KadRTT: Routing with network proximity and uniform ID arrangement in Kademlia

Hidehiro Kanemitsu
School of Computer Science,
Tokyo University of Technology,
Tokyo, Japan
kanemitsuh@stf.teu.ac.jp

Hidenori Nakazato
School of Fundamental Science and Engineering,
Waseda University,
Tokyo, Japan
nakazato@waseda.jp

***Abstract*—Distributed Hash Table (DHT) has been widely applied to peer-to-peer (P2P) applications for efficient content lookup mechanisms. IPFS, one of the distributed systems for sharing files, websites, and data on a worldwide scale, adopts Kademlia as one of the DHTs, that is included in libp2p on the network layer of IFPS. Though DHT-based content lookup can scale in the number of peers, how to control lookup performance, e.g., lookup latency and lookup hop count, is one of the issues for a very large-scale network. Thus, it is necessary to guarantee both the lookup latency and the hop count even if the network scale becomes larger.**

In this paper, we propose a Kademlia alternative, called KadRTT, that reduces both the lookup latency and hop count. KadRTT has two functionalities, i.e., (i) RTT-based lookup target selection not to increase the maximum hop count, and (ii) uniform ID arrangement for each k-bucket to shorten the initial ID distance from content ID. Experimental results by the simulation show that KadRTT outperforms other Kademlia-based DHTs in terms of lookup latency and hop count.

***Index Terms*—Kademlia, DHT, libp2p, RTT, peer-to-peer**

I. Introduction

The current information network requires a large volume of data, including streaming format, a small chunk of data arisen from IoT sensor devices, and so on. Peer-to-peer (P2P) technologies have been spreading out for mainly sharing the data among users. Since each node (hereinafter, we call "peer") interacts with each other without a mediator, it is a natural consequence that how to address the target location, including the counterpart peer itself as well as the target content. In this context, Distributed Hash Table (DHT) has been widely applied to P2P applications for efficient content lookup mechanisms. IPFS [1], one of the distributed systems for sharing files, websites, and data on a worldwide scale, adopts Kademlia as one of the DHTs, that is included in libp2p [2] on the network layer of IFPS. Libp2p consists of many functionalities such as routing, exchange, discovery, naming, and so on. In particular, the routing functionalities in libp2p consists of Kademlia [3], S/Kademlia [4] for secure routing, and Coral DSHT [5] for regional and hierarchy-based DHT. Though DHT-based content lookup can scale in the number of peers, how to control lookup performance, e.g., lookup latency and lookup hop count, is one of the issues for a very large-scale network. Thus, it is necessary to guarantee both the lookup latency and the hop count even if the network scale becomes larger. One idea for optimizing the lookup latency is a proximity-based lookup [6] for optimizing the RTT for each hop. However, if the ID distance between the content ID (CID) and the target peer ID is made larger than the original distance (i.e., shortest ID distance), the total hop count can be increased. Thus, criteria for suppressing the hop count as well as minimizing the RTT for each hop are required.

In this paper, we propose a Kademlia alternative, called KadRTT, that reduces both the lookup latency and hop count. KadRTT has two functionalities, i.e.,

(i) RTT-based lookup target selection not to increase the maximum hop count.
For each iterative lookup, a client tries to send request to the next hops according to increasing order of RTT if it satisfies the logarithmic condition: $\frac{d(p_i^{RTT},CID)}{d(p_i^{kad},CID)} < 2$.

(ii) uniform ID arrangement for each k-bucket to shorten the initial ID distance from content ID.
KadRTT coordinates each k-bucket in order that each ID is uniformly distributed not to bias a specific ID range to suppress redundant lookup hops.

Experimental results by the simulation show that KadRTT outperforms other Kademlia-based DHTs in terms of lookup latency and hop count.

