©2022 Springer Professional. Personal use of this material is permitted. Permission from Springer must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
This is an author manuscript. The final submission is available at : https://www.springerprofessional.de/insights-on-impact-of-distributed-ledgers-on-provider-networks/23857880

Insights on Impact of Distributed Ledgers on Provider Networks

David Guzman[2 1], Dirk Trossen[1], Mike McBride[3], Xinxin Fan[4]

[1] Huawei Technologies, Munich, Germany
{david.guzman,dirk.trossen}@huawei.com
[2] Technical University of Munich, Munich, Germany
[3] Futurewei Technologies, Santa Clara, CA 95050 michael.mcbride@futurewei.com
[4] IoTeX, Menlo Park, CA 94025 xinxin@iotex.io

Abstract. Internet scale distributed consensus takes advantage of randomized algorithms to cope with functional resilience, reliability and access while accomplishing decentralized agreement at expenses of costly unicast communication. This paper provides experimental insights on randomized communication patterns for distributed consensus systems. We define, measure and approximate communication overhead against key performance indicators like latency and communication overhead, leading to a discussion on how possible network innovations may mitigate the identified issues. While this paper does not promote specific solutions or DLT implementations, our initial insights invite the wider community working on DLT and network solutions alike to deepen those insights to aid future research and development into possible solution concepts and technologies.

Keywords: Distributed Consensus · Permissionless · Networking.

1 Introduction

Large scale distributed systems execute functionality, often called *smart contract*, while keeping track of transactional information in distributed databases. It is crucial that the information in those databases represents a consensus among its participants to ensure its veracity. For this, so-called 'permissioned systems' may be utilized where the distribution across authenticated and trusted parties is realized through well-known, e.g., publish-subscribe communication patterns. The trust assumption, however, poses a strong requirement on deploying such systems, often limiting their scalability. For this reason, 'permissionless systems' aim at removing this strong assumption, effectively allowing anybody to join the distributed consensus process, thereby decentralizing the operations across anybody who is willing to commit resources to do so. Blockchains [1], as an example for Distributed Ledger Technologies (DLT) [2], represent a category of solutions for such permissionless systems.

The removal of the strong trust assumption, however, comes with its own challenges. In order to ensure functional resilience and reliability against any

random peer in the system to disappear or stop functioning, randomized algorithms are employed so as to not only utilize highly randomized members of the overall pool of resources but also to spread information across random peers to remove any reliance on any single peer due to the lacking trust into its operations and trustworthiness.

These randomized algorithms find realization in (i) a discovery algorithm that is in charge of assuring a sufficiently large sample of peers from the entire network to be contacted, (ii) a *diffusion multipoint* communication based upon the discovered peer topology, pursuing a so-called atomic broadcast [3] to commit and forward blocks and transactions. Given the randomized nature of both of those key processes, there is a clear impact on the system's key performance indicators (KPIs), such as completion time for a transaction but also costs for those transactions and managing the topology.

Motivated by our preliminary work in [4] we structure, analyze, and discuss the performance and costs of such randomized communication algorithms and their impact on provider networks. Measurement and analysis of the P2P statistical nature have been carried out for Ethereum in [5, 6]. However, we lack studies that analyze peer reachability, privacy, and information propagation while implementing a randomized discovery phase as well as the costs for signaling and the resulting communication establishment latency and efficiency for a diffusion multipoint operation.

Consequently, we summarize our contributions as: (i) striving for a *structured understanding* on communication patterns, and (ii) assessing the *impact and costs* of communication patterns based on KPIs, real measurements, and statistical approximations.. While our work focuses on the Ethereum P2P network for its insights, we are confident that our results can be easily validated, replicated, and scaled to other implementations of large scale decentralized consensus systems.

For this, we first analyze the core interactions and communication patterns between Ethereum peers in the peer discovery, establishment process, and consensus requirements in Section 2. Based on the observed P2P communication patterns, we then identify in Section 3 a number of challenges with respect to peer reachability and their required capabilities, the realization of transaction communication across the DLT, as well as privacy and/or security threats. Through our extensive experiments, we quantify in Section 4 the inefficiencies of the Ethereum P2P network along key performance indicators such as latency communication overhead, and effective data consumption. We further confirm and compare the approximate statistical distributions of our topological measurements with results in [5, 6] . In Section 5, we use these observations and insights to discuss a number of research and innovation opportunities for improving on the observed situation, before concluding the paper in Section 6.

2 Communication in a DLT

There has been ample work, such as [1, 3, 4], including in standardization organizations, such as the IEEE [7, 8] but also within the IRTF/IETF [9], on defining main DLT concepts. We focus our brief introduction on those concepts most important from a communication perspective. We first outline the interactions at a higher level before delving into the communication patterns that result from those.

2.1 DLT Interactions

We can distinguish three core interactions in a DLT:

A client commits a **transaction** to the DLT, which is diffused across a set of DLT miners, which in turn respond separately, adding the transaction to their internal ledger information. Committing the transaction leads to the miners assigning compute and storage resources to the smart contract that underlies the transaction. For this, so-called *proofs* are executed, although some methods for proof require additional communication to take place, e.g., election protocols.

