SOLD: Self-Organizing Lookups in DHTs for better Performance over Unstable P2P Overlay Links

Andreas Berl, Ivan Dedinski, Amine Houyou and Hermann de Meer
Faculty for Mathematics and Informatics
Chair for Computer Networks and Computer Communication
University of Passau, Germany
{berl, dedinski, houyou, demeer}@fmi.uni-passau.de

Abstract

The quality of service (QoS) of a distributed hash table (DHT) lookup is gaining importance with the growing number of services adopting the P2P paradigm. Examples of applications that could largely benefit from an improved timeliness and reliability of message exchange in DHTs are Domain Name System (DNS), or even newer types of distributed location-based services in a mobile environment. The bursty effects of Internet traffic on latency, congestion, and loss can change the short term state of the overlay links in the DHT. The quick changes to overlay link/node states cannot be taken into account while structuring long term P2P routes. This paper proposes self-organizing mechanisms to improve the QoS for DHT lookups, without changing the structure of the DHT network. Different kinds of lookup replication techniques are implemented on top of the DHT to restrict the influence of the heterogeneous capabilities of the overlay routes while offering self-adaptive and robust high performance lookups.

1 Introduction

Distributed hash tables (DHTs) such as Chord [1], Pastry [2] or CAN [3] are theoretical solutions to build large, very scalable, and efficient P2P networks. However, they still face major challenges to be applied to the large Internet and for more industrial applications. Although of being able to provide a "short" overlay route to any peer within a limited number of overlay hops, they hardly cater for the heterogeneity and unstability of Internet end-to-end paths. DHTs structure large networks based on a dry abstract view of the Internet. They rely on overlay routing decisions based on long term abstract identification of logical neighbors. The logical overlay route provided by the DHT is mapped onto a physical route in the underlying physical network, where one overlay link might span over several physical hops. DHTs construct a homogeneous structured overlay network, basing on heterogenous nodes and links. When faced with bursty and highly correlated behaviors like simultaneous requests to one node, or routing requests along the same direction, both overlay links and nodes might impose different delays on these requests, varying according to their capabilities. This means that even with a limited number of overlay hops traversed by DHT lookups, the end-to-end delay might vary strongly, resulting in highly variable quality of the overlay routes.

In this paper the QoS of DHT lookups is improved by adding a new self-organizing lookup mechanism on top of the DHTs without changing its topology. These special DHT lookups are called ***Self Organizing Lookups in DHTs (SOLD)***.

The main purpose of SOLD is to increase the interaction performance between peers in an existing DHT network, where stability of routes and nodes cannot be guaranteed, but where timely messages are essential. Self-organization is applied at the lookup level, to deal with the unstability of the overall structure of the DHT, with the aim of reaching a given peer as fast as possible. This should enable fast transmissions, especially for applications requiring quick reliable control/signalling messages or other small information, like pointers, values, or text. SOLD provides both, a solution to push information to nodes and also to pull information from nodes, with raised performance. The structure offered by DHTs has allowed a new type of middleware-like functionality of the DHT layer. However several industrial scale applications have come short to adopting the P2P paradigm, partly due to the lack of timely assurance, among other reasons.

An example application that requires time-sensitive lookups is the implementation of the Internet's Domain Name Service (DNS) using DHTs. Cox et al. [4] state that besides the advantages of this effort, like fault-tolerance or

load-balancing properties, there is the problem of high latencies in the lookups.

Further example applications are location based services (LBS), requiring spatio-temporal validity of information retrieved. In a P2P-based LBS [5] lookups rely on strict temporal restrictions (e.g. looking up the next petrol station along a motorway). Similarly, in [6] DHTs are used as a signalling mechanism to provide a middleware, capable of organizing heterogeneous wireless cells in 4G scenarios. The P2P model is used to cater for the heterogeneity of the information stored among different location servers. A multiple-radio mobile device discovers the different wireless cells surrounding it (e.g. GSM, UMTS, WLAN, WiMax, etc.) by looking them up through a DHT. The P2P paradigm is explored to bring technologies together, and to provide a common interface to all mobile users, who require looking up information managed by different operators, data-bases, and with various signalling technologies. The state of the wireless cells (blocking rate, used capacity, etc.) would rely on fast lookups. The mobile user looks-up available wireless access points along his movement path.