II. Background

A. Kademlia basics

Firstly we describe the basics of content lookup procedures of Kademlia as shown in Fig. 1. A client peer is supposed to receive the content request including the content ID (CID). Then let the r-th k-bucket be the closest in all k-buckets. To accelerate the lookup procedures, Kademlia allows concurrent lookup by defining α as the degree of lookup concurrency. "FindNode request" message is sent by RPC to α peers included in the r-th k-bucket, then they return the list of next hops as "FindNode response". The number of next hops in a FindNode response is defined as "resiliency" that we denote as β. In Kademlia, each peer has a pool to maintain the list of next hops, and β next hops for each FindNode response are put into the pool, i.e., in total $\alpha\beta$ next hops are put into the

ISBN 978-3-903176-39-3

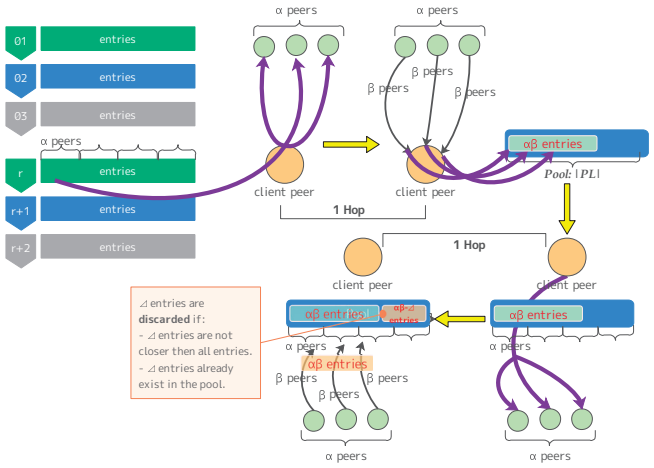


Fig. 1. Lookup procedures of Kademlia in libp2p.

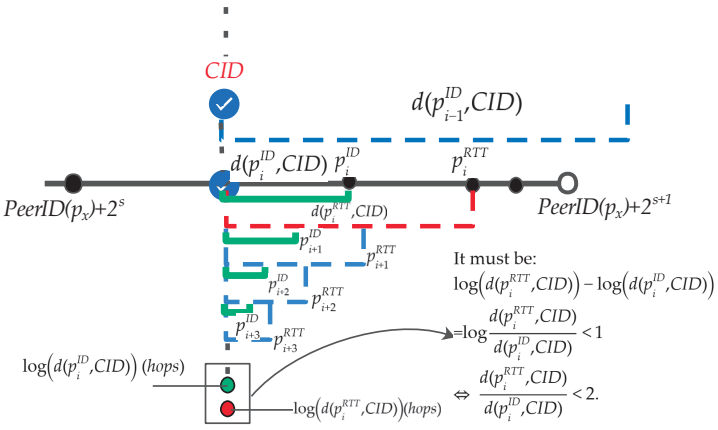


Fig. 2. Condition for RTT-based lookup

pool. If the remaining space of the pool is more than $\alpha\beta$, the $\alpha\beta$ next hops are put into the pool; otherwise, each next hop must be checked whether it has been already asked or not. If so, the next hop is discarded. When all next hops in the pool are traced and CID is not found, the client tries to send FindNode request to the next α peers in the r-th k-bucket.

B. Factors for lookup performance

From lookup procedures presented in Section II-A, we can observe that α affects the lookup latency, and β affects the content hit rate and hop count. However, if ether α or β is increased, the total number of messages in terms of FindNode becomes larger, thereby the available bandwidth for content downloading becomes smaller. The literature [7] presents that $\alpha = 3$ is an optimal value for taking the balance between the lookup latency and the number of messages for the specific network environment. However, such a value depends on the system condition and we investigate the advantage of KadRTT with varying both α and β.

As a more critical factor, the initial ID distance between CID and each peer ID affects the total hop count. Here let define the ID distance between a peer and CID as $d(p_x, CID)$, where p_x is an entry in the specific k-bucket. Since the bound for the hop count is $\log(d(p_x, CID))$, at least such a bound must not be increased if candidate selection criteria are altered by other algorithms such as KadRTT.

