frac{2N}{64}$ nodes per topic on average (that is, $25K$ for $N = 800K$ at the time of writing), yet it serves to transmit messages to an average of only 16 aggregators per committee. This leads to unnecessary network overhead as well as increased dissemination time.

A more efficient design would cluster validators by their actual committee memberships. Since there are 2048 distinct committees per epoch, each with $\frac{N}{2048}$ members (about 390 nodes for $N = 800K$), attestation dissemination could be confined to committee members, reducing network overhead considerably, while preserving anonymity.

Ethereum's current GossipSub implementation requires $\frac{7N^2}{32}$ message forwards per epoch, or about 140 billion forwards for $N = 800K$. Using finer-grained clustering of validators into 2048 distinct overlays, will reduce this to $\frac{7N^2}{2048}$, or 2.1 billion forwards–a 64-fold improvement.

In fact our proposed protocol achieves even better results. By organizing committees into cliques, each attestation dissemination requires only $M - 1$ messages for a committee of M nodes. This totals for $\frac{N^2}{2048}$ forwards per epoch, a 448-fold reduction, totaling 311 million messages for $N = 800K$.

A second concern is that the overlay includes validators that are mostly not part of the committee performing the ongoing dissemination. Consequently, an external validator may lack the incentives to promptly disseminate a vote from

a validator belonging to another committee. Our protocol resolves this by restricting communication to only the relevant committee members.

3 System Model

We consider a set of N nodes connected over a routed infrastructure, enabling the communication between any pair of them based solely on the receiver's network address (i.e., its IP address and port).

Each node has a public/private key pair. A node's public key also serves as its unique ID. Public keys are registered in a public trusted repository, such as a public blockchain. Thus, the identities of the entire set of nodes are globally known. Yet their IP addresses are not known, let alone they may change at any time.

The network address of a node is accessible through *link records* generated by the node itself. A link record, or simply *link*, is a data structure including the node's public key, its network address, and a sequence number to distinguish fresh links from obsolete ones. To prevent masquerade attacks, node links are signed by their owner at creation time.

We assume validators have synchronized clocks, a requirement already enforced by Ethereum clients [4], through the use of the NTP protocol. NTP is currently widely adopted across PoS blockchains, though efforts toward more decentralized time synchronization have been explored [9].

We assume K equal-sized committees, with each node being associated with exactly one of them. We assume a global order on nodes, determining the committee a node is associated with and its rank within that committee. Given a node's public key, anyone may locally infer the node's committee and its rank therein, at a negligible computational cost.

The global order is determined by a swap-or-not shuffling algorithm (see Section 2.2) and a global seed. This seed is updated by some external oracle periodically, with a period corresponding to a consensus protocol's epoch, effectuating a permutation of nodes into a new order. Consequently, nodes are reshuffled across all K committees in each new epoch.

The goal is for nodes to discover *all* other members of the same committee. In other words, the goal is to let nodes self-organize into a topology comprising one clique per committee. At epoch change, nodes should automatically drop their previous committees, and cluster together anew to reflect their new committee associations.

It is crucial for this process to be completed during a *preparation period*, which coincides with an epoch's duration. The shuffling algorithm produces a new permutation at the beginning of each epoch to be used in the subsequent epoch. This gives nodes one epoch's time to build the target topology, forming the corresponding links among themselves.

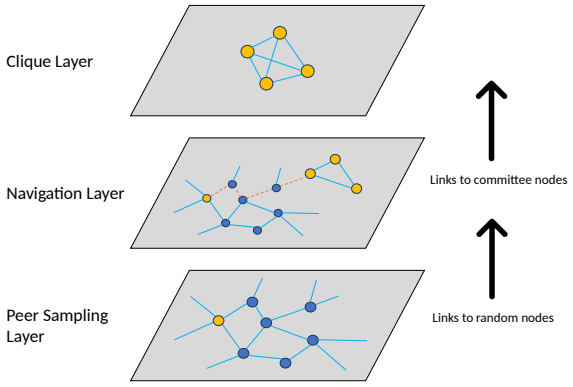


Figure 1. CliqueSensus architecture as a three-layer protocol. Each layer constitutes a distinct gossiping protocol, maintaining its own list of neighbors and gossiping with them. Lower layers help higher layers by equipping them with links from their own neighbors.

4 The CliqueSensus Protocol

The CliqueSensus protocol builds on the system model's objective of enabling nodes to discover and self-organize into one clique per committee. Its design ensures that nodes quickly form these cliques before the end of each epoch, with a new reorganization taking place at the start of every epoch.

This is a three-faceted endeavor.

1. First, nodes need to form and maintain a single connected overlay, preventing them from getting seggregated into disjoint components, and constituting the basis on which to perform peer discovery.
2. Second, there should be a mechanism to facilitate nodes to "navigate" towards nodes of the same committee. That is, a notion of committee ordering and proximity should be introduced.
3. Third, nodes within the same committee should help each other get to know *all* members of their committee, to form a clique.

CliqueSensus addresses the aforementioned goals by deploying three distinct gossiping protocols, depicted as three layers in Figure 1. Each node maintains, separately per layer, a list of links to other nodes, its *neighbors*, collectively known as the node's *view* of the network. On behalf of each layer, each node periodically performs *gossip*, that is, it contacts one of its neighbors in that layer's view and they exchange a few links in an effort to improve their views. In CliqueSensus, the three layers are closely intertwined and operate in a cooperative manner, assisting each other in improving their views and reaching their goals faster.

In a nutshell, CliqueSensus works as follows:

- At the beginning of each new epoch, a node has no other neighbors than a set of links to random peers out of the entire overlay, provided by the **peer-sampling layer**.

This layer performs random peer sampling for discovering peers, which it subsequently feeds to the navigation layer, right above it in Figure 1.

- The **navigation layer** introduces a notion of proximity among committees. With each gossip exchange at that layer, nodes assist each other to discover neighbors at progressively closer committees to their own. Before long, each node gets to discover at least *some* neighbors of the same committee, which it feeds to the clique layer, the top layer in Figure 1.
- By gossiping at the **clique layer**, nodes within the same committee efficiently share missing links, rapidly completing the clique overlay.