This paper is structured as follows. In Section 2 the complexity of providing quality of service (QoS) in DHTs is discussed. Furthermore, the self-organization aspects and design decisions behind SOLD are presented. The detailed implementation and design of SOLD are presented in Section 3. It is shown how SOLD is applied to a vanilla Chord structure to obtain a better QoS for lookups. Subsequently, the emulation of SOLD in the PlanetLab [7], using about 350 real machines, is compared with Chord in Section 4. The emulations carried out on this powerful testbed exploit the real behavior of the Internet linking those machines. In Section 5 other related work is explained and compared to SOLD. The conclusions of this paper are summarized in Section 6.

2 Using Self-Organization to Provide QoS in DHT Routing

In this section the increasingly important role of the QoS of DHT lookups for applications adopting the P2P communication model is exposed. Our design principles are based upon self-organization criteria whose advantage is keeping the complexity of supporting QoS low, while fulfilling the aim of obtaining the best out of an administration free P2P system [8].

QoS is defined as a set of service requirements that need to be met by the network while transporting a flow [9]. This could be traced through some measurable metrics such as delay, jitter, bandwidth, and probability of loss. The need to guarantee or track each of these measurable metrics depends greatly on the application needs. In the open Internet, however, finding better routes is about making an informed decision on which path to take. This usually involves tracking the most recent view of the available resources within an autonomous system, closely administered. A better route should exclude weak links, since delay is an additive metric, and bandwidth a concave one. Dissimilarly, DHTs offer a large administration-free system with its own organization and structure. Adding QoS awareness is rather complex and unscalable. Instead, DHTs create an abstract structure where overlay links exhibit heterogeneous QoS conditions. In the administration free homogeneous DHT, reaching low latency interaction levels between peers is our main goal. This paper solves mostly this issue, in addition to adding reliability to lookups. Depending on the importance of the messages exchanged between peers, some might require a higher reliability or fast delivery. In other terms, the delay sensitive lookups would expect to be routed through the fast routes, which is similar to what QoS routing does in the Internet [9]. In contrast to QoS routing, SOLD provides a mechanism to discover better routes without attempting to discover or measure the state of the overlay links. It treats the underlaying overlay structure, as a black box. However, this black box is not assumed to have a given predictable behavior. For this purpose, the probes sent-in are a heterarchy of independent processes trying to get the best out of this system [10]. This principle is based on self-organization.

Self-organization [11] is the ability for a group of given elements to create structure with a higher goal that needs to be fulfilled. We talk then of an emerged structure. When looking at a group of peers, DHTs create structure with the possibility to reach each peer from any peer while keeping minimum routing information. On top of such a structure, this paper proposes to launch diverse lookup messages that insure exploring concurrent paths along the underlying P2P structure. For each lookup, a group of messages is created to reach the goal of exhorting the system to find out the best path in that structure. The messages progress independently, in a fire-and-forget manner along the overlay hops, to reach the same end of the black box. This diversity of overlay end-to-end paths is limited by the available routing information provided by the DHT (e.g. number of fingers in a finger table). Similar to the self-organizing aspect of an ant algorithm [10], a group of cooperating ants (or messages) independently try to reach, from a given source, their destination through cooperation. The ants leave a noticeable trail along the visited nodes, which allows them to interact and cooperate, by following the strongest trail, i.e. the nodes where more ants have left a trail. However, differently to the ant algorithm, in SOLD the trail left does not strengthen the choice of a given path, but rather lead to the ants making sure to stay away from this path. The group of messages already have a given direction specified by the existing DHT structure. The cooperation between the messages allows the group to detect the best route by

staying diversely spread, but in a directed way (forming a structured flooding). This is assuming that the overlay link quality will not be remembered after the lookup is finished. The emergence of the best path changes to keep the perturbations inside the black box (DHT structure) hidden from the lookup process. Next, the details of how the lookup is generated and how it progresses are explained.

3 SOLD: Self-Organizing Lookups in DHTs

In this section the proposed algorithm "self-organizing lookups in DHTs" (SOLD) is described. SOLD is implemented on top of an existing DHT, adding new functionality without changing its topology. DHTs provide different overlay routes from one overlay node to another. SOLD exploits the diversity of routes by replicating lookups in different ways using ***horizontal replication*** and ***vertical replication***. Horizontal replication aims at limiting the effects of unavailable links or links with low QoS capabilities, and excluding routes spanning over failing or temporarily overloaded overlay nodes. Vertical replication limits the delay of a lookup by replacing lookup datagrams lost between two overlay hops within a single RTT. Compared to a retransmission, when using an ARQ mechanism, the vertical replication can provide a preemptive correction mechanism on a per hop basis. Once the target node is reached, a response is also replicated and returned on various routes to the originator to consider slow (or even not available) direct connections or asymmetric Internet links.

