R/Kademlia: Recursive and Topology-aware Overlay Routing

Bernhard Heep
Institute of Telematics
Karlsruhe Institute of Technology (KIT)
Zirkel 2, D–76131 Karlsruhe, Germany
Email: heep@kit.edu

***Abstract*—Structured peer-to-peer overlays offer a basis for a broad range of applications, such as distributed storage, application layer multicast, and mobility support. This paper introduces R/Kademlia, a low bandwidth and low latency variant of the popular overlay protocol Kademlia. R/Kademlia utilizes recursive overlay routing instead of iterative lookups of keys, thus a higher routing performance can be achieved in network scenarios with churn rates that are characteristic for public KAD networks. Both Proximity Routing and Proximity Neighbor Selection are effectively applicable, which leads to a further decrease of routing latencies. Similar to the original protocol, nodes in a R/Kademlia network meet other nodes during user-triggered routing procedures, hence no expensive periodic tasks are needed. For the simulative evaluation of R/Kademlia, the Performance vs. Cost framework (PVC) and the overlay framework OverSim were employed, where both routing modes—iterative and recursive—were simulated and compared in different network scenarios with varying churn rates.**

I. Introduction

The deployment of peer-to-peer applications (P2P) has continually increased during the last several years. Up to now, various kinds of P2P systems have been developed. They form the basis for decentralized directory services, multi-media applications, and on-line games. Additionally, they can provide application layer multicast (ALM) and ID/locator splitting to solve many problems of today's Internet. Commonly, *overlay networks* i.e. logical networks on top of the physical network structure, provide the basis for most P2P systems.

One focus in research into overlay networks are structured P2P overlays, which are often referred to as distributed hash tables (DHTs). As they all provide a common service for applications—the key-based routing (KBR) service [1]—they are denoted as KBR protocols in this paper. KBR protocols build overlay structures like rings or hypercubes and offer the following service: Delivering messages with a destination key to the overlay node that is currently responsible for the range of keys the destination key belongs to. In dynamic networks, issues like node mobility and churn have to be considered when designing a new KBR protocol, so maintenance is an important task here, as nodes permanently join and leave the overlay or change their IP address, mostly without notifying the nodes they hold in their routing tables (in this paper these nodes are called *peers*).

Kademlia [2] is one of the most popular KBR protocols as it has a simple structure and metric (XOR) and is deployed with widely distributed Internet applications like eMule and several BitTorrent clients that utilize its features as a DHT for decentralized file source searching and tracker-less resource localization, respectively. One of its prominent features is that Kademlia nodes meet their peers during application-triggered lookups and not by performing expensive periodic maintenance tasks. While Kademlia is very stable under churn it suffers from high routing latencies caused by its exhaustive iterative lookup procedure. Additionally, this iterative lookup procedure is problematical in networks using *Network Address Translation* (NAT) or *Port Address Translation* (PAT), as it is unknown to the lookup initiator which of the nodes it has to contact might be inaccessible. Using recursive KBR routing, only nodes that have been successfully contacted before are involved in routing procedures.

The contribution of this paper is as follows: We introduce R/Kademlia, a recursive variant of the widely distributed Kademlia protocol. R/Kademlia has low bandwidth needs while achieving lower routing latencies than the original protocol. Two alternative signaling modes are presented, which show different qualities e.g. in NAT/PAT scenarios. We introduce *Proximity Routing* (PR) for R/Kademlia, based on the prefix metric, and show that PR and *Proximity Neighbor Selection* (PNS) can be applied more effectively to R/Kademlia than to the iterative routing mode, which is shown in the evaluation section.

The rest of the paper is organized as follows: First, the iterative and recursive routing modes in structured P2P overlays are analyzed and compared. In the next section, the original Kademlia protocol with its exhaustive lookup procedure is briefly described. Then, the common topology adaptation methods that can be applied to R/Kademlia are introduced. In the next section, R/Kademlia is described and illustrated in detail. After that, the evaluation method and the results are presented and discussed. The paper ends with a survey of related work and the conclusion with an outlook on future work.

II. Recursive vs. Iterative Routing

In KBR protocols, there are basically two alternative routing modes for locating the responsible nodes for a destination key and delivering a message that can be distinguished: The *iterative lookup* of keys and *recursive routing*.

