Distributed Consensus through Network Support

David Guzman
Technical University of Munich
Munich, Germany
david.guzman@tum.de

Dirk Trossen
Huawei Technologies
Munich, Germany
dirk.trossen@huawei.com

Joerg Ott
Technical University of Munich
Munich, Germany
ott@in.tum.de

***Abstract*—Distributed consensus systems (DCSs) are increasingly important due to the often distributed realization of computational problems like voting systems and cryptocurrencies. The distributed nature of a DCS impacts the latency required to achieve consensus. In this paper, we develop an analytical model with empirically-based parameterization that provides an upper bound for that latency, thus enabling DCS operators to set expectations for its performance. We also propose a departure from the usual peer-to-peer-based DCS realization through an application-layer-based multicast approach. Our comparative analysis shows that our solution can improve convergence times by a factor of four while also ensuring targeted boundaries through the finality of the consensus convergence.**

I. Introduction

In an ever-more-connected world, implementing a computational problem, such as decentralized file storage [24], ledger partake [18], network bootstrap [3] and others, across distributed participants requires to progress over a *state* representing the computational problem.

The distributed realization of the computational problem requires agreement on the current state among the distributed participants; this is commonly described as the *consensus problem*. The insights in [19] show that reaching agreement among the majority of participants is enough to assume consensus; this is referred to as the *majority rule*. A distributed consensus system (DCS) implements the methods needed to reach consensus over a distributed state by achieving a majority agreement among its participants.

A key factor for DCS performance is the *latency bound* within which the system achieves consensus, termed consensus latency in the following. While the costs of maintaining the needed communication relations between DCS participants is another key performance factor, depending on the specific realization of the DCS, we focus in this paper on the latency aspect due to its direct impact on the transactional state changes that the system can agree and confirm on, measured in transaction per second (TPS) and minutes correspondingly. For instance, seven TPS with $\approx$ 60mins confirmation delay and 30 TPS with confirmation delay of $\approx$ 10mins are typical indicators for Bitcoin and Ethereum, respectively [27].

A key aspect in realizing a DCS is its nature of participation. Different from reaching an agreement among well-known entities in a DCS, a *permissionless* DCS lowers the requirements for participation, e.g., by not requiring strong authorization or use of trusted execution platforms. Instead, a simple sign-up procedure suffices together with the use of a distributed code base embodying the consensus protocol. For example, in Bitcoin and Ethereum, anyone can openly join and participate in the process to reach agreement [18], [26]. Through this, such DCS provides a low barrier to entry into the consensus partaking; this is often captured as reaching consensus among potentially non-trustworthy participants.

For these permissionless systems, iterative diffusion [14] based on a flooding algorithm [6] is used in Ethereum and Bitcoin. Here, the desired state over which to reach consensus, is being diffused to an initial set of participants, which diffuses the information to another locally determined set of participants, and so on. The problem lies not only in bounding the latency for the diffusion to the majority of the participants but also in the finality of the diffusion, i.e., judging that majority has been reached.

Our key contribution in this paper is bounding this latency for permissionless DCS platforms. Specifically, we provide an analytical model for the latency to reach consensus, using empirical insights from an existing DCS, namely Ethereum. This contribution enables DCSs to configure parameters, such as fanout, against a desired bound for the system's consensus latency, transaction throughput, and confirmation delay.

Extending this contribution, we compare those insights with a strawman for an application layer multicast delivery model. We use empirical insights of existing Ethereum deployments to parameterize the placement of multicast replication points, to outline the possible performance gains in consensus latency compared to iterative diffusion. Although our strawman leaves details open, specifically the necessary spanning trees for replication, it shows the performance gains once those remaining design aspects are solved. Specifically, we show a latency improvement of four times compared to existing iterative diffusion systems, while the finality of the consensus operation is achieved with higher certainty and less overhead of duplicated diffusion messages. These performance gains are rooted in the fewer diffusion steps required due to the aggregated nature of utilizing spanning trees across the DCS.

In the remainder of this paper, we outline background on DCSs in Section II for the basis for our multicast strawman design in Section III. We then assess the performance of existing DCS implementations, such as Ethereum, against our strawman in Section IV, quantifying the diffusion latency as a key performance indicator (KPI). In Section V, we discuss its benefits, opportunities, and deployment impact before concluding in Section VI.

II. Background on Distributed Consensus Systems

Existing DCS systems like Ethereum and Bitcoin implement a *diffusion-based mechanism* to reach consensus among DCS participants on advancing a common state related to a specific computational problem. Here, each peer samples N peers out of a set S of possible recipients. Since peers are generally unreliable or untrusted, owing to the permissionless nature of the DCS participation, the sample N is drawn at random. After disseminating the state advance to the N peers, those peers will, in turn, perform the same randomized selection of N peers and so on. This divide and conquer approach leads to iteratively disseminating information across the DCS [14].