The result of the proof is a **block** (of ledger information) that miners in turn commit to a set of (other) DLT miners, which each receiving miner adds to their internal blockchain.

A client may query the latest **blockchain** from a set of miners to which the query is sent. The longest returned blockchain represents the most trustworthy ledger information.

In the following subsection, we focus on the communication patterns that are utilized to achieve the aforementioned interactions. Special attention is given on the establishment of the needed peer-to-peer network, used in the randomized multipoint operations that are executed for each interaction, be it a transaction or the commitment of a newfound block.

2.2 Resulting Communication Patterns

Key for the communication within the DLT P2P network is to establish and maintain a *pool of DLT peers*, which is used to exchange information. This pool represents a subset of the larger set of all DLT peers. We differentiate two types of peers, namely those actively initiated by a peer and those maintained as a result of a received request to initiate a relation due to other peers also actively seeking relations. We call the former *outgoing peer* and the latter *incoming peer* in the following.

The details on how this pool is established and maintained are obtained from Ethereum [10], containing platform-specific aspects, which likely differ from those used in other platform; however, the main concepts remain the same.

Figure 1 outlines the necessary steps, divided into a) *discovery* of outgoing DLT peers, and b) *establishment* of the pool of DLT peers, ultimately used to for the randomized interaction patterns outlined in the previous sub-section.

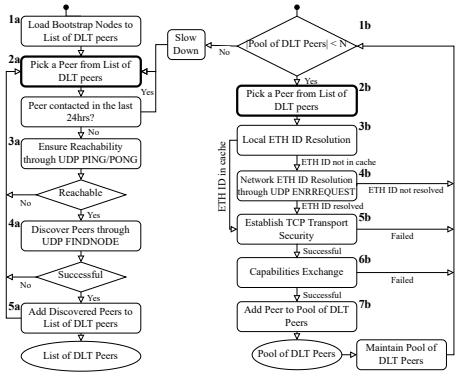

Fig. 1. (a) Discovery of Outgoing Peers, (b) Establishing Pool of Peers

More specifically, the discovery phase in Figure 1a aims at building a *list of outgoing DLT peers* through actively seeking relationships with other peers. For this, the DLT peer first seeds the selection of first step neighbours through information obtained from pre-configured so-called *bootstrap peers*. This set of neighbours is randomized (Step 2a) and tested for *reachability* through a simple UDP-based PING/PONG exchange with a default timeout of 500ms. Any non-reachable peer, i.e., those timing out, are removed from the list of outgoing peers.

All remaining nodes are now tested for *neighbourhood information* by sending a UDP-based FINDNODE request, which is responded positively with a list of (usually 16, as defined by configuration) DLT peers in the form of a NEIGHBORS response, while removing any timed-out DLT peer from the list. The response includes a list of IP addresses and supported ports of the (outgoing) DLT peers.

Through repeating the neighbour peer discovery with the above steps in intervals of 500ms, the list of outgoing DLT peers grows. Upon reaching a configured length N (set to a default of 50 in a standard Ethereum configuration), the peer discovery is slowed down with discovery intervals growing up to 30 minutes. The final size of the list of outgoing DLT peers is a matter of local limitations, with Section 4.3 outlining typical list sizes obtained from our experiments.

The list of outgoing DLT peers, i.e., candidates for relationships, is now extended into a *pool of DLT peers* as active, i.e., maintained, TCP connections to a subset of DLT peers. Figure 1b) shows the necessary steps for the creation of that pool.

Key difference to the list of outgoing DLT peers in step 2a is that the pool of DLT peers includes those actively created during the discovery phase as well as any incoming peers, i.e., connections initiated by other peers that are undergoing the same process of DLT pool establishment. The limit for the desired DLT pool size is equal to N, i.e., the same number for the initial limit for the list of outgoing peers. However, the pool establishment limits the number of outgoing peers to

one third of this pool size, while the rest is reserved for incoming peers. Hence, a candidate for adding to the pool of DLT peers can either be selected from the list of outgoing peers (as long the configured limit has not been reached) or through accepting an incoming peer connection.

For both cases, the Ethereum ID, uniquely identifying the DLT peer in the Ethereum P2P network, is being resolved next, either through obtaining a locally cached value (for any DLT peer that has been previously resolved) in step 3b or through sending a UDP-based ENRREQUEST request to the chosen DLT peer in step 4b. Next, transport layer security is established, utilizing TLS-based mechanisms [11] [10]. If successful, a capability exchange takes place, determining bilateral support for, e.g., hash functions, needed hardware capabilities (such as GPUs) or others, to successfully complete any requests. If capabilities match, the DLT peer (and its connection information) is added to the pool of DLT peers. For all operations, a default timeout of 35s is configured that when reached, sees the contacted DLT peer being discarded (i.e., removed from the pool of DLT peers) and the process returning to the initial step.