In iterative mode, usually $\mathcal{O}(logN)$ nodes have to be asked sequentially for other nodes closer to the destination key y until the corresponding destination node Y is identified. Then the message is sent directly to Y. To find all nodes that are close to y with respect to the identifier space, usually more overlay nodes have to be contacted and asked for the closest

978-1-4244-8172-9/10/$26.00 ©2010 IEEE

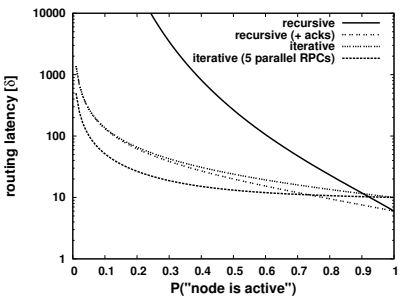


Fig. 1. Routing latencies depending on node activity

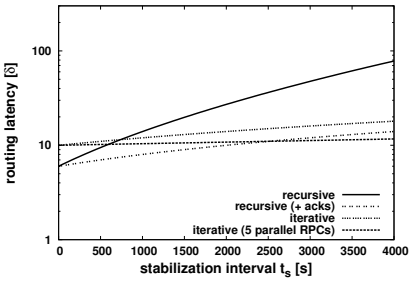


Fig. 2. Routing latencies depending on stabilization interval

nodes they know. Iterative lookups can be accelerated by using parallel *Remote Procedure Calls* (RPCs) i.e. sending lookup messages to different nodes at the same time. This way, the total latency of the lookup can be decreased, as failed nodes do not delay the lookup procedure due to the fact that other asked nodes are still alive and respond. Additionally, since physically closer nodes respond faster and so new met nodes can be asked earlier, the lookup gets further accelerated. Since the initiator of an iterative lookup can control parallelism, he is able to ensure that each parallel RPC is sent to a node that has not been contacted before (if needed), thus avoiding duplicates. The main drawback of using parallel RPCs is that they come with a higher bandwidth consumption.

In the recursive mode, $\mathcal{O}(logN)$ overlay hops are needed to reach the destination node Y, the message is sent from hop to hop—where the next hop is always the closest known node according to the KBR metric (greedy routing)—until the destination is reached. After forwarding the message to its first hop A_1, the message's originator loses control over the message and its routing path. Issuing parallel queries in recursive mode is possible, but it is difficult to ensure that these queries do not converge, i.e. contact the same set of nodes, hence this can lead to duplicate messages. Hop-by-hop acknowledgements on the routing path are essential for low routing latencies, as—when using acks—the occurrence of failed nodes on the path does not results in a rollback of the whole routing procedure but only in forwarding the message to an alternative node.

A detailed analysis of both routing modes can be found in [3], where the authors show, that in moderate churn scenarios recursive routing with activated hop-by-hop acknowledgements is superior in most network scenarios to iterative routing using parallel RPCs regarding routing latencies. Fig. 1 illustrates some of their results: Here, the latency of a single routing procedure (an RPC to a key and the corresponding response) with 5 hops is plotted on the y-axis with different values for the probability of a node being alive on the x-axis. The routing latency is stated as a multiple of the average one-way latency δ in the concerned network. The effect of parallel RPCs in iterative mode and of hop-by-hop acknowledgements in recursive mode is noticeable.

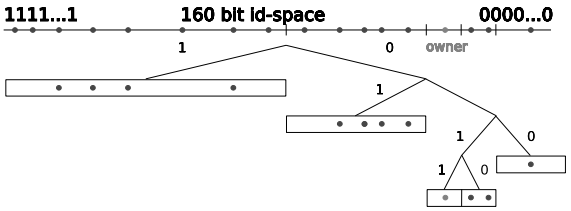


Fig. 3. k-buckets filled with NodeHandles

We used the model for the lifetime of overlay nodes in public KAD networks proposed in [4][5] (Weibull distributed session time: $k = 0.5, \lambda = 5,000$) and combined it with the analysis of churn discussed before. The results are shown in Fig. 2, which illustrates the routing latencies of recursive and iterative routing modes for different stabilization intervals t_s i.e. the amount of time passing by until a peer is probed or contacted. It is observable that using this lifetime model, recursive routing with hop-by-hop acknowledgements achieves lower routing latencies compared to iterative routing with 5 parallel RPCs up to a stabilization interval of $t_s \approx 2,500\,s$. The effective stabilization interval caused by continuously operated application-triggered overlay routing usually goes far below this value, thus the usage of a KBR protocol which supports recursive routing is reasonable in these network scenarios.