Here, the number of iterations needed to reach the majority of peers, $m \in \mathbb{N}^+$, in a time interval $t_m \in \mathbb{R}^+$, determines the DCS consensus performance in the form of the transaction completion latency and, therefore, the transaction throughput of the overall system. The critical challenge for this diffusion lies in completing the process, i.e., reaching the majority rule, possibly within a desired time or even ever.

The necessary communication for the diffusion is implemented as a peer-to-peer network over IP by utilizing randomized multipoint relations that are established and maintained as a *pool of unicast relations* at each DCS peer. For this, a protocol is realized for pool discovery, establishment, and maintenance for relations being actively sought by each DCS peer while simultaneously handling requests from other peers. Locally generated random seeds assist in peer selection and discovery, while the unicast relations are constantly replenished for reasons discussed in [13].

These mechanisms are prone to issues, as studied in [13], [16], [25]. First, it is expensive to maintain updated availability information of peers joining and leaving the DCS network, causing discovery and reachability tests to fail, necessitating the constant pool replenishment of N peers. Second, validating capabilities requires an end-to-end unicast negotiation to support end-node heterogeneity. Third, timeouts and delays due to geographic distribution cause requests and responses to be dropped. Additionally, frequent changes in end-user addressing, firewalls, and network address translators (NATs) perturb the P2P relation establishment and cause imbalance in the communication [13]. Hence, *any bounding of the size of multipoint relations N needs to consider those issues to ensure consensus and convergence performance*.

Apart from the costs for those mechanisms, the randomized and highly dynamic nature of the peers' pools impacts the convergence latency, often taking many minutes for large-scale systems. As a consequence, DCS platforms require methods to work on *inconsistent state* during the consensus finding phase. Here, information is partially operated over in a so-called *proof* operations to generate temporary *system states*, representing an inconsistent yet agreed ephemeral state until consensus is achieved. The required proof methods, in turn, are responsible for much of the reported cost factor of popular DCS platforms, particularly for decentralized cryptographic currencies [10].

III. Strawman for an Overlay Multicast DCS

Let us now contrast the iterative diffusion in existing DCS systems with a packet replication approach akin to multicast. Unlike IP multicast, introduced in [8], we do not assume an in-network approach but propose an overlay network instead.

Key to our strawman are dedicated *Replication Points* (RPs) forming the overlay network, with the option to be deployed in ISP networks as a virtualized network function or through a compute node hosted in existing cloud platforms. RPs implement two essential functions, namely *overlay topology formation* and *packet replication*. We foresee peers obtaining the RPs to announce through, e.g., sign-up methods.

For the topology formation, RPs listen to peer announcements to join the DCS. Timed announcements can be used as a heartbeat mechanism to ensure the freshness of peer membership information in the overlay network. Criteria for selecting specific RPs may be geography, latency, or others.

For extending beyond a single RP, we suggest building spanning trees between RPs that allow for the distribution of service information from an RP-local peer to a given number of other peers in the tree. This procedure of sending to a given number of peers (in the spanning tree) is enabled by each RP aggregating peer announcements during the construction of the spanning tree by counting the number of downstream peers at each interface to another RP before propagating the announcements further to upstream RPs. For this, the *next hop information base* (NIHB) in each RP is extended with an entry *# peers*. As a result, when eventually being propagated to the peer-facing RPs, the total number S of downstream DCS peers can be retrieved from the peer-facing RP by summing up all interface entries for *# peers*, albeit only approximately due to possible churn in the set of peers connected to the RPs. This active spanning tree approach provides more up-to-date information on active peers in the overall DCS, replacing the pool creation and maintenance methods [13] done in each individual peer in today's DCS realizations.

For the spanning tree, a shared tree may be used. Alternatively, a (full or partial) mesh of RPs may build source-specific multicast (SSM) trees for each RP, e.g., utilizing methods outlined in [2] albeit adjusted to our strawman's overlay nature. Furthermore, we may utilize improvements to Protocol-Independent Multicast (PIM) that use unicast tunneling between replication points akin to our strawman. The authors in [1] have shown that the costs for signaling messages to build the resulting spanning tree(s) can be significantly reduced, thus supporting Internet-scale deployments aligned with the goal of our strawman. Overall, we foresee a significant reduction of control traffic compared to the pool maintenance methods outlined in [13] through our approach of peer announcement and building an RP spanning tree overlay, particularly when considering a low churn in the DCS. However, future work will be required to assess that cost aspect further.

The aggregation of peer announcements and the RP-local determination of the overall DCS peer number S, is key for the RPs' packet replication function since, unlike IP multicast,

packets are not replicated to *all* clients but to a dynamic number d of peers. Here, d is defined through the majority rule as at least half, i.e., $S/2$, of all announced peers, plus some overhead δ to account for possible packet losses and churn among DCS peers. To realize this semantic of d *out of all peers* multicast, a peer that initiates a send operation first queries S from its local RP, which in turn determines this number from the aggregated *# instances* entry in its next hop table, which then allows the peer to determine $d = S/2 + \delta$.

