Peer Clustering for the InterPlanetary File System

Yannis Thomas
Department of Informatics
Athens University of Economics and Business
Athens, Greece
thomasi@aueb.gr

Nikos Fotiou
Department of Informatics
Athens University of Economics and Business
Athens, Greece
fotiou@aueb.gr

Iakovos Pittaras
Department of Informatics
Athens University of Economics and Business
Athens, Greece
pittaras@aueb.gr

George Xylomenos
Department of Informatics
Athens University of Economics and Business
Athens, Greece
xgeorge@aueb.gr

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

George C. Polyzos
Department of Informatics
Athens University of Economics and Business
Athens, Greece
polyzos@aueb.gr

ABSTRACT

Distributed Hash Tables (DHT) are once again attracting research interest, 20 years after their inception, as a promising solution for scalable and decentralized content storage. A prominent example is the InterPlanetary File System (IPFS), a distributed peer-to-peer storage system with more than 20K online peers, which uses the Kademlia DHT. We design and implement H-Kademlia, a hierarchical version of Kademlia for IPFS, where peers are distributed into disjoint sets of users, or clusters. Peer clustering can offer resilience to network partition, privacy of in-cluster content lookups, as well as improved caching, content filtering and access control. We assess the performance of IPFS over H-Kademlia via simulations that use real traces from the IPFS network. Our findings show that our design delivers the benefits of peer clustering, without significant performance penalties.

CCS CONCEPTS

• **Networks** → **Routing protocols**; **Network simulations**.

KEYWORDS

IPFS, Kademlia, DHT, Clustering

ACM Reference Format:
Yannis Thomas, Nikos Fotiou, Iakovos Pittaras, George Xylomenos, Spyros Voulgaris, and George C. Polyzos. 2023. Peer Clustering for the InterPlanetary File System. In *2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing (FIRA '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3607504.3609289

1 INTRODUCTION

Serving millions of downloads daily and supporting dozens of third-party applications, the *InterPlanetary File System* (IPFS) [5] is an increasingly popular solution for decentralized storage. Currently, the IPFS network is present in over 150 countries and more than 2500 autonomous systems and is growing rapidly [20]. IPFS is a fully decentralized storage system which achieves high reliability via content replication across multiple storage nodes; it is widely used to build highly resilient web sites and web-based applications.

Under the hood, IPFS is an open-source, content-addressable *Peer-to-Peer* (P2P) content storage network. IPFS uses a *Distributed Hash Table* (DHT) for content routing, that is, correlating "who stores" with "what content;" specifically, IPFS relies on a tailored version of *Kademlia* [13]. Similar to other DHTs, in Kademlia each participating node, or *peer*, maintains a number of links to other peers, so as to be able to locate a content item in a logarithmic number of steps. Kademlia is a "flat" DHT in that each peer can maintain links to any other peer. As a result, a single overlay link between two peers can span a large distance over the actual Internet.

An alternative to the flat DHT organization is to start with clusters of peers, for example, a cluster for the peers in each autonomous system, and set up a separate DHT for each cluster. To interconnect these clusters, we can use the *Canon* design paradigm [9], which enables the construction of multi-level, DHT-based overlay networks, through a progressive merging of individual DHT clusters, thus creating a hierarchical DHT.

The resulting network allows locating content and peers in any participating cluster, but with two additional properties: *Locality of intrA-Cluster Paths* (LACP) and *Convergence of intEr-Cluster Paths* (CECP). LACP assures that *lookups* for content located inside a cluster will never exit the cluster. This enhances content availability in case of cluster partitioning, since "local" content remains reachable, and improves privacy, since *lookups* for local content are essentially hidden from the rest of the network. CECP assures that *lookups* for *the same* "global" content item exit the cluster from the same peer. This creates chances for caching, since any request for that content will go through that peer, as well as other network optimizations (e.g., overlay mutlicast delivery trees), and also enables deploying security mechanisms, such as content filtering

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

FIRA '23, September 10, 2023, New York, NY, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0276-1/23/09…$15.00
https://doi.org/10.1145/3607504.3609289

and access control. For example, assuming that all EU peers form a cluster, the LACP property ensures that lookups from peers in the EU for content that resides at peers within the EU will never leave the EU cluster (which is a legal requirement for some types of content, according to the GDPR); the CECP property ensures that all lookups from peers in the EU for a blocked content item that resides outside the EU, can be stopped at a unique EU peer.