III. KadRTT algorithm

A. Overview of KadRTT

To address the requirements described in Section II-B, KadRTT has two points, i.e., (i) hop count-aware RTT-based lookup to guarantee that the maximum hop count equals that of original Kademlia, and (ii) ID arrangement for each k-bucket to make the initial ID distance shorter. In the following sections, we present details of (i) and (ii), then how to implement KadRTT is presented.

B. Hop count-aware RTT-based lookup

In Kademlia, α peers are selected from the k-bucket with the specific index, where their peer ID is close to CID. The problem is that such a selection does not consider the latency for each lookup. Thus, as a conventional approach such as R/Kademlia [6] selects α peers according to RTT by exchanging peers when each peer receives FindNode responses. Though each peer in a k-bucket has a shorter RTT than them in the case of Kademlia, the initial ID distance, i.e., $d(p_x, CID)$ may be longer. As a result, the total lookup latency may be optimized.

In KadRTT, each RTT during a lookup is made shorter by equalizing the maximum hop count with Kademlia in order to suppress the total lookup time. Here, we formulate the difference of the maximum hop count between KadRTT and Kademlia. Let the client peer as p_x, and CID is the content ID. At the i-th iterative hop count, let p_x determines the s-th k-bucket record is the nearest among all k-bucket records in the routing table. Thus, suppose

$$2^s \leq d(p_x, CID) < 2^{s+1}. \tag{1}$$

p_i^{kad} is the entry in s-th k-bucket record selected by Kademlia. p_i^{RTT} is the one selected by KadRTT. Then we assume

$$\begin{aligned} &2^s \leq d(p_x, p_i^{kad}), d(p_x, p_i^{RTT}) < 2^{s+1}, \\ &d(p_i^{kad}, CID) \leq d(p_i^{RTT}, CID), \end{aligned} \tag{2}$$

because p_i^{RTT} is the selected one by RTT, not by ID distance such as Kademlia. Fig. 2 shows bounds on the difference in terms of the ID distance among Kademlia and KadRTT. From Fig. 2, a bound on the difference in terms of the ID distance can be formulated as follows. The condition for equality of the maximum lookup hop count between Kademlia and KadRTT

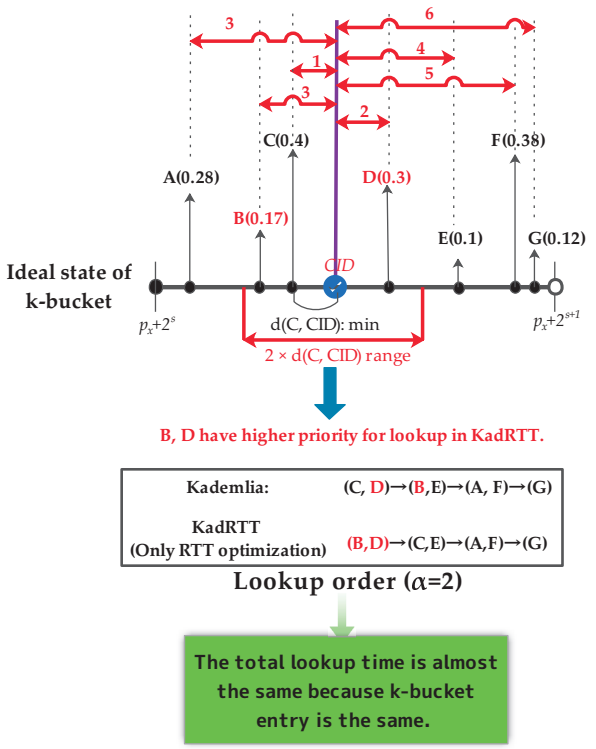


Fig. 3. Problem in RTT-based lookup

is:

$$\begin{aligned}&\log(d(p_i^{RTT}, CID)) - \log(d(p_i^{kad}, CID)) \\ &= \log \frac{d(p_i^{RTT}, CID)}{d(p_i^{kad}, CID)} < 1 \\ &\Leftrightarrow \frac{d(p_i^{RTT}, CID)}{d(p_i^{kad}, CID)} < 2. \end{aligned} \tag{3}$$