The peer then provides d as a request parameter to its local RP, based upon which any traversing RP can now implement the required packet replication function as follows: The RP first determines all NIHB entries for a given service, with the total being w, and retrieves the NIHB-specific *# instances* entries for each, defined as $\texttt{inst}_i$. It then distributes the provided parameter d of the incoming request as a ratio,a_i, of the *# instances* entries to the sum of all downstream instances, i.e., the RP calculates $d_t = \sum_{n=1}^{w} \texttt{inst}_n$ with $a_i = (d \cdot d_i)/d_t$. It then replaces the incoming parameter d with the determined ratio a_i and forwards the request down the NIHB interface i to the next level RP if the determined ratio is at least 1. To avoid topological bias towards paths with a strong presence of peers, the RP may choose to send at least one request downstream of those NIHB entries where the ratio is determined as zero.

IV. Comparing Iterative with Multicast Diffusion

When considering the benefits of our multicast strawman, it becomes apparent that the random diffusion is built on the fly by the RPs based on the availability announcements of peers. This RP-assisted approach contrasts against existing mechanisms described in [13], [16], [25], where peers must actively (and independently) discover and negotiate sessions with other peers. Also, the RP-based randomization avoids consensus centralization since it dodges the super-peer's higher connection degree, contrary to the central behavior shown in [13] for Ethereum, and in [12] for Bitcoin.

These aspects of building and maintaining the DCS overlay, positioned as the *control plane* of a DCS, are key to comparing both approaches regarding their cost and overhead in terms of messaging, as done in [13] for iterative diffusion.

However, we keep our evaluation in line with the previously expressed focus of our paper in studying what we call the *data plane* performance, as chiefly represented by the *convergence time* t_m towards a consistent state, including the possibility to judge the finality of that convergence. To do so, we develop suitable diffusion models for both dissemination approaches.

A. Model for Iterative Diffusion

We consider a DCS with S peers. Each peer in this DCS diffuses recursively to N peers. As outlined in Section II, and in more detail in [13], those N peers are sampled from the respective peer's local pool. A diffusion step to N peers requires $t_d(N)$ [ms]. The process aims at reaching at least half of the peers m, the majority rule in t_m [ms], and S peers in t_S [ms], as in Fig. 1. Moreover, at each iteration, the diffusion

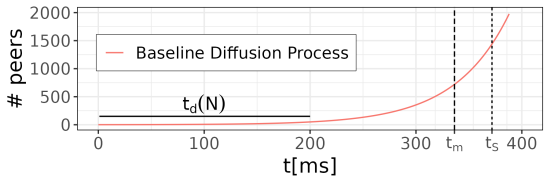

Fig. 1. Model example for a baseline diffusion in a DCS with $S = 72k$ peers, $m = 36k$ peers, $N = 50$ peers, $t_d(N) = 0.200$s, and $u(t) = 1$

sets are disjoint with expected probability $u(t)$ for a fixed N where $u(t) = 1$ signifies uniqueness across all diffusions.

We derive from insights into existing DCSs, [16] that a peer at each iteration parallelizes its invocations. Thus, we estimate the individual iteration latency, $t_d(N)$, as diffusion to a single peer within the set N. With this, we can formulate the evolution of an iterative diffusion in a DCS over time t as

$$m < (u(t)N)^{t/t_d(N)}, \quad \forall m, N, S \in \mathbb{N}^+, t, t_d(N) \in \mathbb{R}^+, \quad (1)$$

where m unique peers realize the majority rule at time $t_m = t_d(N) \cdot (\log_N(m) - 1)$.

We outline a baseline for Eq. 1 using typical Internet P2P system latencies of $t_d(N) = 0.200$s [15], an empirical average latency from a long-tailed distribution, and assume a complete uniqueness at each iteration, i.e., $u(t) = 1$. Fig. 1 shows an illustrative example for this baseline for the default $N = 50$ in an Ethereum system with 72k active peers with the dataset obtained from the authors in [13], [25].

We refine our baseline utilizing empirical insights for the variables diffusion latency $t_d(N)$ and uniqueness $u(t)$.

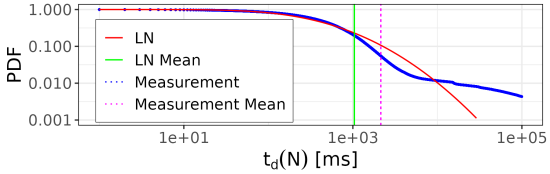

Fig. 2. Latency distribution for $t_d(N)$ from Internet-scale DCS measurements with mean 2.1s and its LN approximation with mean 1.04s in a log-log scale

1) Distribution of the Diffusion Latency: To refine $t_d(N)$ from the baseline, we use insights from [13], [16] into the complexities of communication in a DCS. We approximate $t_d(N)$ based on 1.8M data points, gathered at a public probe by the authors in [13], shown as the blue line in Figure 2.

