Behind Farcaster: Characterizing a Hybrid Decentralized Social Network

Qiming Ye
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
qiming@connect.hkust-gz.edu.cn

Wen Yang
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
wyang330@connect.hkust-gz.edu.cn

Saidu Sokoto
City St George's, University of London
London, UK
saidu.sokoto@city.ac.uk

Leonhard Balduf
TU Darmstadt
Darmstadt, Germany
leonhard.balduf@tu-darmstadt.de

Michał Król
City St George's, University of London
London, UK
michal.krol@city.ac.uk

Onur Ascigil
Lancaster University
Lancaster, UK
o.ascigil@lancaster.ac.uk

Gareth Tyson*
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
gtyson@ust.hk

Abstract

Decentralized social networks present independent alternatives to centralized counterparts, yet there are open questions about their efficiency and long-term viability. Recently, a new class of hybrid social network has emerged, which combines blockchain-based and off-chain operations to balance availability, performance, and sustainability. Farcaster was the first and largest hybrid social network to adopt this strategy. It combines on-chain identity management with peer-to-peer content dissemination via independent servers (aka *hubs*). In this paper, we present its first large-scale empirical study, analyzing 240 million user-generated messages (aka *casts*) across the platform and focusing on 3,000 independent hubs. We uncover asymmetries that challenge the platform's core goals of availability, performance, and sustainability. Many hubs exhibit repeated periods of non-responsiveness, and the network struggles to maintain a consistent state across hubs. Moreover, infrastructure centralization emerges due to hosting concentration and network topology skew. While gossip-based dissemination enables rapid initial delivery, achieving full consistency remains challenging: on average, 80 % of hubs' casts reach half of other hubs within 9.2 minutes, whereas the remaining 20 % require up to 57 minutes. Finally, despite collecting over $2.6M in storage fees, there is no redistribution to hub operators, raising concerns about long-term sustainability. Our findings provide pointers for the design of hybrid and decentralized social platforms.

*Also with Queen Mary University of London.

IMC '26, October 12–16, 2026, Karlsruhe, Germany

ACM ISBN 979-8-4007-2327-8/26/10
https://doi.org/10.1145/3777912.3809136

CCS Concepts

• **Networks** → **Network performance analysis**; **Network measurement**; • **Information systems** → **Social networks**.

ACM Reference Format:
Qiming Ye, Wen Yang, Saidu Sokoto, Leonhard Balduf, Michał Król, Onur Ascigil, and Gareth Tyson. 2026. Behind Farcaster: Characterizing a Hybrid Decentralized Social Network. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26), October 12–16, 2026, Karlsruhe, Germany.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3777912.3809136

1 Introduction

Major social platforms have faced growing criticism regarding their monopolistic control of user data [33, 61], their algorithmic amplification of misinformation, and their oversight of content moderation [63, 64]. These limitations have spurred the development of several decentralized social networking alternatives, each adopting a unique architecture with its own trade-offs. For example, the Mastodon microblogging platform employs a federated model [3], where users create accounts on one or more servers, which then communicate in a federated fashion; Nostr takes a relay-based approach, where users are not associated with individual servers but, instead, can post messages on any server in an unstructured way, to mitigate censorship [66]; Bluesky operates a modular social media architecture [43], allowing any operator to contribute alternative implementations; and memo.cash records all user social data on a public blockchain to guarantee data transparency [72]; These systems vary significantly in their architectural choices, each presenting distinct trade-offs in terms of scalability, performance and sustainability.

These trade-offs have led others to propose *hybrid* architectures that combine on-chain components with off-chain execution in an attempt to fuse the benefits of both approaches. Our work focuses on the largest hybrid microblogging social network: Farcaster. The

system adopts three design objectives typical of this architectural class, while operating atop a highly decentralized network [59]. First, *Availability* is delivered by replicating the entire user data across decentralized servers (aka hubs) that anyone can deploy [22]. This relies on a peer-to-peer (P2P) [44] network of hubs, which employ GossipSub [54] and synchronization via Diff Sync [32] to disseminate data (*e.g.* user posts). The design strives towards eventual consistency, whereby each hub hosts an entire replica of the global state. This contrasts sharply with federated off-chain platforms like Mastodon [3], where users must register with specific servers, granting administrators significant control [55]. Second, *Performance* is achieved by dividing functions into two categories: (1) frequent actions (*e.g.* message posting) employ a lightweight set of P2P hubs to store and disseminate the data, whereas (2) infrequent actions (*e.g.* account creation) employ a heavyweight blockchain to handle the data. This is intended to ensure that common activities attain high performance, while not compromising the security required for the infrequent governance actions. Finally, *Sustainability* is supported by a combination of on-chain storage rentals and token rewards. Specifically, the Farcaster administrators charge users an annual fee and allocate it to USDC rewards that not only incentivize high-quality content creators but also compensate third-party operators who contribute to the network's functionality [24]. This design aims to sustain the ecosystem without relying on centralized funding. In contrast, prior blockchain social networks (*e.g.* Lens) do not directly compensate operators, instead relying on broader chain-level economics [45] to sustain participation [41, 46, 60].

By April 2025, Farcaster had onboarded over 1M users and recorded 231K monthly active users (MAU). Despite these achievements, Farcaster's design has not yet been evaluated. We argue that this provides an ideal opportunity to examine how hybrid platforms operate in-the-wild and whether this approach is viable for broader decentralized social networks. We therefore evaluate delivery upon the above three design goals by conducting the first large-scale measurement study, encompassing 1M users, 240M posts (*i.e.* "casts"), and 3,000 hubs. In summary, we make the following major contributions:

- To the best of our knowledge, we present the first measurement study of the largest hybrid decentralized social media platform, Farcaster. We gather a dataset covering 1 million users, 240 million casts, and 3,000 hubs. [1]
- We evaluate the ability of the Farcaster's hybrid model to deliver high availability, performance, and sustainability. The full replication approach does increase availability. However, it is threatened by emerging hot spots, with over half of the hubs located in a single AS. We also discover some performance issues. While GossipSub effectively replicates data, hubs that miss messages must rely on a heavyweight synchronization mechanism that creates substantial delays.
- We claim that the bottlenecks above pose key sustainability challenges. Farcaster's current fee model allocates 93 % of registration revenue to unused storage quotas, leaving 99 % of hub operators unprofitable (see § 6), which underscores the need to redesign storage fees and incentive mechanisms.
- Based on our measurements, we identify design pain points in Farcaster and outline future directions to improve hybrid on/off-chain architectures.

2 A Primer on Farcaster

Farcaster introduces a hybrid model, with data spread between blockchain smart contracts and a dedicated unstructured P2P network. To participate, each user must create an account, which they can then use to share posts (*i.e. casts*). We refer interested readers to the full documentation [25].

Farcaster On-chain Actions. Farcaster manages its user accounts via Optimism, a Layer-2 (L2) blockchain solution built on top of Ethereum [11, 18, 51, 67]. Optimism reduces transaction costs by batching transactions before final settlement on Ethereum. This makes it cheaper to manage identities while also ensuring that no party can control or ban users. Users create their accounts by interacting with a dedicated Optimism smart contract, which assigns the user's blockchain account (*i.e.* custody address) a unique numerical Farcaster ID (FID). They can subsequently register a human-readable Farcaster User Name (Fname) (*e.g.* @alex) linked to the same account. To reduce exposure of the private key associated with the custody address, users authorize additional keys called *signers*. They generate *add signer* messages, sign with the custody key, and submit to the Optimism smart contract [25]. User messages (*i.e.* casts) are signed with these signer keys before being submitted. After registration, users can optionally bind additional blockchain addresses as wallets and then participate in token-related economic activities (*e.g.* tipping).

Farcaster Off-chain Actions. The majority of remaining user actions (*e.g.* liking, posting) are then stored and disseminated via a P2P network of servers called *hubs*. Each hub replicates the entire state of the platform, meaning that the whole network survives even if only one hub remains online. Anyone can run a hub, start synchronizing with the network, and validate the messages without gatekeepers. Each hub must also connect to an Optimism endpoint to sync and cache the relevant on-chain data (*e.g.* FIDs, Fnames). Following this, the new hub synchronizes *all* prior user data state (*e.g.* message casts) from one or more other hubs. Tech-savvy users interact with the system by submitting their casts to any hub of their choice using HTTP [6, 8] or gRPC [35]. Others typically do this via a convenient front-end app (aka Clients). Warpcast[2] is the most popular client, yet anybody can build their own. Before accepting a message, a hub verifies the user's on-chain storage rental payment and authorization via registered signer keys. If any individual hub attempts to censor a user, the messages can still be propagated through the remaining independently operated hubs, preventing any single operator from controlling user content.

Synchronization. Recall, each hub maintains a full replica of all users' data. If a hub receives updates (*e.g.* casts), it is responsible for sharing those messages globally. Thus, each hub maintains a *Delta Graph* [65] to represent all user messages it knows. This graph is stored as *Merkle Patricia Trie* [14] and synchronized across hubs

[1] https://zenodo.org/records/19480833

[2] Note, Warpcast (https://warpcast.com/) has been rebranded to Farcaster (https://farcaster.xyz/) in May 2025 to align with the protocol's name.

through two mechanisms: GossipSub [54] and Diff Sync [32]. GossipSub is a gossip-based publish-subscribe protocol that is used for disseminating two types of data: (1) Metadata about hubs, *e.g.* communication endpoints; and (2) User messages, *e.g.* casts. Hubs continuously exchange gossip messages to synchronize this information, but GossipSub provides no guarantees on timeliness or delivery. As a probabilistic broadcast protocol, it may fail to deliver certain updates even when all hubs are online. To address this limitation, hubs periodically perform unidirectional point-to-point synchronization utilizing the Diff Sync protocol every 2 hours. This operates by comparing the underlying Merkle Patricia Trie between a pair of hubs. For each mismatched node in the Merkle Patricia Trie, Diff Sync initiates a background task that recursively traverses the underlying sub-tree, retrieving and comparing all descendant nodes with those of the peer. This procedure is repeated until all discrepancies are resolved. Though time-consuming, Diff Sync enables hubs to retrieve missing historical messages.

3 Measurement Methodology

We gather five datasets: (1) *User Data:* we gather a complete snapshot of all user social posts and interactions. (2) *Hub Metadata:* using each hub's public API, we collect a longitudinal dataset of each hub's profile, storage summary, and responsiveness; (3) *Cast Propagation Data:* via a set of controlled cast injections, we monitor how long it takes the hubs to replicate the casts globally; (4) *Grafana Data:* using each hub's public Grafana dashboard [1], we gather each Hub's gossip network topology, throughput and message source details; (5) *On-chain Data:* we gather Farcaster's on-chain data from its two Ethereum Layer 2 networks, Optimism (for registration and identity management) and Base (for social transaction activities, *e.g.* tipping and token transactions). All data collection procedures adhere to the ethical guidelines (see Appendix A for details).

3.1 User Data

To ensure completeness, we deploy two hubs in geographically distinct regions (*i.e.* one in Asia and one in Europe). This setup helps capture user data from diverse replication routes within the network, ensuring more comprehensive coverage. Both hubs have been continuously synchronized with the global Farcaster network since October 24, 2024. We extract and merge all messages[3] retrieved by the two hubs. As of March 20, 2025, our dataset contains 1,024,225 registered FIDs, 240,871,656 casts and 702,798,258 reactions.