The drawback of clustering is that by restricting the links between peers in different clusters to achieve the LACP and CECP properties, paths to content can be made longer and it may be easier for paths to break as peers come and go. Our goal in this paper is to establish whether the cost of clustering is detrimental to IPFS. Therefore, we apply the Canon paradigm to Kademlia, introducing *Hierarchical-Kademlia* (H-Kademlia), and adapt it to IPFS. Our main contributions include (i) the first technical specification of H-Kademlia, (ii) an open-source implementation of H-Kademlia for the PeerNet Simulator,[1] (iii) the integration of H-Kademlia to IPFS and (iv) an evaluation of IPFS over H-Kademlia based on real IPFS traces, showing that the H-Kademlia delivers the benefits of clustering, without significant performance penalties.

The remainder of the paper is organized as follows. In Section 2, we briefly describe IPFS, Kademlia, and hierarhical DHTs. In Section 3, we detail our H-Kademlia design. In Section 4, we introduce the setup of our evaluation. In Section 5, we evaluate the performance of IPFS over H-Kademlia. Finally, in Section 6, we conclude and discuss future work.

2 BACKGROUND WORK

The *InterPlanetary File System* (IPFS) is a P2P system that stores content items in a distributed, decentralized, and collaborative way. Each IPFS node, or peer, can store and lookup content, also participating in the routing of content storage and lookup requests. Pointers to files are replicated on multiple peers in order to provide resilience to failures. Redundancy is driven by a demand-oriented replication policy that correlates replication to popularity, thus enhancing scalability. Currently, the IPFS network consists of roughly 20K active peers.[2] Recent research indicates that IPFS suffers from high rates of churn: 87.6% of user sessions last for less than 8 hours [20]. It also suggests that IPFS is challenged by slow content publication, which takes over 33s for 50% of the publications, since the high churn rate leads to inaccurate routing state and timeouts.

DHTs are fundamental components for many P2P systems, mapping *identifiers* (IDs) to *values* and supporting storing and looking up content in a distributed manner. IPFS uses a tailored version of the *Kademlia* DHT [13], adding the Coral [8] and S/Kademlia [4] extensions. Kademlia uses the XOR metric to organize the network. Every peer (and content) is assigned a unique ID of size s (256 bits in IPFS). Kademlia peers maintain s *KBuckets*, storing information about remote peers. The x^{th} KBucket of a peer stores information about peers that have a common ID prefix of length $x-1$ with the peer's ID. Given an ID, Kademlia locates the (XOR-based) "closest" peers to that ID in a logarithmic number of steps.

The Kademlia protocol introduces four types of messages: *PING, STORE, FIND_PEER*, and *FIND_VALUE*. A PING message is sent to a receiver peer in order to verify that it is alive. A STORE message conveys a request for storing a <ID, value> pair at the receiver peer. A FIND_PEER message asks the receiver peer for the closest peers to a given ID. A FIND_VALUE message requests the value associated with a given ID which is stored at the receiver peer.

For content *Lookup*, the *inquiring* peer, which starts the process, sends in parallel kad_A (3 in IPFS) *FIND_VALUE* messages to the peers closest to the content's ID, until either the entire network is searched or the ID is found. When an inquired peer does not have the content, it returns the closest peers to the requested ID based on its local KBuckets, thus allowing the inquiring peer to move closer to its target. Content *Store* is based on the same recursive process. The difference is that the inquiring peer first locates the kad_K (20 in IPFS) closest peers to the content's ID and then sends a STORE message to each of them. A peer stops its search when the lists of IDs in the responses do not contain any peer IDs closer to the content ID compared to those already discovered.

When a peer *Joins*, it uses a "bootstrapping peer," acquired through an out-of-band mechanism. The new peer inserts the bootstrapping peer into the appropriate KBucket, and then performs a Lookup for the new peer's ID. The new peer forwards to the bootstrapping peer a Lookup for its local ID. During the Lookup process, the new peer adds to its KBuckets information about the peers with IDs closest to its own, and vice versa. Note that the KBuckets are updated every time a new peer ID is found through any received message. When a peer *Leaves*, it does not explicitly notify anyone; Kademlia removes a peer from a KBucket after a message to it remains unanswered for a predefined period.

Canon is a design paradigm that enables the construction of multi-level DHT-based overlay networks, through a progressive merging of individual DHTs, assuming a hierarchical structure of the different peer clusters [9]. A Canon-based DHT achieves the LACP and CECP properties mentioned in the previous section. A brief discussion regarding the application of the Canon paradigm to Kademlia is presented in [9]; to the best of our knowledge, this paper is the first work to document, implement and experimentally evaluate such a design.

Literature on hierarchical DHTs demonstrates specific advantages over traditional flat designs [1]. Specifically, studies that systematically compare hierarchical and flat DHTs suggest that hierarchical designs present reduced cost of maintenance and size of routing state [11], better fault isolation and enhanced provisioning of content placement, at the cost of higher intercluster traffic [14] and longer routing paths [10].