This approximation preserves the primary behavior of the pool maintenance mechanism, i.e., the constant replenishment of N peers, resulting in frequently establishing connections to remote peers, thus incurring latencies over the Internet, while still considering rare events such as finding an unreachable peer or a long-termed capability/parameter negotiation albeit with low probability only. These considerations lead us to a log-normal (LN) approximation for $t_d(N)$, shown in Fig. 2 with log-mean μ_d and σ_d, denoted as:

$$t_d(N) \sim \ln \mathcal{N}(\mu_d, \sigma_d^2) \quad (2)$$

As we can see from Eq. 2 and Fig. 2, there is some, albeit small, probability for long diffusion times $t_d(N)$ and thus also long convergence time t_m for achieving distributed consensus.

2) Distribution of the Uniqueness of Peers: We turn to the uniqueness distribution in Eq. 1, i.e., $u(t)$. Selecting a set of random peers at each iteration impacts this uniqueness factor. This peer selection occurs in the discovery and establishment of N peers [13] as follows, exemplified in Figure 3 for Ethereum and further detailed in [17] for other systems:

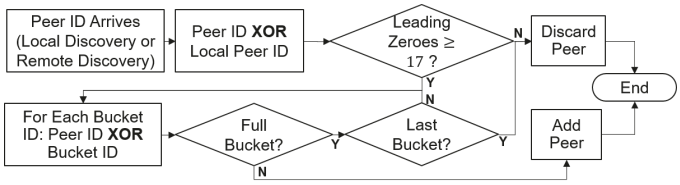

Fig. 3. Sorting Peers based on Binary Log Distance

Each peer aims at locally creating a unique and independent *ID*, determined from the *most significant bits* (MSBs) of the digest over its public key, which in turn is derived from local pseudo-random-number generators [11]. Any incoming or outgoing ID, i.e., either those actively discovering a peer or those being discovered by the peer, are then sorted into random *buckets of tuples*, which include the peer's IP address and its ID, and arranged by the *binary log distance* (XOR metric) relative to the local ID [17]. Thus, each peer discovers and sorts IDs based on its ID and samples from this sorted list to create a pool of peers of size N.

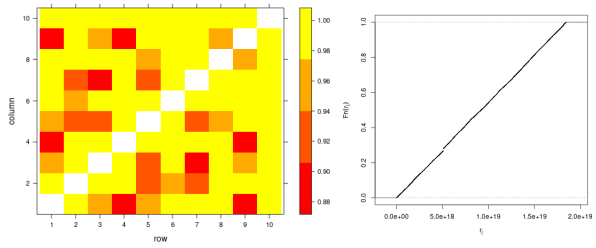

Fig. 4. (left) Sample of $u(t)$, and (right) ECDF for, p_j, selection probability for a peer j out of S from empirical data based on peer ID crawls

We determine the uniqueness of the randomized sets (across the diffusing peers) to derive the duplication that harms the convergence time t_m, based on the dataset in [13] with 88 experiments. Each experiment passively probes an ID for five hours. The probe crawls the DCS with an ID per experiment. This mechanism is the same across the DCS, and with each ID, a peer intends to sample periodically from the *entire* DCS.

The example in Fig. 4-left shows $u(t)$ based on set intersections of IDs over a small sample of ten selections of $N = 50$ peers (a default fanout for *geth* [11]). If a peer diffused to unique peers at each iteration, the entire matrix in Fig. 4 would be yellow. Instead, we notice the correlation between chosen sets as red squares in Fig. 4. Such high correlation is inherent to the random selection approach, worsened by highly active super-peers [16] because the size N of selections is larger than the default configuration, e.g., thousands [16].

Further, we reason about the distribution of those sets from iteration to iteration. For this, we consider the sequence of choices made by a peer r_0 as an independent and identically distributed (i.i.d.) selection since the ID sorting scheme ensures that the entire DCS ID space is explored, while the sampling process ensures random picks from sorted IDs [17].

With this in mind, a peer r_0 diffuses information to N random peers drawn from the set S of active peers. Each of the N peers diffuses again independently to other N remote peers, randomly selected from the same set of active peers.

This process randomly executes k selections in a set of S peers, where k grows exponentially from initially $k = N = 50$ with each iteration since each selected peer, in turn, will cause executing N selections again.

We model this process through a discrete random variable $\mathbf{b}$, which counts the number of peer selections $\mathbf{r} = [r_1, \ldots, r_S]$ after k i.i.d. random trials. Then, a peer r_i is picked b_i times; hence, the set of peers $\mathbf{r}$ are randomly hit or selected as $\mathbf{b} = [b_1, \ldots, b_S]$ times. Consequently, the probability of observing *occurrences* $\mathbf{b}$ follows a multinomial distribution, parametrized by $\mathbf{p} = [p_1, \ldots, p_S]$ [21]:

$$\mathbb{P}(\mathbf{b}|\mathbf{p}, k) = \frac{k!}{b_1! \ldots b_S!} \prod_{i=1} p_i^{b_i}, \quad \forall k, b_i \in \mathbb{N}^+, p_i \in [0, 1]\,, \tag{3}$$

where p_i is the probability of choosing b_i times a peer r_i out of k trials, with $\sum_{i=1}^{S} p_i = 1$.