In this section the reader is expected to be familiar with the basic concepts of Chord [1] and its terminology because it is used as an example DHT to explain the mechanisms of SOLD.

3.1 Definitions

The node initiating a lookup is called the ***originator node*** of the lookup and the node which is responsible for the required information is called the ***target node***. Other nodes belonging to the overlay route are called ***intermediate nodes***. If an originator node queries information within the DHT the whole process is called a ***lookup process***. Every lookup process consists of several simultaneously launched ***lookup threads*** traversing different overlay routes in a cooperative manner and transporting the same request. A single message passed over an overlay link is called a ***lookup message***. The response from the target node to the originator node is called a ***response process***, which also consists of several ***response threads*** and ***response messages***. Every message has a ***message identifier*** identifying it as corresponding to a certain thread of a process. Every thread stores one message identifier at every overlay node it passes by, to enable communication and therefore cooperation in between the different threads. This message identifier is stored with a ***soft state*** of a few seconds.

3.2 Horizontal Replication - Autonomous Cooperating DHT Lookup Threads

Within an DHT different redundant routes to a target node with different performance attributes are provided. The mechanism of utilizing these different routes by performing autonomous cooperating DHT lookup threads is called horizontal replication. The threads are autonomous because they are able to succeed the lookup process individually. They are cooperating by storing and reading identifiers at the passed overlay nodes and making sure to be routed along separated overlay routes by bouncing off each other. Since the physically fastest lookup thread succeeds first, the originator node receives a response with the lowest delay of all cooperating lookup threads. A lookup process, indicated by an originator node, consists of τ autonomous cooperating, simultaneously launched lookup threads transporting the same request. The lookup threads are sent to the τ best fitting overlay successors of the originator node. In Chord this would be the best fitting fingers of the originator nodes finger table. If there are not enough unused successor nodes left, the amount of autonomous lookup threads is reduced. Every single lookup thread performs a full lookup procedure, starting at the originator node and finishing at the target node in a usual case. If more lookup threads of one lookup process are routed over the same intermediate node, they are bouncing off each other, utilizing different routes to the target node. The most promising successors of the intermediate node are chosen to forward the lookup threads. If there are not enough appropriate successors left for to find a separated route for a lookup thread, the thread terminates itself. A concrete example of this mechanism is depicted and explained later in this paper in Figure 1.

The target node usually receives more than one lookup thread belonging to the same lookup process. But only the fastest, first arriving lookup tread of this lookup process triggers the response to the originator node. All other messages are discarded.

The amount of cooperative lookup threads τ is a positive number greater or equal to 1. If τ is set to 1 no horizontal replication is performed. The maximum of τ is the amount of overlay successors of the originator node, e.g. the number of fingers in Chord. The appropriate value for τ has to be determined especially for a certain time-sensitive application and its environment as it depends on the structure of the physical network, the number of nodes in the DHT environment, the traffic constraints and the desired lookup latency. If available, τ can be adapted to the priority of the lookup. A higher value of τ increases the probability of a smaller latency and the robustness of the lookup. In Fig-

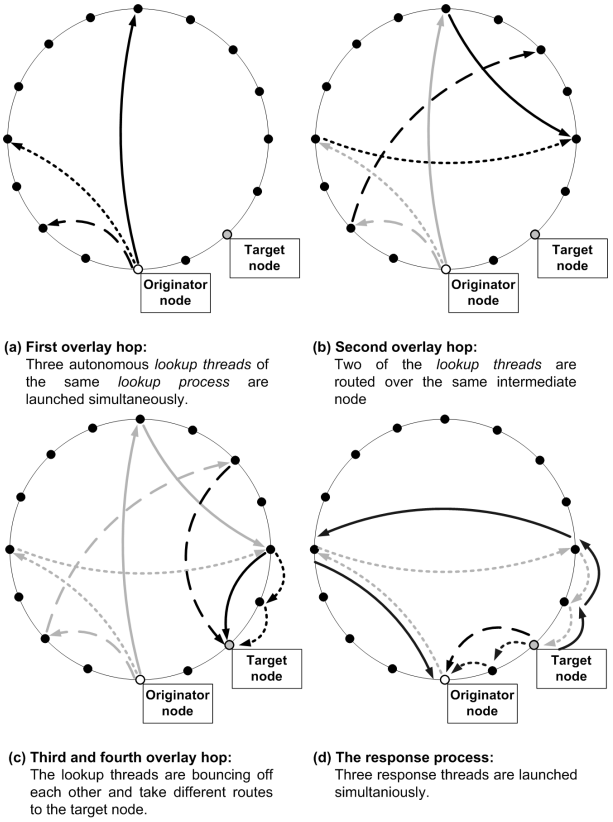

Figure 1. An example of autonomous cooperating DHT lookup and response threads in Chord.