III. Iterative Lookups in Kademlia

Kademlia in its original version is only able to perform iterative lookups. During these lookups, information about nodes is sent back in RPC responses and these nodes A_i are put into a sorted vector of the closest nodes (the result vector $\overline{V_y}$) to y, according to the XOR distance $d_{\mathrm{XOR}}(A_i, y)$ between the nodes' identifier and the destination key. Lookups are either application-triggered or part of the maintenance protocol, which is only needed if not enough application-triggered lookups have to be performed.

All nodes in a Kademlia network store their peers represented as *NodeHandles* into the leaves of a binary tree, the so called k-buckets. A NodeHandle comprises of a node's IP address, the transport protocol port and its 160-bit identifier (the *nodeId*). The k-buckets are organized in a similar way as Pastry's routing table [6] (with $\mathrm{BpD} = 1$ and k instead of one redundant node at each position): For each l, where l is the length of the common nodeId prefix, there is one k-bucket of size k, which is filled with nodes sharing this nodeId prefix e.g. a node with nodeId 0011 holds buckets for nodeIds starting with 1, 01, 000 and so on (s. Fig. 3).

Kademlia uses RPCs for lookups, data handling and overlay maintenance. For the KBR service, which is discussed here, only the FIND_NODE RPCs are important: In order to find the nodes that are currently responsible for a specific key y, first the initiator node browses its own k-buckets for

closer nodes to y to initialize the result vector $\overline{V_y}$ with these nodes. Then, depending on the degree of parallelism p, it sends FIND_NODE RPCs to the first p nodes of $\overline{V_y}$. If a node receives a FIND_NODE RPC, it searches through its bucket that belongs to the longest common prefix with the destination key. From all the nodes found in that bucket, those A_i with the smallest XOR distance ($d_{\mathrm{XOR}}(A_i, y) = \min$) to the destination key y are chosen and returned to the initiator where the nodes are put into $\overline{V_y}$. Then again, the initiator sends RPCs to the first p nodes in $\overline{V_y}$, until no new nodes are returned that fit into $\overline{V_y}$. Then the initiator sends RPCs to all nodes in $\overline{V_y}$ that have not been asked so far until all nodes in $\overline{V_y}$ have been asked and finally the lookup terminates. This means, although the responsible node for y is found, the lookup does not instantly terminate, which leads to an immense amount of network traffic due to the additional RPCs that have to be sent.

In this paper, this behavior is called *exhaustive iterative lookup*. All nodes that are returned to the initiator during the lookup are put into the initiator's k-buckets, if there is space left in the appropriate bucket. Otherwise, the least recently used (LRU) node of this bucket is probed and, in case of failure, replaced.

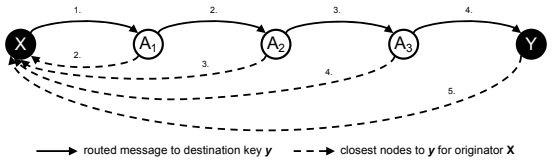
IV. Topology Adaptation

To achieve low latencies in KBR protocols like Kademlia, the overlay nodes must be aware of the underlying network topology. Without any knowledge, single overlay hops have the average latency of average end-to-end transmissions in the underlaying network. Castro et al. [7] distinguish three kinds of topology adaptation mechanisms for structured P2P overlays:

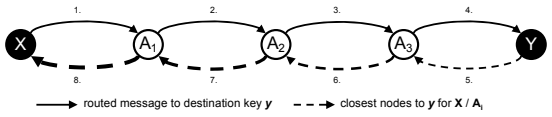
1) Proximity Routing: PR considers proximity during routing procedures and is a trade-off between progress in ID-space and routing messages through nearby nodes. For each routing hop, several candidates $(A_1, ..., A_n)$ must be available, where the one is chosen that is close in the physical network as well as close to the destination key in ID-space using an alternative routing metric d_{pr}. For a decision, the latency of all potential next hops must be determined in advance. As PR only takes effect on local routing decisions, it is only applicable to recursive routing procedures.