Intuitively, the marginal distribution b_j follows a binomial distribution, where $\mathbb{E}(b_j) = kp_j$ is the expected times a peer r_j is chosen, with variance $\mathbb{V}(b_j) = kp_j(1 - p_j)$.

Here, $\mathbf{p}$ is vital for the uniqueness definition. If selecting a peer r_j were equally likely for all j, then $p_j = \frac{1}{S}$. However, there are super-peers to pick with higher probability than $\frac{1}{S}$ since those super-peers spread their ID to more than the default $N = 50$ peers [16]. We show in Fig. 4-right that the assumption $p_j = \frac{1}{S}$ is nonetheless plausible.

With the above, it follows that $\mathbb{E}(b_j) = \frac{k}{S}$ with variance $\mathbb{V}(b_j) = \frac{k}{S}(1 - \frac{1}{S})$. Further, we define the expected unique peers *up to* a certain diffusion iteration x from the expected times a peer r_j is selected, as:

$$\mathbb{E}(u_j) = \frac{k^{x+1}}{\ln N}(1 - \frac{k^{x+1}}{S \ln N}), \tag{4}$$

where $k^{x+1}/\ln N$ denotes peers selected up to iteration x. Now, each iteration can be described as a time-dependent variable for a diffusion size N with

$$\mathbb{E}(u_j) = \frac{N^{1+t/t_d(N)}}{\ln N}(1 - \frac{N^{1+t/t_d(N)}}{S \ln N}) \tag{5}$$

Then, we derive the factor $u(t)$ as a uniqueness upper bound over the total number of peers up to iteration $N^{t/t_d(N)}$ as

$$u(t) = (1 - \frac{N^{1+t/t_d(N)}}{S \ln N}), \quad \forall u(t) \in [0, 1]\,, \tag{6}$$

refining our initial model in Eq. 1 to

$$m < (N - \frac{N^{2+t/t_d(N)}}{S \ln N})^{t/t_d(N)}, \tag{7}$$

with $N^{2+t/t_d(N)}/S \ln N \leq 1$.

The latter implies $t_S \approx t_d(N)(\log_N S - 1)$, outlining the existence of a moment in the diffusion after which all peers are exhausted, continuing with information propagation to all peers only using expensive duplicated communication. In other words, the iterative process continues diffusing to all S peers until $t > t_S$, with diffusion to most peers until t_m to ensure consensus. However, for the diffusion at each timestep t, the system draws exponentially growing trials, and, as time passes, the cost for repeated sampling of peers grows high.

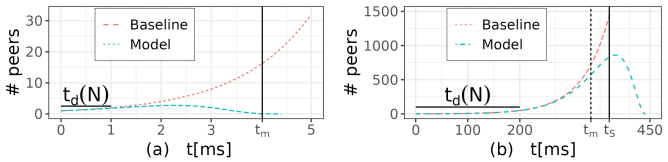

Fig. 5. Baseline diffusion, i.e., Eq. 1 with $u(t) = 1$, against the randomized diffusion model, i.e., Eq. 7 with $u(t) < 1$, with parameters (a) $S = 63, N = 2, t_d(N) = 0.001$s, and (b) $S = 72$k, $N = 50, t_d(N) = 0.200$s

We illustrate this effect in comparison to ensuring complete uniqueness of peers throughout the iterations, with Fig. 5a illustrating a small scale DCS with $S = 63$ peers and Fig. 5b for measured DCS parameters, derived from [13]. There is a marked decline already before reaching t_m, which is even higher for $t > t_m$, particularly for Fig. 5b, ultimately slowing down finality, reaching all S peers, of the diffusion process at time $t > t_S$ as illustrated in Fig. 5b.

From Fig. 5, we derive a performance indicator g which compares the model, i.e., non-uniqueness diffusion, against a baseline diffusion based on a completely unique selection of peers at each step in a given time t_m through:

$$g = \int_0^{t_m} \frac{(u(t)N)^{t/t_d(N)}}{N^{t/t_d(N)}} dt \tag{8}$$

In Fig. 5a, this performance factor is $g = 3.06$, which implies that a DCS system in a best-case scenario with a baseline diffusion, i.e., uniqueness $u(t) = 1$, finds three times more unique peers in a given time compared to the randomized selection approach; hence, the diffusion up to the majority rule is delayed by the same factor $g \cdot t_m$.