ure 1 the cooperation of autonomous cooperating lookup threads is shown. Chord is chosen as DHT environment to illustrate the cooperation because of its popularity. As a simple example τ is set to 3. A complete lookup process with the overlay hops of all threads is shown in (a), (b) and (c). In each Figure the lookup messages of the current steps are drawn in black color and the lookup messages of previous steps are drawn in gray color. The three lookup threads can be distinguished by their different line patterns. Figure 1 (a) shows the launch of the cooperating lookup threads at the originator node. In Figure 1 (b) it can be seen that at the intermediate nodes the lookup threads take the route to the best fitting fingers. In this concrete situation two of the lookup threads are routed over the same intermediate node. The second lookup thread recognizes the situation because of the stored message identifiers at the intermediate node. In Figure 1 (c) it is shown that this second thread is routed to a different finger, keeping itself separated from the first one. All autonomously launched lookup threads finally reach the target node. It is important to see that the three lookup threads are likely to reach the target node at different times since the arrival time does not depend on the number of overlay hops. However, only the first arriving lookup thread triggers the response to the originator node while the other lookup threads terminate themselves.

3.3 Horizontal replication of the response

The transmission of the response from the target node to the originator node is also replicated in a response process to keep the latency of the entire lookup as small as possible. However, the horizontal replication mechanism is slightly modified for the response. The first of the τ horizontal replicated response threads is sent directly to the originator node, since its IP-address is known. The direct route is assumed to be the fastest in vanilla Chord, but other routes should be taken in account. The direct route can be congested, slow or even unavailable if one or both of the nodes are firewalled. The second lookup thread is sent back over the route that has been used to send the lookup. This route is expected to be fast since it was the fastest found route towards the target node. However, there can be asymmetric links involved in this route, slowing the traffic down in the response direction. The other $(\tau - 2)$ response threads are horizontally replicated like described in Subsection 3.2. In Figure 1 (d) the horizontal replication of a response is shown. The line in the grey color depicts the first succeeding lookup thread. The back colored lines depict the different response threads. They can be distinguished by their different patterns. As explained in this section we find a direct routed thread and a thread routed back on the lookup route. The third thread is routed to the originator node, avoiding the direct way, as it is already used.

3.4 Vertical Replication - Machine-Gun Messaging

With this mechanism lookup and response messages are repeated in a fast manner. In this section response messages are referred to as lookup messages for simplicity. If a lookup message fails to arrive properly at the next hop, one of its backup messages arrives after short time, within one RTT. This mechanism is called vertical replication or machine-gun messaging.

In common implementations of DHTs TCP is used to send keep alives and to transmit lookups. If a lookup message gets lost or is corrupted a retransmission of the message is necessary. In TCP the corresponding segment will be resent after a timeout, which depends on the measured RTT on the physical end-to-end link. Since one overlay hop may span over several physical hops this timeout adds additional delay to the lookups. To lessen the latency of the lookups this paper suggests the use of replicated UDP packets for DHT lookups.

Every node sends μ equal lookup messages of the same lookup thread to a successor node where μ is a number greater or equal to 1. If μ is set to 1, there is no vertical replication. Let x be the probability of one single message getting lost or corrupted. It is assumed as a simplification that these events are independent from each other. If k equal messages are sent, the probability that all messages get lost is x^k, where $1 \leq k \leq \mu$ and $0 \leq x \leq 1$. According to this exponential behavior of the probability it is important to resend the first few messages over each hop very fast to have a backup message arrived a few moments after the last sent message.

On the other hand, the receiver and the network should not be flooded with messages. When problems occur, like congested networks or overloaded nodes, a back-off mechanism is needed. To solve this problem the machine-gun messaging mechanism increases the time in between two repeated lookup messages exponentially. Message k is sent after $\gamma * (2^{k-1} - 1)$ ms, where γ ms is the initial gap in between the first and the second message with $1 \leq k \leq \mu$ and $\gamma > 0$. If e.g. $\gamma = 1$ and $\mu = 6$ the sending times would be $[0, 1, 3, 7, 15, 31]$ ms. Additionally acknowledgements are used to reduce the amount of sent machine-gun messages sent over one overlay hop. Every first arriving lookup message of a thread is acknowledged by the receiving node. If the acknowledgement reaches the sender, it stops immediately sending further machine-gun messages over this hop. If the acknowledgement does not reach the sender for some reason, the whole μ machine-gun messages are sent. These acknowledgements do not confirm the lookup itself. They are just used to reduce the number of sent machine-gun messages.