2) Proximity Neighbor Selection: PNS considers proximity while constructing the routing tables. If a node A_i is found that fits in the routing table of X, an existing entry A_j is displaced if A_i is closer to X than A_j. If there is space for k entries for this position in the routing tables, the closest k nodes are kept. When routing a message, only progress in ID-space is considered. The latency of all peer candidates must be determined before they can be put into the routing tables. PNS is applicable to recursive as well as iterative routing, but more effectively in recursive mode.

3) Topology-based NodeId Assignment: TbNA modifies the overlay nodes' identifiers (or parts of them) to map the overlay's ID-space directly onto the underlying network topology. This way, progress in ID-space causes routing towards the destination node. This kind of topology adaptation usually leads to a non-uniform nodeId distribution, thus it is not further discussed in this paper.



(a) *Direct Mode*



(b) *Source-routing Mode*

Fig. 4. Signaling modes for R/Kademlia

V. R/Kademlia

R/Kademlia is designed to meet the following demands:

- Low key-based routing latencies
- Avoidance of connection problems in NAT/PAT scenarios
- Effective deployment of PNS and PR
- Meeting new peers by application-triggered lookups
- Simple recursive routing and maintenance
- Robustness and resilience against node failures

Basically, R/Kademlia behaves similar to the original Kademlia: It maintains k-buckets as routing table, peers are met during lookups and not by using periodic tasks, and the XOR metric d_{XOR} is used to determine distances between keys and nodes. The main difference of R/Kademlia is the routing and lookup procedure, both done in a recursive manner. In this context, a *recursive lookup* is similar to a recursive routing procedure, with the difference that no message is delivered but the node Y responsible for the lookup key y and (optionally) other close nodes to y return a list of close nodes to y to the lookup initiator X.

A. Recursive routing in Kademlia

Recursive routing in Kademlia is done in the same greedy way (s. Sect. II) as e.g. in Pastry. The problem here is that in Kademlia the originator of a message needs to get information about nodes that are directly or indirectly involved in the lookup procedure to maintain and fill up its k-buckets. While in iterative mode a lookup initiator meets new peers when receiving FIND_NODE responses, this is not possible in plain recursive mode, as contacted nodes do not respond to the lookup initiator. The solution for this problem presented here is to send additional messages from all nodes on the routing path back to the initiator. Here, two alternative signaling modes for this task are presented (s. Fig. 4):

1) Direct Mode: In this mode, all nodes on the routing path forward the message to that node from the local k-buckets that is closest to the message's destination key. In addition they send n nodes from their k-buckets that have the closest nodeIds to the destination key y according to d_{XOR}, directly back to the message's originator X. For this, an appropriate message type is used. This way, the originator gets the same amount of information about potential peers he would get in iterative mode. A drawback here might be that the originator and the nodes on the routing path are not peers, which could

lead to connection problems in NAT/PAT scenarios. Fig. 4(a) illustrates the *Direct Mode*.

2) Source-routing Mode: In this mode, the destination node Y of a key-based routed message sends back the n closest nodes from his k-buckets to the last hop on the routing path. This node acts the same way: It merges the closest nodes to the destination key y it holds in its k-buckets into the message and forwards it along the source route. This is done until all nodes on the routing path have merged their closest nodes into the message and it reaches the originator. This way, the originator and additionally all nodes on the routing path get all the information about other nodes the originator would get in iterative mode. This mode has the advantage that all nodes only communicate with peers i.e. known nodes. Additionally, nodes on the routing path meet more nodes than in *Direct Mode*, but this comes with a higher bandwidth consumption. In high churn scenarios it is possible that nodes on the return path fail and so the originator does not receive the message. Fig. 4(b) illustrates the *Source-routing Mode*.