Furthermore, we notice $g \approx 1$ for the short time regime in Fig. 5a and Fig. 5b. $g = 1.1$ when comparing a baseline diffusion against a randomized one for an Internet-scale DCS with $S \gg N$ up to t_m in Fig. 5b, and it increases up to $g = 1.25$ when considering diffusion up to t_S. The latter means *the randomization process slows down diffusion on average to* 25% *per single request.*

Combining our insights into the diffusion latency and the uniqueness distributions, we can contrast the above baseline derived from Eq. 1 against the empirical diffusion model

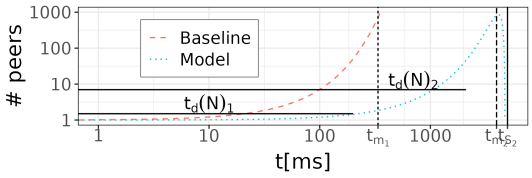

Fig. 6. Comparison for baseline diffusion, i.e., Eq. 1 with $u(t) = 1, t_d(N)_1 = 0.200$s [15], against the randomized diffusion model, i.e., Eq. 7 with $t_d(N)_2 = 2.1$s, both with $S = 72$k, $N = 50$, resulting in $t_{m_1} = 0.336$s, $t_{m_2} = 3.962$s and $t_{S_2} = 4.977$s in a log-log scale

provided in Eq. 7. Through our results shown in Fig. 6 as the times for achieving the majority rule t_{m_1} for the baseline model against t_{m_2} for the randomized empirical model, we conclude that our empirical insights have stretched the latency bound for achieving the majority rule by a factor of 11.7, which conversely means that *a system without randomization can be at least* 11.7 *times faster at diffusing information to achieve consensus at Internet scale.*

B. Model for a Multicast-based Diffusion

To outline a model for multicast-based diffusion, we detail the peer clustering behavior and its complexity from the measurements previously outlined. The clusters are identified based on an infrastructure provider, i.e., autonomous system (AS), cloud provider, or hyper-scaler, and its complexity is based on a geographical spread for each cluster.

A map between an ID, an IP address, and an autonomous system number (ASN) shows the infrastructure clusters. In Fig. 7, we characterize the concentration size of peers in well-known infrastructure providers. This clustering is also reported in [12] for Bitcoin. For example, the top infrastructure provider, HOGH in Fig. 7, hosts $10k$ peers $\approx 9.6\%$ of the total peers in the DCS. Indeed, 47.9% of the peers rely on 10 out of 3795 infrastructures.

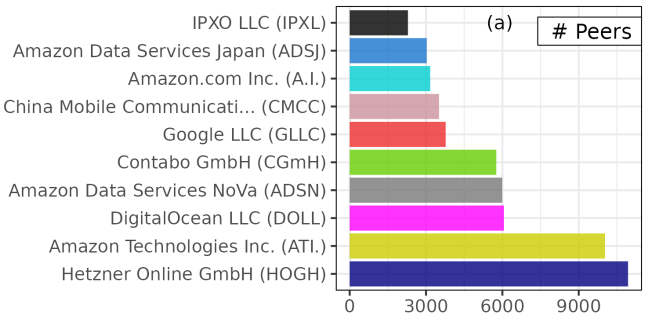

Fig. 7. Distribution and clustering of peers for an Internet-scale DCS based on measurements for Ethereum with cluster size for the top 10 infrastructures

To estimate the geo spread for each infrastructure, we map each peer IP address to a geographic location to describe the convex hull of an infrastructure cluster, where the geolocation mapping is guaranteed with 99% accuracy [4]. This assumption allows us to compare each cluster's geographic span based on country centroids from empirical results.

The geometric approach is not a topological representation of the infrastructure provider. Instead, the convex hulls relate to physical connectivity in terms of locations of peers, geographical distances, and aspects of logical clustering, bounding the connectivity span in private backbone infrastructures. We use this geo span as a weight for the infrastructure complexity and latency to design the DCS overlay network.

We assume an RP overlay deployed with one RP per infrastructure, located in a focal area within the convex hull given by the nationwide cluster size and serving peers distributed in those providers, with Ethereum as the example. A country within an infrastructure convex hull hosting most peers defines a focal area. For example, in the case of HOGH, the RP serving $6k$ out of $10.9k$ peers is located in Germany.

RPs are connected in a partial mesh with peers announcing their services to the nearest infrastructure RP, using unique established relations as described in Section III that avoid the latency variances induced by the processes for reachability checks, connection establishment, and capability exchange that iterative diffusion approaches must rely on.

The majority rule demands sending information to m peers. As outlined in Section III, the initiator inquires the DCS size from its local RP, being able to thus determine m and provide it as part of the initiating request to its RP, which distributes to its neighboring RPs until reaching their local peers. Hence, two tiers of the overlay multicast mesh support the diffusion, where the overall diffusion delay, t_{d_m}, is bounded through

$$t_{d_m} = 2 \cdot t_a + 2 \cdot t_p + t_e + t_c, \quad \forall t_a, t_p, t_e, t_c \in \mathbb{R}^+, \quad (9)$$

$t_a \leq 0.200$s is the delay between a peer and the nearest RP at the originator and destination RP side. t_p is the maximum processing packet latency at an RP, and t_e a maximum peering delay between infrastructures with the most extended inter-infrastructure delay, bounding this delay to $t_e \leq 0.100$s [7], [15], and t_c being a maximum propagation delay in infrastructures as outlined in [23] with the highest delay for GLLC equal to $t_c \leq 0.333$s [15].