The values of γ and μ depend on the RTTs measured in the transport network of a certain application environment and the desired lookup latency for the application. γ should be a value less than the RTT, so that the first few lookup messages are sent within one RTT. μ should not be chosen too small, because it would be more likely that all of the lookup threads get lost. On the other hand, it would be not reasonable to send further machine-gun messages, if a lookup resulting from it would exceed the required latency time. Thus, μ should be chosen in a way, that $\gamma * (2^{\mu-1} - 1)$ ms, which is the time of sending the last machine-gun message, is smaller or equal to the tolerated lookup latency of the application.

It is important to see that lookup messages are transported in very small UDP packets, near the lower bound of an ethernet frame size. Replication of this packets does not demand too much additional performance from the network, compared to "usual" traffic. The overhead of the machine-gun messaging is evaluated in Section 4.

The target node usually receives more than one lookup message belonging to the same lookup process. But only the fastest, first arriving lookup message of the first lookup thread triggers the response to the originator node. All other messages are discarded.

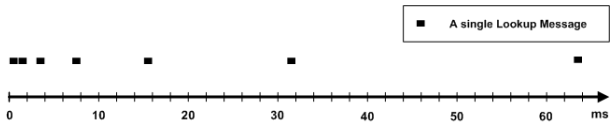

Figure 2. The machine-gun messaging mechanism with $\mu = 7$ and $\gamma = 1$.

In Figure 2 the machine-gun messaging mechanism is visualized, μ is set to 7 and γ is set to 1. The figure depicts that the first message is sent at time 0 and the last one after 63 ms.

3.5 Soft State

Every lookup thread stores a message identifier at every node it passes by. It is stored with a soft state for σ seconds. When the soft state times out, the identifier is deleted. The soft state is reset to σ every time a repeated message of a certain thread arrives at a node. The identifiers enable the cooperation of the different threads and messages. Threads communicate by examining the identifiers of a node. Different threads of the same process are bouncing off each other. Messages belonging to a thread that already visited the node terminate. σ has to be chosen to be long enough for the identifier, to be by the latest arriving lookup message of the corresponding lookup process. If σ is chosen to be too short, a late arriving lookup message is not able to identify itself as a replicated message and can cause some overhead. This message would be routed on the best fitting route to the target node and trigger a new response, which would be finally discarded by the originator node.

4 Evaluation

In this section Chord and SOLD are evaluated on PlanetLab [7]. Chord in its simple form is also used to create the structure of the P2P network on which SOLD has been tested. The goal of the experiments is to show the performance of lookups in both vanilla Chord and comparing SOLD lookup techniques used top of it.

4.1 Methodology

PlanetLab has been used as the evaluation environment, to create a large globally distributed P2P network, where the influence of real network heterogeneity is essential for evaluating the performance of SOLD. The testbed included

about 360 PlanetLab nodes, located all over the world, with different bandwidth limits and CPU power. On each physical PlanetLab node 10 virtual DHT nodes were started in order to reach a total DHT node count of about 3600 nodes. The hash identifiers of the DHT nodes were distributed randomly, not depending on their physical neighborhood, in order to have more long distance (non-local) routes.

Several virtual DHT nodes had to be shut down during the experiments. Some nodes turned out to be blocked by firewalls allowing connectivity only in one direction, or not to every IP address, thus being unable to build up their finger tables. Other nodes were running under heavy stress conditions (heavy CPU workload, high packet dropping rate), loosing all their fingers. All these nodes were discovered and shut down automatically in order to have a stable set of nodes for a clear comparison of SOLD to vanilla Chord. During the experiments the number of active nodes varied. Overall $863/739$ DHT nodes were shut down for Chord/SOLD,respectively, leading to $2740/2860$ DHT nodes that remained functional until the end of the experiments.

Two experiments were performed for comparison, one with SOLD and one with standard Chord. Each experiment lasted for about 12 hours to cover network traffic changes with their effects on SOLD and Chord. The experiments scenarios were generally the same for both runs, except for the load changes on PlanetLab. For SOLD the following values were set:

- The amount of cooperating treads τ was set to 4.
- The amount of machine-gun messages μ was set to 9.
- The initial gap between the first two machine-gun messages γ was set to 10 ms.
- The soft state σ of the placed information at every passed node was set to 10 seconds.