The steps of adding a single peer to the pool is repeated until the target size N is reached. At this point, the pool is maintained rather than further extended. For this, any failing DLT peer connection is being removed from the pool and another peer is being added, either actively as an outgoing DLT peer or by waiting for another incoming establishment request, depending on the type of the failed peer and committed to the proportion of incoming and outgoing peers. Furthermore, any incoming DLT establishment request is being responded to, leading to removing an existing incoming DLT peer from the pool if the connection to the new incoming DLT peer is ultimately successful. Outgoing peers are replenished after a time interval, depending on the type of transaction, after which the DLT peer is being replaced with another from the list of outgoing DLT peers. Through these methods, the DLT pool is constantly replenished with randomly renewed DLT peers albeit with a continuing cost for doing so.

Transactions, realizing the interaction patterns outlined in Section 2.1 utilize the pool of DLT peers by sending their requests to all peers in the pool. This sending of transactions may even start already while the pool creation is still ongoing, i.e., once a suitable peer can be obtained from the overall (and still growing) pool of DLT peers. However, transactions will not finish until the requests will have been sent to the number of DLT peers defined as the pool size, i.e., N. With this, latencies for initial transactions are directly dependent on the time it takes to fully establish the pool.

3 Challenges for Users and Provider Networks

Considering the observed communication patterns, we can identify a number of challenges that need better understanding; our results in Section 4 will quantify some of the aspects related to those challenges.

1. **Costs for pool maintenance:** Reachability information is required to interact with other peers. As illustrated in Figure 1, new DLT peers need to

continuously establish and maintain suitable reachability information, resulting in each DLT peer maintaining a constantly changing pool of DLT peers as active (TCP) connections.

2. **Costs for resilience and reliability:** During the pool establishment and maintenance, peers may never reply to connection attempts, resulting in additional latency due to timeouts, prolonging the establishment of the pool of peers to communicate with. Furthermore, peers added to the pool may fail throughout their lifetime, leading to (timeout-inducing) removal from the pool, requiring to replenish the pool with another suitable peer.
3. **Need to match capabilities:** Peers do not know about other peers' capabilities to serve requests when contacting them or being contacted by them. While the discovery in Figure 1a ensures possible reachability, only the pool establishment (Figure 1b) guarantees successful communication with a matching peer, while a capability mismatch leads to discarding the peer and attempting a connection with a new peer albeit having already created costs for the connection to the mismatching peer.
4. **Unicast replication:** We can see from the interactions in Section 2.1 that communication in a DLT is *multipoint* in nature, i.e., transactions or information (such as blocks) are sent to the entire pool of DLT peers, not just a single one. Given the mechanisms explained in Section 2.2, this pool of DLT peers is a randomized subset of the larger overall DLT overlay with the intention to achieve *diffusion* among many DLT peers. This avoids repeated communication with a fixed set of DLT peers, reducing the threat of collusion of information through a malicious set of DLT peer, and ultimately ensure consensus. The consequence of that varying random nature of the multipoint diffusion, however, is that repeated unicast replication (at the initiating DLT peer) is used instead of efficient network-level multicast, which negatively impacts efficiency and transaction completion time.
5. **IP address privacy:** DLT peers need to expose their IP address to the DLT system, replicated to the bootstrap nodes and DLT peers by virtue of the discovery process outlined in Figure 1. This leads to privacy and/or security issues, enabling geo-identification of peers, DoS attacks on particular parts of the DLT, and others.

4 Experimental Insights

Our insights were obtained through an Ethereum based experiment, using the *go-ethereum release V1.10-stable* on a Linux-based machine, operating out of Munich, Germany. For our experiments, we used the same setup as described in [4], with more details and initial insights to be found in [4].

The experiment aims to complete a first transaction that any new peer will initiate as a requirement to be part of the DLT network, namely to download the latest agreed blockchain from the DLT network. For this, the peer initiates a so-called *full synchronization* to replicate the latest full blockchain from the DLT network, where full (and light) synchronization are configurations for a

peer to obtain replicas of the blockchain. The latest blockchain is identified by a *checkpoint*, which is agreed and disseminated in the DLT system.

For this, the initiating peer realizes the steps in Figure 1 to build the pool of DLT peers, over which to execute requests such as *commit* transactions, *diffuse* blocks, as well as to execute a full synchronization. The necessary discovery and communication establishment steps are performed over a geographic spread which included all continents albeit with an expected clustering of nodes across North America, Europe, Asia, and Australia, with only few discovered in South America and Africa as shown in Figure 2a.

Aligned with the process in Figure 1, we categorize the type of communication relation for an initiating peer into outgoing (OUT) and incoming (IN). Furthermore, we define a synchronization process as a *local warm start* when caches for the outgoing relations are re-used from a previous execution, as opposed to a *local cold start* which empties any caches before starting. Note that a local cold start does not assure a remote cold start, leading to the possibility of stale information in the remote cache compared to any newly discovered information in the local cache; we did not specifically explore the impact of this issue, e.g., on security of the overlay.