The following sections detail the gossiping layers comprising CliqueSensus, discussing their rationales and reasoning on their design choices. We present them in a bottom-up order.

4.1 Peer Sampling Layer

Peer Sampling protocols [21] form a well-studied family of gossiping protocols known for maintaining robust P2P overlays in a self-organizing manner and with strong self-healing properties.

The Peer Sampling layer serves two primary functions. First, it guarantees that the entire set of nodes remains connected in a single connected component. Second, it constitutes a continuous source of random peer samples, that is, fresh links to randomly chosen peers among all alive nodes in the network. The continuous link provision ensures the systematic replacement of old—and possibly obsolete—links by newly acquired, fresh links. This is key to peer discovery.

We have opted to use the SecureCyclon [8] version of the Cyclon protocol [30], a popular representative of the peer-sampling protocol family. Cyclon is highly scalable and lightweight in network resources, as each node is only required to retain a very small, fixed-sized view and to engage in small gossip exchanges once per cycle, irrespectively of the network size.

The SecureCyclon variant of the protocol enforces nodes to operate in a predefined manner. Specifically, it mandates that every node in the overlay periodically generates a limited number of fresh links, as dictated by the protocol. This mechanism prevents malicious nodes from manipulating the connections to and from legitimate nodes, thereby inhibiting the formation of hubs by adversaries and ensuring they cannot dominate legitimate node communications.

SecureCyclon requires a third-party Sybil-resistance mechanism to be in place. This requirement is inherently met by blockchains that employ Proof-of-Stake (PoS) protocols, where each consensus node must stake a portion of its wealth to participate in the protocol, and whose identity is publicly registered on the blockchain.

As shown in [30], the links in Cyclon have an average lifespan equal to the configured view length. To achieve a high link refresh rate, we opted for a low view length. As shown in our experiments, a Cyclon view length of 8 is sufficient for achieving rapid convergence in our protocol.

It should be noted that the Peer Sampling layer is the only one of the three layers that retains its view when a new epoch starts. The other two layers discard their views and start rebuilding them from scratch. This is thanks to the self-healing properties of the Peer Sampling layer, removing stale links of departed nodes, fully incorporating newly joined nodes into the overlay, replacing old and invalid links of nodes (e.g., if their IP address changed) with fresh ones, and keeping an equal representation of all nodes in the overlay with uniformly random connectivity.

4.2 Navigation Layer

The Navigation layer helps nodes navigate through the overlay to reach nodes of the same committee in a timely manner.

Rather than letting nodes come across same-committee peers relying purely on random encounters, which could take unpredictably long given the high number of 2048 committees, the Navigation layer defines a proximity metric by which nodes progressively get *closer* to their same-committee peers, and eventually connect to them.

This layer employs Vicinity [31], a gossiping protocol that organizes an initially arbitrarily connected overlay into a structured topology. The target structure is defined through a proximity metric, which quantifies some notion of proximity between any pair of nodes.

Following the classic gossiping paradigm, each node maintains a small number of neighbors and periodically gossips with one of them to exchange a few links. Both nodes aim to discover neighbors of as close proximity as possible. Assuming a transitive property in the specified proximity metric, the closer a node gets to its neighborhood of proximal peers, the higher its chances to discover even more of them, as its current neighbors must have also discovered other nodes in that same vicinity.

In CliqueSensus, the Navigation layer models committees as vertices in a ring structure, enumerated from 0 to 2047. The proximity between two committees is defined as the Euclidean distance between them in this ring, in modulo arithmetic. Key to the operation of this layer is the swap-or-not shuffling algorithm (Section 2.2), which enables nodes to *locally* infer any other node's committee for a given epoch.

A node's goal is to pick neighbors belonging to committees increasingly closer to its own, and eventually its own. With each node sharing the same objective, the probability of a node acquiring neighbors from its committee steadily rises with the progression of cycles, as it gradually gets connected with neighbors closer to its committee.

More specifically, when two nodes are gossiping, the sender first incorporates all its current Peer Sampling neighbors in

its Navigation view, and subsequently picks to send the peers that are closest to the receiver's committee. This way, every time a node gossips, either on its own initiative or in response to the gossip request of another node, it is exposed both to the other node's accummulated proximal link concentration, as well as to its own and the other node's random entropy stemming from the links in their Peer Sampling layers. The former accelerates clustering once the topology has started forming, while the latter plays a crucial jumpstart role in the early stages of clustering, by offering potential shortcuts leading nodes in the vicinity of their committees.

We deviate from the standard Vicinity protocol in two ways. First, given the ephemeral nature of clustering in our use case, nodes do not need to discard any neighbors at the Navigation layer while clustering is underway. They are altogether discarded, however, when a new epoch starts and the clustering procedure commences all over again.

The second deviation is that, although nodes get to acquire neighbors of the same committee, they do not store them in the Navigation layer view. Instead, they hand them directly over to the Clique layer, discussed in the following section. The rationale behind this decision is that the Navigation layer's goal is to build and strengthen inter-committee paths, to point nodes of other committees in the right direction, rather than getting isolated within one's committee.

Finally, a node decides whom to contact for gossiping by picking its closest proximity neighbor. However, to maximize mingling diversity, a node does not contact the same neighbor twice before all other neighbors have also been contacted, approximating a round-robin iteration.

As far as network resources are concerned, the Navigation layer is extremely lightweight. As we will see in our evaluation (Section 5), a surprisingly small value of 3 links sent to the other party in a gossip exchange is sufficient for CliqueSensus to converge fast to the target clustering, effectively exchanging negligibly small messages.

4.3 Clique Layer

The Clique layer enables nodes of the same committee to share their knowledge on committee membership, effectively letting nodes of the same committee self-organize into a clique structure.

The operation of the Clique layer can be outlined as follows. Each node maintains in its Clique layer's view *all* peers of the same committee it has discovered so far. This discovery is initially triggered by the Navigation layer, which, as explained in the previous section, feeds all same-committee peers it gets to see into the Clique layer's view. Given that the Navigation layer incorporates all links acquired from the Peer sampling layer, the Clique layer also peeks into the acquired random links for potential matches with its committee. Gossiping is triggered periodically, by the Clique layer picking one of its neighbors at random.