If not enough application-triggered lookups or routing procedures have to be performed, recursive lookups on random keys fitting in the k-buckets that have not been used for an interval of t_b are performed as maintenance tasks to keep the buckets up-to-date. To deal with failed nodes on the routing path, hop-by-hop acknowledgments are used. If an acknowledgement is missing, the node is removed from the k-buckets and the message is forwarded to the secondary closest node according to d_{XOR}. The time-out interval t_o needed to detect a node failure is calculated in a TCP-like manner, using the mean and variance of the measured round-trip-times:

$$t_o = \overline{\mathrm{RTT}_X} + 4 \times \sigma_{\mathrm{RTT}_X}$$

If *active probing* is enabled, all unknown nodes that are met during a lookup or routing procedure are probed if no information about their proximity and availability is given. Not until then are these nodes put into the k-buckets. *Active probing* can also be enabled in iterative mode.

R/Kademlia can be used in a mixed mode, parallel to the original protocol. Though, the signaling modes introduced in this paper only work if all nodes on the routing path are capable of using R/Kademlia.

B. PR in R/Kademlia

For the application of Proximity Routing in R/Kademlia, we propose a different routing metric $d_{\mathrm{route}} = d_{\mathrm{KadPR}}$ instead of $d_{\mathrm{route}} = d_{\mathrm{XOR}}$ for the decision at each node on the routing path to which peer a message should be forwarded:

$$d_{\mathrm{KadPR}}(X,Y) = d_{\mathrm{prox}}(X,Y) + d_{\mathrm{prefix}}(X,Y)$$

$$d_{\mathrm{prefix}}(X,Y) = \begin{cases} 0 & , X_i = Y_i \, \forall \, 0 \leq i < m \\ m-n & , \exists \, n : X_i = Y_i, X_{n+1} \neq Y_{n+1} \\ & \forall \, 0 \leq i \leq n < m \end{cases}$$

with $X = X_1X_2...X_m$, $Y = Y_1Y_2...Y_m$, , and $d_{\mathrm{prefix}}(X,Y) \in [0;m] \subset \mathbb{N}$ (here with $m = 160$). $d_{\mathrm{prox}}(X,Y) \in [0;1)$, where a small value indicates a low latency. The use of d_{KadPR} results in forwarding the message to the node that has the longest common prefix with the destination key and is physically close ($d_{\mathrm{prox}}(X,Y)$) to the current node on the routing path. As peers in k-buckets are organized with respect to the common prefix with the local node, this exchange of the metric does not increase the number of routing hops: If A_{i+1} and A_{i+2} are peers (from the same k-bucket) of A_i with $d_{\mathrm{prefix}}(A_{i+1},y) = d_{\mathrm{prefix}}(A_{i+2},y)$, y would be in the same k-bucket of A_{i+1} and A_{i+2}, respectively.

To determine d_{prox}, all peers have to probed in advance, otherwise d_{prox} is set to 0.99. Alternatively, network coordinate systems can be utilized for estimating the latencies. New met nodes only have to be contacted, if they fit into a local k-bucket, thus no probing is needed if the appropriate k-bucket is full.

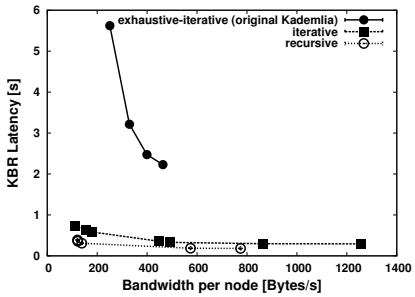
C. PNS in R/Kademlia

For Proximity Neighbor Selection in R/Kademlia, the LRU strategy used for maintaining the k-buckets is replaced by the policy of filling up all buckets with the physically closest nodes that are met. For this, all potential peers must be probed to detect their proximity. This applies to all unknown nodes that are met during application-triggered routing or lookup procedures. If an overlay node is found that is physically closer than any node in the bucket, the farthest node from the bucket owner is replaced by the new one. Since new peers also have to be checked for availability before they are put into the k-buckets, the usage of network coordinate systems is here inappropriate, as these systems cannot be used for estimating availability.