3.2 Hub Metadata

To analyze hub characteristics, we collect longitudinal operational status snapshots through the API exposed by each hub [26], as this is recommended practice [19]. All times reported below are in Coordinated Universal Time (UTC).

Hub Discovery. First, we discover the full set of hubs. For this, we maintain an evolving set of known Hubs $\mathcal{H}_{\text{known}}$. Starting from a single official seed hub, we conduct a breadth-first search (BFS) via the P2P gossip network, from Jan 15, 2025 to Mar 20, 2025. To reduce crawler overhead, each iteration queries the *currentPeers* endpoint of a randomly selected hub $h \in \mathcal{H}_{\text{known}}$, retrieving peers observed by h within the past 24 hours. Previously unseen peers are added to the set for subsequent exploration. If an iteration yields no new hubs, we skip the round and proceed to the next iteration. During this period, we identify 3,180 distinct IP-port pairs and 4,614 peer IDs associated with Farcaster Hubs. [4] The hubs also publish the FID of the hub operator, revealing a total of 3,889 unique Hub operators' FIDs. This allows us to link each hub with its operator's casts.

Operational Status. For each $h \in \mathcal{H}_{\text{known}}$, we query its GETINFO API endpoint at fixed 10-minute intervals. Each interval defines a round, r, which serves as the unit for our later responsiveness analysis. In each round, we collect the peer ID, operator FID, gRPC address, GossipSub address and storage usage from live hubs. Each Hub exposes the same IP address for both GossipSub and gRPC, which we annotate with its Autonomous System (AS) and geographical location using IP-based lookups from the IP2Location [40].

Responsiveness. To evaluate hub availability, we define a *responsive session* as a sequence of consecutive rounds during which a hub responds to GETINFO API queries. Hub responsiveness directly impacts user experience, as clients rely on these hubs for accessing messages (*i.e.* casts). A window starts with the first observed response and ends when the hub becomes nonresponsive. We consider a hub to be nonresponsive if it fails to respond for longer than a specified number of rounds, k, where each round spans 10 minutes. We choose 10 minutes per round to balance granularity and noise: shorter intervals may capture transient issues rather than true downtime, while longer intervals could obscure short but meaningful outages. To reduce the effects of measurement artifacts, we test $k = 1$ to 3 and find that larger k values do not significantly alter overall responsiveness (detailed in Appendix C). We therefore adopt $k = 1$ in our main analysis. A *responsive session* is a maximal sequence of consecutive responsive rounds. A *nonresponsive interval* is the time gap between two consecutive responsive sessions. We define a *responsive session* as a maximal sequence of consecutive responsive rounds within each window. A *nonresponsive interval* is the time gap between two consecutive responsive sessions.

3.3 Cast Propagation Data

Recall, a cast refers to a social post that is shared via Farcaster. When a cast is injected by a user into a specific hub, it is propagated to the remaining hubs via GossipSub and Diff Sync. To measure propagation latency, we perform active experiments by injecting uniquely identifiable casts into all known hubs and monitoring the remaining reachable hubs to record when each receives the injected cast. Algorithm 1 illustrates the measurement steps in detail. Specifically, we use three user accounts, each with an on-chain FID registered over three months prior,[5] to post on Farcaster's network. The measurements are conducted across all accessible hubs in the system and repeated every 24 hours.

We randomly assign each hub to one of the above three FIDs (resulting in three subsets of hubs). For each hub, we then generate and inject a new *unique* cast (details in Appendix B) using the FID

[3] Note, a message is stored within (and exchanged across) each hub, and can contain any data item, *e.g.* casts, reactions and user relationships.

[4] Note, the number of peer IDs and operator FIDs are higher than the number of hubs, because these can be changed over time.

[5] This is to make sure the credentials (*i.e.* secret keys of custody address) are well synchronized across hubs.

assigned to it. This ensures each hub receives a different message while keeping per-user storage within limits to avoid automatic pruning (see Appendix E). Once the hub has accepted the cast, we then monitor every other hub's GETCAST API to check if it has received the cast. To improve scalability, we employ an exponential backoff strategy (see Algorithm 1) for these queries; the polling interval starts at 1 second and increases with each attempt up to a maximum cap of 64 seconds. If not, we retry using exponential backoff (detailed in Appendix B), with a 300-second timeout per attempt, to balance timeliness with the desire not to burden hubs. We continue retrying until one of the following conditions is met: (1) the cast is successfully received; (2) 10 consecutive API requests are rejected, indicating the hub is offline; or (3) the total retry duration exceeds 24 hours. For each cast injected into a hub h_i, we record the timestamp t_{s_i} when the cast is submitted and confirmed by h_i, and the timestamp t_{r_j} when it is received by another hub h_j. We then define the hub-to-hub cast dissemination delay between two hubs h_i and h_j as $\Delta t_{ij} = t_{r_j} - t_{s_i}$. We repeat the above experiment 34 times, resulting in 28,116 unique casts being injected into 2,267 hubs.

Algorithm 1 Propagation Measurement Steps

```
Fetch all active Hubs
for hub h_i in Hubs do
    Submit unique cast c_i to h_i and record t_{s_i}
end for
for cast c_i in Casts do
    for hub h_j in Hubs where h_j ≠ h_i do
        retry ← 0
        while not received and retry ≤ 10 and elapsed < 24h and h_j online do
            Query h_j
            if received then
                Record reception time t_{r_j}; break
            end if
            Wait min(2^retry, 64) seconds
            retry ← retry + 1
        end while
        Compute Δt_ij = t_{r_j} − t_{s_i}
    end for
end for
```

3.4 Grafana Data

According to Farcaster's official deployment instructions, each hub can be deployed with *Grafana* [1] and *Graphite* [2], as specified in the reference configuration [31]. This dashboard exposes several useful performance metrics, which we gather. We connect to each known dashboard and crawl historical records every 5 days between February 10, 2025, and March 20, 2025.[6] For each dashboard, we collect retrospective time-series data at a 10 minute resolution[7] by retrieving all available metrics. The earliest observable data goes back to March 11, 2024, while over 50 % of hubs provide historical records starting from at least January 14, 2025. We obtain the following information for each hub. (1) *Storage:* We retrieve the total disk size and message count of the Merkle Patricia Trie at each point in time. These metrics are used to assess storage resource utilization. By analyzing these values across hubs, we examine operational behavior under varying storage conditions. (2) *Communication:* We collect the number of peers (used to infer network topology), message throughput (measured in message count), and processing latencies. We use this data to analyze the dissemination efficiency and delay characteristics of the GossipSub and Diff Sync protocols. In total, our dataset covers 2,102 unique dashboards, comprising 907M average and 986M summed metric values per interval.[8]

3.5 On-chain Data

Farcaster's on-chain data is primarily stored and accessed on two Ethereum Layer 2 networks: Optimism (for registration and identity verification) and Base (for token transactions). We gather data across these two chains, detailed below.

Account Management. We retrieve all data related to smart contracts stored on Optimism by operating a full Optimism node. Multiple smart contracts deployed on the Optimism chain collaboratively handle account creation (*e.g.* FID generation), verification (*e.g.* user modifications of readable usernames and profiles), and authorization processes (*e.g.* users delegating message-sending permissions to clients for hub communication). Note, for each user, all contract invocations are bound to the user's FID, which is in turn bound to a custody address. We record all invocations by each user, alongside their FID and associated custody address, which are logged in transaction logs (where each transaction corresponds to a contract call event). This tracking covers: (1) All FID registration events; (2) Storage rental transactions; (3) Identity verification and authorization events; (4) Associated blockchain transaction costs (*i.e.*, gas fees); (5) Registration fees deposited into the *StorageRegistry* contract. These data points are used for analysis in § 6.1. In total, we gather 3,196,428 contract invocations.

Token Transactions. Within Farcaster's multi-chain ecosystem, user-bound Ethereum addresses maintain interoperability across all EVM-compatible chains (*e.g.* Base, Optimism, Polygon, and BSC). However, empirical data reveal the dominance of Base chain utilization in Farcaster: (1) all top-ten tokens by daily transaction volume originate from Base chain deployments [48]; (2) Base chain transactions constitute nearly 90 % of total cross-chain activity among Farcaster users [53]; and (3) Farcaster's native reward mechanism exclusively employs Base-chain USDC for weekly distributions to qualified users and developers [24]. This establishes Base as the de facto standard for Farcaster's transactional layer. Therefore, we utilize the FID-to-wallet mapping data from § 3.1, combined with the Alchemy API [4], to obtain historical transfer records of users' bound wallets on the Base chain. This allows us to measure the inflows and outflows generated by different users and hub operators in § 6.2. As of May 31, 2025, we have collected a total of 87,687,791 transaction records, covering 662,006 Ethereum wallet addresses associated with 489,824 distinct FIDs.

[6] We do this every 5 days to account for new hubs joining the network.

[7] This is to align with the chosen resolution for measuring hub responsiveness.

[8] For each dashboard, metrics are sampled at 10-minute resolution. The *average* value is the mean within each interval, whereas the *sum* represents the total accumulated metric within that interval.

3.6 Limitations

We wish to flag three key data limitations. First, our data collection is limited to publicly accessible hubs and Grafana instances. This potentially introduces a selection bias, as private hubs are excluded. Consequently, our findings only reflect the public portion of the Farcaster network. While private hubs may show different stability or propagation patterns, the public portion remains the most important component. Second, the cast propagation measurements employ exponential backoff to minimize network disturbance (see Appendix B). Instead of continuously probing hubs, our system periodically increases the interval between requests when no updates are detected. This approach reduces network load, but it also lowers temporal resolution. We would be keen to increase the resolution of these checks (*e.g.* to every 10 seconds), but chose not to due to the load impact it would have on hubs. Finally, our findings of course capture a system during a particular period, and may not represent the entire network configuration.

4 Availability

A core feature in Farcaster is the independently operated hubs that distribute full replicas, thereby offering high redundancy and availability. We evaluate the goal of high *availability*, by measuring the hub's responsiveness and deployment patterns.

4.1 Responsiveness Across Time

Responsive Session Duration. We begin by analyzing the *responsive duration* for all monitored hubs. This is defined as the continuous period of time during which we can elicit a response from each hub. Note, hubs may not reply because they are offline or simply overloaded. Figure 1a presents both the overall and per-hub average distributions of responsive sessions. The majority of hubs exhibit short and unstable periods of continuous responsiveness: 80 % of hubs remain responsive for less than 1.68 hours, on average. In contrast, only a small minority (3 %) exhibit significantly longer and more stable responsiveness, with average sessions exceeding 24 hours. However, even among these stable hubs, only 13 % maintain low variability of session durations (coefficient of variation (CV) < 0.3 [58]), indicating unpredictable durations. This indicates that the network is disproportionately sustained by a very small subset (4 ‰) of stable hubs, while the broader Farcaster infrastructure remains vulnerable to churn.