Clique view synchronization leverages the global ordering provided by the swap-or-not shuffling algorithm (Section 2.2), and more specifically the fact that for any given node, one can infer: (i) which committee it currently belongs to, and (ii) what its rank within that committee's ordering is. The rank, specifically, allows nodes to encode in the shortest possible data structure which committee members they already know and which ones they are still missing, by creating a bitmap where each committee node's presence is represented by a mere single bit. When gossiping in this layer, nodes also exchange these bitmaps, reporting which committee members they know, and allowing them to fill each other's gaps.

A gossip exchange involves three messages. A first one from the gossip initiator carrying its bitmap, a second one returning the receiver's bitmap and any links the initiator is missing, and finally a third message from the initiator providing links the receiver is missing.

The selection of which neighbor to gossip with is made in a round-robin fashion, in a random initial order. This maximizes the diversity of contacted nodes, thereby optimizing information exchange. Moreover, it accelerates the incorporation of nodes that were slow in discovering committee members.

Finally, the Clique layer discards its entire view by the start of a new epoch and starts building it all over again. This is an obvious decision, as the links of a past and disbanded committee are of no use in a new epoch.

4.4 Security Discussion

CliqueSensus' security is based on two pillars: (1) the on-chain registration of nodes' public keys, which also serve as unique identifiers, and (2) the SecureCyclon protocol. SecureCyclon utilizes unique links signed by their owners, preventing malicious nodes from arbitrarily forging connections. As a result, an adversary cannot fake which committee a node currently serves, cannot impersonate a legitimate node, and cannot flood any committee with numerous malicious nodes.

Provided that the proportion of links to malicious nodes remains within safe bounds, each node will begin every epoch with a valid sample of random peers. However, in the Navigation layer, malicious nodes could refuse to respond to gossip requests, or could intentionally provide links of low utility to legitimate nodes, slowing down the overlay network's convergence. Similarly, in the Clique layer, malicious nodes may withhold committee links to delay clique formation. Nonetheless, nodes unable to fully connect to their committees can still find legitimate members via the Navigation layer. As a result, these disruptions are partially mitigated, as nodes which possess a link to a malicious committee member, typically already also possess links to nodes of nearby committees.

In summary, systematic security evaluations should investigate the extent to which malicious nodes can disrupt

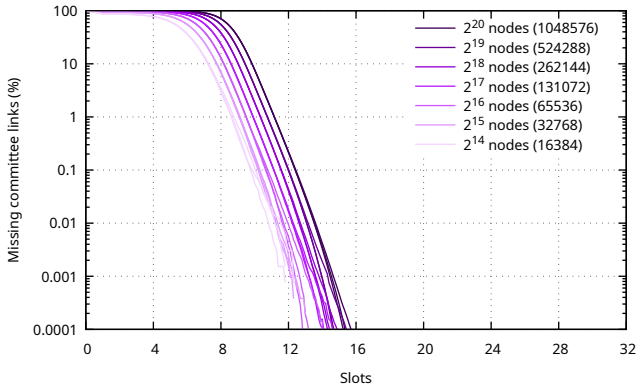


Figure 2. Percentage of committee links still missing from the network. Seven network sizes are considered (2^i nodes for $i \in [14, 20]$) , with three individual runs for each. Different runs of the same configuration exhibit hardly any deviation, demonstrating the protocol's highly predictable behavior.

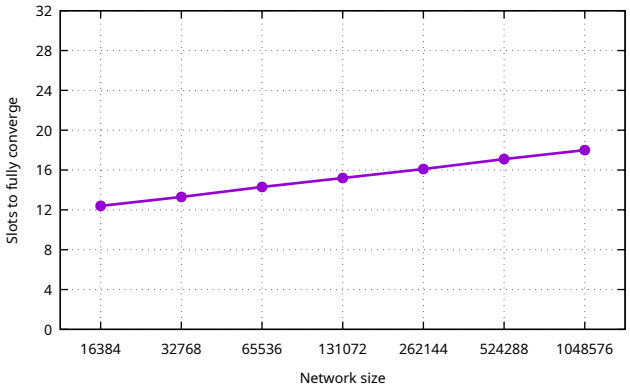


Figure 3. Number of slots required for all cliques to be formed, for network sizes from 16K to 1M nodes (2^i with $i \in [14, 20]$).

legitimate operations in the navigation layer. Although, this paper does not provide such assessments, we extensively evaluate our protocol resistance to churn in Section 5.3.

4.5 Challenges of Integrating with Ethereum

Seamless integration with the Ethereum ecosystem presents the following implementation challenges:

1. Multi-Client Support: Ethereum supports multiple clients implemented in different programming languages. Deploying the protocol would require simultaneous adoption by all clients, demanding standardization efforts.
2. Malicious Behavior Proofs: SecureCyclon generates deterministic proofs of malicious behavior, enabling penalties for misbehaving nodes. Ethereum 2.0 employs a similar mechanism, known as *slashing*, at the consensus layer. Extending this mechanism from the consensus layer to the network layer would require protocol-level integration.

5 Evaluation

We implemented both CliqueSensus and GossipSub on the *PeerNet Simulator* [1], a fork of the popular event-driven *PeerSim Simulator* [26], and we conducted an extensive set of experiments to assess their individual and comparative performance.

5.1 Experimental Setup

In order to emulate realistic network latencies between nodes, we acquired a real-world latency trace publicly available by WonderNetwork [2], and we used it to model the latencies between Ethereum nodes.

We implemented and simulated GossipSub based on the specifications detailed in [6]. We parameterized GossipSub by adopting the values provided by the Ethereum specifications [3], as described in Section 4.1. We simulated the core GossipSub version 1.0, and incorporated flood publishing from version 1.1, as it had a positive impact on the message dissemination speed of GossipSub.

The current peer discovery mechanism in Ethereum is `Discv5` [5], which utilizes a Kademlia-based recursive lookup operation for peer sampling. Instead of implementing and deploying Kademlia, we initialized each GossipSub node with 10 links to other nodes in its committee. Given that links are bidirectional, nodes ended up having on average 20 random neighbors each within their respective committees, which is sufficient to fulfill the requirements of the GossipSub protocol.