Hierarchical DHTs can be grouped into two categories [1, 21]: *horizontal*, in which all peers are equal, and *vertical*, in which a relatively small group of "super" peers behave as proxies that interconnect clusters. In a horizontal system, hierarchy is built by merging clusters; during this process, all peers add links to a number of peers in sibling DHTs. In a vertical system, a tree-like hierarchy is created from self-contained DHTs, where transition from a lower-level cluster to an upper-level one is realized only through super peers which are members of both clusters. Artigas et al. [1] verify the claim of [9] that horizontal hierarchical DHTs achieve better load balancing and greater resilience to failures compared to vertical designs. In addition, horizontal systems, which can be built by slightly modifying existing flat DHT designs, allow for code

[1] https://github.com/PeerNet/PeerNet.git
[2] https://trudi.weizenbaum-institut.de/ipfs_analysis.html

reuse and (possibly) incremental deployments (i.e., co-existenceof a flat DHT and its corresponding hierarchical variant). Furthermore, they do not impose any requirements on how peers and content identifiers are constructed. Finally, they do not require special purpose, powerful super peers to take on extra responsibilities. Such a requirement would contradict the decentralized nature of IPFS, introducing security and privacy concerns.

A vertical design for H-Kademlia is introduced in [12]; it uses super-peers for inter-cluster communication and correlates clusters with peer identifiers. That work does not delve into the implementation specifics of the protocol and only examines the design analytically. Our H-Kademlia design follows instead the horizontal paradigm, with cluster-independent peer IDs and no super peers. This is critical for IPFS, where peers and content identifiers are cryptographically associated with specific physical peers and content items, respectively, thus preventing the use of [12].

H-Pastry [7] is probably the most similar hierarchical DHT to H-Kademlia. H-Pastry is a multi-level DHT implemented by adapting the Pastry DHT following the Canon approach and by adding support for multihoming and peering relationships. H-Pastry cannot be easily used for IPFS which is deeply intertwined with Kademlia. In contrast, H-Kademlia can introduce hierarchical clustering into IPFS, without altering the IPFS implementation itself.

3 H-KADEMLIA DESIGN

Our H-Kademlia design introduces a modified KBucket maintenance process which differentiates *local* peers (from the same cluster) and *remote* peers (from a different cluster). In H-Kademlia, a peer can maintain links to *any* local peers. However, it can only maintain a link to a remote peer *if* it is the *XOR-wise closest* one to that remote peer among *all* peers in its cluster. Therefore, a peer A performs the following checks before inserting a candidate peer C in its KBuckets:

- If peer C is remote, then peer A searches for the peer in its cluster that is the closest one to C.
  - If that is *not* A, then A does *not* store a link to C.
  - Otherwise, A inserts a link to C in its KBuckets; we say that A becomes its cluster's *gateway* peer for C.
- If peer C is local, then it is always added in A's KBuckets; in addition, A removes any links to remote peers that are closer to C than to A, as C should become their new gateway peer.

Figure 1 shows the H-Kademlia operations during a Store and a Lookup, using a network of four peers in two clusters; the degree of replication (kad_K) is 2, meaning that each content item is stored in 2 peers, and the degree of parallelism (kad_A) is 1, meaning that each lookup is not sent in parallel to multiple peers. We also assume that the peers have already populated their KBuckets with peer IDs.

Figure 1(a) shows the resolution of a Store request where peer 010 wants to store some content with ID 110. The cluster's gateway peer for ID 110 is peer 111, hence only local peer 111 has a link to remote peer 110 in its KBuckets. Peer 010 starts by sending a *FIND_PEER* message to peer 111, which is the closest peer to the content ID in its KBuckets (message 1); peer 111 sends back peer ID 110 and the ID of one of the other two peers which are close to the content ID (message 2); peer 010 sends a request to the next closest non-inquired peer, that is, peer 110 (message 3), which finds and returns the IDs of the kad_K closest peers to the content ID (message 4). As no new peers were discovered, peer 010 concludes that the process is done and sends *STORE* requests to the kad_K peers closest to ID 110, namely, peers 111 and 110.