Non-responsive Intervals. The above leads us to analyze the intervals for which hubs remain unresponsive. Figure 1b presents the ECDFs of downtime durations between consecutive responsive sessions. We plot all intervals, alongside the per-hub average. 80 % of non-responsive intervals are shorter than 30 minutes, suggesting that most hubs alternate between responsive and non-responsive states within short periods (under 2 hours). Indeed, over the measurement period, hubs experience a high median of 782 distinct sessions, with the most unstable 20 % experiencing at least 1,463 sessions. This frequent oscillation confirms the lack of consistent availability. For end users selecting those hubs, such unresponsiveness directly translates into delayed or missing messages.

4.2 Per-Hub Message Replication

We next examine the availability of user-generated messages on each hub. Although Farcaster aims to fully replicate messages across all hubs, we conjecture that the above responsiveness issues and operational overhead may create problems [57], causing them to lag behind the global state. Thus, we analyze the message count recorded in each hub to assess the actual availability of messages across the network.

Message Volume. We begin by looking at the number of messages stored across all hubs (from the Grafana of each hub). Daily mean message count across all the hubs grows from 559M on October 1, 2024 to 637M on March 21, 2025 (≈ 457K new messages per day). However, the gap between the 25th and 50th percentiles (≈ 12M) consistently exceeds that between the 50th and 75th (≈ 5M),[9] indicating that intended global state replication is not yet fully effective. Motivated by this, we further investigate whether certain hubs consistently fall behind in synchronization.

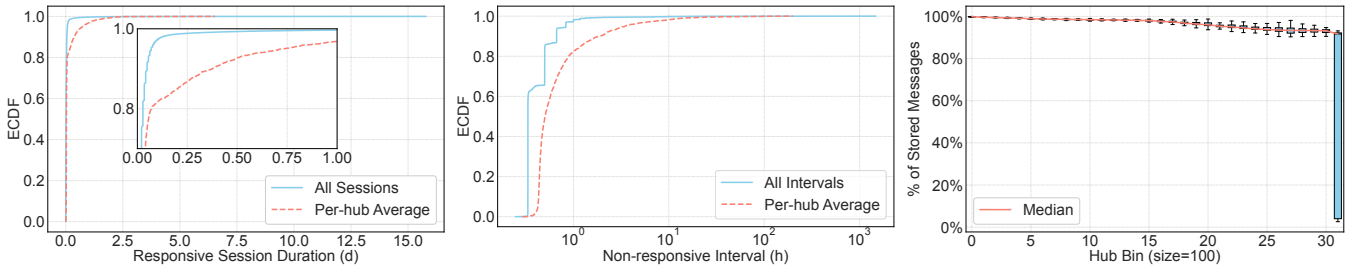
Per-Hub Differences. To quantify storage disparities, we calculate the percentage of messages stored by each hub relative to the hub with the global highest message count in the network.[10] Figure 1c plots these percentages per hub. The x-axis sorts hubs by their average message count ranking,[11] while the y-axis shows the percentage of messages stored by all hubs in each bin. Hubs in the top 100 maintain nearly identical message volumes, *i.e.* median coverage > 99 %. Note, on average, a 1 % difference corresponds to approximately 6.6M messages over the measurement period. Across the top 75 % of hubs (rank ≤ 2300), message coverage remains relatively high (median > 95 %). However, the bottom 100 hubs show wide disparities in message coverage, ranging from 4 % at the 25th percentile to 92 % at the 75th percentile. On average, the top-100 hubs store 1.45× more messages than the bottom-100. This suggests that these bottom hubs may be cold-starting or recovering from an out-of-date snapshot, resulting in incomplete replication. These patterns align with our earlier percentile-based findings on message volume: persistent disparities exist in message replication, with the same bottom 25 % of hubs consistently lagging behind the network. Such behavior raises concerns about global state consistency and the availability of messages across the network.

4.3 Hub Centralization

Prior decentralized architectures often experience tacit centralization [55]. Here, we ask if Farcaster faces similar pressures.

Message Sources. We begin by examining whether centralization emerges in terms of the messages received by each hub. Farcaster relies on three protocols for dissemination. The first is gRPC where client applications (*i.e.* user-facing interfaces) submit user-generated messages directly to a hub via gRPC. Once a hub receives a new client message, it replicates the data to other hubs using either GossipSub or Diff Sync. GossipSub disseminates messages through a probabilistic gossip mechanism, whereas Diff Sync serves as a deterministic fallback protocol that reconciles missing data between hub

[9] Hub with the highest message count is ranked at the 100th percentile.
[10] The message counts are derived from hub metadata described in § 3.2.
[11] Each hub is ranked at every 10-minute interval based on its message count in descending order (*i.e.* the hub with the most messages ranks first), and the ranks are averaged over time.



(a) **(b)** **(c)**

Figure 1: (a) ECDF of responsive session durations (unit: day); (b) ECDF of non-responsive intervals (unit: hour). Both (a) and (b) show overall and per-hub averages. (c) Percentage of messages stored relative to the hub with the highest message count. Hubs are sorted by average message count ranking and grouped into bins of 100.

pairs but incurs substantially higher overhead. To understand which sources dominate hub-level traffic, we analyze message contributions using metrics extracted from Grafana (see § 3.4).

GossipSub is the dominant dissemination mechanism, accounting for 95 % of the total traffic, measured by message count, across all observed hubs in the network. By contrast, gRPC is responsible for just 0.1 %. The top 20 % of hubs receive 91 % of all gRPC client message injections, likely driven by pre-configuration in popular apps. This clearly raises centralization concerns, as the 20 % of "write" hubs play a disproportionately important role in the system. Despite the high proportion of GossipSub traffic, 46 % of hubs exhibit a success rate below 80 %. Here, the message success rate is the fraction of received GossipSub casts that are successfully merged into a hub's local Merkle Patricia Trie. This is caused by the very high number of redundant messages that GossipSub disseminates: over 90 % of failures are caused by message duplications (*i.e.* a hub receiving the same message multiple times). This generates large volumes of redundant traffic and imposes extra load on the fallback reconciliation mechanism (*i.e.* Diff Sync), which we further analyze in § 5.2.

Inter-Hub Connectivity. Next, we explore the number of P2P inbound connections received by each hub. This is a proxy for centralization, as hubs that receive many requests serve a disproportionate role in the network. Figure 4a plots the average inbound Diff Sync requests vs. the average inbound GossipSub peer number per-hub. We observe a strong positive correlation between these two metrics ($P < 0.001$, $r = 0.88$). We conjecture that this relationship is partly driven by the number of messages stored on each hub, *i.e.* for hubs with more messages, a greater number of peers are motivated to initiate GossipSub connections. Indeed, the top-10 hubs have an average of 40 inbound connections, compared to 17 for the bottom-100. Moreover, hubs are more likely to perform Diff Sync with peers that possess a larger volume of messages. This behavior stems from Diff Sync's threshold policy, whereby a hub only initiates Diff Sync with a peer that has at least 0.95× of its own message count. Overall, 20 % of hubs possess 50 % of all GossipSub inbound peers and account for 47 % of the Diff Sync inbound requests. Therefore, user experience may increasingly depend on these hubs, which reduce resilience.

Hub Hosting. Another source of centralization arises from the physical co-location of infrastructure. To study this, Figure 2 plots the distribution of the top five hosting autonomous systems (ASes) and countries. We find that more than 50 % of unique IPs are hosted by a single AS, Contabo GmbH, while each of the remaining ASes contributes less than 20 %. This skew is likely driven by cost economic factors, as users often report concerns about the cost of running a hub (see Appendix G). To estimate this, we take the top five hosting providers used by Farcaster hub operators, as well as three providers (Google, Digital Ocean and AWS) introduced in Farcaster's official tutorial and developer forum [20, 22]. We then extract the minimum system requirements from Farcaster's official documentation [21] (see Table 1), and map it to the cheapest configuration offered by these providers, as detailed in Table 2.

Table 1: Hub Minimum Requirements

Requirement	Details
Memory	16 GB
CPU	4 Cores
Storage	1 TB
Network	Public IPs on ports 2281–2283
RPC Endpoints	Ethereum and Optimism

This table suggests that costs are, indeed, a factor in hub operator decisions. For instance, Contabo offers VPS plans at $260 per year, about 13 % of the cost of AWS, likely making it a more attractive choice. Community guides further reinforce this preference. Deployment guides [9, 16, 62, 73] often recommend Contabo, leading new users to adopt the same provider. This pattern also drives our country-level observations. More than 50 % of hub IP addresses are located in Germany, largely due to the presence of Contabo GmbH and Hetzner GmbH in that region. Such co-location at both network and geographic levels introduces additional centralization risks and increases the impact of regional outages.

Takehomes

1) Hubs have repeated periods of non-responsiveness. Only a small minority (3 %) exhibit good continuous responsiveness (>24 hours). 2) Farcaster has high levels of replication, but with disparities across hubs. On average, the top 100 hubs hold 1.45× as many

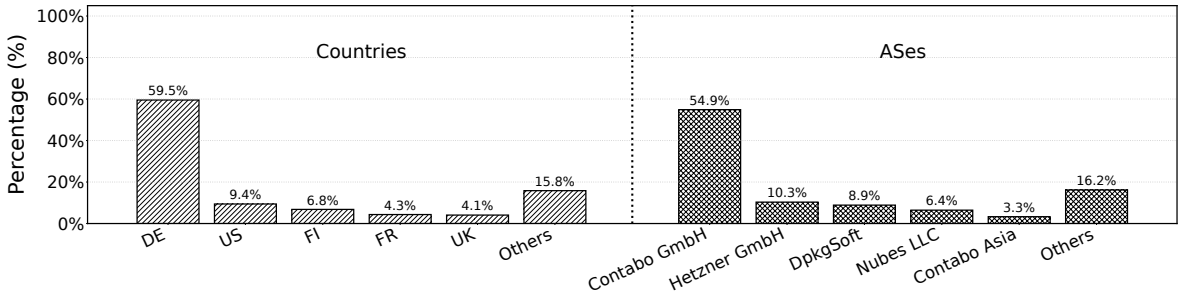


Figure 2: Distribution of Autonomous Systems and geolocations where hubs are hosted.

Table 2: Hub Operating Costs

AS	Package	Annual Cost (USD)
Contabo GmbH/Asia	VPS 30	$260
Hetzner GmbH	CPX42	$261
DpkgSoft	R9-16	$293
Nubes LLC	a3s.8c16g	$612
Google Cloud	e2	$1,354
Digital Ocean	Droplet	$1,512
Amazon AWS	m5.xlarge	$1,972

Note: Costs are estimated based on minimum requirements and reflect a snapshot as of Sept 2025; actual prices may vary over time.

messages as the bottom 100. 3) The network exhibits high topological centralization, with 20 % of hubs accounting for > 90% of user message submissions and half of all inbound GossipSub connections and Diff Sync requests. 4) Farcaster experiences physical centralization of hubs, with over half based in one AS.

5 Performance

Another key innovation introduced by Farcaster is its hybrid dissemination architecture, which combines a low-latency gossip layer with a point-to-point differential synchronization mechanism. While the general pattern of pairing best-effort broadcast with slower state-repair protocols is common in distributed systems [7, 15, 39, 52], its realization using *GossipSub* and *Diff Sync* is unique. This hybrid model is designed to ensure both fast propagation and eventual consistency across hubs. Thus, we evaluate its performance from two perspectives: (1) the propagation delay of casts across the hubs; and (2) the respective efficacy of the GossipSub and Diff Sync protocols at attaining this propagation. We analyze these performance aspects in the context of the availability challenges identified earlier.