We established a vanilla configuration of our protocol as a point of reference, which aims for message efficiency by achieving rapid convergence with lightweight messages. In our vanilla configuration, every layer operates at the same frequency, gossiping once per cycle. The cycle is defined to coincide with an Ethereum slot (i.e., 12 seconds). The Peer Sampling layer is configured with a small view length of 8 neighbors. Finally, when gossiping, nodes in the Peer Sampling and Navigation layers send as few as $g_{PS} = 2$ and $g_N = 3$ links, parameters known as the respective *gossip lengths*.

5.2 Convergence Evaluation

Timely convergence is the most important requirement in our use case. We show that, even at its most lightweight configuration, CliqueSensus manages to organize committees into cliques in just half of the required time, that is, within half an epoch.

Figure 2 shows the number of committee links nodes are still missing on average in the course of time, for network sizes of up to one million nodes (specifically, for 2^i nodes with $i \in [14, 20]$). A node should end up having links to all other members in its committee. Initially all nodes have

no committee links (missing all 100% of them), but they gradually discover committee neighbors and they eventually get to know them all. The evolution of all experiments is very fast.

Two important observations can be made in these experiments. First, different runs of the same configurations exhibit almost identical convergence, which demonstrates a highly predictable behavior for our protocol. Second, convergence time increases linearly at an exponential increase of the network size, demonstrating a strong scalability.

Figure 3 presents the time required for a fully converged overlay to emerge, as a function of the network size (in logarithmic scale). The time required for full convergence is shown to closely follow the function $f(x) = log_2 x + c$ (with c being approximately -2), confirming CliqueSensus' logarithmic scalability. This indicates successful utilization in extremely large networks, achieving full convergence for several million nodes long before the end of an epoch.

5.3 Convergence in the Presence of Churn

The set of validators during an epoch is fixed, as new validators can only register or deregister at epoch boundaries. Additionally, validators' main concern is to remain uninterruptedly online, in order to participate in the protocol and earn rewards. Failure to do so incurs penalties.

In this context, churn concerns the temporary absence of nodes, out of this fixed registered set, due to technical issues (e.g., network delays or transient failures), with nodes making every effort to reconnect and deliver their votes.

We evaluate churn in two scenarios. The first scenario assesses the impact of independent node failures. The second one considers nodes that join the network overlay construction significantly later than expected.

In the first scenario, we simulate sporadic node absence by randomly removing a percentage p of the nodes in each cycle, and keeping them disconnected for c cycles, after which they resume the protocol. We refer to p as the *churn rate* and to c as the *disconnection duration*. At any given moment, a percentage of $p \cdot c$ of the nodes are disconnected.

Removed nodes do not participate in gossiping, neither by initiating or by responding to messages, however they do retain their previous state. Active nodes that attempt to communicate with disconnected nodes will, instead, try to connect to the next available node.

Figure 4a shows the results for $N = 262144$ nodes, with churn rates up to 10% per slot, and disconnection durations from 1 to 5 slots. Relatively low churn rates (1–2% per slot) incur negligible delays in convergence, compared to the no-churn scenario. With higher churn rates, nodes still converge to the target topology, albeit with increased delays. Note that the scenarios that take more than 25 slots to converge represent extreme churn conditions. For instance for a churn rate of 10% and disconnections of 5 slots (rightmost line), 50% of the nodes are offline at any moment. These plots highlight the Navigation layer's ability in guiding disconnected nodes via paths that were established while they were absent.

Figure 4b presents experiments in which 2048 nodes have been held back, joining late, at slot 20. We observe that these late-joining nodes converge to their committees faster than the early nodes, within approximately 8 cycles instead of 12-16 (Figure 2). This is expected, as the already formed overlay navigates them swiftly to their target committees, much faster compared to a full network cold start. As observed, all committees are formed well before the epoch concludes at slot 32.

5.4 Gossip Frequencies

In this set of experiments we explore the effect of each individual layer's gossiping frequency on the overall convergence speed. We do so by modifying the gossiping frequency of one layer at a time, while keeping the other layers' frequencies constant and equal to their default values in the vanilla version, that is, one cycle per slot.

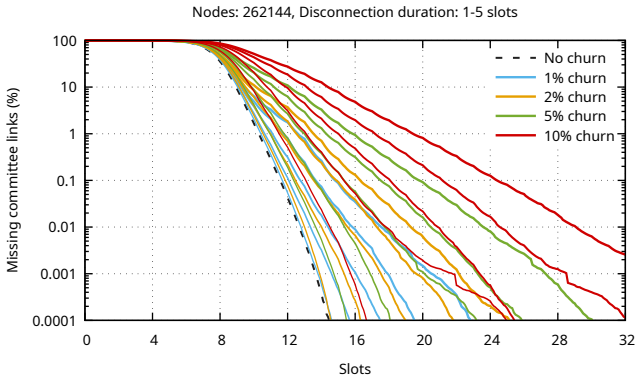
Figure 5 presents these experiments, for $N = 262144$. It is clear that the highest improvement is exhibited when the Navigation layer's gossiping frequency is boosted. Conversely, the Peer Sampling layer's gossiping frequency does not appear to affect the overall convergence speed. This observation suggests that the random links provided by the Peer Sampling layer at the start of a new epoch constitute a sufficient basis to build the target overlay, however, in later stages convergence depends on exchanging strategically chosen links rather than sampling more links at random.

When increasing the frequencies of the Navigation and Clique layers, we initially observe a sharp improvement that smooths out for even higher frequencies. The flattening of the slopes underlines the importance of cooperation between both the Navigation and Clique layers in achieving swift convergence. Having a high Clique layer frequency but low Navigation layer frequency, will lead to the slow discovery of initial committee members, with a direct negative impact on the overall convergence speed. The opposite will lead to a fast initial discovery, but a slow completion of each individual clique afterwards.