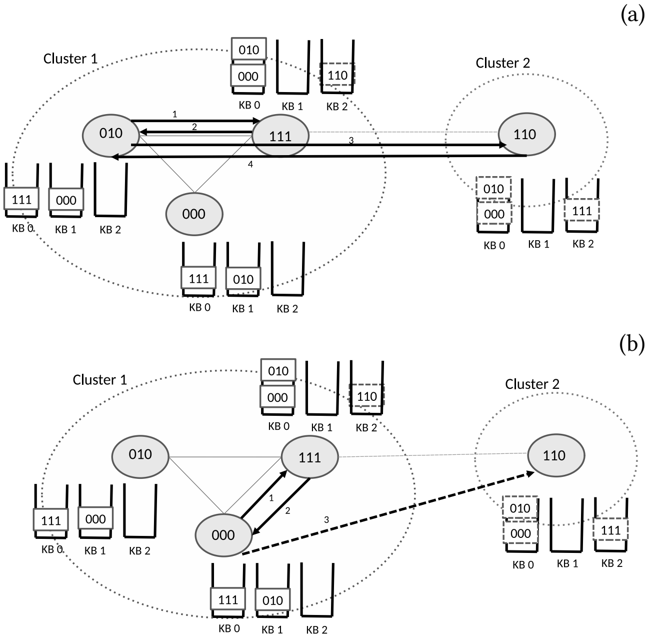

Figure 1: H-Kademlia resolving (a) a Store and (b) a Lookup request in a network of four peers (circles) in two clusters (dotted circles). The lines indicate overlay links, the arrows indicate messages and the numbering shows the sequence of the messages.

Figure 1(b) shows the resolution of a Lookup request where peer 000 searches for content ID 110; the gateway peer for content ID 110 is peer 111. If peer 000 does not find a locally stored copy of the content, it sends a *FIND_VALUE* request to the closest peer to that content ID in its KBuckets, peer 111 (message 1). Peer 111 searches locally for the content, and if the search fails it replies with the IDs of the kad_K closest peers in its KBuckets (message 2). Peer 000 sends a request to peer 110 (message 3), which satisfies the request.

These examples illustrate the two properties of the Canon paradigm (and, by extension, H-Kademlia). First, all requests that concern a non-local peer ID exit the cluster through the same gateway peer (CECP). This property is useful for caching, as it ensures that all lookups for a content ID will go through the gateway peer. Second, all requests that can be resolved locally, never exit the cluster (LACP). This property is useful for implementing traffic control policies, including security and access control mechanisms. Both properties arise due to the decision to only store a link to a remote peer at the peer closest to it. Note that as the only change to flat Kademlia is the policy of inserting links to the KBuckets, H-Kademlia can be gradually adopted by the IPFS network.

Peers need a mechanism to decide whether other peers are local or remote, in order to implement H-Kademlia. This can be an out-of-band mechanism, such as an online service, or a distributed procedure that each peer can perform. Depending on the clustering scheme, different mechanisms may be appropriate. For a fixed clustering method, e.g., based on the peer's country of origin, an out-of-band bootstrap mechanism can assign cluster IDs to the new peers and respond to cluster ID queries. For a dynamic clustering method, e.g., based on inter-peer latency, the peers should exchange clustering-specific messages in order to infer and share changes in network topology. Without loss of generality, we assume that each peer knows the cluster it belongs to and shares this information with other peers by including it in Kademlia messages.

Caching can deliver an additional performance gain to hierarchical DHTs by resolving requests within a cluster. In our system, each peer opportunistically stores the traffic it forwards. Without loss of generality, we assume a *First In First Out* (FIFO) replacement policy, according to which each received content item is inserted in the cache, evicting the oldest content item when cache is full. Thereafter, upon the receipt of a *Lookup* request, each inquired peer searches its cache and, if the item is found locally, it returns it to the inquiring peer. Responses from remote peers are also cached at the gateway peer for their content ID; future requests for the same content will always flow through the gateway peer, due to the CECP property.

H-Kademlia can be used as a defense mechanism against common DHT threats. For example, due to the LACP, H-Kademlia guarantees that Lookup requests for locally stored content will always succeed, even when the DHT network is under an external *Eclipse* attack [18], i.e., an attacker "eclipses" some peers by "poisoning" the routing protocol. H-Kademlia is resilient to this type of attack since routing entries for "local" peers have higher priority.

The clustering scheme is orthogonal to the design of H-Kademlia since, as mentioned previously, the only requirement for H-Kademlia to operate is that peers should be able to decide whether another peer is local or remote. However, the clustering scheme determines what we can achieve by clustering the peers. For instance, a *latency-based clustering*, that "minimizes" the intra-cluster latencies, is expected to mitigate the lookup latency when the content source and the peer that emits the Lookup request reside at the same cluster. Note that, in H-Kademlia, a latency-based clustering policy would act as the second routing metric, akin to Pastry's ability to consider the link latencies when updating the routing table [15].

In our experiments, clusters were based on the peer's country of origin. This policy is appropriate for supporting security mechanisms and access control policies for multi-level DHTs. For example, due to international content distribution agreements, content availability can be restricted within a country; the LACP property ensures that these requests will be satisfied locally. Peers within the same country are likely to experience lower latency, thus combining management and performance considerations. At a second level, we could group all EU countries into a merged cluster, ensuring that EU originated requests for EU hosted content do not leave the EU, as mandated by the GDPR for some types of content.