Thus, if the above condition is satisfied by selecting p_i^{RTT}, the maximum hop count is the same as the one in Kademlia. Furthermore, RTT for each iterative lookup is reduced by selecting p_i^{RTT} for each i-th lookup. Thus, in KadRTT, in the s-th k-bucket, the peer satisfying (3) is selected with the highest priority for the lookup target.

Though the firstly selected peers have shorter RTT to the client, the difference between KadRTT and Kademlia is the lookup order because every entry in the k-bucket is equal to each other. Fig. 3 shows the effect by varying the lookup target selection order, where $\alpha = 2$ and each value in brackets is the RTT from p_x. The selection order in Kademlia is $(C, D), (B, E), (A, F), (G)$ according to the increasing order of the distance from CID. We assume that C has the shortest ID distance from CID and E is the shortest RTT from p_x. As for KadRTT, since B and D satisfy the condition defined at (3), they become the first targets for the lookup. thus, the lookup order is, $(B, D), (C, E), (A, F), (G)$. From this result, we can see that only the lookup order is varied from Kademlia to KadRTT, thus the difference in terms of the lookup time depends only on the iteration index. As long as the entry in the k-bucket is not changed, the lookup time is not drastically changed.

C. ID arrangement for each k-bucket

In Kademlia, a new entry is added during an iterative lookup if the k-bucket is not full. That is, the original Kademlia accepts any IDs and it drops the new entry if the k-bucket is full. As a result, there is a possibility that many IDs are located in the specific ID range in the k-bucket and if a CID is near from a "void" area, the lookup up hop count may take higher. Thus, it is necessary to make IDs distributed uniformly in order not to create a void area. At the same time, such IDs must have short RTTs with the client peer. In KadRTT, if a new entry p_{new} having the following condition, is exchanged with the existing one p_{old}.

$$\begin{aligned}&T_{rtt}(p_x, p_{new}) < T_{rtt}(p_x, p_{old}), \\ &\sigma^2(b_s^x/\{p_{old}\} \cup \{p_{new}\}) = \min_{p_i \in b_s^x} \left\{\sigma^2(b_s^x/\{p_i\} \cup \{p_{new}\})\right\}. \end{aligned} \tag{4}$$

where $T_{rtt}(p_x, p_i)$ is the RTT from p_x to p_i, p_x is the lookup client and b_s^x is the s-th k-bucket of p_x. And $p_{old} \in b_s^x$. σ^2 is the variance in terms of the ID distance among adjacent peers $d(p_i, p_{i+1})$, where $p_i, p_{i+1} \in b_s^x$. In case that a new peer p_{new} is being put into the k-bucket, the procedure for (4) is performed. Fig. 4 shows the result of the ID arrangement in KadRTT. The ID arrangement procedures start when the k-bucket is full, i.e., the number of entry is k. If p_{new} satisfies the condition of (4), it is switched with p_{old} and thereby each adjacent ID distance approaches to $\frac{2^s}{k}$.

D. Implementation

Other than the above presented two functionalities, KadRTT assumes that each k-bucket entry has the RTT from the client; that is, the client obtains the latest RTT value for each k-bucket entry with one of the existing measurement method for RTT. Since Kademlia does not have any methods for measuring RTT, additional communication for obtaining the RTT is needed. In R/Kademlia[6], each peer obtains the RTT through active PING messages, but the total message number is increased, thereby the frequency should be adjusted in order not to affect the available bandwidth. Though KadRTT does not have any requirements for an RTT measurement method, RTT should be obtained without depending on Kademlia protocol for not affecting on findNode procedures. Network coordinate systems [8], [9], [10], [11].