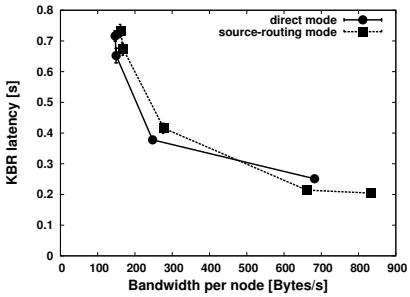
VI. Evaluation

For the evaluation of R/Kademlia the overlay framework OverSim [8] was deployed to simulate overlay networks with 5,000 nodes. For this, OverSim's implementation of Kademlia was extended to support recursive routing, PR, and PNS. For the simulation of churn, nodes are assigned a Weibull distributed session time with shape $k = 0.5$ and a mean lifetime of $10,000$ seconds. This resembles the churn behavior in the KAD file sharing network as it has been observed by [4] and [5]. Alternatively, the mean lifetime is varied between 1,000 and 30,000 seconds for the simulation of high and low churn scenarios. On each node a test application periodically sends a 100 bytes probe RPC messages to random nodeIds of currently alive nodes with a normal distributed interval of 60 seconds. Here, latency means the interval between sending the RPC message and receiving the response, which is sent directly to the originator. Failed routing attempts are counted as successful with a latency of 10 s. For the underlaying network model, we chose OverSim's *Simple Underlay*, with network latencies calculated from network coordinates, which are based on Skitter Internet measurements [9]. Each protocol and parameter combination is simulated 20 times for 1,800 s after building up the network and a transition phase of 1,800 s.

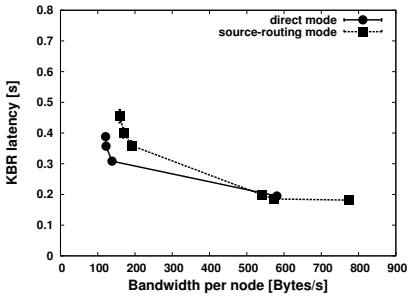
The simulation results are prepared using the Performance vs. Cost framework (PVC) [10] to identify the best parameter combinations regarding routing latencies and bandwidth consumption. The 95% confidence intervals are always calculated, but they are often too small to be visible in the plots. In addition to the original Kademlia protocol and R/Kademlia a third mode—the "simple" iterative mode—is evaluated, where lookups terminate immediately if a contacted node notifies the initiator that it is responsible for the destination key. Both iterative modes are evaluated with 1, 3, and 5 parallel RPCs for



(a) All parameter combinations of Kademlia and R/Kademlia



(b) Signaling modes (lifetime mean = 1,000 s)



(c) Signaling modes (lifetime mean = 10,000 s)

Fig. 5. Performance vs. Cost (PVC) / convex hulls

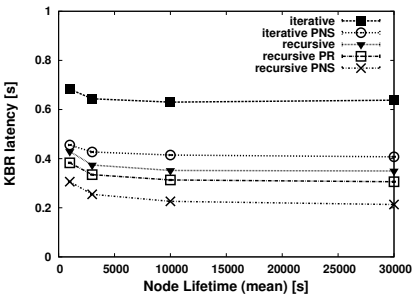


Fig. 6. KBR latency (RPCs)

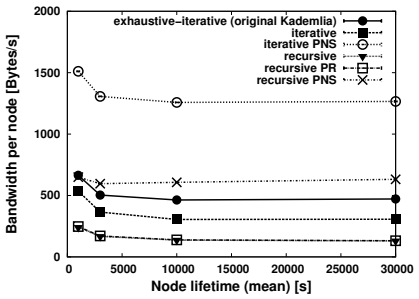


Fig. 7. Bandwidth consumption per node

lookups. PNS is applied to R/Kademlia and the simple iterative mode, PR only to R/Kademlia. The bucket size k is always set to 8, t_b to 1,000 s. All routing modes where evaluated with *active probing* enabled and disabled.