Further, to bound t_p, we consider that each RP handles a maximum of 10k peers, derived from Fig. 7b. Thus, an RP processes 10k requests in $t_p \leq 0.025$s given that steering mechanisms handle ≈ 400000 reqs/s as in [20] where packet processing relies on in-kernel implementations.

With the above, we compare the time to diffuse to m peers in an iterative diffusion fashion, t_{m_2} as per Fig. 6, against the time achieved by our strawman approach, t_{d_m} from Eq. 9, as $t_{m_2}/t_{d_m} = 3.962/(0.4 + 0.05 + 0.1 + 0.333) = 4.486$, *improving thus the time to reach consensus by a factor of* 4.5. This is a minimal performance improvement since we upper-bounded the latencies for our strawman approach. Moreover, the significant convergence times in rare cases caused by the long tail delay distributions in the iterative diffusion model are removed in our strawman approach. Overall, this result aligns with the expected performance improvement noted in the previous section as part of the *baseline* iterative diffusion.

From the insights in Fig. 6, we can compare the time for diffusing to the entire DCS t_{S_2} against the upper bounded latencies for the strawman approach as $t_{S_2}/t_{d_m} = 4.977/(0.4 + 0.05 + 0.1 + 0.333) = 5.636$. However, the real nature of the improvement here lies beyond 5.636 since the distributed nature of the iterative diffusion makes the judgment of finality, i.e., the completion of the diffusion process, impossible to judge after the 4.977s provided in Fig. 6. Instead, diffusion attempts will continue among peers until the peer-local selection process will ultimately stop the diffusion; even at that point, an individual peer may not be able to judge finality of the diffusion without an explicit coordination among the peers.

Here, the controlled sending to d peers in the strawman provides an enormous qualitative advantage over the iterative diffusion in that finality can be judged through the finality of the performed overlay multicast operation; thus, the bound of $(0.4 + 0.05 + 0.1 + 0.333) = 0.883s$ provides not just a limit for diffusing to the majority of peers but also for judging the achievement of consensus, i.e., finality.

We note that RP-supported multicast diffusion avoids the high percentage of duplicated communication found in the randomized approach due to actively maintaining service instances at each RP and performing the replication only once. We leave the evaluation of the resulting cost savings due to avoiding unnecessary communication to future work.

The key takeaway from a randomized iterative diffusion compared to a multicast approach is that the latter cannot only improve in factors of around 4 for the majority rule and around 5 for the finality of consensus, but it also ensures that the diffusion ends and the completion time is known.

V. Discussion

Let us discuss a few aspects concerning our model insights and assumptions made throughout its development.

We assert *transferability* of our empirical insights to other DCSs, such as Bitcoin and Algorand, since its consensus mechanism applies randomized communication [9]. Furthermore, it is extensible to other restrictions, such as for Byzantine faults in [5], where higher constraints are required to tackle failures and malicious behavior.

Another impact of our results is on the methods to *work on inconsistent state*, required to continue operations during an ongoing convergence process, as observed in [18], [26]. They often incur system instabilities through decision forking and consuming expensive computational and communication resources [10]. We argue that our evaluation insights provide DCS providers with better tools to limit the convergence time, thus allowing them to tune DCS operations towards desired operational parameters as well as provide better capabilities to estimate the necessary costs for the proof methods for those periods where operations on inconsistent state are needed.

Further, we assert that our multicast strawman may push a DCS towards much *shorter periods of inconsistency*. More so, the insights on bounding the latency may allow DCS operators to entirely avoid PoW or PoS methods if the latency lies beyond a desired bound for the operations of the larger

computational problem, although this situation depends on the specific DCS application and its required convergence time boundaries. A more profound study across various DCS applications may provide the needed insights on this potential to improve the overall environmental impact of the DCS.

Further insights are required into *deployment aspects* of our strawman. Our initial proposal in this paper outlines an infrastructure-based RP deployment, leading to a simple partial mesh of RPs to handle the complexity of the private backbone. Other models may include city-wide deployments or a mix depending on the geographical distribution of peers. What remains unchanged is the strict time bound on convergence; merely additional bounded latencies may need to be accounted for when adding tiers of distribution into the multicast system.

Furthermore, the move towards an RP overlay may allow for revisiting key *security aspects*, such as establishing confidential channels. Here, confidentiality may be merely ephemeral, e.g., between RP and peer at the local diffusion step. Similar considerations may apply to IP address confidentiality, where announcing to RPs instead of diffusing one's IP address to possibly many if not all, peers for iterative diffusion may change the private nature of the DCS, possibly requiring a trust relation with the DCS platform but not with individual peers. This removes a fundamental weakness of existing DCSs where individual peers may misuse the peer discovery process to scrape other peers' public IP address for, e.g., denial of service attacks [22]. Moreover, such trust relation is independent of the enhancements to the consensus convergence latency since adversaries (Byzantine peers) remain the same, i.e., $2/3$ of total peers [5] or the majority while achieving consensus [19].