As for hop count-aware RTT-based lookup in Section III-B, the target selection criteria for FindNode requests must be modified. Fig. 5 shows an algorithm for hop count-aware RTT-based lookup. The parts to be modified are: the client sends FindNode requests (a) to the selected α peers from the k-bucket, and (b) to the selected α peers from the pool. Both cases require the sorted list of next hops by increasing the order of RTT. Then from the first entries, the logarithmic condition defined at (3) is checked. If an entry satisfies the condition, it becomes a candidate for the FindNode target.

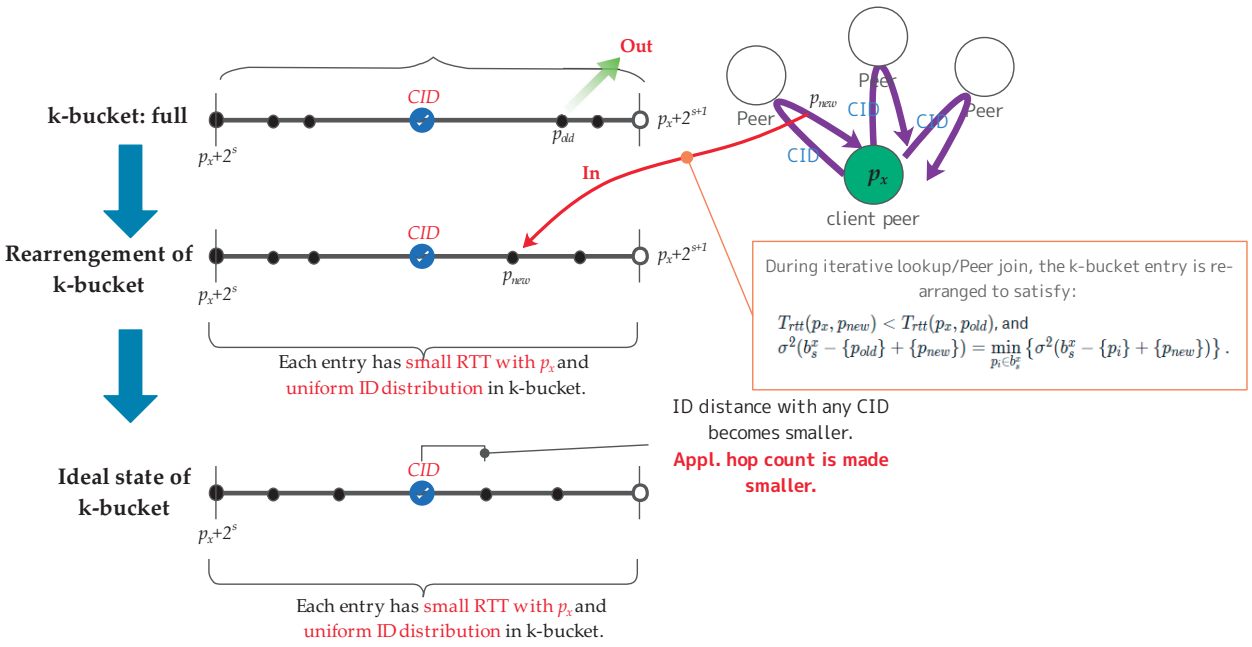


Fig. 4. ID re-arrangement if KadRTT

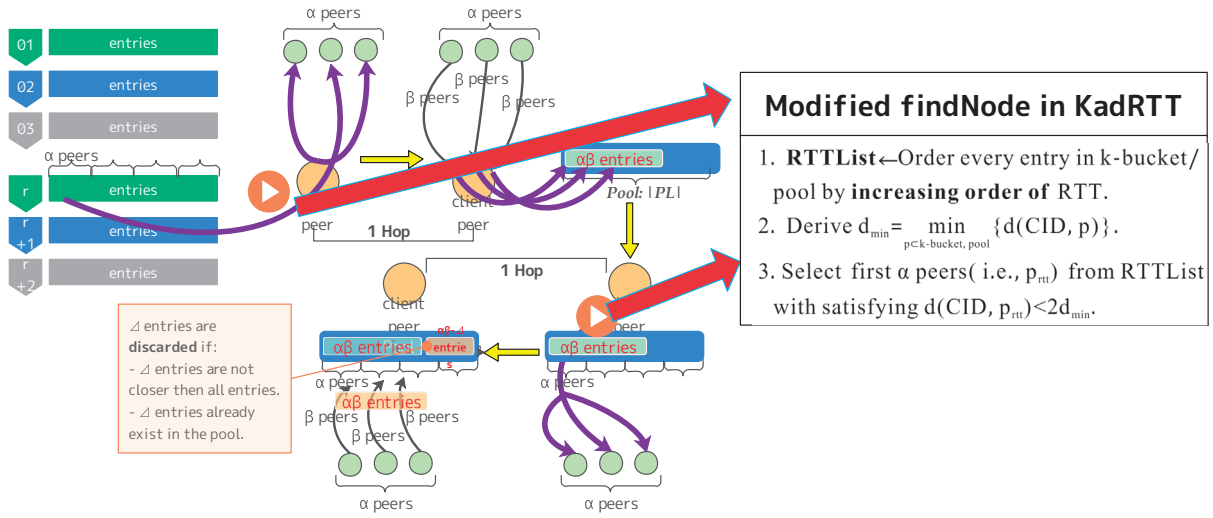


Fig. 5. Problem in RTT-based lookup

IV. Experimental Results

In this section, we present experimental results by the simulation in terms of the lookup latency, hop count, and the number of messages with varying several factors.