The first results shown in Fig. 5 are plotted according to the PVC rules. In Fig. 5(a) the convex hulls of the results of the original Kademlia protocol, the simple iterative mode, and R/Kademlia in a churn scenario with a nodes' mean lifetime of 10,000 s using all possible parameter combinations are shown. R/Kademlia shows the best trade-off between routing latencies and bandwidth consumption while the original Kademlia protocol cannot compete at all due to the high costs of its exhaustive lookup procedure. The simple iterative mode shows average results. The two different signaling modes are also compared using the PVC framework in simulated networks with different lifetime means of overlay nodes. The corresponding convex hulls are shown in Fig. 5(b) and Fig. 5(c): They illustrate that for R/Kademlia the *Direct Mode* is superior to the *Source-routing Mode* in moderate (s Fig. 5(c)) and low churn scenarios (with a lifetime mean $\geq$ 10,000 s), as it comes with lower traffic per node. In networks with high churn rates (s. Fig. 5(b)), the combination of PNS and the *Source-routing Mode* achieves the best results regarding routing latencies. The reason for this effect is that more nodes are met and so more nodes have to be probed, which leads to k-buckets filled up with physically close nodes that have been recently checked for availability. For all results presented in the following, the *Direct mode* has been used.

Fig. 6 shows the RPC latencies of routed RPC messages in scenarios with different churn rates. Here, *active probing* is activated for all routing modes. It is observable that R/Kademlia without topology adaption is faster ($\approx$ 350 ms) than the simple iterative variant ($\approx$ 630 ms), here using 5 parallel RPCs. The original protocol—not plotted here—achieves latencies between 2-5 s depending on the churn rate. The effect of PR is small in comparison to the results of PNS, but as described below it comes with almost no additional traffic. Activated PNS leads to an immense decrease of routing latencies in all churn scenarios—in iterative mode ($\approx$ 415 ms) but especially with R/Kademlia ($\approx$ 225 ms). A combination of PNS and PR does not lead to a further speed-up, hence it is not plotted here. R/Kademlia with PR activated and the simple iterative variant using PNS achieve nearly equal routing latencies, but with a significant difference in the amount of network traffic.

Fig. 7 shows the measured bandwidth consumptions: In all different churn scenarios R/Kademlia without topology adaptation and R/Kademlia with PR activated come with a significant lower bandwidth consumption ($\approx$ 140 Bytes/s) than both simple iterative routing modes ($\approx$ 300 and 1250 Bytes/s). The amount of network traffic needed by PNS is about 4 times as much as in R/Kademlia as well as in the simple iterative mode. It is obvious that less bandwidth is needed when PR is activated instead of PNS, since if a node's bucket is full newly met nodes do not have to be probed when using PR. In contrast, all newly met nodes have to be probed to decide whether a node is put into a bucket or not when using PNS.

Fig. 8 and 9 show the results with *active probing* disabled, which leads to a bigger effective stabilization interval t_s (s. Sect. II), hence recursive routing here achieves better results only in low and moderate churn scenarios. Iterative routing combined with PNS is faster in high churn scenarios, but again, this comes with a very high bandwidth consumption.

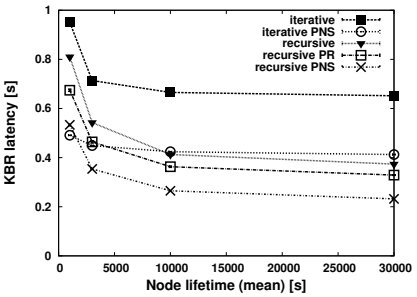


Fig. 8. KBR latency (RPCs) without *active probing*

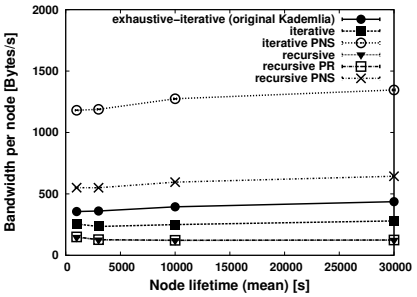


Fig. 9. Bandwidth consumption per node without *active probing*

VII. Related Work

Li et al. [11] present a recursive KBR protocol, where overlay nodes also meet peers during lookup procedures. It differs from the here proposed R/Kademlia approach in the organization of routing tables as R/Kademlia derives from the well-established bucket-based Kademlia protocol. Bamboo [12] is a KBR protocol designed for high routing performance in dynamic networks. It also uses recursive routing, but new peers are only met by periodic tasks. Instead of the XOR metric two different metrics are applied at the same time. Limited redundancy in the routing tables is supported, thus only one node per routing table entry is effected by PNS. Kaune et al. [13] utilize topology adaptation methods like PNS for the original iterative Kademlia protocol, which leads to a noticeable decrease of routing latencies. However, due to the fact that PR and PNS optimize local routing tables or local routing decisions, respectively, PR and PNS are more effective in recursive routing mode as shown in the evaluation section.