In our setup, we configure our peer according to a default configuration, thereby seeking to establish relations with $N = 50$ peers out of which $O = 17$ are outgoing, and $I = 33$ are incoming relations. With each run, we execute an unsynchronized *geth* client in warm-start with the goal to ultimately download a full blockchain. Hence, each run re-uses the caches from previous runs, and we set an interval of 2mins between each new run during 01-2022 and 06-2022.

4.1 Topological Measurements

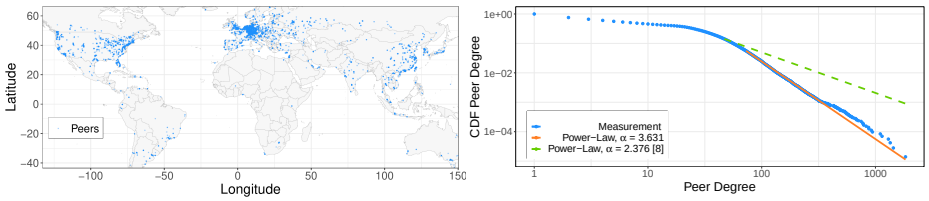

Fig. 2. (a) Peer Geographic Distribution, (b) Peer Degree Distribution

The experiment allowed us to discover 71.5k *active* peers, spread globally as depicted in Figure 2a. The scale of this result is comparable to the one reported in [5, Section IV-A], 74k during 12-2018 and 01-2019, where 30 parity nodes were used only for peer discovery. In Figure 2a, we identify two main clusters conformed of around 200 peers in North America and in Europe. Indeed, clear tails on the geographic distribution in the system can be visualized for Asia, South America, Africa and Australia ranging from clusters of $10 - 20$ peers.

This geographic topology impacts and shapes the statistical properties of the distributed system KPIs, this is discussed in the following sections.

We measure the random variable (RV) *peer degree* over the discovered topology, and show in Figure 2b the resulting distribution, its heavy-tail approximation, and its comparison to the one in [6, Section 4]. This distribution is characterized by a small-degree (≤ 71) phase and a tail (≤ 1844), confirming a plausible Power Law (PL) distribution as already reported in [5, Section 4.C], and [6, Section 4]. In order to formalize these observations we follow [12], arriving to the conclusion that in fact a PL distribution approximation fits our empirical data tail. Hence, we believe our measurements are representative while capturing key communication patterns, and can be replicated, validated and scaled.

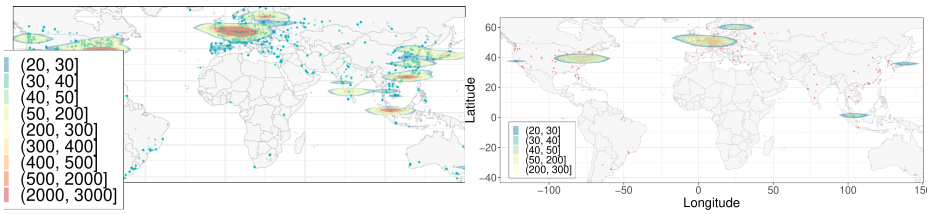

Fig. 3. (a) Incoming Connection Reachability, (b) Outgoing Connection Reachability

In Figure 3, the *long tails* are exposed as a measure of reachability, manifesting the same clustering behaviour as in the geographic topology. But, in this case noticing that the incoming communication relations in Figure 3a are $\approx 10x$ greater than the outgoing communication relations in Figure 3b. We believe that during pool establishment, it is geo-policy dependent to reach certain peers in specific locations. Clearly the peers in Asia are less likely to establish successfully an outgoing communication relation than peers in North America and Europe.

4.2 Types of DLT Peers

To further understand the previously exposed system behaviour we differentiate types of DLT peers that stem from the communication patterns in Figure 1. Specifically, we came to differentiate the following types:

Non reachable peers include all peers that never positively responded to Step 3a of the discovery or step 4b in pool establishment, i.e. the PING/PONG to determine reachability or ENRREQUEST. Reasons here may include that those peers may be located behind a firewall, may be intermittently available (and switched off during the connection attempt), or may simply have left the DLT while still remaining in the list of DLT peers maintained at the bootstrap peers or other peers.

Signalling peers include peers that respond positively to reachability but do not positively succeed in the various signalling steps for the DLT pool inclusion, i.e. step 4a in the discovery phase or the various steps during the pool

establishment phase (ETH-ID resolution in Step 4b, TCP handshake and secure context establishment in Step 5b, or capability exchange in Step 6b).

Dropped data peers successfully complete all discovery and establishment steps, thereby end up in the pool of peers, but still do not provide suitable data upon request (here a valid blockchain information). The reasons here are unavailable information or not completing the transfer of information; blockchain information can be several GB large, so that DLT peers may run out of BW budget or decide to sever the connection because of switch-off or other reasons during the transfer. Hence, while communication does take place, it is not successful with respect to the intended communication and therefore counted as being wasted.

Data peers successfully complete all steps in Figure 1, particularly the intended transfer of DLT-relevant data.