VI. Conclusion

Achieving the majority rule over the distributed state is at the heart of any DCS. The mechanism's efficacy to diffuse to at least the majority of all DCS peers drives a vital portion of the DCS's cost and usefulness regarding latency bounds.

In this paper, we shed light on the impact of the randomized algorithms in existing DCSs on the time to converge to a single state. For this, we formalized an exponential growth model that captures the iterative diffusion process in those systems. We bounded the key parameters of this model through experimental insights derived from real systems, here Ethereum, to lead to expectations for the convergence in those systems. We specifically showed that convergence times achieved through iterative diffusion exceed those of ideal diffusion (achieved through unique peer distribution) by at least 25%.

While our model and results aim to guide DCS platform operators in their configurations alongside quality-of-operation expectations, it also allows for comparison against other diffusion approaches. Here, we specifically showed that a system that applies overlay multicast for diffusion might lead to significant improvements in convergence time of a minimum of 4x for the majority rule and 5x for the finality, with an ensured finality of the consensus.

In future work, we envision quantifying the practical cost savings from this key design difference at the Internet scale.

References

[1] T. Bartczak and P. Zwierzykowski, "Performance evaluation of source-specific multicast routing protocols for ip networks," in *2012 8th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2012, pp. 1–6.

[2] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," IETF, RFC 3569, Jul. 2003. [Online]. Available: http://tools.ietf.org/rfc/rfc3569.txt

[3] R. Bless, M. Zitterbart, Z. Despotovic, and A. Hecker, "Kira: Distributed scalable id-based routing with fast forwarding," in *IFIP*, 2022.

[4] I. Brand Media, "Online Tools," March 2023. [Online]. Available: https://tools.iplocation.net/

[5] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. USA: USENIX Association, 1999.

[6] É. Coulouma, E. Godard, and J. Peters, "A characterization of oblivious message adversaries for which Consensus is solvable," *Theoretical Computer Science*, pp. 80–90, 2015.

[7] T. K. Dang, N. Mohan, L. Corneo, A. Zavodovski, J. Ott, and J. Kangasharju, "Cloudy with a chance of short RTTs: Analyzing cloud connectivity in the internet," *ACM SIGCOMM, IMC*, 2021.

[8] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans." ACM, may 1990, p. 85–110.

[9] N. Dimitri, "Proof-of-stake in algorand," *Distrib. Ledger Technol.*, 2022.

[10] D. Drusinsky, "On the high-energy consumption of bitcoin mining," *Computer*, vol. 55, no. 01, pp. 88–93, jan 2022.

[11] Ethereum, "Go ethereum official client," 2022. [Online]. Available: https://github.com/ethereum/

[12] M. Grundmann, M. Baumstark, and H. Hartenstein, "On the Peer Degree Distribution of the Bitcoin P2P Network," *IEEE ICBC*, 2022.

[13] D. Guzman, D. Trossen, M. McBride, and X. Fan, "Insights on impact of distributed ledgers on provider networks," in *Blockchain – ICBC*, 2022.

[14] D. Guzman, D. Trossen, and J. Ott, "If iterative diffusion is the answer, what was the question?" in *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, ser. FIRA '23.

[15] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the internet," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. ACM, 2016.

[16] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring ethereum network peers," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. ACM, 2018.

[17] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*, 2002.

[18] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Journal for General Philosophy of Science*, vol. 39, no. 1, pp. 53–67, 2008.

[19] J. V. Newman, "Probabilistic Logics and the Synthesis of Reliable Organism from Unreliable Components," *Automata Studies*, 1956.

[20] F. Parola, R. Procopio, R. Querio, and F. Risso, "Comparing user space and in-kernel packet processing for edge data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 53, no. 1, p. 14–29, apr 2023.

[21] C. E. Rasmussen, "Discrete Categorical Distribution [Lecture]. Cambridge," pp. 1–8, 2016.

[22] M. Rasolroveicy and M. Fokaefs, "Impact of ddos attacks on the performance of blockchain consensus as an lot data registry: An empirical study," in *Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '22. USA: IBM Corp., 2022, p. 71–80.

[23] L. Salamatian, S. Anderson, J. Mathews, P. Barford, W. Willinger, and M. Crovella, "A Manifold View of Connectivity in the Private Backbone Networks of Hyperscalers," *Communications of the ACM*, 2023.

[24] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and evaluation of ipfs: A storage layer for the decentralizedweb," *SIGCOMM 2022 - Proceedings of the ACM SIGCOMM 2022 Conference*, 2022.

[25] D. Trossen, D. Guzman, M. McBride, and X. Fan, "Impact of Distributed Ledgers on Provider Networks," no. 935, 2021.

[26] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, pp. 1–32, 2014.

[27] M. Zhou, L. Zeng, Y. Han, P. Li, F. Long, D. Zhou, I. Beschastnikh, and M. Wu, "Mercury: Fast Transaction Broadcast in High Performance Blockchain Systems," *Proceedings - IEEE INFOCOM*, 2023.