5.1 Cast Propagation Performance

We begin by measuring the propagation delay of casts across hubs, following the steps detailed in § 3.3. We target all active hubs that are responsive and able to receive our submitted casts.

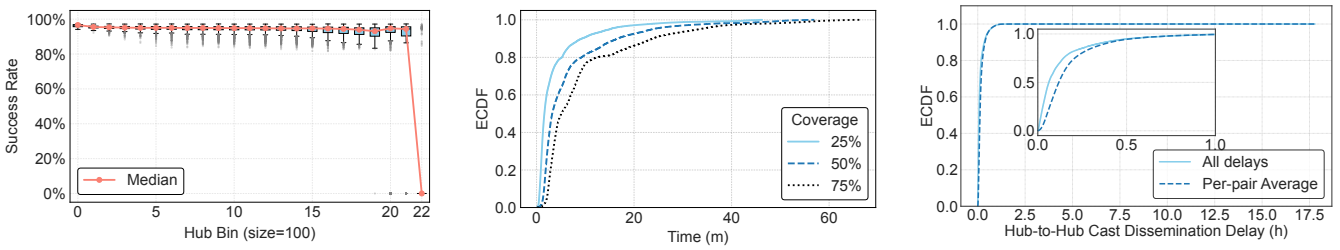
Successful Reach. To measure the ability of hubs to globally replicate data, we inspect the percentage of hubs that receive the casts we inject into the network. For each injected cast, we define its *success rate* as the percentage of other hubs that receive it within 24 hours.

Figure 3a presents the distribution of success rates. The x-axis plots the hubs that we inject casts to, sorted by their average success rate and grouped into bins of 100. The y-axis plots the corresponding success rates of the hubs across the measurement period. We see that not all casts reach the full network within 24 hours: on average, each hub's casts reach just 92 % of the other hubs across the measurement period.[12] In the most extreme case, we find that 1.4 % of casts fail to propagate to *any* other hub, despite being accepted by the origin hub we inject the message into. These numbers contrast sharply with the expectations of typical user-facing applications, *e.g.* X expects 99.99 % reliability [12].

Cast Coverage Latency. We further examine the time required for a cast to reach different shares of the remaining hubs (referred to as *coverage latency*). This directly impacts the message availability of decentralized systems. Thus, we compute the time it takes for each cast to reach 25 %, 50 %, and 75 % of the remaining hubs. Figure 3b plots the distribution of the time required for each cast to reach 25 %, 50 %, and 75 % of the remaining hubs from the injecting hubs. For those successfully delivered casts, coverage latency varies substantially. Focusing on 50 % coverage, 80 % of casts reach half of the hubs within 9.2 minutes, whereas the remaining 20 % require up to 57 minutes (×6.2 slower). Interestingly, the coverage latencies vary greatly by the hub from which the cast is sent: only 12 % of hubs show low variability in their 50 % coverage times (measured by CV $\leq$ 0.3 [58]), suggesting that most hubs experience unstable dissemination performance across rounds. This disparity highlights the unpredictable pace of propagation across the network.

Hub-to-hub Propagation Delay. One possible explanation for the observed disparities and instability is that certain hubs (used for injection) experience higher fluctuations and delays to other specific hubs. To better understand this behavior, we analyze pairwise hub-to-hub cast propagation delays across the network (see § 3.3). For each hub that we inject a cast to, we measure the time it takes for the cast to reach each other hub. Figure 3c presents the cast propagation delays. We plot both the per-cast distribution and the per-hub distribution

[12]We compute the CV for each hub's per-cast success rates (based on its success at disseminating its casts within 24 hours). 84 % of hubs have a CV below 0.2, suggesting low variability of success rate across experimental rounds.



(a) (b) (c)

Figure 3: Message propagation performance across hubs: (a) Boxplot of message receipt success rate by hub bin (size=100); (b) ECDFs of cast coverage latency, showing how long casts take to reach different shares of the remaining hubs from their injecting hubs; and (c) ECDFs of cast propagation delay Δ_{ij} measured between injected hubs h_i and receiving hubs h_j.

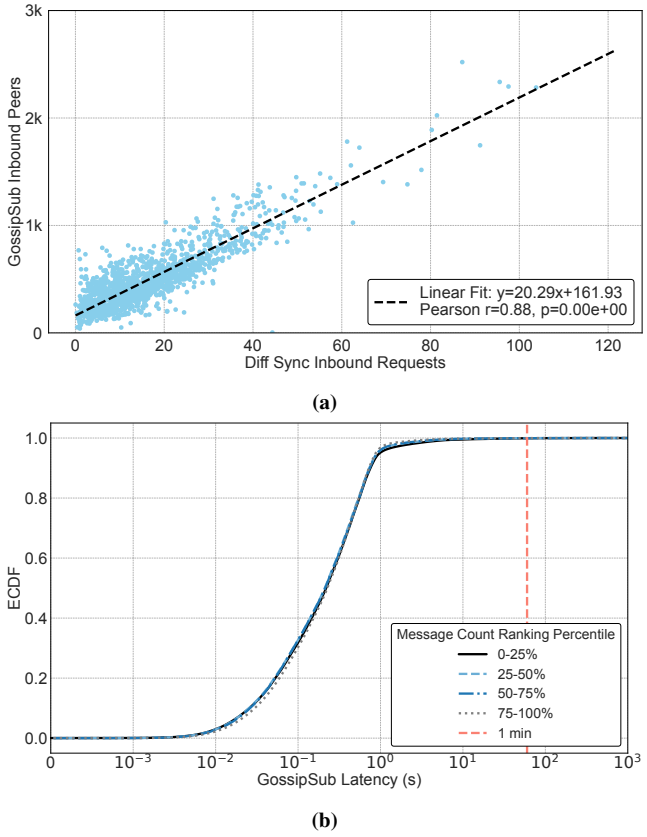
(based on the average taken between all pairs of hubs). Echoing the cast-level variance, only 22 % of the CV for these delays per hub pairs remains below 0.3. Moreover, the 75th of CVs reaches 1.12 (3× of 25th), highlighting unstable dissemination behavior in a subset of hub pairs. Averaged over hub pairs across repeated rounds, 80 % of pairs synchronize within 15 minutes. However, the distribution exhibits a pronounced long tail: the remaining 20 % require up to 17.5 hours for cast delivery. These observations indicate that cast propagation is unstable and highly dependent on the propagation path (*i.e.* the submitting and receiving hubs). We conjecture that this behavior is rooted in the underlying dissemination protocol, which we further examine in § 5.2.

We then investigate whether the hubs that are slow to disseminate their casts to others also experience longer delays as recipients of casts. For that, we compute the Pearson correlation between each hub's average reception delay vs. the average time required by casts injected into the hub to reach 25 %, 50 % and 75 % of the remaining hubs. We find strong positive correlations at all thresholds ($p < 0.001$): $r = 0.64$, 0.68, and 0.70 for 25 %, 50 %, and 75 % coverage, respectively. This suggests that the problem is indeed based on the individual hub, confirming that slower hubs not only lag in message receipt but also hinder propagation performance when they serve as the initial injection point for casts. Users connected to such hubs face systemic disadvantages: their casts arrive later and disseminate more slowly across the network. This observation motivates a deeper investigation into protocol-level performance (see § 5.2).

5.2 Protocol Performance

Next, we aim to uncover potential causes of the poor data dissemination performance. Farcaster uses GossipSub to broadcast casts as they arrive in the system (received from the clients). It then falls back to Diff Sync to recover casts lost by GossipSub. Thus, we break down the performance based on which protocol is used to receive the message.

GossipSub Latency. First, we calculate the average time required by each hub to receive casts over GossipSub (see § 3.4 for methodology).[13] Here, we only consider messages that are successfully



(a)

(b)

Figure 4: (a) The average number of inbound Diff Sync requests against inbound GossipSub peer count per hub. (b) ECDFs of the GossipSub latency for successful message dissemination. Hubs are ordered by their average message count ranking (*i.e.* hubs in 75 %-100 % percentiles have more messages).

received and merged into the local Merkle Patricia Trie.[14] Figure 4b

[13] Note, this analysis differs from § 5.1, which aggregates delays across all protocols instead of isolating GossipSub.

[14] GossipSub circulates many duplicate messages which may have already been merged into a hub's data stores.

depicts GossipSub latencies across the measurement period.[15] Surprisingly, successfully disseminated messages exhibit consistently low delay: 99 % of per-hub average latencies fall below one minute. This observation contrasts sharply with our earlier findings in § 5.1, where 20 % of hub pairs experienced mean propagation delays exceeding 15 minutes and extending up to 17.5 hours with high variability. These results confirm that GossipSub is effective in quickly spreading new messages. This is intuitive, as hubs maintain an average of 13.9 GossipSub outbound connections, suggesting that messages could (theoretically) reach the entire network within 3 hops. We emphasize that this comes at a cost though: the majority of GossipSub messages are redundant with many duplicates forwarded, thereby resulting in large volumes of unnecessary traffic (see § 4.3).

Diff Sync Completion Time. Given the high performance of GossipSub, we next examine the efficacy of Diff Sync to help explain the high propagation delay observed earlier (§ 5.1). Each hub initiates a Diff Sync session with a single peer every 2 hours, and each session can last up to 110 minutes, as enforced by the implementation. Figure 5a presents the ECDFs of all Diff Sync sessions observed per hub, based on metrics collected from Grafana (see § 3.4). We divide hubs into four quartiles based on their average message count ranking, computed by averaging their ranks across 10-minute intervals. The top quartile (75 %–100 %) includes hubs with the highest average message counts, indicating stronger replication performance. We observe that 40.4 % of hubs take more than 60 minutes on average to complete a single Diff Sync session. Interestingly, performance degrades for low-ranking hubs: only 41 % of sessions for hubs in the bottom 0–25 percentile finish before the cut off threshold. Since a cast may require multiple rounds of Diff Sync to fully disseminate, this confirms that the Diff Sync is a major factor behind the high and fluctuated delay observed in § 5.1. This motivates a deeper investigation into how message count ranking, used as a proxy for replication performance, influences Diff Sync efficiency.

Explaining Diff Sync Performance. Finally, we analyze Diff Sync efficacy metrics from the public Grafana dashboards to investigate the causes of the observed disparity across hubs. We hypothesize that this disparity stems from differing Diff Sync workloads. To evaluate this, we compute the pairwise local-to-peer relative message count ratio for each Diff Sync pair, defined as the initiating hub's message count divided by that of its peer. For instance, if a hub with 100 messages initiates syncing with a peer holding 200, the ratio is 50 %. This metric serves as an estimation of the volume of messages requiring reconciliation during Diff Sync. Figure 5b shows the distribution of these ratios per hub against their message count ranking. We find that top-ranking hubs typically have over 99 % message parity with their Diff Sync peers, indicating near-identical message volumes and therefore low complexity synchronization. In contrast, lower-ranked (> 1,600) hubs show a drop, with average ratios falling to 91 %. While these figures may seem satisfactory, they correspond to ≈ 59.4M missing messages (as estimated from § 4.2). These findings suggest that even small differences in message volume (*e.g.*, a 6 %–9 % deficit) can impose a substantial workload on Diff Sync. Despite relatively high parity ratios, the low efficacy of Diff Sync likely stems from its structural inefficiency: because historical messages are scattered across leaf nodes in the Merkle Patricia Trie, reconciling even minor discrepancies may require traversing a large number of subtrees [25].