First, in a stabilization period of about $500s$ the DHT ring bootstrapped and each DHT node filled its fingertable. Most of the nodes that had to be shut down were discovered during this period. After the initial stabilization interval, each DHT node started sending lookups for randomly chosen hash IDs in the hash ID range from 0 to 2^{14}. Every node sent one lookup every 1000 seconds leading to about 3.5 lookups per second. This low amount of lookups was chosen to decrease the impact of the CPU workloads of the PlanetLab nodes on the experiment. On one hand the CPU workloads varied dynamically due to other simultaneous done experiments by other research groups. On the other hand the workload was additionally increased by the simulation of about 10 virtual nodes on each physical node.

During the experiments the following data was measured:

- the latency for each lookup,
- the hop count on the succeeding path,
- the average number of sent machine-gun messages before receiving an acknowledgement in SOLD, and
- the number of virtual nodes that had to be shut down.

4.2 Results

Altogether $74895/73592$ lookups were started for Chord/SOLD. When a lookup lasted longer than 60 seconds it was considered as a failed lookup since we wanted to analyze fast lookups.

SOLD turned out to be far more robust than Chord. In Chord 15576 failed lookups were encountered which is a failed lookup rate of 20.7971%. SOLD instead had only 84 failed lookups which is a failed lookup rate of $0,1141\%$.

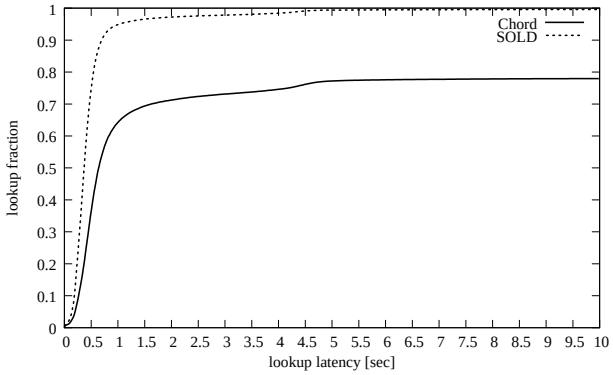

Figure 3. CDFs of Chord and SOLD.

SOLD managed much lower lookup latencies than Chord. Figure 3 shows the cumulative distribution function of the number of lookups distributed by the total time the lookup took (CDF) of Chord, compared to the CDF of SOLD. Its can be seen that the latencies of SOLD are lower in all ranges. About 75% of all lookups have a latency less than 0.5 ms in SOLD, in Chord it is only about 40% which is nearly a factor of two. Before 1 about 95% of SOLDs lookups are completed,in contrast to only 65% in Chord. The average latency of Chord is 1.2359 seconds, the average latency of SOLD is 0.5363 ms excluding the failed lookups for both protocols. This is a factor of more than two in the average latencies. If the failed lookups are considered by assuming a lookup time of 60 seconds, the situation gets much worse for Chord. Its average latency is then 13.4873 seconds, SOLDs average latency remains in the same range with 0.6392.

For the response messages, as we have expected for an environment like PlanetLab, the direct way to the origina-

tor node has been the fastest way in most cases. Approximately 2.4% of the response messages were not routed on the direct way, but took one of the provided alternative routes. 1.3% were routed back on the fastest lookup path. 1.1% were routed on completely new routes. The average lookup latency of these overlay-routed response messages was 0.6417 which is in the same range than the overall average latency. This advantage does probably not compensate the overhead produced by this response message mechanisms. However, the results may get better in other testbeds, involving models of asymmetrical connections and more unreliable access networks.

To evaluate the machine-gun mechanism in SOLD, every node measured the number of sent machine-gun messages, before receiving an acknowledgement (machine-gun messages within the RTT of an overlay link). It turned that the average number of sent machine-gun messages was 3.9955, before being acknowledged.

4.3 Overhead of SOLD

To calculate the theoretical upper bound for the number of messages per lookup sent by SOLD, that for the messages sent by Chord is evaluated first. The number of hops from the originator to the target is limited to $O(log(n))$ in Chord, where n is the number of nodes in the ring. Chord sends one message over each hop, which is then acknowledged (e.g. implicitly with TCP). Additionally, the target node sends a response to the originator, which also is acknowledged. This leads to

$$2 * O(log(n)) + 2$$

messages per lookup for Chord.