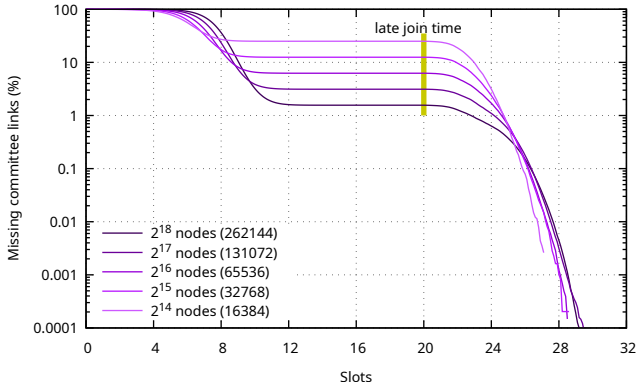
5.5 Gossip Lengths

Figure 6 presents the convergence time for various values of the Navigation layer's gossip length, g_N, for a network size of $N = 262144$ nodes. As expected, when nodes exchange more neighbors per gossip at the Navigation layer, the target overlays emerge faster.

However, when comparing this against Figure 5, we observe that there is a higher improvement to gain by doubling the gossip frequency of the Navigation layer than by increasing its gossip length from 3 to 20, while the former introduces less network overhead. Specifically, a gossip length of 20 leads to the transmission of 40 links for a node per cycle, on average, considering that each node gossips twice



(a) Convergence under churn. Churn rate $p\%$ means that $p\%$ of the nodes fail per slot and remain disconnected for a duration of 1 (leftmost, thinnest lines), 2, 3, 4, or 5 (rightmost, thickest lines) slots, when they resume.



(b) Convergence for nodes joining late. In these experiments 2048 nodes do not join until slot 20.

Figure 4. CliqueSensus overlay convergence under churn conditions.

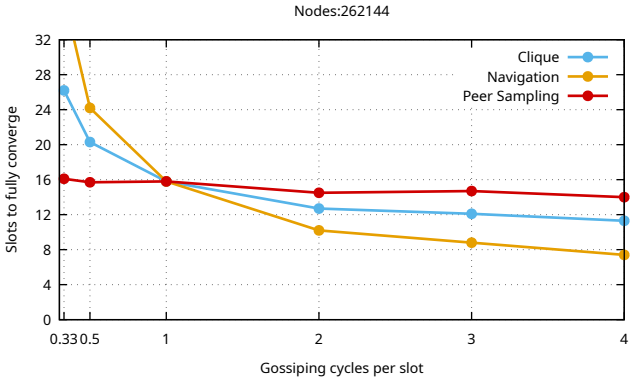


Figure 5. Number of slots required for all cliques to be formed, for various execution frequencies of the three layers. The x-axis indicates the number of gossiping cycles a given layer executes during a slot. In our vanilla configuration, each layer executes one cycle per slot.

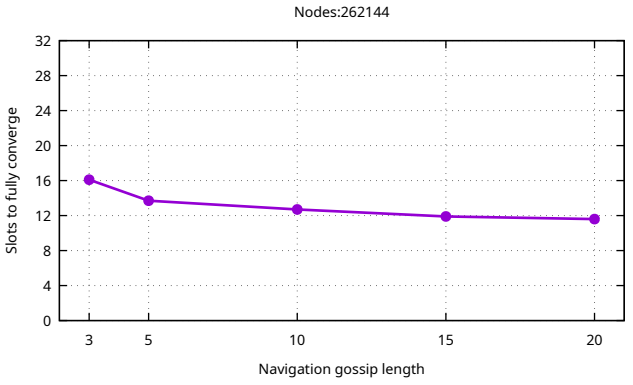


Figure 6. Number of slots required for all cliques to be formed, for various Navigation layer gossip lengths.

in that period, once as the initiator and once as the recipient. Similarly, by doubling the gossip frequency with a gossip length of 3, the node will transmit 12 links per cycle, on average, as it will gossip four times, leading to 28 fewer links transmitted.

This observation leads us to the conclusion that, taking many, small, iterative steps towards a node's committee is more effective than making fewer, but larger iterations.

5.6 Overlay Construction

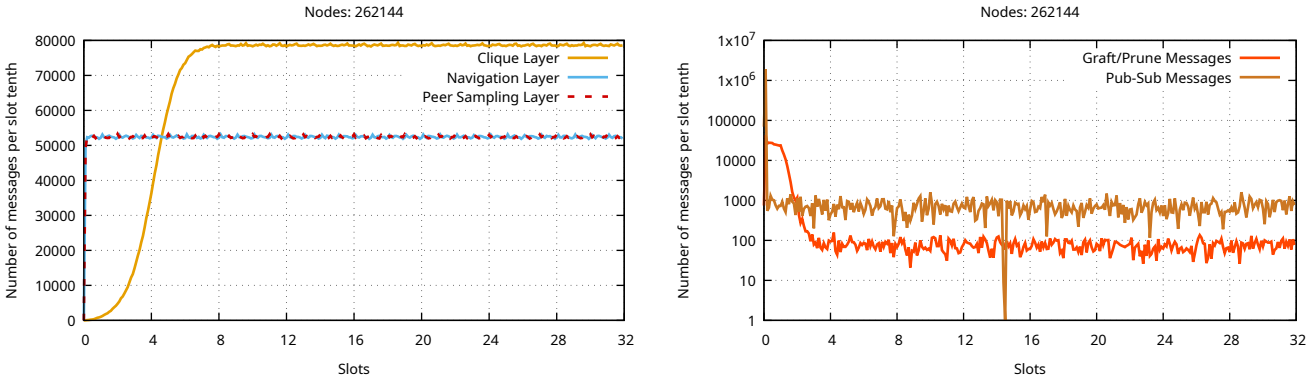
In this set of experiments we assess the number of messages sent for overlay construction in CliqueSensus and GossipSub, considering the network size of $N = 262144$.

With respect to the CliqueSensus experiments, we initiate the preparation phase as soon as each node has established links with 8 random other nodes through the Peer Sampling layer.

In GossipSub, validators learn about each other by exchanging *pub/sub* messages that contain the topics in which each validator participates. Following the pub/sub message exchange, each node attempts to establish approximately 8 mesh neighbors for its topic by exchanging *graft*/*prune* messages. We allow each topic to achieve a stable state, and when the dissemination of pub/sub, and graft/prune messages ceases, we initiate the preparation phase.