[15] Latencies are sampled and averaged at 10-minute intervals across the measurement periods (see § 3.4 for methodology).

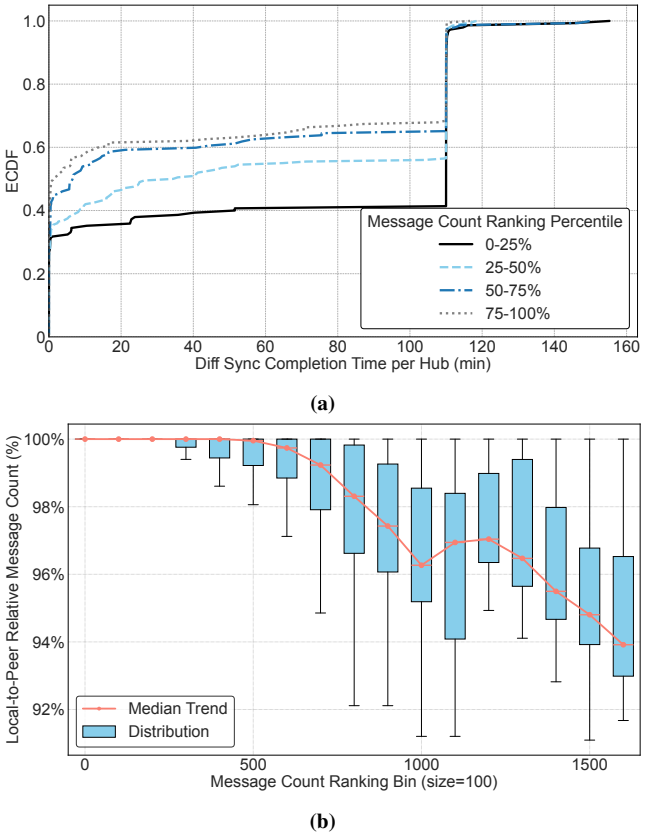


Figure 5: (a) ECDF of Diff Sync session durations. Hubs are ordered by their average message count ranking (*i.e.* hubs in 75 %-100 % have the most messages); (b) Local-to-Peer Message Coverage Ratio: Hub's message count as a percentage of its peers during Diff Sync.

Takehomes

1) The network exhibits significant propagation time differences based on where casts are injected: 80 % of casts reach 50 % of hubs within 9.2 minutes, while the remaining 20 % take up to 57 minutes. Moreover, hubs that disseminate casts slowly also tend to receive them with higher delays. 2) GossipSub is effective at ensuring timely delivery, with 99 % of successful dissemination latencies below one minute. However, its probabilistic delivery model places pressure on fallback reconciliation (Diff Sync). 3) Diff Sync is significantly slower and more variable: 40.4 % of hubs require over 60 minutes per session, on average. Only 41 % of sessions for the bottom 25 % of hubs complete before the 110-minute cutoff threshold. This is caused by the inefficient design of the set reconciliation procedure rather than fundamental issues with the dissemination pattern. This is very problematic, as Diff Sync is responsible for delivering 5 % of messages.

6 Sustainability

Another important goal is to incentivize active and sustained participation in both maintaining hub infrastructure and contributing to the social layer of the network. Accordingly, we next evaluate key sustainability factors in Farcaster.

6.1 Exploring User Account Sustainability

To participate in the network and mitigate spam, users are required to rent *storage units*, each granting a one-year quota for storing messages (*e.g.* casts, reactions, etc.) replicated across all hubs. The specific limits per message type are listed in Appendix E. This storage rental fee is initially paid during the user registration process, can be optionally renewed each year, and is deposited into a protocol-managed smart contract on Optimism [27]. This payment-based storage model raises two concerns: (1) users may overpay for underutilized storage; and (2) expired leases can result in unsustainable and irreversible data loss.

User Registration Costs. We first assess the fees associated with the user registration process. The fee covers the rent for the storage unit, which is paid into the *StorageRegistry* contract, and the gas costs incurred during all contract interactions (*e.g.* registering user identity and signing keys). Farcaster has adjusted the client registration fee four times, ranging from $2 to $7 (see Appendix E) and offers two onboarding methods: (1) Tech-savvy users can interact directly with contracts, incurring gas and storage fees; (2) Regular users make a one-time fiat payment via a client app, which subsequently converts the fiat currency to ETH and manages all blockchain interactions on their behalf. By May 15, 2025, a total of 808 ETH (≈ $2.5M) had been deposited into the *StorageRegistry* contract, with 66.5 ETH (≈ $182K)[16] in cumulative gas fees paid for all smart contract interactions.

In the registration process, storage fees constitute 93 % of user expenditure (≈ $2.8 per user), while gas fees account for only 7 % (≈ $0.2 per user). This breakdown highlights a key sustainability benefit of Farcaster's hybrid on-chain/off-chain architecture: only a small fraction of user activity incurs on-chain costs, in contrast to prior blockchain-based social networks that store a majority of social data on-chain [47, 71]. Yet, the storage fees that are deposited into the *StorageRegistry* contract have so far remained unused, and Farcaster has no defined plans for the use of these funds.[17] Although storage fees could support hub operators who store and disseminate user data, this has not occurred in practice, raising questions regarding the financing of hub operations. Further, replicating data across all hubs raises questions regarding how such fees could be split.

Storage Activity Status. We next explore the user rental behavior in detail. Here, we separate storage units into three categories: (1) **First Rental**: the initial storage purchase (*i.e.* the user rents one storage unit for the initial registration purpose); (2) **Expansion**: an additional storage unit purchase by existing users; (3) **Contraction**: a storage unit expiration, leading to data deletion. Figure 6a plots the number of storage units that fall into each category over time. The majority of purchases (93 %) correspond to first rentals, reflecting new user onboarding. We also observe a significant number of storage contraction events beginning in Oct 2024, peaking in Feb 2025. This pattern aligns with two key periods: (1) The initial permission-less user registrations starting in Oct 2023; (2) The first major wave of user growth in February 2024. In 90 % of contractions, the user is left with no units (as they only owned a single unit). This is concerning as such users would lose all data storage. As of March 31, 2025, a total of 258,338 users (29 %) had reached zero storage units. Notably, 92 of these users later renewed their storage only after expiration. We conjecture that these users unintentionally allowed their storage to lapse, resulting in irreversible message purging. This raises concerns about long-term content availability.

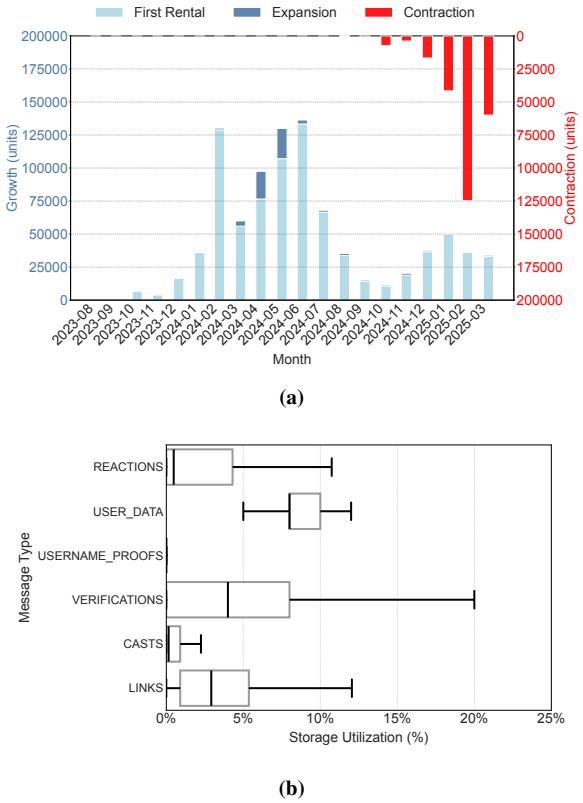


Figure 6: Storage Behavior: (a) Trends in storage rental activities: first-time rentals, storage expansions, and contractions; and (b) Utilization rate by message type.

Storage Utilization Rate. Given the observed annual contraction of user storage, we next inspect how users utilize their storage units. Since 7 November 2023, annual quotas are set at 5,000 *Casts*, 2,500 *Links* (*i.e.* Follows), 2,500 *Reactions* (*i.e.* Likes), 50 *User Profile Updates*, 25 *User Verifications*, and 5 *Username Proofs*. On 28 August 2024, after the registration fee dropped from $3 to $2, quotas for *Casts*, *Links*, and *Reactions* were cut to 2,000, 1,000, and 1,000, respectively; the other quotas remained unchanged (detailed in Appendix E). To assess usage, we compute the storage utilization rate for each FID as the ratio of stored messages to the corresponding

[16] Fiat is estimated by USD; exchange rate is based on transaction time.
[17] https://warpcast.com/dwr.eth/0xba528e14

type-specific quota, using a snapshot taken on May 11, 2025. Figure 6b displays storage utilization rates by message type. We observe that over 89 % of users consume under 20 % of their allocated capacity (regardless of message type). Utilization varies significantly by message type: Verifications show higher usage (median: 4 %), while Casts are the most underutilized (median: 0.1 %). A small subset (1 % of users) fully exhaust their storage quota for Links (*i.e.* Follows), whereas the remaining 99 % exhibit low usage with a median consumption of only 3 %. This indicates that the 1 % of users must purchase additional storage units to expand their follow graphs, while their unused quota for other message types becomes wasted capacity. Moreover, while expiring the oldest messages is a reasonable strategy for posts (as their practical value perhaps decays over time), this approach seems unsuitable for Links (*i.e.* Follows) and Reactions (*i.e.* Likes). Nevertheless, Farcaster's enforcement of fixed storage thresholds per message type compounds challenges within its storage model.

6.2 Exploring Hub Sustainability

Finally, we analyze the operational costs of running a hub and potential revenue streams (other than storage fees) for hub operators to evaluate operational sustainability.

Hub Operating Costs. Operating a hub incurs non-trivial infrastructure costs. We focus our cost analysis on the providers that dominate both observed deployments and official recommendations. Specifically, the configurations listed in Table 2 account for 88 % of all observed hubs in our dataset. Although alternative, lower-cost setups may exist (*e.g.* self-hosted servers), we exclude them because they are not widely adopted. Therefore, the providers we analyze represent a representative cost spectrum grounded in the actual deployment patterns. Additionally, we exclude fees for Ethereum and Optimism RPC endpoints, which is justified by Alchemy's free tier being sufficient to meet the hub's setup requirements [5, 22].