Alternatively, by forming clusters based on trust metrics, rather than topology-based metrics, H-Kademlia can provide resilience against *Denial-of-Service* (DoS) attacks, where attackers refuse to forward or respond to a query [6]. Such a trust-based clustering approach can also be used as a building block for a query privacy mechanism [2]: in H-Kademlia a query is initially routed through intra-cluster (trusted) peers and only if no result is found, it exits the cluster through the gateway peer, which can act as an obfuscation point for the query.

4 EVALUATION SETUP

We have implemented Kademlia and H-Kademlia on *PeerNet*, a well-established simulator for P2P networks, using real IPFS traces to drive the evaluation. We selected the following metrics to explore the performance of H-Kademlia:

- Path length (*hops*): the distance that a request travels until it reaches its destination, in *overlay* hops.
- Latency (*ms*): the time between the emission of a request and its completion, that is, when one peer is found for a Lookup, or all peers are found for a Store.
- Success ratio (%): the ratio of Lookups that are successfully resolved.
- Routing state (*peers*): the number of peers that are maintained in the KBuckets.
- Store receivers (*peers*): the number of peers that are selected to store a content item in a Store.
- Inter to Intra-Cluster Ratio: the ratio of inter-cluster to intra-cluster messages in Lookups and Stores.

To evaluate the performance of H-Kademlia under realistic conditions, we started with traces of actual IPFS network traffic provided by Nebula, an IPFS crawler.[3] We analysed the traces and inferred the value distributions of the parameters critical to our evaluation, such as peer churn, network topology, network size, content popularity, and request rate. Subsequently, we fed values from these distributions to our simulations. This allowed us to simulate diverse scenarios, rather than the specific scenarios provided by the traces. As the generation of the input parameters was stochastic, we tested multiple input data sets and we present the average measurements.

Churn. To describe churn, we consider three peer session parameters: the overall up-time, the number of sessions (or re-connections) and the down-time between sessions. We infer the up-time distribution from [16] (Fig. "peer classification"). Traces in [19] (Fig. "inter-arrival time") depict the CDF of the unavailability period (between two consecutive sessions) in the IPFS network. From the same traces, we determined the number of sessions that the peers conducted within the trace period and inferred the distribution that relates the percentage of peers to the number of re-connections. For the simulations, we generated a random churn profile for each peer according to these distributions.[4]

Network topology. Network topology consists of peer location and overlay link latency. We inferred the peer location distribution from [17] (Fig. "geolocation of on peers"). Then, we synthesized a link-latency matrix at the country level (for scalability) based on publicly available global RTT measurements made by cloud providers.[5] By combining the location distribution and the latency matrix, we approximated the latency between every pair of peers.

[3] https://github.com/dennis-tra/nebula

[4] As we combine the results of crawls with different timespans, we scale them down to a two-day duration, the minimum one among the crawls.

[5] https://learn.microsoft.com/azure/networking/azure-network-latency

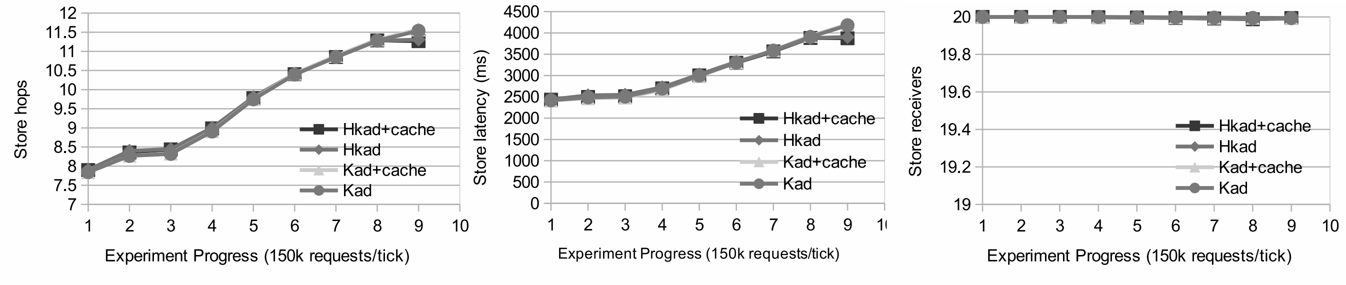

Figure 2: Performance comparison of H-Kademlia and Kademlia: Store requests.