Figure 7a presents the number of messages transmitted by CliqueSensus, sampled ten times per slot. The Navigation and Clique layers are observed to be generating a stable flow of messages, which peaks at approximately $520K$ messages per slot. The Clique layer starts without transmitting any messages but catches up by cycle 8, when each validator has acquired some of its committee members.

Likewise, Figure 7b presents the corresponding messages for GossipSub. These are the messages triggered by the 16 aggregator nodes per committee, which join the topics of their



(a) CliqueSensus **(b)** GossipSub

Figure 7. Messages exchanged during each slot of the preparation phase, for both GossipSub and our protocol, sampled ten times per slot. The overlay of GossipSub and the Peer Sampling layer were let converge before initiating the experiments.

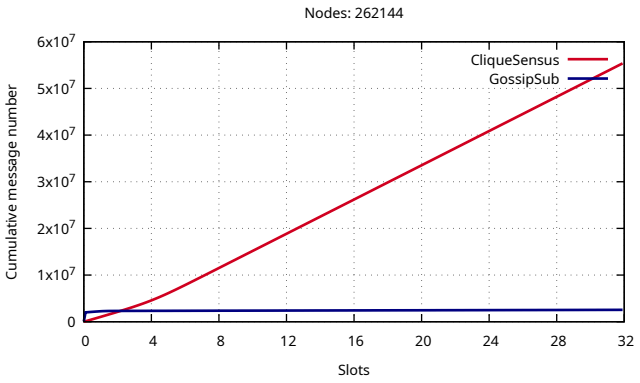


Figure 8. Overlay construction messages (cumulative) for CliqueSensus and GossipSub, for the entire network.

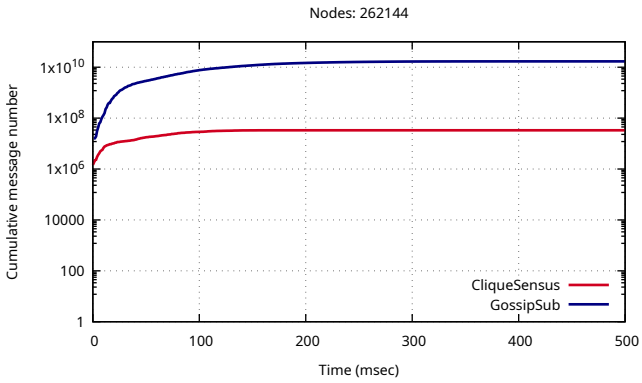


Figure 9. Attestation dissemination messages (cumulative) for CliqueSensus and GossipSub, for the entire network.

committees (appearing as a steep rise at the beginning of slot zero). Additionally, they include messages that correspond to nodes which leave their topics and join two new, random ones once every 256 epochs, according to the Ethereum specifications.

The cumulative message count of both protocols is presented in Figure 8. It is evident that our protocol is more expensive, with respect to overlay construction, requiring 5.5×10^7 messages per epoch compared to the Gossipsub's 2.5×10^6 messages..

This result was expected, as CliqueSensus strives to create fine-grained overlays on the fly, while GossipSub relies on mostly static, large overlays. As we will see in the following sections, the benefits derived from our protocol during attestation dissemination exceeds by far the cost of overlay construction.

5.7 Attestation Dissemination

Figure 9 presents the cumulative disseminated attestation count for all 2048 committees of an epoch, for both CliqueSensus and GossipSub, within a network of 262144 nodes. In these experiments, both protocols were left to form their fully converged topologies. In CliqueSensus, attestation dissemination occurred through the corresponding committee cliques, while in GossipSub, each validator was equipped with 10 gateway-links to random nodes from their committee topic. Upon the completion of topology convergence for both protocols, all validators concurrently initiated the dissemination of their attestations.

CliqueSensus outperforms GossipSub by a significant difference. More precisely, CliqueSensus and GossibSub concluded the dissemination with a total of 16, 256 and 8.29×10^6 messages, respectively, per committee. This accounts to a total of 3.3×10^7 and 1.7×10^{10} messages, respectively, for the whole network for the entire epoch (Figure 9).

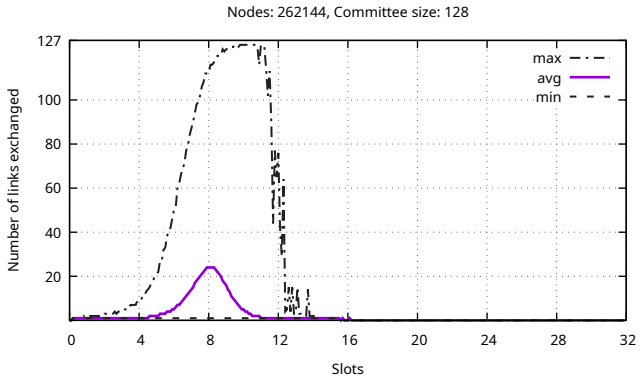


Figure 10. Number of links exchanged in the Clique layer per node, per bidirectional gossip exchange.

5.8 Overall Message Improvement

Factoring in both the messages for overlay construction and attestation dissemination, CliqueSensus achieves attestation dissemination for each epoch by utilizing 191.5 times fewer messages than GossipSub. This demonstrates that the higher network overhead required to construct a fine-grained overlay is compensated by significantly reduced network usage during attestation dissemination, making it orders of magnitude more efficient.

To assess the broader impact of our protocol on Ethereum block dissemination, we analyze its message efficiency across all three block generation phases discussed in Section 2.3.

In both the *Proposing* and *Aggregating* phases, GossipSub employs global topics, where all validator nodes participate. Although our protocol lacks a dedicated mechanism for these actions, a simple broadcast using 8 links from each node's peer-sampling layer achieves equivalent results with an approximately equal number of messages.

In the Proposing phase, a single node disseminates a message to all N validator nodes. For $N = 262144$, this results in $262144 \times 7 \times 32 = 5.87 \times 10^7$ messages per epoch.

In the Aggregating phase, approximately 16 aggregator nodes from each of the 64 committees disseminate attestation aggregation messages to all N nodes. For $N = 262144$, this amounts to $16 \times 64 \times 262144 \times 7 \times 32 = 6 \times 10^{10}$ messages per epoch.