As discussed earlier, the lowest annual fee charged by Contabo GmbH ($260) is nearly one-seventh that of the most expensive option AWS ($1,972). At an annual cost of $260 per hub, $2.5 million in storage revenue could theoretically subsidize the operation of nearly 9,600 Contabo hubs. Even if an AWS plan were used, this revenue stream could still sustain the operation of approximately 1,300 hubs. Nevertheless, Farcaster explicitly rules out economic rewards for running hubs, instead encouraging deployment based on necessity or interest with self-sustainability through alternative revenue [29]. The co-founders justify this lack of direct incentives by citing "incentive farming" risks and absent reliable proof-of-service mechanisms [20]. They maintain that ≈ 100 independent hubs suffice for network health [20, 28]. This contrasts with July 2024's surge to 10,000 hubs, driven by reward expectations [28] (see discourse on hub incentives in Appendix G). As demonstrated in § 5, this growth introduces critical sustainability challenges linked to hub performance and network overhead. Because the protocol requires each hub to replicate the full state history and engage in intensive Diff Sync operations, low-performance nodes impose a disproportionate burden on the rest of the network. This design creates a negative feedback loop: the presence of underpowered hubs increases aggregate network-wide overhead, which in turn raises operating costs for all participants. This compromises the long-term economic viability of the ecosystem.

Hub Operator's Revenue. By analyzing posts related to hub operation in Farcaster, we identify several potential hub operator revenue streams (see Appendix G), including user tips, subscriptions, alongside purchases from clients, mini-apps [23], APIs or crypto-service development. Due to their public and transparent nature, the hub operators' blockchain wallets allow us to directly study this revenue, to investigate whether their Farcaster participation could fully support hub costs.

To isolate user-specific economic activities, we analyze transactions exclusively between Farcaster users (*i.e.* where both sender and receiver wallets are bound to FIDs). We further filter out all intra-FID transfers (transactions between wallets owned by the same FID). Our study period spans November 2023 through 2024, capturing Farcaster's inaugural year of permissionless operation, enabling annual cost-revenue comparisons across hubs. We focus on the four dominant traded currencies: *USDC*, *USDT*, *ETH*, and *DEGEN*, which collectively represent > 99.9 % of total transferred value among hub operators' inter-FID revenues. Given the substantial USD volatility of *ETH* and *DEGEN*, we apply USD conversion rates at the time of receipt by hub operators.

Surprisingly, we find that 97 % of hub operators have no income to their wallets, leaving only 116 hub operators (3.4 %) with positive earnings. To study the sustainability of hub operations, we benchmark these earnings against the annual costs of Contabo and AWS cloud configurations. These serve as the lower and upper bounds of hub operation costs (see Table 2). This reveals viability challenges: only 0.52 % (N=20) of operators generate ≥$260 in token revenue — the required minimum for covering basic Contabo costs. This proportion drops to 0.13 % (N=5) when considering the $1,972 required for AWS infrastructure. This reveals that most hubs fail to generate enough revenue from on-chain user transactions within the Farcaster ecosystem to cover their costs, highlighting significant sustainability concerns.

Takehomes

1) Farcaster allocates 93 % of user registration fees (≈ $2.5M) to Storage Fees, with the remainder (≈ $182K) covering on-chain transaction costs. 2) However, most users significantly underutilize their storage quotas, resulting in overpayment for unused capacity. 3) While Farcaster deliberately withholds $2.5M in Storage Fees from hub operators to disincentivize low-quality nodes, this may exacerbate sustainability challenges: the $260/year minimum hub operating cost exceeds the on-chain revenue of 99 % of operators. 4) Consequently, optimizing storage fee structures and re-evaluating potential subsidies (*e.g.* set up eligibility criteria for hub operators) is a clear area for improvement.

7 Related Work

Decentralized Social Networks (DSNs) are not a new concept. As early as 2009, PeerSon [10] leveraged a Distributed Hash Table (DHT) for decentralized content discovery and user lookups. Since, there have been several general approaches to building these DSNs.

Most common are federated platforms, such as Mastodon [49], distribute content across independent servers, enabling user autonomy but concentrating control within instance administrators [41, 56]. Purely on-chain solutions like memo.cash enforce micropayments per interaction to deter spam, yet face prohibitive costs and limited throughput [72]. Meanwhile, hybrid architectures (such as Farcaster) have emerged to balance decentralization with performance and cost efficiency. These try to combine off-chain content storage with on-chain identity [38, 41]. Early hybrid systems like Steemit [60] anchored social metadata on-chain while storing content externally, introducing token-based incentives to align operator and user interests [50]. These designs typically store minimal metadata (*e.g.* identity credentials, content hashes) on-chain via smart contracts, while delegating bulk content to InterPlanetary File System (IPFS), NoSQL databases, or P2P networks [13, 42, 69]. Research demonstrates that such separation can achieve 4–18× throughput gains compared to blockchain-only storage [69], with reported performance ranging from 2k Transactions Per Second (TPS) in consensus simulations [70] to 250k records/second in specialized deployments [13]. Advanced verification mechanisms, including zero-knowledge proofs and authenticated data structures, further reduce on-chain verification costs by up to 550× while maintaining trust guarantees [37].

Despite these advances, existing evaluations predominantly rely on simulations or small-scale prototypes rather than real-world deployments [36, 70]. Empirical studies of production platforms remain scarce [50, 72], while standardized benchmarks for latency, storage cost, privacy, and moderation effectiveness are absent [38].

Our work addresses these gaps by presenting the first large-scale measurement study of a hybrid decentralized social network combining blockchain-based identity with peer-to-peer content dissemination. Unlike prior simulation-driven or prototype-focused research, we analyze 240 million user-generated messages across 3k independent hubs in a production environment. We quantify the real-world trade-offs among availability (hub responsiveness, message replication disparities), performance (gossip-based versus synchronization-based dissemination), and sustainability (storage fee utilization, operator profitability).

8 Conclusion, Implications & Future Work

This paper has presented the first large-scale empirical study of *Farcaster*, a hybrid decentralized social network built on independently operated hubs. Our findings indicate that, although the hybrid architecture mitigates several limitations of prior designs, it introduces unexpected trade-offs.

Availability. Farcaster replicates the full body of user data across thousands of decentralized hubs, mitigating the impact of incidental failures or attacks on the overall system. However, the full replication model introduces unexpected availability challenges from other perspectives. We observe frequent oscillations in hub responsiveness, with only a small minority (3 %) maintaining continuous availability for more than 24 hours (§ 4.1). Hubs also exhibit persistent disparities in message volume, with the top 100 hubs storing on average 1.45× more messages than the bottom 100 (§ 4.2). For end users relying on less responsive or message-deficient hubs, this instability directly results in delayed or missing messages. Moreover, Farcaster demonstrates *physical* centralization, with over half of all hubs located within a single AS, and *topological* centralization, where 20 % of hubs account for half of all inbound GossipSub connections and Diff Sync requests, raising concerns about network resilience (§ 4.3).

To mitigate these challenges, hubs could first adopt fallback retrieval strategies. When a hub fails to locate a requested message locally, it may query peers with higher message availability and cache the retrieved content for future use. Although this mechanism introduces additional retrieval latency, it should improve overall data completeness and therefore reduce the likelihood of user-facing inconsistencies. Our measurements also indicate that the current load imbalance is dominated by a small number of top hubs receiving the majority of user traffic, while many others consume resources without effectively contributing. Overall availability could therefore be improved by implementing load-balancing mechanisms or enabling users to explicitly select their preferred hub. Such rebalancing could transform the high number of hubs from a potential liability into an operational advantage.

Performance. Farcaster confines highly secure but complex blockchain transactions to infrequent governance actions, while delegating frequent operations (*e.g.* posting) to lightweight P2P hubs. Such separation aims to improve performance, but introduces unevenness. Our measurements reveal substantial propagation time disparities depending on where casts are injected: 80 % of casts reach half of hubs within 9.2 minutes, whereas the remaining 20 % require up to 57 minutes (§ 5.1). Further analysis identifies the heavyweight Diff Sync protocol as the primary cause of long-tail delays. GossipSub remains effective for timely dissemination, with 99 % of latencies under one minute; however, hubs may fail to accept GossipSub messages if they are non-responsive. Slower or intermittently offline hubs must rely on Diff Sync for state repair and message retrieval, which is substantially more expensive: 40.4 % of hubs require over 60 minutes per session on average (§ 5.2). This asymmetry creates a reinforcing feedback loop in which hubs that disseminate casts slowly or are frequently unresponsive also experience significantly higher reception delays.

To address this, our results suggest that the reconciliation protocols warrant further optimization. A short-term solution is for hubs to adjust the Diff Sync concurrency dynamically: lower it when few messages are missing to reduce bandwidth use, and raise it when many are missing to speed up reconciliation. A long-term solution could focus on optimizing the underlying data structure (*i.e.* Merkle Patricia Trie) that Farcaster relies on. For example, recent work on rateless Invertible Bloom Lookup Tables [68] offers a promising way to reduce computation time by enabling hubs to incrementally reconcile missing messages without comparing full state.

Sustainability. Farcaster aims to achieve sustainability by integrating fees for storage rental and financial transactions that incentivize participation, such as tipping hub operators. Empirical observations indicate that most users under-utilize their storage quotas, resulting in overpayment for unused capacity: over 89 % of users consume less than 20 % of their allocated capacity, regardless of message type (§ 6.1). These payments have accumulated a substantial surplus (≈ 2.5 M), theoretically sufficient to sustain 1,200 AWS hubs or 9,600 Contabo hubs. Despite suggesting potential economic sustainability, the funds remain largely unutilized, leaving 99 % of hub

operators to cover operational costs independently (§ 6.2). This dynamic may unintentionally favor centralization, as only well-funded hubs can reliably persist over the long term. Accordingly, sustainability concerns arise for both users and hub operators: users pay for storage they rarely consume, and hub operators must maintain infrastructure to support long-term network operation.

To address these challenges, Farcaster could emphasize intelligent storage allocation and new incentivization. From the user perspective, instead of enforcing fixed storage thresholds per message type, the system could adopt a tiered grid allocation scheme. Users would rent a fixed number of storage grids at different priority levels, with each message type occupying grids proportional to its significance. Higher-priority grids would store user-edited or critical data, such as submitted casts and following relationships; this ensures these messages are less likely to be purged. Lower-priority grids could hold ephemeral content, which can be automatically removed when space is needed. From the hub operator's perspective, a reputation system based on historical message dissemination or availability could reward well-managed hubs using storage fees.

Future Work. Notably, the network has recently initiated a transition to *Snapchain* [30], a sharding-based architecture designed to mitigate the scaling challenges consistent with those highlighted in this study. Looking ahead, our work sets the stage for assessing this new architecture. We aim to measure *Snapchain* to evaluate whether the performance bottlenecks identified here are mitigated. Building on our implications, we will further translate our empirical findings into actionable protocol improvements. Specifically, this will initially focus on two critical areas: (1) investigating the security implications of Snapchain model, exploring lightweight cryptographic proofs (*e.g.* zk-STARKs [34]) that enable "light nodes" to verify the integrity of social actions without local access to the full global state, thereby mitigating the risk of data corruption in a decentralized, high-throughput environment; and (2) exploring alternative data sharding mechanisms and robust peer scoring algorithms that jointly optimize economic sustainability and data availability, thereby preventing "lazy" hubs from skipping shard verification while maintaining the high performance required for real-time social interactions. We expect that these efforts will improve the underlying data storage and better incentivize participation in the system.

Acknowledgments

This work is in part supported by Guangzhou Municipal Science and Technology Project (2024D03J0008), Guangdong Provincial Project (2023ZT10X009), the Guangzhou Science and Technology Bureau (2024A03J0684), the Guangzhou Municipal Science and Technology Project (2023A03J0011), the Guangdong provincial project (2023QN10X048), the Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007), and the 111 Center (No. D25008).