Network size. Due to churn, the number of (unique) peers *seen* differs from the number of *up* peers at any specific time. We determined the number of *unique* peers from the number of *up* peers and the mean up-time of peers. The number of up peers is 10K to 12K [19] (Fig. “crawl time series"). The mean up-time of peers in [16] (Figs. “peer classification” and “reliability”) suggests a 3.7 ratio of active to up peers, hence our simulations use 37K unique peers; this is consistent with the 54K unique peers in 7 days from [19].

Content Popularity and Request Rate. Content popularity in IPFS is assessed in [3], which reports the total number of Lookups received for a particular content item over a given period; this study indicates that the IPFS network handles roughly 10M requests daily. Hence, we generated a traffic workload with size equal to the overall requests, i.e., 20M for 2 days, and roughly 20K individual content items. Due to computational and memory constraints, each experiment simulates 1.5M requests (3.6 hours of IPFS operation). We are not aware of any studies characterizing Stores for IPFS or similar P2P networks, hence we also use the distribution of [3] for Stores, assuming an 1:4 ratio of Stores to Lookups.

Our setup does not consider the correlation between different input parameters, e.g., different churn rates for different countries; peers are randomly assigned location, reliability and other parameters. The configuration parameters of (H-)Kademlia are those used in IPFS: 256 bit IDs, kad_A=3 and kad_K=20. The error detection timeout is 2s and we explore four different IPFS modes: Kademlia, Kademlia with caching, H-Kademlia and H-Kademlia with caching.

5 PERFORMANCE EVALUATION

5.1 Store requests

In Fig. 2 we present the metrics related to Stores: *path length*, *latency* and *store receivers*. All figures show the evolution of the metrics as the 1.5M requests are being issued. Hierarchy and caching does not seem to affect any of the metrics. In parallel, we notice a increasing trend in hops and latency suggesting that the metrics may have not yet converged, which is in accordance with the findings of paper [20] that reports that “publication” latency can reach up to 150-s. The application of the hierarchical scheme and caching does not improve the performance of the Store process but does not make it worse either.

5.2 Lookup requests

In Fig. 3 we present the metrics related to Lookups: *path length (in hops)*, *latency* and *success ratio*. The results indicate that the number of hops grows by approximately 0.5 hops with H-Kademlia, but is reduced by roughly 0.2 hops with caching. The difference is more evident for latency, where the hierachical design increases latency by roughly 150 ms and caching offers approximately 50 ms latency reduction. The success ratio is affected as well: H-Kademlia has an approximately 5% lower success ratio; caching increases this success ratio by approximately 3% (slightly less for flat Kademlia).

Since the difference in path length is not significant, we infer that the latency overhead of H-Kademlia can be attributed to presenting more re-transmissions. The hierarchical design reduces the number of acceptable peers in the Kbuckets (see also Fig. 4, where the KBucket size in hierarchical mode is consistently lower), making it harder to locate peers under peer churn. To verify our inference, we ran the experiment without churn and found that H-Kademlia and Kademlia present similar latency.

Finally, we observe that caching does not have a substantial effect on the Lookup performance of H-Kademlia for two reasons. First, Lookups are sent in parallel to kad_A=3 “one-hop-away” peers, therefore content can be found at two other local peers besides the gateway peer, hence caching cannot greatly exploit the CECP property. Second, the cached responses in our experiments comprise roughly 23% of the successful Lookups, but they are only shorter by around 0.5 hops (compared to the origin peer). Caching should have more prevalent effects on larger networks, but since in Kademlia path length grows logarithmically to the network size, this would mean *very* large networks.

5.3 Topology awareness and clustering

In Fig. 4 we present metrics related to topology awareness and clustering: *KBucket size* and *Inter:Intra-cluster ratio* for Stores and Lookups. The routing state (KBucket size) is consistently smaller for H-Kademlia; there are roughly 20 fewer peers compared to Kademlia. Although this issue does not severely challenge operation, it explains the small performance disadvantages of H-Kademlia seen in previous experiments. With no churn and complete topology awareness, the number of peers in the KBuckets would be 200 peers, hence the churn observed in the IPFS traces penalizes the topology awareness of the peers by 20% to 30%.