A. Simulation setup

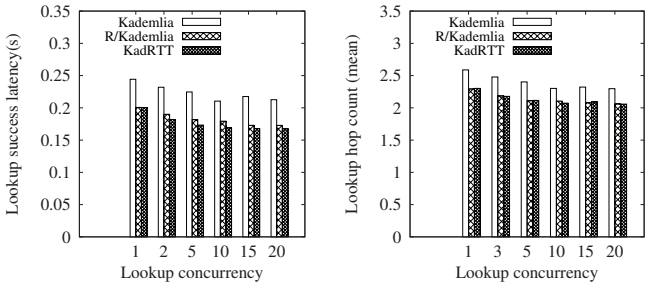
As comparison targets for KadRTT, we selected Kademlia and R/Kademlia. We compared on OverSim [13] that can simulate underlay communication as well as overlay. In OverSim, many parameter can be defined before the simulation, we set the parameter as shown in Table I.

TABLE I
Simulation Parameters

Parameter	Value
♯ of peers	3000
Sim. Time	500 (s)
Churn model	Pareto churn [12]
Lookup concurrency (α)	variable
Resiliency (β)	variable
k-bucket size (k)	20
Lookup type	Iterative lookup
Application type	KBRTest

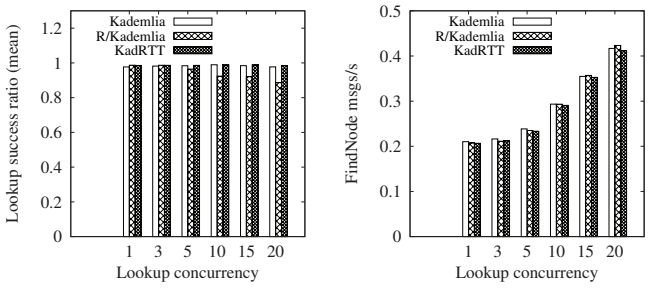
B. Comparison by varying lookup concurrency

Fig. 6 and 7 show the comparison results with varying α and $\beta = 1$; that is, it is a worst-case in terms of content



(a) Lookup success latency (b) Lookup hop count

Fig. 6. Comparison of lookup by varying α with $\beta = 1$.