References

[1] Grafana documentation. https://grafana.com/docs/.

[2] Graphite documentation. https://graphiteapp.org/. Accessed: 2024-10-14.

[3] Mastodon blog. https://blog.joinmastodon.org/. Accessed: 2025-04-14.

[4] ALCHEMY. Alchemy official document. https://www.alchemy.com/docs/reference/transfers-api-quickstart.

[5] ALCHEMY. Alchemy's free tier for ethereum and optimism rpc endpoints. https://www.alchemy.com/pricing#what-are-compute-units-cu.

[6] BELSHE, M., PEON, R., AND THOMSON, M. Hypertext transfer protocol version 2 (http/2). Request for Comments: 7540, Internet Engineering Task Force, 2015.

[7] BENET, J. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* (2014).

[8] BISHOP, M. Hypertext transfer protocol version 3 (http/3). Request for Comments: 9114, Internet Engineering Task Force, 2022.

[9] BREIZHNODE. How to easily help decentralize farcaster with your node. https://medium.com/@breizh-node/how-to-easily-help-decentralize-farcaster-with-your-node-028d536843d6.

[10] BUCHEGGER, S., SCHIÖBERG, D., VU, L.-H., AND DATTA, A. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems* (New York, NY, USA, 2009), SNS '09, Association for Computing Machinery, p. 46–52.

[11] COLLECTIVE, O. Optimism bedrock: Technical specifications. Tech. rep., Optimism Foundation, 2023. Whitepaper.

[12] CORP., X. Api documentation. https://docs.x.com/x-api.

[13] DAS, A. K. Swarmed: A high-throughput interoperability architecture over ethereum and swarm for big biomedical data. *International Journal of Computer Science and Information Technology 14*, 4 (2022), 19–34.

[14] DE OCÁRIZ BORDE, H. S. An overview of trees in blockchain technology: merkle trees and merkle patricia tries. *University of Cambridge: Cambridge, UK* (2022).

[15] DECANDIA, G., HASTORUN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P., AND VOGELS, W. Dynamo: Amazon's highly available key-value store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles* (2007), ACM, pp. 205–220.

[16] DEPINSPIRATIONHUB. Farcaster deployment tutorials. https://www.youtube.com/watch?v=b73C4q2You4.

[17] DITTRICH, D., AND KENNEALLY, E. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. Tech. rep., U.S. Department of Homeland Security, August 2012.

[18] ETHEREUM. Ethereum whitepaper. https://ethereum.org/en/whitepaper/.

[19] FARCASTER. Api instruction. https://github.com/farcasterxyz/hub-monorepo/tree/main/packages/hub-web.

[20] FARCASTER. Discussion on hub costs and rewards. https://hackmd.io/@farcasterxyz/ry0QL4M4o#Hub-Costs.

[21] FARCASTER. Farcaster hub installation configuration. https://docs.farcaster.xyz/hubble/install.

[22] FARCASTER. Farcaster hub setup tuturial. https://docs.farcaster.xyz/hubble/tutorials.

[23] FARCASTER. Farcaster miniapp document. https://miniapps.farcaster.xyz/.

[24] FARCASTER. Farcaster offical usdc reward. https://warpcast.notion.site/Warpcast-Rewards-15f6a6c0c10180339fcdee6c9a18c338.

[25] FARCASTER. Farcaster official document. https://docs.farcaster.xyz/.

[26] FARCASTER. Farcaster official http documentation. https://docs.farcaster.xyz/reference/hubble/httpapi/httpapi.

[27] FARCASTER. Farcaster storage registry contract. https://optimistic.etherscan.io/address/0x00000000fcce7f938e7ae6d3c335bd6a1a7c593d.

[28] FARCASTER. Opposition to hub rewards. https://farcaster.xyz/dwr.eth/0x2d7823b.

[29] FARCASTER. Rejection of hub rewards. https://farcaster.xyz/dwr.eth/0xe4780508.

[30] FARCASTER. Snapchain official document. https://snapchain.farcaster.xyz/.

[31] FARCASTER. Farcaster official documentation - hub monorepo. https://github.com/farcasterxyz/hub-monorepo, 2023.

[32] FARCASTER. Diff-sync description. https://docs.farcaster.xyz/learn/architecture/hubs, 2024.

[33] GILLESPIE, T. *Custodians of the Internet: Platforms, Content Moderation, and the Hidden Decisions That Shape Social Media*. Yale University Press, 2018.

[34] GONG, Y., JIN, Y., LI, Y., LIU, Z., AND ZHU, Z. Analysis and comparison of the main zero-knowledge proof scheme. In *2022 International Conference on Big Data, Information and Computer Network (BDICN)* (2022), pp. 366–372.

[35] GOOGLE. grpc: A high performance, open source universal rpc framework. https://grpc.io. Accessed: 2023-10-21.

[36] GOWDA, N. R., AND VENKATESH. BDOSN: Privacy-aware blockchain based decentralized OSNs. In *International Conference on Recent Advances and Innovations in Engineering* (2022).

[37] GU, B., AND NAWAB, F. zk-oracle: Trusted off-chain compute and storage for decentralized applications. *Distributed and Parallel Databases 42*, 4 (2024), 525–548.

[38] HISSEINE, M. A., CHEN, D., AND YANG, X. The application of blockchain in social media: A systematic literature review. *Applied Sciences 12*, 13 (2022), 6567.

[39] HODGSON, M. The Matrix Protocol: Architecture and Overview. https://spec.matrix.org/latest/, 2021.

[40] IP2LOCATION. Ip2location database. https://www.ip2location.io/.

[41] JEONG, U., NG, L. H. X., CARLEY, K. M., AND LIU, H. Navigating decentralized online social networks: An overview of technical and societal challenges in architectural choices. *arXiv preprint arXiv:2504.00071* (2025).

[42] JIANG, L., AND ZHANG, X. BCOSN: A blockchain-based decentralized online social network. *IEEE Transactions on Computational Social Systems* (2019).

[43] KLEPPMANN, M., FRAZEE, P., GOLD, J., GRABER, J., HOLMGREN, D., IVY, D., JOHNSON, J., NEWBOLD, B., AND VOLPERT, J. Bluesky and the at protocol: Usable decentralized social media. In *Proceedings of the ACM Conext-2024 Workshop on the Decentralization of the Internet* (New York, NY, USA, 2024), DIN '24, Association for Computing Machinery, p. 1–7.

[44] LABS, P. libp2p: A modular network stack. https://libp2p.io.

[45] LABS, P. Polygon description. https://polygon.technology/.

[46] LENS. Lens official documentation. https://lens.xyz/docs/chain/overview.

[47] LI, C., AND PALANISAMY, B. Incentivized blockchain-based social media platforms: A case study of steemit. In *Proceedings of the 10th ACM conference on web science* (2019), pp. 145–154.

[48] LUC. Dashboard of farcaster token. https://www.dune.com/0xluc/farconomy.

[49] MASTODON. Mastodon official documentation. https://docs.joinmastodon.org/. Accessed: 2025-05-10.

[50] NGUYEN, H. H., BOZHKOV, D., AHMADI, Z., NGUYEN, N. M., AND DOAN, T.-N. SoChainDB: A database for storing and retrieving blockchain-powered social network data. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[51] OPTIMISM. Optimism official documentation. https://docs.optimism.io/.

[52] PBLLC, B. ATProto: Authenticated Transfer Protocol. https://atproto.com, 2023.

[53] PIXELHACK. Dashboard of farcaster users transactions by chain. https://dune.com/queries/3027146/5036846.

[54] PROJECT, L. Gossipsub protocol - js-libp2p. https://github.com/libp2p/js-libp2p.

[55] RAMAN, A., JOGLEKAR, S., CRISTOFARO, E. D., SASTRY, N., AND TYSON, G. Challenges in the decentralised web: The mastodon case. In *Proceedings of the Internet Measurement Conference* (New York, NY, USA, 2019), IMC '19, Association for Computing Machinery, p. 217–229.

[56] ROSSARO, A. Supporting the portability of profiles using the blockchain in the mastodon social network. Master's thesis, Politecnico di Torino, 2019.

[57] SAITO, Y., AND SHAPIRO, M. Optimistic replication. *ACM Comput. Surv. 37*, 1 (Mar. 2005), 42–81.

[58] SNEDECOR, G. W., AND COCHRAN, W. G. *Statistical methods: by george w. snedecor and william g. cochran*. Iowa State University Press, 1968.

[59] SRINIVASAN, V. Sufficient decentralization for social networks. https://www.varunsrinivasan.com/2022/01/11/sufficient-decentralization-for-social-networks.

[60] STEEMIT. Steemit official documentation. https://steemit.com/guide/@steemitblog/steemit-a-guide-for-newcomers, 2025. Accessed: 2025-07-29.

[61] SUZOR, N. P. *Lawless: The Secret Rules That Govern Our Digital Lives*. Cambridge University Press, 2019.

[62] TECHJOCKEY. Guide to running farcaster nodes with contabo. https://www.techjockey.com/reviews/contabo.

[63] TUCKER, J. A., GUESS, A., BARBERA, P., VACCARI, C., SIEGEL, A., SANOVICH, S., STUKAL, D., AND NYHAN, B. Social media, political polarization, and political disinformation: A review of the scientific literature. *Political Polarization, and Political Disinformation: A Review of the Scientific Literature* (2018).

[64] TUFEKCI, Z. Algorithmic harms beyond facebook and google: Emergent challenges of computational agency. *Colorado Technology Law Journal 13* (2015), 203–218.

[65] VAN DER LINDE, A., LEITÃO, J. A., AND PREGUIÇA, N. Δ-crdts: making δ-crdts delta-based. In *Proceedings of the 2nd Workshop on the Principles and Practice of Consistency for Distributed Data* (New York, NY, USA, 2016), PaPoC '16, Association for Computing Machinery.

[66] WEI, Y., AND TYSON, G. An empirical analysis of the nostr social network: Decentralization, availability, and replication overhead. *Proc. ACM Netw. 3*, CoNEXT4 (Nov. 2025).

[67] WOOD, G., ET AL. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper 151*, 2014 (2014), 1–32.

[68] YANG, L., GILAD, Y., AND ALIZADEH, M. Practical rateless set reconciliation. In *Proceedings of the ACM SIGCOMM 2024 Conference* (New York, NY, USA, 2024), ACM SIGCOMM '24, Association for Computing Machinery, p. 595–612.

[69] YU, J., ET AL. Robust and trustworthy data sharing framework leveraging on-chain and off-chain collaboration. *Computers, Materials & Continua* (2024).

[70] ZUO, J., GUO, W., AND LING, L. Nexonet: Blockchain online social media with user-centric multiple incentive mechanism and poap consensus mechanism. *Applied Sciences (2076-3417) 14*, 21 (2024).

[71] ZUO, W., RAMAN, A., MONDRAGÓN, R. J., AND TYSON, G. A first look at user-controlled moderation on web3 social media: The case of memo. cash. In *Proceedings of the 3rd International Workshop on Open Challenges in Online Social Networks* (2023), pp. 29–37.

[72] ZUO, W., RAMAN, A., MONDRAGÓN, R. J., AND TYSON, G. Set in stone: Analysis of an immutable web3 social media platform. In *Proceedings of the ACM Web Conference 2023* (New York, NY, USA, 2023), WWW '23, Association for Computing Machinery, p. 1865–1874.