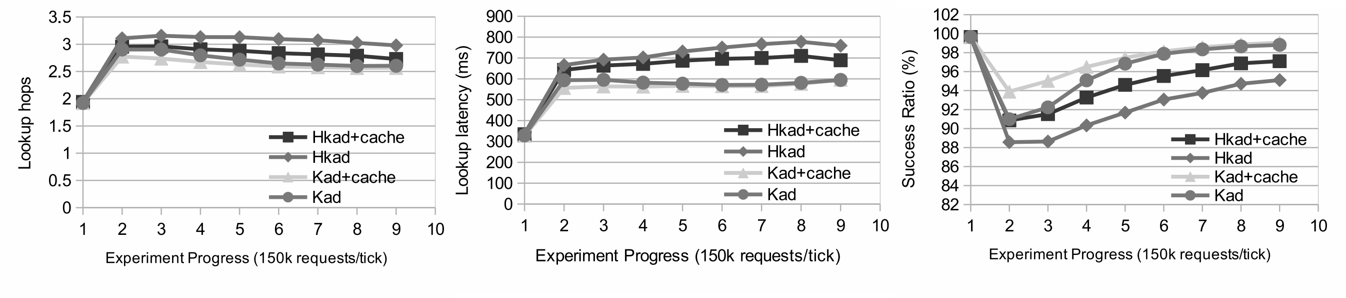

Figure 3: Performance comparison of H-Kademlia and Kademlia: Lookup requests.

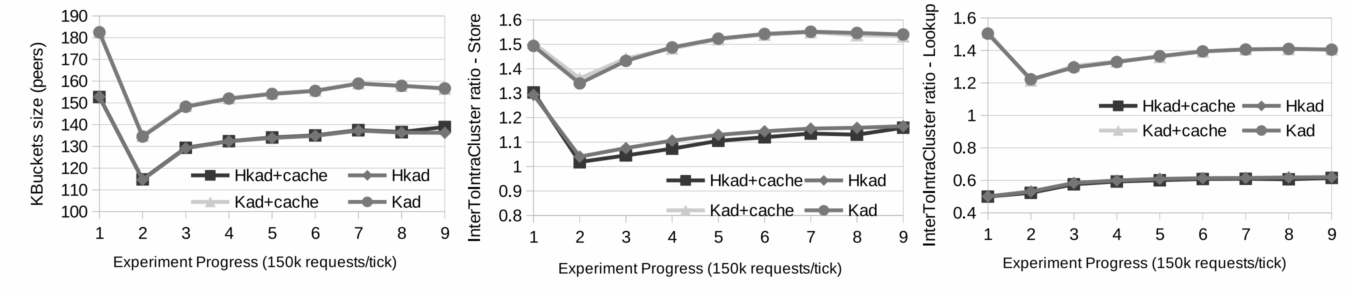

Figure 4: Performance comparison of H-Kademlia and Kademlia: Topology awareness and clustering.

For the "Inter:Intra-cluster" message ratio for Lookups and Stores, note first that the clustering for flat Kademlia is virtual; the metrics are estimated a posteriori based on the clustering used in the equivalent H-Kademlia experiments. We observe that H-Kademlia significantly increases message locality for both Stores and Lookups, respectively. Specifically, for H-Kademlia the ratio of inter-cluster to intra-cluster messages is 1.15 and 0.6 for Stores and Lookups, respectively, compared to 1.55 and 1.4 for flat Kademlia, thus a significant portion of traffic is confined within the cluster.

In addition, we observe that caching can reduce the overall distance to discovered peers; the discovered network "shrinks" around the reference peer because caching reduces the length of the paths. This assumption is indicated by the inter-to-intra cluster ratio of messages sent during Stores, shown in Fig. 4 for H-Kademlia, where caching reduces the number of inter-cluster messages, thus favoring "shorter" intra-cluster links.

5.4 Discussion

Our evaluation suggests that H-Kademlia offers the benefits of clustering to IPFS without significant performance penalties. Even though the parallelism and extensive content replication of Kademlia reduce the performance gains promised by the CECP property, the LACP property is prevalent in H-Kademlia. In the following, we summarize these effects:

- The routing state of H-Kademlia is smaller, since the inter-cluster peers are filtered.
- The performance of Lookups is slightly worse for H-Kademlia, due to routing through the gateway peer.
- Locality is significantly increased by H-Kademlia, for both Lookups and Stores.
- Caching is not effective in H-Kademlia, due to the parallelism of requests.
- Caching can reduce the topological awareness of Kademlia nodes by reducing the length of paths.
- Our findings show similar trends with the actual measurements of IPFS in [20], i.e., timeouts due to unresponsive peers penalize the latency of Stores but not Lookups.

6 CONCLUSIONS AND FUTURE WORK

This paper introduced H-Kademlia, a hierarchical DHT design based on Kademlia and the Canon paradigm. We designed, implemented and evaluated H-Kademlia in PeerNet, as a candidate DHT for the growing IPFS network. In our simulations, we considered real IPFS traces that capture user activity in terms of user location, churn, and content requests. Our findings show that in the case of IPFS, H-Kademlia can enhance the locality of requests, reducing the cost of inter-cluster traffic and facilitating the implementation of several security mechanisms, at the cost of a minor performance loss in Lookup latency and success ratio.