Considering the combined number of messages across all three phases—Proposing, Attesting, Aggregating—we conclude that CliqueSensus sends 22% fewer messages than GossipSub, achieving significant network savings.

5.9 Dissemination Speed

Finally, we compare CliqueSensus and GossipSub with respect to their performance in attestation dissemination speed.

Figure 11a presents the percentage of not-yet-delivered attestations ("missing messages"), as well as the percentage of nodes that have not received all attestations yet ("incomplete nodes"), in the course of time, for networks of 262144 nodes. Protocol initialization was conducted in the same way as in Section 5.7. It is evident that the single-hop dissemination through our clique overlay accelerates attestation dissemination, resulting in the faster acquisition of votes by all validators.

With our approach, all messages have been disseminated within 216 msec, in contrast to GossipSub's 273 msec. This represents a 21% reduction in dissemination time.

Figure 11b presents aggregate results of this type of experiment for various network sizes. It is evident that CliqueSensus consistently outperforms GossipSub across all network sizes, with more substantial advantages at lower sizes, reaching up to approximately a 70% drop in dissemination time for networks of 16384 nodes.

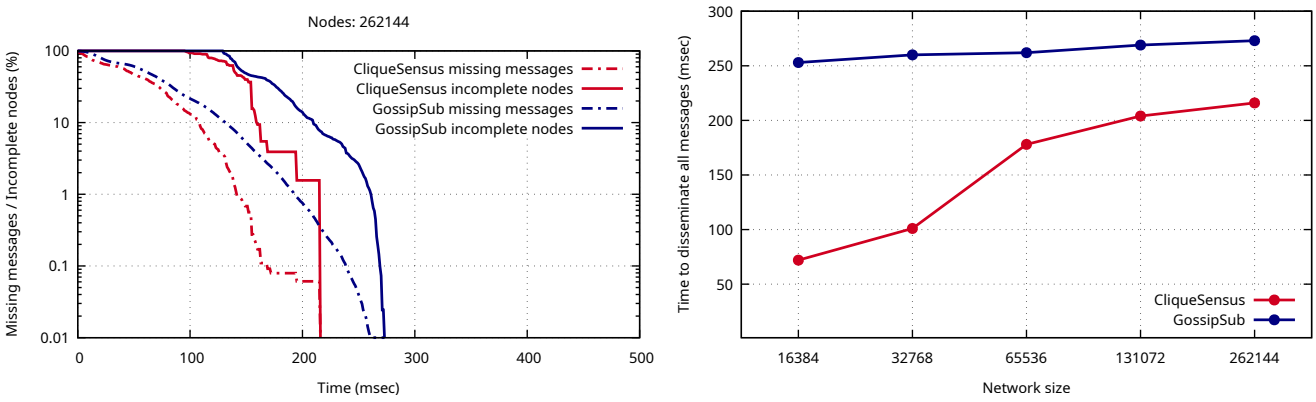
These findings imply that CliqueSensus excels in performance, especially when clique sizes become smaller. Consequently, a consensus mechanism utilizing our protocol could flexibly adapt to smaller committee sizes, particularly when rapid dissemination is imperative for achieving consensus.

6 Related Work

The closest literature to our work can be found in the domain of topic-based distributed pub/sub protocols (e.g., PolderCast [28], Spidercast [14], Tera [10], and VCube [16]). These works apply the pub/sub paradigm [18] in distributed settings, establishing topics/channels among nodes without relying on a centralized orchestration entity. Typically, such solutions exhibit remarkable scalability, combining the scalability inherent in P2P approaches, where peers have the role of both the client and the server, and the sender-receiver decoupling of the pub/sub paradigm [23]. Application-level multicast approaches, such as Scribe [13] and Bayeux [34], share similar semantics to distributed topic-based pub/sub, and often fall under the same category.

The diversity of existing approaches shows that there is no one-fits-all solution. Instead, a group-based messaging solution should be tailor-made, or carefully adapted to meet the needs of each individual system.

The work most closely aligned with ours is presented in PolderCast [28], a scalable distributed publish/subscribe approach aimed at ensuring guaranteed connectivity within topics, particularly in environments characterized by high churn. Similar to our design, PolderCast is structured as a three-layer protocol, employing vanilla Cyclon for peer sampling and establishing ring topologies at the top layer. Despite the structural similarities between the two protocols, their design and implementation differ significantly as they address distinct objectives. For instance, (CliqueSensus) leverages contrived proximity and is fully optimized to enable a node to locate all nodes within a group efficiently. In contrast, PolderCast matches nodes to ensure that each node



(a) The course of the dissemination through the voting period.

(b) Dissemination times for different network sizes,

Figure 11. Comparison between CliqueSensus and GossipSub on the speed of attestation dissemination, for a single committee. Both protocols are left to converge, and in GossipSub, validators are equipped with 10 links to random members of their committee topic.

maintains a sufficient number of links per topic, while keeping the total number of links at a moderate level. PolderCast is designed for scenarios involving multiple topics, where a node must establish efficient connections across various topics, even if this process is not rapid. Conversely, CliqueSensusaims to form clique clusters as quickly as possible.

Another study which focuses on establishing topic connectivity, with scalable average node degree and topic diameter, for environments with churn is Spidercast [14]. In this approach, each node employs two heuristics to establish neighbors: first, selecting peers with the most topics in common with the node, and second, choosing random peers with at least one topic in common. To maintain an adapted view length, nodes adjust their neighbor count by adding or removing links, while considering the view length of their neighbors. This approach requires each node to retain descriptors for 5% of the total number of nodes, a sharp contrast to our approach, which accumulates descriptors for up to approximately 0.05% of all nodes. Moreover, the experiments in Spidercast were conducted with a limited node count of up to 8K nodes, which is notably low given our use case.

Scribe [13] and Bayeux [34] are two approaches which facilitate multiple individual multicast groups on top of the Pastry and Tapestry DHTs, respectively. These approaches utilize efficient multicast trees that get formed by merging DHT routes, enabling rapid, non-redundant message dissemination within each group. Nonetheless, such approaches suffer from a single point of failure, in the form of multicast roots, and the message dissemination of a group may be facilitated by nodes that are not part of the specific group.