A single thread, created by SOLD, needs $O(log(n))$ hops to reach the target node, since it is routed via the same way as it would be in Chord. Two threads might in a worst case scenario (which is actually not possible) bounce off $O(log(n))$ times from each other. The (physically) fastest one of the threads at a certain node (this may change from hop to hop) takes the best overlay route, the other one bounces off. Since both of the threads transport the same lookup, and both threads finally reach the target node, the threads can be mixed up. It is of no matter for this evaluation, which one of the threads bounces off. Thus, it is assumed for this analysis that the k^{th} thread with $1 \leq k \leq \tau$ bounces off all $k-1$ previous threads. With these assumptions the first thread succeeds after $O(log(n))$ hops. The second thread bounces off $O(log(n))$ times and is free after that to route to the target node over an usual Chord route, leading to a worst case hop count of $2 * O(log(n))$. Likewise the k^{th} thread needs a maximum of $k * O(log(n))$ hops. An amount of k threads, routed to the target node and back to the originator, together need a maximum of $2 * \frac{k*(k+1)}{2} * O(log(n))$ hops. On every hop a maximum of μ machine-gun messages are sent and every first message is acknowledged. This leads to a number of

$$k * (k+1) * O(log(n)) * (\mu + 1)$$

messages for SOLD, if all acknowledgements have arrived too late. This means, that as a rough upper bound, SOLD produces at most

$$\begin{aligned} & \frac{k * (k+1) * O(log(n)) * (\mu + 1)}{2 * O(log(n)) + 2} \\ \leq & \frac{k * (k+1) * O(log(n)) * (\mu + 1)}{2 * O(log(n))} \\ = & \frac{k * (k+1) * (\mu + 1)}{2} \end{aligned}$$

times more messages than Chord.

To evaluate the traffic overhead in the PlanetLab caused by SOLD in comparison to Chord, another experiment was started. It lasted only 2 hours, because a huge amount of data had to be collected, and was done under the same conditions as the other experiment. Every message that was produced by a lookup on any hop was counted. The result was, that an average Chord lookup caused 13 messages and an average SOLD lookup caused 246 messages, which is an overhead factor of $18,92$. The theoretical upper bound for this experiment with $\tau = 4$ and $\mu = 9$ would have been a factor of

$$\frac{4 * (4+1) * (9+1)}{2} = 100.$$

5 Related Work

One of the major goals of P2P research in the area of DHTs today is performance optimization, needed by a variety of potential real life DHT applications, like location aware services for mobility, file sharing, instant messaging and many others. Optimization research includes a decrease of signalling traffic, routing delay optimization, topology optimization or improving resilience to network dynamics like failed nodes and connections, churn or congestions.

When looking at the performance optimization work done by the P2P community, three different categories of related work can be distinguished. The first category is considering more adequate metrics than just logical proximity during the construction of the overlay topology. The second category of related work aims at replicating the information itself, to bring it closer to where it is most popular. Finally the third category uses more than one route to improve the DHT routing performance, and therefore is the closest to SOLD.

Under the first category, Pastry [2] and Tapestry [12]create more neighborhood-aware P2P structures. A given node collects passively discovered close neighbors

and use them in a heuristic way as forwarding alternatives. Proximity could be expressed in physical hop count, RTT, or even geographic distance. As a further variation, Kademlia [13] improves this principle by integrating the neighborhood optimization into its finger table. In Coral [14], hierarchically ordered DHT clusters effectively reduce long distance lookups and data access using milestones. Whereas in [15] the gathering of lookup latency provides some information to find fingers with low latency for every range of the key space in a Chord ring. In a survey on combining parts of further restructuring techniques, in [16], a combination different techniques shows some promising results. This category can be seen as complementary to SOLD. Unlike SOLD, these solutions modify the overlay structure constructed by the DHT. SOLD, in contrast, applies an algorithm on top of an existing DHT, which can both apply to either simple DHT protocols (such as vanilla Chord) or to more complex distance and delay-aware DHT topologies. The difference lies in the fact that SOLD is robust to any kind of high frequency perturbations relating to delay changes or to the stability of a given DHT route. Even if given an ideal DHT route, SOLD tries to explore other routes, which might in that moment experience less traffic demand for instance. This assumption can be said about links at the access network, where high frequency unpredictable changes can occur due to restricted network capacities.

The second category of work brings the information closer to the node starting the lookup [17]. It mostly uses cached replicates of DHT key-data pairs along the paths, where the most lookups originate from. In other words, the most popular keys are exactly replicated there, where they are most needed. Similarly in [18], a more static replication scheme hashes the same key by using multiple hash functions. The lookup addresses simultaneously all hash identifiers of a given key. Caching techniques are also explored for a DHT-based Domain Name System in [19]. SOLD in its current state does not deal with replicated content and so does not need to cope with problems like replication inconsistency. However, as SOLD is a lookup strategy it can easily be enhanced to take advantage of the availability of replicated keys in the DHTs but also be used for fast interactions to increase the consistency of the replicas.