Due to space limitations, we did not consider alternative clustering approaches. To enhance performance, a fine-tuned clustering policy that leverages Kademlia's parallelism and content replication could deliver better results. To enhance security, clustering mechanisms based on peer profiles, i.e., white lists and cryptography support, can instead be considered. Finally, an important topic for future work is evaluating the performance of IPFS under an incremental deployment of H-Kademlia to selected peer clusters.

ACKNOWLEDGEMENT

We would like to thank Yiannis Psaras and Dennis Trautwein for their comments on the operation of IPFS and their insights on the Nebula logs. This work was supported in part by the Research Center of the Athens University of Economics and Busines

REFERENCES

[1] Marc Sanchez Artigas, Pedro Garcia Lopez, and Antonio F. Skarmeta. 2007. A Comparative Study of Hierarchical DHT Systems. In *Proc. of the IEEE Conference on Local Computer Networks (LCN)*. IEEE, New York, NY, USA, 325–333.

[2] Michael Backes, Ian Goldberg, Aniket Kate, and Tomas Toft. 2012. Adding query privacy to robust DHTs. In *Proc. of the ACM Symposium on Information, Computer and Communications Security*. ACM, New York, NY, USA, 30–31.

[3] Leonhard Balduf, Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. 2022. Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS. arXiv:2104.09202 [cs.NI]

[4] Ingmar Baumgart and Sebastian Mies. 2007. S/Kademlia: A practicable approach towards secure key-based routing. In *Proc. of the International Conference on Parallel and Distributed Systems (IPDPS)*. IEEE, New York, NY, USA, 1–8.

[5] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. arXiv:1407.3561 [cs.NI]

[6] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S Wallach. 200f2. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review* 36, SI (200f2), 299–314.

[7] Nikos Fotiou, Konstantinos V. Katsaros, George Xylomenos, and George C. Polyzos. 2015. H-Pastry: An inter-domain topology aware overlay for the support of name-resolution services in the future Internet. *Computer Communications* 62 (2015), 13–22.

[8] Michael J. Freedman, Eric Freudenthal, and David Mazières. 2004. Democratizing Content Publication with Coral. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 18.

[9] Prasanna Ganesan, Krishna Gummadi, and H. Garcia-Molina. 2004. Canon in G major: designing DHTs with hierarchical structure. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, New York, NY, USA, 263–272.

[10] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. 2003. SkipNet: A Scalable Overlay Network with Practical Locality Properties. In *Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS)*. USENIX Association, Berkeley, CA, USA, 14 pages.

[11] Yuh-Jzer Joung and Jiaw-Chang Wang. 2007. Chord2: A two-layer Chord for reducing maintenance overhead via heterogeneity. *Computer Networks* 51, 3 (2007), 712–731.

[12] Isaias Martinez-Yelmo, Ruben Cuevas, Carmen Guerrero, and Andreas Mauthe. 2008. Routing Performance in a Hierarchical DHT-based Overlay Network. In *Proc. of the Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, New York, NY, USA, 508–515.

[13] Petar Maymounkov and David Mazieres. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. of the International Workshop on Peer-to-Peer Systems*. Springer, Berlin, Heidelberg, 53–65.

[14] Josyl Mariela B. Rocamora and Jhoanna Rhodette I. Pedrasa. 2016. Evaluation of hierarchical DHTs to mitigate churn effects in mobile networks. *Computer Communications* 85 (2016), 41–57.

[15] Antony Rowstron and Peter Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, Berlin, Heidelberg, 329–350.

[16] Zhenyang Shi. 2021. Nebula Crawlings. https://github.com/wcgcyx/nebula-crawler/tree/812f33515342321461e39b17ac02a91858926e14.

[17] Zhenyang Shi. 2021. Nebula Crawlings. https://github.com/wcgcyx/nebula-crawler#6-correlation-between-uptime-and-geolocation.

[18] Emil Sit and Robert Morris. 2002. Security considerations for peer-to-peer distributed hash tables. In *Proc. of the International Workshop on Peer-to-Peer Systems*. Springer, Berlin, Heidelberg, 261–269.

[19] Dennis Trautwein. 2021. Nebula Crawlings. https://github.com/dennis-tra/nebula-crawler-reports/blob/main/2021/calendar-week-44/ipfs/README.md.

[20] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. 2022. Design and evaluation of IPFS: a storage layer for the decentralized web. In *Proc. of the ACM SIGCOMM Conference*. ACM, New York, NY, USA, 739–752.

[21] Peng Wang, Julong Lan, Yuxiang Hu, and Shuqiao Chen. 2015. Towards locality-aware DHT for fast mapping service in future Internet. *Computer Communications* 66 (2015), 14–24.