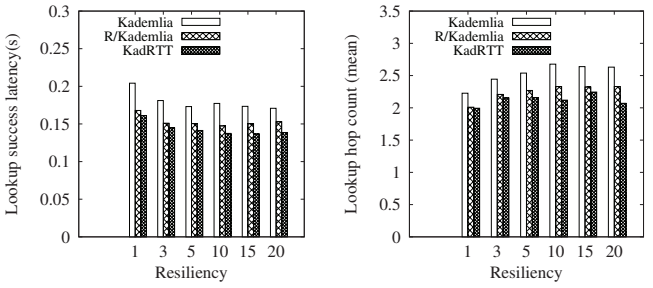
(a) Lookup success ratio (b) FindeNode message ♯

Fig. 7. Comparison of success ratio and message ♯ by varying α with $\beta = 1$.

hit ratio for each α. The comparison results shown in those figures are lookup latency, hop count, lookup success ratio, and the number of FindNode messages. From Fig. 6(a), KadRTT outperforms others, but the difference is slight to R/Kademlia. At Fig. 6(b), the hop count of KadRTT is almost the same as R/Kademlia. However, From Fig. 7(a), the lookup success ratio of R/Kademlia gets worse with increasing α, while the ratio of KadRTT is stable in all α. Since the selected peers for FindNode request in R/Kademlia are based on RTT, not ID distance. Thus, the next hop included in a FindNode response may not be appropriate in terms of the ID distance. However, KadRTT imposes the constraint for the ID distance defined by (3), thereby the ID distance during FindNode procedures can be made shorter per hop even if $\beta = 1$. From Fig. 7(b), the number of FindNode messages in KadRTT is the smallest, because the content hit is achieved with shorter RTT and smaller hop count, thereby the required number of FindNode messages becomes smaller than that of both Kademlia and R/Kademlia.

From the obtained results by Fig. 6 and 7, we can conclude:

- If β is too small with varying α, the difference of both the lookup latency and hop count between KadRTT and R/Kademlia is slight, but KadRTT outputs a shorter lookup latency.
- The content hit ratio by R/Kademlia gets worse if α is larger and β is too small.



(a) Lookup success latency (b) Lookup hop count

Fig. 8. Comparison of lookup by varying β with $\alpha = 10$.

C. Comparison by varying resiliency

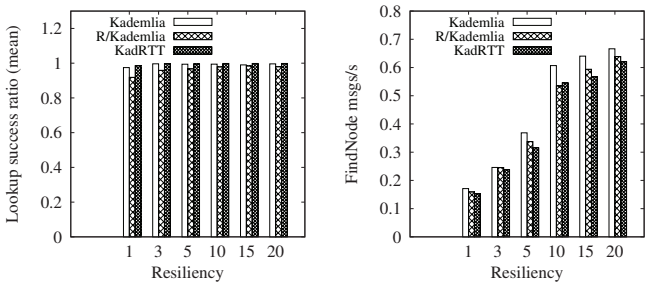
In this comparison, we varied β with $\alpha = 10$ because libp2p sets α set the default value as 10 and Kademlia output the shortest lookup latency in own result shown in Fig. 6(a). Varying β can affect the content hit ratio as well as the lookup latency, however, the number of FindNode messages may be increased drastically.

Fig. 8(a) shows the comparison results in terms of the lookup latency. We can see that KadRTT outperforms others and the difference between KadRTT and R/Kademlia is larger than that of the case of $\beta = 1$, i.e. Fig. 6(a). Thus, β should be taken to a value higher than "1". Af for Fig. 8(b), KadRTT outperforms others in terms of the hop count. Thus, if $beta$ is higher, the content hit with smaller hop counts in KadRTT. From Fig. 9(a) the lookup success ratio in R/Kademlia is higher than the case of $\beta = 1$, i.e., Fig. 7(a). However, the ratio of R/Kademlia is still the worst in every value of β. In Fig. 9(b), the number of FindNode messages increases with higher $beta$, but that of KadRTT is the smallest. From results in Fig. 8 and 9 we can conclude:

- If β is higher, KadRTT outperforms others in terms of the lookup latency and the difference between KadRTT and R/Kademlia becomes larger.
- If β is higher, KadRTT outperforms others in terms of the hop count.
- If β is higher, the success ratio of R/Kadmelia is improved, but is still the worst.
- The number of FindNode messages can be suppressed if either the lookup latency or the hop count is improved.