Under the third category, Epichord [20] and Kademlia [13] use lookup replication closest to the scheme used in SOLD, while adding to that some caching strategies. Unlike the adaptive recursive approach of SOLD, where a lookup progresses as a self-organizing structured flood, Epichord requires the originator node to centrally manage the progress of the flooded lookup by obtaining routing information back, at each hop, in an iterative way. In Epichord, after a horizontal replication each of the chosen better hops must contact the originator node. This is repeated per each hop, while in SOLD this is done at the start only. Furthermore, and in comparison with SOLD the number of overlay hops in each lookup increases through this lengthy conversational feedback to the originator. Additionally the whole lookup process depends heavily on the the originators node network connection and routing capability.

6 Conclusion

In this paper SOLD proposed a replication mechanism aiming at providing QoS for DHT lookups, while using the principle of self-organization to keep the complexity low. SOLD adds new functions to an existing DHT without changing its available functionality or topology.

The evaluation work carried on the PlanetLab shows that the overlay routes can not be considered to provide a constant level of QoS. The experienced quality of the overlay routes varied in a bursty manner. SOLD deals with this situation by utilizing different overlay routes in the DHT in a cooperative manner. The diversity of routes is exploited to fire a message from one peer to another, rather than gathering information to optimize routes.

The results of the experiments on the PlanetLab revealed that SOLD is far more robust than Chord, with about $99,9\%$ of succeeding lookups for the former compared to $79,2\%$ with Chord. SOLD achieved an average latency more than twice as good as vanilla Chord. About 95% of SOLD's lookups had a latency of less than 1 second and 75% less than 0.5 seconds.

There are some issues with SOLD that have to be addressed in future work. SOLD has to be evaluated under different conditions than the PlanetLab testbed, used for this paper. The effects of the "last mile" of the Internet's topology (i.e. the access networks) have to be evaluated, with its firewalls, air interfaces and asymmetric links. Such environments introduce problems to which SOLD should offer promising solutions.

Acknowledgement: This project was partly funded by the German Research Foundation (Deutsche Forschungsgemeinschaft - DFG), contract number ME 1703/4-1 and by EPSRC, contract number GR/S69009/01. It was also supported by the Euro-NGI - Network of Excellence, European Commission grant IST-507613.

References

[1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001. `http://www.acm.org/sigcomm/sigcomm2001/p12-stoica.pdf`.

[2] A. I. T. 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 Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[4] R. Cox, A. Muthitacharoen, and R. Morris. Serving dns using a peer-to-peer lookup service. In *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 2002.

[5] M. Scholl, M. Thilliez, and A. Voisard. Location-based mobile querying in peer-to-peer networks. In *OTM Workshops*, pages 166–175. Springer, 2005.

[6] A. M. Houyou, H. de Meer, and M. Esterhazy. P2p-based mobility management for heterogeneous wireless networks and mesh networks. In *Proceedings of EuroNGI Joint Workshops IA 8.3 and IA 8.2 LNCS 3883*. Springer Verlag, 2005.

[7] Planetlab an open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org`.

[8] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2003*, volume 2, pages 1499–1509, San Francisco, California, United States, March-April 2003.

[9] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for qos routing in the internet, August 1998.

[10] H. de Meer and C. Koppen. Characterization of self-organization. In Ralf Steinmetz and Klaus Wehrle, editors, *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 227–246. Springer, 2005.

[11] H. de Meer and C. Koppen. Self-organization in peer-to-peer systems. In Ralf Steinmetz and Klaus Wehrle, editors, *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 247–266. Springer, 2005.

[12] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, EECS Department, University of California, Berkeley, Berkeley, CA, USA, 2001.

[13] P. Maymounkov and D. Mazi. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[14] M. Freedman and D. Peres. Sloppy hashing and self-organizing clusters. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[15] H. Zhang, A. Goel, and R. Govindan. Incrementally improving lookup latency in distributed hash table systems. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 114–125, New York, NY, USA, 2003. ACM Press.

[16] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, California, March 2004.

[17] R. Cox, A. Muthitacharoen, and R. Morris. Serving dns using chord. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.

[18] J. W. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In *IPTPS*, pages 80–87. Springer, 2003.

[19] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the web from dns. In *1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, March 2004.

[20] B. Leong, B. Liskov, and E. D. Demaine. Epichord: Parallelizing the chord lookup algorithm with reactive routing state management. Technical Report MIT-LCS-TR-963, MIT, Cambridge, MA, August 2004.