VIII. Conclusion & Future Work

In this paper we presented R/Kademlia, a recursive variant of the popular Kademlia protocol. R/Kademlia achieves very low latencies—especially when using topology adaptation—while keeping the overlay traffic small. The here proposed Proximity Routing mechanism for R/Kademlia achieves notable results, nearly without additional network traffic. The original Kademlia protocol cannot compete at all regarding routing latencies and bandwidth consumption due to its exhaustive iterative lookup procedure. R/Kademlia, as a fast and economic KBR protocol, can be the basis of future P2P applications in the Internet with high demands on routing latencies. The deployment of R/Kademlia in established Internet applications that utilize the original Kademlia protocol is reasonable, as the users of these applications would benefit from R/Kademlia's higher routing performance and its NAT/PAT compatibility. The source code of R/Kademlia's implementation is included in the OverSim framework and is available for download at *http://www.oversim.org/*.

In the future, we plan to evaluate both proposed signaling modes of R/Kademlia in comparison to the original protocol in NAT/PAT scenarios to prove R/Kademlia's advantages in these networks. Another task for the future is a simulative comparison with other KBR protocols, especially with Bamboo. Here, first results show a better latency/bandwidth trade-off for R/Kademlia in low churn scenarios. We also intend to apply other topology adaptation mechanisms [14][15] to R/Kademlia to achieve a further decrease of KBR routing latencies.

References

[1] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, vol. 2735/2003, Berkeley, CA, USA, Feb. 20–21, 2003, pp. 33–44.

[2] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems: First International Workshop (IPTPS 2002). Revised Papers*, vol. 2429/2002, Cambridge, MA, USA, Mar. 7–8, 2002, pp. 53–65.

[3] D. Wu, Y. Tian, and K.-W. Ng, "Analytical Study on Improving DHT Lookup Performance under Churn," in *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, Cambridge, UK, Sep. 6–8, 2006, pp. 249–258.

[4] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Rio de Janeiro, Brazil, Oct. 25–27, 2006, pp. 189–202.

[5] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1371–1384, Oct. 2009.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware 2001 : Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, vol. 2218/2001, Heidelberg, Germany, Nov. 12–16, 2001, pp. 329–350.

[7] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in distributed hash tables," in *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, Bertinoro, Italy, Jun. 3–7, 2002, pp. 52–55.

[8] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, Anchorage, AK, USA, May 6–12, 2007, pp. 79–84.

[9] B. Huffak, D. Plummer, Daniel, D. Moore, and K. Claffy, "Topology Discovery by Active Probing," in *SAINT-W '02: Proceedings of the 2002 Symposium on Applications and the Internet (SAINT) Workshops*, Nara, Japan, Jan./Feb. 28–1, 2002, pp. 90–96.

[10] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, Miami, FL, USA, Mar. 13–17, 2005, pp. 225–236.

[11] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, Massachusetts, May 2–4, 2005.

[12] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, Boston, MA, USA, Jun./Jul. 27–2, 2004, pp. 127–140.

[13] S. Kaune, T. Lauinger, A. Kovacevic, and K. Pussep, "Embracing the Peer Next Door: Proximity in Kademlia," in *Proceedings of the Eighth International Conference on Peer-to-Peer Computing (P2P'08)*, Aachen, Germany, Sep. 8–11, 2008, pp. 343–350.

[14] F. Hartmann and B. Heep, "Coordinate-based Routing: Refining NodeIds in Structured Peer-to-Peer Systems," in *Proceedings of the International Conference on Ultra Modern Telecommunications & Workshops (ICUMT'09)*, St. Petersburg, Russia, Oct. 12–14, 2009.

[15] B. Heep, "dCBR: A Global View on Network Coordinates for More Efficient Peer-to-Peer Systems," in *Proceedings of the Second International Conference on Ubiquitous and Future Networks (ICUFN 2010)*, Jeju Island, South Korea, Jun. 16–18, 2010, pp. 372–377.