In our experiments, we determined that 13.09% are of the last type, while 0.64% are of the third category, 1.38% are non routable peers and 85.61% are signalling peers. In other words, almost 90% of all attempted relations fail either because of the lack of reachability or mismatching capabilities. Our following experimental insights focus on outlining the costs associated with those failures.

4.3 Costs for Pool Establishment & Maintenance

For outlining the potential costs for the communication patterns in Figure 1, we focussed on the following key performance indicators (KPIs) of random nature:

1. *Pool Establishment Time* for the initiating peer to achieve a default ($N = 50$) pool size.
2. *Pool Establishment Cost* as the relation of the number of peers which established successful communication compared to the total amount of attempted peers until the time of reaching full pool size.
3. *Effective Data Consumption* as the amount of useful data being retrieved from the DLT (i.e. a valid blockchain) versus the total downloaded data.

We now discuss our insights along those KPIs.

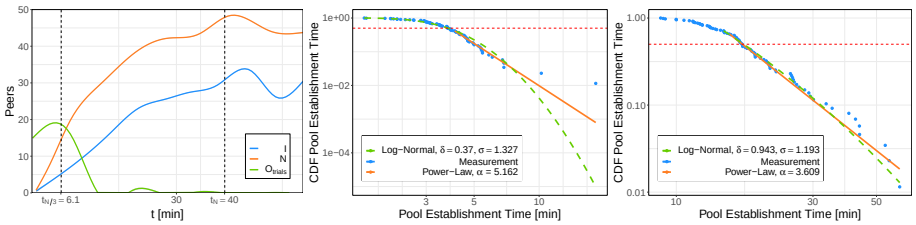

Fig. 4. (a) Single Sample for Pool Establishment Time, (b) $t_{N/3}$, (c) t_N

Pool Establishment Time Figure 4a shows an example iteration that runs until the pool of DLT peers is completed. Here, we identify two main phases when establishing such pool. First, the interval $0 < t \leq t_{N/3}$, where attempts for outgoing connections (O_{trials}) are executed until the number of outgoing connections hit a third of the total amount of connections at time $t_{N/3}$. Second, the interval $t_{N/3} < t \leq t_N$, where the main contributors to the pool N are inbound connections I. After the initiating peer built its contingent outgoing connections, these are maintained during this second period as needed. For instance, we can see the initiation of another outgoing connection at $t = 25$min, likely due to an existing connection failing at that time. In the sample, the first interval is reached after 6.1min, and the second one at 40min.

The RVs $t_{N/3}$ and t_N are measured and approximated for the entire experiment in Figure 4b and Figure 4c respectively [12]. Over the total amount of iterations, the outgoing establishment limit, $t_{N/3}$, is reached at 4min with 50% probability, while the time to complete a pool of DLT peers is reached at 20min with 50% probability, as depicted in Figures 4a and 4b respectively. It is worth to notice that for both RVs a PL and a Log-Normal(LN) approximation are a plausible fit for our measurements.

Pool Establishment Cost For this KPI, we divide costs into those for peer discovery and for pool establishment:

Peer Discovery. For the sample in Figure 4a, a total of 1.9k peers were contacted (Figure 1-Step 2a, UDP PING) out of which reachability with 1.6k peers was ensured (Figure 1-Step 2a, UDP PONG), while 151 peers replied to the discovery (Figure 1-Step 4a, UDP NEIGHBORS). Overall, about 5k peers were added to the list, significantly less than the size of active DLT peers [13].

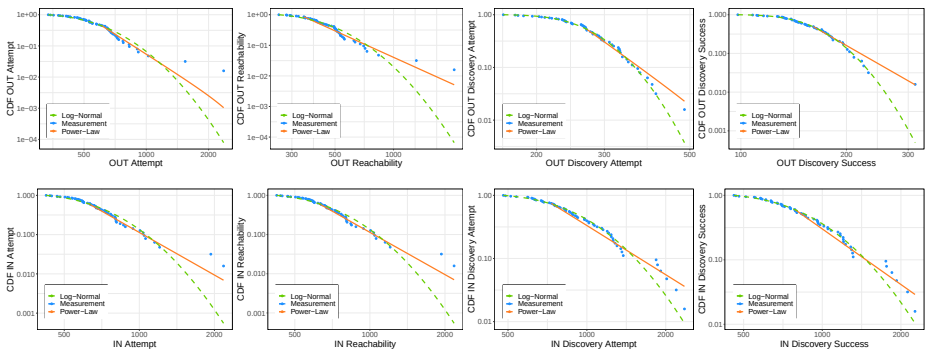

Fig. 5. (a-d) Outgoing, and (e-h) Incoming Discovery KPI Distributions

As before, we analyze the short and long term distribution behaviour for the RVs part of the pool establishment cost in the peer discovery protocol (Figure 1a). And, in Figure 5 we show a compilation of the outgoing and incoming

Table 1. Discovery Cost KPI Summary

	Outgoing						Incoming					
	Power-Law			Log-Normal			Power-Law			Log-Normal		
RV	α	x_{min}	p	μ	σ^2	x_{min}	α	x_{min}	p	μ	σ^2	x_{min}
Attempts	5.24	604	0.2	6.31	0.38	336	4.6	620	0.58	6.47	0.37	423
Reachability	3.87	346	**0.03**	6.04	0.41	257	4.58	619	0.59	6.47	0.37	423
Disc. Req.	6.22	269	0.79	5.61	0.21	164	3.59	732	0.16	6.79	0.41	480
Disc. Success	6.16	161	**0.04**	5.07	0.21	98	3.86	785	0.11	6.72	0.43	459

communication relation distributions. Here, we notice that only the outgoing reachability and discovery success RVs are non-plausible PL distributions in its tails, due to its p-value in Table 1. This, due to its truncated protocol execution bounded by the communication relations rate $\frac{out}{in} = \frac{1}{3}$.