[73] ĐẠT, P. Q. Guide to running farcaster nodes with contabo. https://blogtienao.com/huong-dan-chay-node-farcaster-voi-contabo/.

A Ethics

The user datasets comprise casts and on-chain transactions, which may raise privacy concerns. To address potential data mishandling, we strictly collect publicly accessible data while adhering to well-established ethical guidelines for social data research [17]. We only collect broadcasted messages (*i.e.* casts), onchain transaction records and hub-level metadata. All of this data is publicly accessible. To further protect user privacy, we apply data anonymization by hashing unique user identifiers (*i.e.* FIDs and FNames) before analysis. Also, we paid for the storage used during the experiments and our crawl is non-intrusive by design with respect to hub resources. To avoid intrusive impact on the hub, we implement rate limiting, exponential backoff for retries, and concurrency control (see § 3.3). These measures ensure that our data collection does not affect the performance or availability of the hub. In addition, we obtained a waiver from the ethics committee at the author's institution.

B Cast Propagation Experiments

Cast Uniqueness. Each submitted cast is made unique by generating a random text payload in the format: "Timestamp [UTC]: five random numbers", with each number drawn independently from a uniform distribution (ranging from 0 to 10). The message is then encapsulated in a `MessageData` structure [26], which includes a unique Farcaster timestamp (Unix time plus a random offset up to 600 seconds relative to epoch 1609459200). A BLAKE3 hash is computed over the message data, including the Farcaster ID (FID) and timestamp, and paired with the FID to uniquely identify the message across all hubs during reception checks.

Exponential Backoff. Message reception is verified by querying each hub via HTTP GET (`/v1/castById`). For each cast, the query is retried up to 10 times within a 24-hour window (86,400 seconds). After a failed attempt, the process waits for $\min(2^{\text{retry}}, 64)$ seconds, where retry ranges from 0 to 9. This implements an exponential backoff capped at 64 seconds, balancing timely detection with minimal load on the hubs.

C Session Tolerance Window

To refine our availability analysis in § 4.1, we evaluate the impact of varying the *Nonresponsive Tolerance Window* (k), which defines the maximum allowed duration of inactivity before a hub is considered nonresponsive. Figure 8 presents the ECDF of nonresponsive intervals for $k = 1$, 2, and 3. The similar curve shapes across different k values indicate stable responsiveness behavior. As k increases, the CDF shifts right by approximately 10 minutes per increment. These results suggest that most nonresponsive intervals are brief, and increasing k mainly filters out transient outages. For simplicity, we adopt $k = 1$ in the main analysis.

D GossipSub Acceptance Rate

We analyze message acceptance in *GossipSub*, where a message is considered *Accepted* if it is validated and can be further propagated; otherwise, it is *Rejected*. Rejected messages are not propagated, posing potential availability concerns. Figure 7 shows the acceptance rate averaged over 10-minute intervals. Top-hubs with larger message volumes (< 1000) maintain high acceptance rates (median: 0.98), while the bottom-hubs show a marked decline.

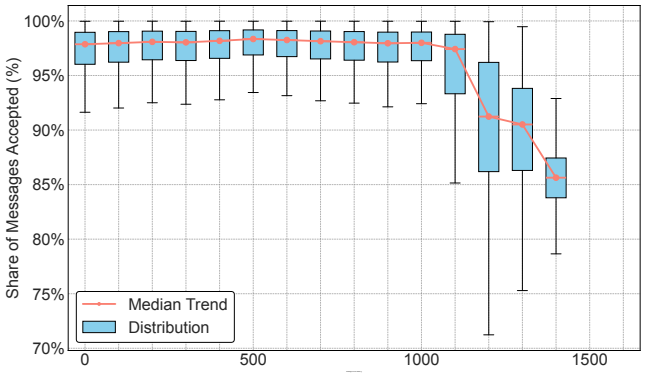


Figure 7: GossipSub acceptance rate per hub, aggregated into bins of 100 by message count ranking.

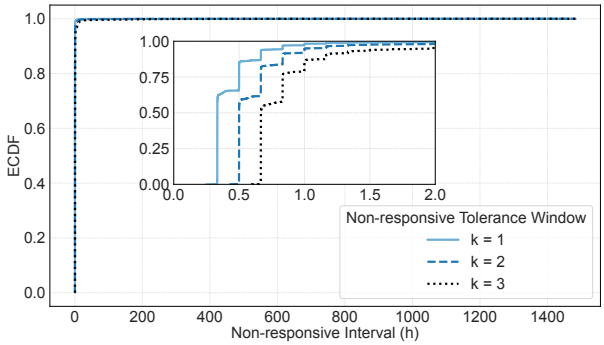


Figure 8: ECDF of non-responsive interval under different *Tolerance Window* values k across all hubs.

E Storage Unit Price

We provide additional information on message capacity per unit (Table 3) and storage unit rental price in Table 4.

F Hub Operating Cost and Revenue Estimation

Table 1 outlines the minimum hardware requirements for operating a Farcaster Hub, as specified in the official documentation [21].

Table 3: Message Quota per Storage Unit

Type	Quota (2023.11.07)	Quota (2024.08.28)
Casts	5,000	2,000
Links (Follows)	2,500	1,000
Reactions	2,500	1,000
User Profile	50	50
User Verification	25	25
Username Proof	5	5

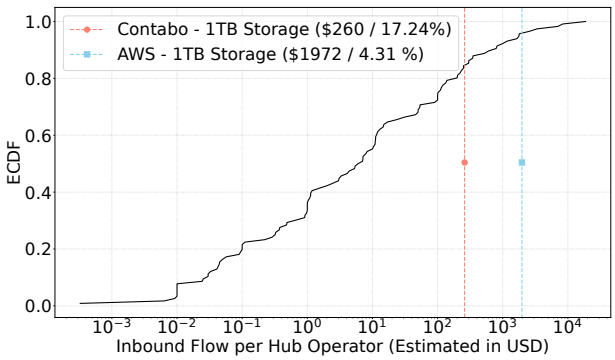


Figure 9: ECDF of revenue by hub operator.

Table 2 summarizes estimated annual costs across cloud providers that meet these minimum specifications (excluding Ethereum and Optimism RPC Endpoints, which are covered by Alchemy's free tier). Our findings show that annual expenditures vary significantly between providers, ranging from $260 to $1,972 — a nearly sevenfold difference. This cost disparity likely explains why 54.9% of hub operators opt for Contabo, while 10.3% choose Hetzner GmbH, with minimal adoption of officially recommended providers like DigitalOcean, Google Cloud, and AWS.

Figure 9 presents the empirical cumulative distribution function (ECDF) of on-chain revenue across hub operators, with benchmark thresholds denoted by dashed lines representing the annual operational costs for hubs configured with Contabo and AWS infrastructures, respectively. Note, only operators with positive on-chain revenue ($N = 116$) are included in Figure 9. The full dataset contains 3,855 hub operators. The data reveals that only 0.52% (N=20) of operators generate sufficient token revenue ($\geq$ $260) to offset basic Contabo expenses. This percentage drops to 0.13% (N=5) required for AWS infrastructure ($1,972). These results demonstrate that the vast majority of hubs operate at a financial deficit, unable to recover their operational costs through on-chain transaction revenue within the Farcaster ecosystem — a finding that raises questions about the platform's economic sustainability.

[1]https://contabo.com/en/pricing/
[2]https://www.hetzner.com/managed-server/
[3]https://cloud.google.com/products/calculator?hl=en
[4]https://www.digitalocean.com/pricing/droplets
[5]https://aws.amazon.com/ec2/instance-types/

Table 4: Storage Unit Rental Price

Effect Date	Value (USD)
2023-08-29	$5
2023-10-11	$7
2023-12-15	$5
2024-01-18	$3
2024-08-23	$2

G Farcaster Discourse on Hub Operations

To gather contextual qualitative data for understanding the cost-revenue dynamics of hub operations in Farcaster, we initially extracted $\approx$ 70,000 public Farcaster posts referencing hubs. We then refined this dataset by focusing specifically on posts discussing storage fees, operational challenges, and incentive mechanisms. Following manual review of approximately 3,000 relevant posts, our analysis identified seven predominant thematic patterns: (1) Operational Costs; (2) Incentive Appealing; (3) Alternative Funding; (4) Incentive Policy; (5) Storage Fee Management; (6) System Sustainability; (7) Incentive Farming (with Misinformation).

We list representative examples in Table 5. The table has been anonymized, abridged, and paraphrased to preserve the core meaning while removing any personally identifiable information. All links or symbols that could reveal individual identities have been redacted, except for posts by Farcaster co-founder @dwr.eth, which serve as the official Farcaster statement. These findings reveal a tension between user expectations and platform governance. Some users explicitly call for official financial subsidies to support hub operations, while others spread misinformation, exaggerating potential rewards. Several hub operators report that the current reliance on user tips does not cover operational expenses. This contrasts with the official Farcaster position, which rules out hub operation rewards to prevent incentive farming and avoid network performance degradation from low-quality hubs.

Table 5: Examples of Farcaster Discourse on Hub Operations

Theme	Key Content
(1)(2)	"Farcaster is experiencing explosive growth, and as a Hubble node operator, I need to scale hardware capacity at significant expense. We require support and incentives from the Farcaster team." "I'd consider running another node, but some direct or indirect incentives would be essential. Operating a node/Hub usually involves more complexity than expected. Even $20 or alternative benefits like reputation certificate would help."
(1)(3)	"My mini-apps have processed an impressive 9M requests from 90K unique users. Yet only <2K users have offered tips. I now maintain multiple hub servers and pay hundreds per year for on-chain data access. I've been working full-time, relying exclusively on user tips to keep this project alive"
(4)	@**dwr.eth**: "A reminder: there are zero economic rewards for running a Hub. There won't be a surprise one later. For example, this YouTube video has 7.4K views teaching people how to operate a Hub on a $50/month VPS. If you don't actually have a use for the Hub—or aren't in it for ideological/altruistic reasons—stop wasting your money." https://youtu.be/8VLHXRJ6wcQ?t=224 https://farcaster.xyz/dwr.eth/0xe4780508"
(5)	@**dwr.eth**: "(Storage Fee) are sitting in a multi-sig (blockchain wallet). When the amount gets large enough, we'll figure out what to do with it. Not a pressing problem right now." https://farcaster.xyz/dwr.eth/0x2d7823b4
(4)(5)(6)	@**dwr.eth**: "Paying for storage is too much friction to onboard new users, but not enough to slow down spam [...] The incentives to run a Hub are misaligned and causing congestion instead of capacity [...] Adding incentives would increase the number of Hubs. The team is exploring how to reduce congestion—possibly by increasing the economic cost of running a Hub (since access to the network is valuable). Nothing definitive yet. See also my cast below on Hubs: https://farcaster.xyz/dwr.eth/0x2d7823b"
(7)	"I have a foolproof method to generate spam casts and manage hubs. I'll disclose it for 7 USDC via Farcaster's on-chain pay feature. I guarantee it works for any project that meets one specific requirement." "Manual for Operating a Farcaster Hubble. Opportunity to Earn Over $1000"

Note: the contents are anonymized, abridged and paraphrased while preserving the original meaning.