V. Conclusion

In this paper, we proposed a Kademlia alternative, called KadRTT to optimize the lookup latency. KadRTT has two approaches to achieve the objective. The first one is hop count-aware RTT-based lookup target selection. And the second one is ID arrangement for each k-bucket so that each adjacent ID becomes located uniformly to make the initial ID distance shorter. From the experimental comparisons by the simulation, we found that KadRTT outperforms other approaches in terms of both lookup latency and the hop count. Since KadRTT does not impose any additional communication among peers,



(a) Lookup success ratio (b) FindeNode message ♯

Fig. 9. Comparison of success ratio and message ♯ by varying β with $\alpha = 10$.

the number of FindNode messages is not increased. Thus, KadRTT has a practical nature for P2P applications.

As our future works, we will simulate in a larger network, where 10000 or more peers to confirm the practicality of KadRTT. Then, we explore the optimal value for both the lookup concurrency and the resiliency to adjust them dynamically depending on network structures.

ACKNOWLEDGMENT

The work leading up to this paper has been supported by a gift from Protocol Labs under the grant titled as “RFP 7: Multi-Level DHT Design and Evaluation”.

REFERENCES

[1] J. Benat, “IPFS - Content Addressed, Versioned, P2P File System, ” *ArXiv*, 11 pages, 2014.
[2] libp2p Web site: https://libp2p.io/.
[3] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the XOR metric,” *in Lect. Notes Comput. Sci.,* , vol. 2429, pp. 53–65, 2002.
[4] I. Baumgart and S. Mies, “S/Kademlia: A practicable approach towards secure key-based routing,” *in Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 2, 2007, doi: 10.1109/ICPADS.2007.4447808.
[5] M. J. Freedman and D. Mazieres, “Sloppy hashing and self-organizing clusters,” *in Lect. Notes Comput. Sci.*, vol. 2735, pp. 45–55, 2003.
[6] [1] B. Heep, “R / Kademlia : Recursive and Topology-aware Overlay Routing,” *in Proc. 2010 Australasian Telecommunication Networks and Applications Conference*, pp. 102–107, 2010.
[7] D. Stutzbach and R. Rejaie, “Improving lookup performance over a widely-deployed DHT,” *in Proc. IEEE INFOCOM2006*, 2006.
[8] P. Francis et al., “IDMaps: A global Internet host distance estimation service,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 5, pp. 525–540, 2001, doi: 10.1109/90.958323.
[9] M. Costa, M. Castro, A. Rowstron, and P. Key, “PIC: Practical internet coordinates for distance estimation,” *in Proc. Int. Conf. Distrib. Comput. Syst.*, vol. 24, pp. 178–187, 2004, doi: 10.1109/icdcs.2004.1281582.
[10] P. Sharma, Z. Xu, S. Banerjee, and S. J. Lee, “Estimating network proximity and latency,” *Comput. Commun. Rev.*, vol. 36, no. 3, pp. 39–50, 2006, doi: 10.1145/1140086.1140092.
[11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: A decentralized network coordinate system,” *Comput. Commun. Rev.*, 2004, vol. 34, no. 4, pp. 15–26, doi: 10.1145/1030194.1015471.
[12] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, “Modeling heterogeneous user churn and local resilience of unstructured P2P networks,” *in Proc. Int. Conf. Netw. Protoc. ICNP*, pp. 32–41, 2006, doi: 10.1109/ICNP.2006.320196.
[13] I. Baumgart, B. Heep, and S. Krause, “OverSim: A Flexible Overlay Network Simulation Framework,” *in Proc. 2007 IEEE Global Internet Symposium*, pp. 79–84, May, 2007.