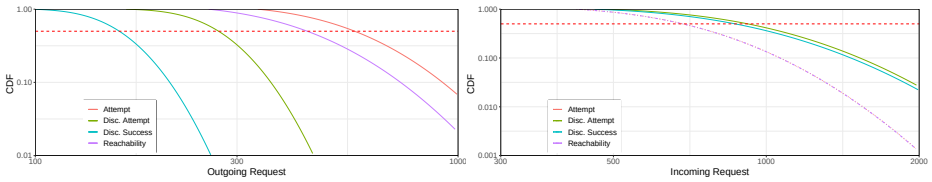

Fig. 6. (a) Outgoing, and (b) Incoming Discovery Cost

Aligned with the results in Figure 5 and parameters in Table 1, we procede to compare these RVs for outgoing communication relations in Figure 6a, and for incomming communication relations in Figure 6b. First, for outgoing communication relations with probability 50%: $\approx$ 77% of the contacted peers are reachable, out of these peers $\approx$ 63% are used to further topology discovery, and $\approx$ 57% successfully executed a complete discovery protocol. Second, for incoming communication relations almost all the reachability requests were positively replied by our peer, and $\approx$ 94% of the discovery requests are successfully completed. Finally, for the long tail, based on the PL approximation we estimate that for outgoing relations with probability 25%: $\approx$ 77.5% of peers were reached, with $\approx$ 33.9% selected for requesting a discovery and $\approx$ 31.46% successfully completing such discovery. Here, we can observe that only about one third of reachable peers are used to perform an actual discovery request.

Pool Establishment. For the sample in Figure 4a, 6.7k peers establish incoming communication to our peer, out of which 1.5k peers were successfully added to the pool. Furthermore, outgoing attempts to 1.8k peers were created, out of which 153 peers were added to the pool.

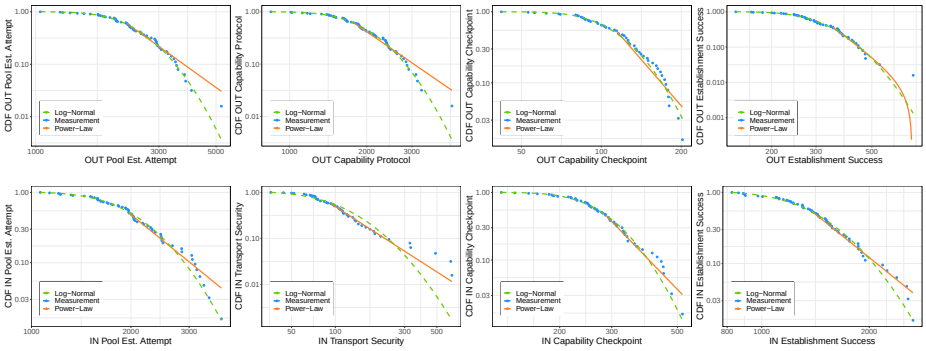

Fig. 7. (a-d) Outgoing, and (e-h) Incoming Pool Establishment KPI Distributions

Table 2. Pool Establishment KPI Cost Summary

	Outgoing						Incoming					
	Power-Law			Log-Normal			Power-Law			Log-Normal		
RV	α	x_{min}	p	μ	σ^2	x_{min}	α	x_{min}	p	μ	σ^2	x_{min}
Attempts	4.52	2250	0.33	7.78	0.29	1041	4.81	1973	0.43	7.6	0.29	1063
Transport Security	3.6	23	0.69	3.09	0.56	5	3.03	93	0.62	4.63	0.62	36
Capability Protocol	4.37	1777	0.25	7.57	0.3	845	6.71	83	0.86	4.32	0.26	44
Capability Checkpoint	5.31	114	0.25	4.73	0.31	42	5.52	286	0.46	5.63	0.28	127
Establishment Success	6.21	350	0.26	5.74	0.28	144	5.04	1383	0.75	7.24	0.3	825

For the entire experiment, the estimated minimum number of incoming peers was 1k peers, out of which 0.8k peers were added to the pool, i.e. a 77.6% success rate for a LN approximation. For outgoing peers, 1.04k peers were attempted, out of which 144 peers were added to the pool, i.e. a success rate of only 13.83% as exposed in Table 2. Furthermore, for the long tail phase, Figures 7e and 7h, the success rate for incoming peers is 70% based on a PL approximation. And, for outgoing peers based on a PL approximation and Figures 7a and 7d, the success performance is equal to 15%.