An approach that utilizes clique-graphs in the domain of P2P networks is eQuus [25]. This work introduces a DHT storage system designed for environments with high churn. In eQuus, nodes that are close in terms of physical network proximity form cliques, sharing the same ID, and facilitating the storage of identical items. The utilization of cliques in this approach yields several desirable properties: a good degree of redundancy, as node failures are offset by other clique members possessing identical data; swift consistency, facilitated by the close proximity among clique nodes, which leads to fast item synchronization; and resilience to churn, accomplished by restricting communications to the clique-level during node entry and exit from the network overlay.

7 Conclusions

Considering the problem of distributed voting in PoS-based blockchains, with a focus on Ethereum 2.0, we have introduced CliqueSensus, a protocol that organizes consensus nodes into clique-structured, ephemeral overlay clusters. Through CliqueSensus, each node begins with an initial set of randomly selected, up-to-date overlay links, and progressively navigates through the network to eventually obtain links to the members of its committee. This behavior arises through self-organisation, rendering our solution inherently decentralized and highly scalable.

We evaluate our solution through extensive simulations, scaling up to 2^{20} nodes, using Ethereum's current approach based on GossipSub as a benchmark. Our results show that CliqueSensus achieves rapid convergence, even under extreme churn conditions, ensuring timely committee formation and communication.

Moreover, CliqueSensus outperforms Ethereum's existing method by transmitting far fewer messages and achieving faster overall message dissemination.

Acknowledgements

This work has been funded and supported by the Ethereum Foundation, in the context of the *Eclipse and DoS-Resilient Overlays for High-Performance Block Dissemination* project.

References

[1] 2011. *PeerNet Simulator Repository.* https://github.com/PeerNet

[2] 2020. *Wonder Network Traces.* https://wonderproxy.com/blog/a-day-in-the-life-of-the-internet/

[3] 2024. *Ethereum 2.0 p2p Specifications.* https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/p2p-interface.md

[4] 2024. *Ethereum Geth Client Specifications.* https://geth.ethereum.org/docs/fundamentals/peer-to-peer

[5] 2024. *Ethereum Peer Discovery Service.* https://github.com/ethereum/devp2p/blob/master/discv5/discv5.md

[6] 2024. *GossipSub Specifications.* https://github.com/libp2p/specs/tree/master/pubsub/gossipsub

[7] Ailidani Ailijiang, Aleksey Charapko, and Murat Demirbas. 2016. Consensus in the cloud: Paxos systems demystified. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–10.

[8] Alexandros Antonov and Spyros Voulgaris. 2023. SecureCyclon: Dependable Peer Sampling. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1–12.

[9] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2019. Ouroboros chronos: Permissionless clock synchronization via proof-of-stake. *Cryptology ePrint Archive* (2019).

[10] Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. 2007. TERA: topic-based event routing for peer-to-peer architectures. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*. 2–13.

[11] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph. D. Dissertation. University of Guelph.

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

[13] Miguel Castro, Peter Druschel, A-M Kermarrec, and Antony IT Rowstron. 2002. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications* 20, 8 (2002), 1489–1499.

[14] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. 2007. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*. 14–25.

[15] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. 2013. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)* 31, 3 (2013), 1–22.

[16] João Paulo de Araujo, Luciana Arantes, Elias P Duarte Jr, Luiz A Rodrigues, and Pierre Sens. 2019. VCube-PS: A causal broadcast topic-based publish/subscribe system. *J. Parallel and Distrib. Comput.* 125 (2019), 18–30.

[17] Nour Diallo, Weidong Shi, Lei Xu, Zhimin Gao, Lin Chen, Yang Lu, Nolan Shah, Larry Carranco, Ton-Chanh Le, Abraham Bez Surez, et al. 2018. eGov-DAO: A better government using blockchain based decentralized autonomous organization. In *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*. IEEE, 166–171.

[18] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM computing surveys (CSUR)* 35, 2 (2003), 114–131.

[19] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*. 51–68.

[20] Viet Tung Hoang, Ben Morris, and Phillip Rogaway. 2012. An Enciphering Scheme Based on a Card Shuffle. In *Advances in Cryptology – CRYPTO 2012*. Springer Berlin Heidelberg, 1–13.

[21] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. 2007. Gossip-Based Peer Sampling. *ACM Transactions on Computer Systems* 25, 3 (2007), 8–es.

[22] Christoph Jentzsch. 2016. Decentralized autonomous organization to automate governance. *White paper, November* (2016).

[23] Anne-Marie Kermarrec and Peter Triantafillou. 2013. Xl peer-to-peer pub/sub systems. *ACM Computing Surveys (CSUR)* 46, 2 (2013), 1–45.

[24] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*. Springer, 357–388.

[25] Thomas Locher, Stefan Schmid, and Roger Wattenhofer. 2006. eQuus: A provably robust and locality-aware peer-to-peer system. In *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*. IEEE, 3–11.

[26] Alberto Montresor and Márk Jelasity. 2009. PeerSim: A scalable P2P simulator. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, 99–100.

[27] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*. 305–319.

[28] Vinay Setty, Maarten Van Steen, Roman Vitenberg, and Spyros Voulgaris. 2012. Poldercast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. In *Middleware 2012: ACM/IFIP/USENIX 13th International Middleware Conference, Montreal, QC, Canada, December 3-7, 2012. Proceedings 13*. Springer, 271–291.

[29] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. 2020. Cockroachdb: The resilient geo-distributed sql database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM New York, NY, USA, 1493–1509.

[30] Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. 2005. CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *Journal of Network and Systems Management* 13, 2 (June 2005), 197–217.

[31] Spyros Voulgaris and Maarten Van Steen. 2013. Vicinity: A pinch of randomness brings out the structure. In *Middleware 2013: ACM/IFIP/USENIX 14th International Middleware Conference, Beijing, China, December 9-13, 2013, Proceedings 14*. Springer, 21–40.

[32] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. 2020. GossipSub: Attack-resilient message propagation in the Filecoin and ETH2.0 networks. *arXiv preprint arXiv:2007.02754* (2020).

[33] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[34] Shelley Q Zhuang, Ben Y Zhao, Anthony D Joseph, Randy H Katz, and John D Kubiatowicz. 2001. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*. 11–20.