In Table 2, we further divide the attempts and failures of communication based on the sequences in Figure 1b.

We first list the number of incoming or outgoing attempts, i.e. how many peers undergo the discovery steps and how many outgoing and incoming attempts trigger the establishment steps in 1. The second row shows the amount of peers which failed either resolving an ETHID, establishing a three-way-handshake TCP or negotiating RLPx security [14]. Further, we mark peers not meeting a capability requirement, e.g., the ETH protocol, while row 4 shows peers at which the checkpointed blockchain was not available, i.e. which signalled to not have

the right blockchain information available. Ultimately, the last row shows the number of peer that were included into the pool of DLT peers.

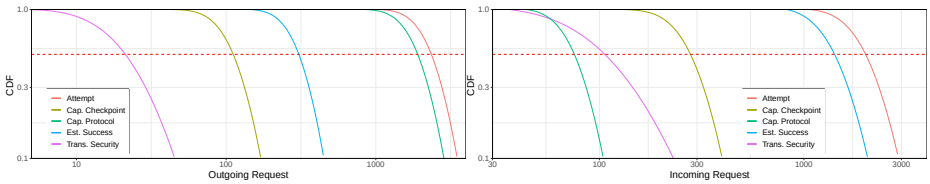

Fig. 8. (a) Outgoing, and (b) Incoming Pool Establishment Cost

Noteworthy here are the discrepancies between failures of incoming and outgoing connections in Figure 8. For instance, outgoing peers have significantly higher transport and security failures, where 82% of errors occur while trying to decrypt the remote secret and find the proper blockchain checkpoint, and 18% I/O errors (invalid ciphertext length, unexpected end of file); we currently explain this issue with stored but outdated cypher information, while outgoing connections (of a new peer) start afresh in their cypher information.

Effective Data Consumption During the experiment, the total downloaded data amounted to 278453.42MB, out of which 88026.44MB were useful data added to the local blockchain, while 190426.98MB were dropped. We interpret this as an effective data consumption ratio of 32%.

5 Discussion

Let us take the insights from the previous sections to outline an agenda for future investigations.

Increasing insights: The results and insights in this paper, e.g., in respect to communication patterns and incurred costs for networks, are only the start. As outlined in Section 2.2, the multipoint replication incurs significant costs on the network; specifying those transmission costs would provide a basis for any potential improvements on any network mechanism addressing this aspect of DLTs. Also, insights on topology dependence, e.g., for localized DLTs in comparison to wide-area DLTs, would be another route for investigation.

Extension to other DLT types: Our insights in this paper are limited to a PoW-based system, i.e. Ethereum, which in turn defines the communication patterns to realize the interactions within such system. Extending our insights into other types of DLTs, particularly those utilizing different ledger validation techniques, such as proof-of-stake, is key. Our ultimate goal here is to develop a methodology that allows for looking at DLTs more generally, e.g., in the form the overlay topology is maintained, the way transactions are communicated and

others, and compare DLTs in terms of the costs they incur when realizing those general aspects.

Network innovations: While the results in this paper may lead to improvements in the realization of specific DLTs at the level of their platform realizations, we also see opportunities for possible network innovations that allow for mitigating the impact of DLT operations. For instance, new constraint-based routing mechanisms [15] may reduce the number of disconnects in the pool establishment due to non-matching capabilities between peers. Also, the use of service-oriented routing [16] may address the disclosure of peers' IP addresses to the overlay system and therefore improve privacy but also security of the participating peers. Moreover, addressing the multipoint replication inefficiency provides a significant, albeit challenging opportunity for network innovations.

Relation to ongoing standardization efforts: Standardization organizations, such as the IETF or the IEEE, have been investigating various aspects of DLTs as a technology enabler for decentralized network and application functions. We believe, however, that our insights from this paper provide useful input into those standardization efforts in providing experimental evidence into some known, albeit so far underspecified impacts that DLTs may bring, supporting the development of future DLTs and their supporting network functions to ensure efficient communication in real-life deployments. For instance, side meetings or dedicated contributions reporting on those insights is a vehicle for bringing those experimental insights into the standardization organizations.

6 Conclusion

The increasing use of DLT has led us to question the possible impact on provider networks. When looking closer at the interactions in a typical DLT, we have identified that inefficiency and waste are important factors that may negatively impact further adoption of DLTs. Understanding the reasons behind both is paramount for thinking of alternatives, both for the use of DLTs (if that is possible within the remit of the considered application) and the realization over provider networks through novel network solutions. With our experimental insights, albeit limited to a specific proof-of-work based DLT system, this paper serves as a starting point for a wider dialogue on the use and impact of DLTs, the relation with and use within provider networks, and the opportunities for addressing the identified problems.

Bibliography

[1] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, and A. Zohar, "Survey on blockchain networking: Context, state-of-the-art, challenges," ACM Comput. Surv., vol. 54, no. 5, may 2021. [Online]. Available: https://doi.org/10.1145/3453161

[2] B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, and A. Mezrioui, "Sok: A comprehensive survey on distributed ledger technologies," in 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2022, pp. 1–16.

[3] H. Howard, "Technical Report," no. 935, 2019.

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

[5] Y. Gao, J. Shi, X. Wang, Q. Tan, C. Zhao, and Z. Yin, "Topology Measurement and Analysis on Ethereum P2P Network," 2019 IEEE Symposium on Computers and Communications (ISCC), pp. 1–7, 2019.

[6] T. Wang, C. Zhao, Q. Yang, S. Zhang, and S. C. Liew, "Ethna : Analyzing the Underlying Peer-to-Peer Network of Ethereum Blockchain," pp. 1–15, 2021.

[7] IEEE, "P2958: Standard for a decentralized identity and access management framework for internet of things," Tech. Rep., 2020.

[8] ——, "P3210: Standard for blockchain-based digital identity system framework," Tech. Rep., 2020.

[9] IETF, "Decentralized internet infrastructure (dinrg)," Tech. Rep., 2022.

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

[11] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," IETF, RFC 8446, Aug. 2018. [Online]. Available: http://tools.ietf.org/rfc/rfc8446.txt

[12] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," SIAM Review, vol. 51, no. 4, pp. 661–703, 2009. [Online]. Available: https://doi.org/10.1137/070710111

[13] Ethereum, "Etherscan," 2022. [Online]. Available: https://etherscan.io/nodetracker

[14] ——, "Ethereum developers," 2022. [Online]. Available: https://ethereum.org/en/developers/docs/

[15] R. Glebke, D. Trossen, I. Kunze, Z. Lou, J. Rüth, M. Stoffers, and K. Wehrle, "Service-based forwarding via programmable dataplanes," in 1st Intl. Workshop on Semantic Addressing and Routing for Future Networks, 2021.

[16] B. C. S. Jiang, G. Li, "A new approach to a service oriented internet protocol," in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020.

7 Protocols

An example for a successful outgoing discovery process is presented below.

```
1  2022-03-03 23:21:19.634361 IP port-217-110-131-84.str.dial.de.
       colt.net.30303 > ec2-13-212-218-222.ap-southeast-1.compute
       .amazonaws.com.30303: UDP, length 134
2  2022-03-03 23:21:19.796806 IP ec2-13-212-218-222.ap-southeast
       -1.compute.amazonaws.com.30303 > port-217-110-131-84.str.
       dial.de.colt.net.30303: UDP, length 157
3  2022-03-03 23:21:20.298709 IP port-217-110-131-84.str.dial.de.
       colt.net.30303 > ec2-13-212-218-222.ap-southeast-1.compute
       .amazonaws.com.30303: UDP, length 171
4  2022-03-03 23:21:20.461449 IP ec2-13-212-218-222.ap-southeast
       -1.compute.amazonaws.com.30303 > port-217-110-131-84.str.
       dial.de.colt.net.30303: UDP, length 1057
5  2022-03-03 23:21:20.461496 IP ec2-13-212-218-222.ap-southeast
       -1.compute.amazonaws.com.30303 > port-217-110-131-84.str.
       dial.de.colt.net.30303: UDP, length 425
6  --
7  2022-03-03 22:57:43.199378 IP 47.243.120.93.42620 > port
       -217-110-131-84.str.dial.de.colt.net.30303: Flags [S], seq
        2245802468, win 64240, options [mss 1460,sackOK,TS val
       2274098869 ecr 0,nop,wscale 7], length 0
8  2022-03-03 22:57:43.199467 IP port-217-110-131-84.str.dial.de.
       colt.net.30303 > 47.243.120.93.42620: Flags [S.], seq
       3694169839, ack 2245802469, win 65160, options [mss 1460,
       sackOK,TS val 4097244013 ecr 2274098869,nop,wscale 7],
       length 0
9  2022-03-03 22:57:43.401346 IP 47.243.120.93.42620 > port
       -217-110-131-84.str.dial.de.colt.net.30303: Flags [.], ack
        1, win 502, options [nop,nop,TS val 2274099063 ecr
       4097244013], length 0
10 2022-03-03 22:57:43.564131 IP 47.243.120.93.42620 > port
       -217-110-131-84.str.dial.de.colt.net.30303: Flags [P.],
       seq 1:548, ack 1, win 502, options [nop,nop,TS val
       2274099234 ecr 4097244013], length 547
11 2022-03-03 22:57:43.564186 IP port-217-110-131-84.str.dial.de.
       colt.net.30303 > 47.243.120.93.42620: Flags [.], ack 548,
       win 505, options [nop,nop,TS val 4097244377 ecr
       2274099234], length 0
12 2022-03-03 22:57:43.584184 IP port-217-110-131-84.str.dial.de.
       colt.net.30303 > 47.243.120.93.42620: Flags [P.], seq
       1:361, ack 548, win 505, options [nop,nop,TS val
       4097244397 ecr 2274099234], length 360
```

Listing 1.1. UDP OUT Discovery Example