EXPLOITING THE OVERHEAD IN A DHT TO IMPROVE LOOKUP LATENCY

Gerald Kunzmann, Rüdiger Schollmeier
Institute of Communication Networks, University of Technology, 80333 Munich, Germany

Abstract: Third generation Peer-to-Peer (P2P) networks are characterized by the fact, that they are based on Distributed Hash Tables (DHT). Thus, it is intended to reduce the high signaling overhead observed in unstructured P2P networks like in Gnutella. However especially if the churn rate in the considered approach is high, i.e. nodes leaving and joining the network frequently (small session duration) the signaling traffic needed to maintain the DHT structure increases considerably. In the Chord protocol, which is investigated in this work, so called fingers are used to establish shortcuts though the DHT structure to achieve a scalable routing performance. To keep these fingers up to date in regular Chord, a significant amount of signaling traffic is necessary, especially in high churn scenarios. Therefore we propose a new scheme which provides additional finger update methods without causing additional traffic.

Keywords: chord, finger update, P2P, proactive routing

1.INTRODUCTION

In today's networks the majority of traffic is caused by P2P applications like Gnutella, BitTorrent and Kazaa [Gnutella, 2002][BitTorrent, 2004][Kazaa, 2003]. However, a large amount of this P2P traffic is not caused by the transmission of user data, but by signaling traffic. The reason is that in unstructured P2P networks a large amount of the messages is flooded through the overlay network. Additionally, upon each request a route to the requested object has to be determined in unstructured P2P networks. Therefore, such an approach is classified as a reactive routing protocol. It is especially advantageous in frequently changing environments

as in Mobile ad hoc scenarios and even in file sharing applications where the session duration is comparably small [Kazaa, 2003].

However, in more stable scenarios, e.g. in a Voice over P2P applications like Skype [Skype, 2004], such a high signaling overhead, like the one in unstructured P2P networks, seems to be unnecessary. Further on, in voice applications it is necessary to receive an answer from the network whether a certain object is available or not. Therefore proactive routing protocols like Chord or CAN present a promising approach to provide dedicated answers with an additionally comparably low signaling overhead [Stoica et al., 2001][Ratnasamy et al., 2001]. These networks are also referred to as structured P2P networks, because the topology of the overlay network is determined by the protocol.

In this work we concentrate on the routing performance and possibilities to reduce the signaling traffic of the Chord protocol. In a Chord network a ring structure is established upon which nodes and content are arranged in the same identifier space. Thus nodes as well as content become routable. To provide a scalable routing approach, concerning the number of participants, additional routing shortcuts through the ring structure are established by any node in the network. These shortcuts are called *fingers*. We propose a new scheme which provides Chord with additional fingers at no additional cost. We therefore call these fingers *freebie fingers*. Using these freebie fingers together with a symmetrical routing algorithm can reduce the average path length by 25%. Further related work can be found in S-Chord where a symmetrical routing algorithm is presented [Mesaros et al., 2003]. F-Chord proposes a routing scheme based on Fibonacci numbers [Cordasco et al., 2004]. Both papers try to improve the maximum and average number of hops for lookups.

The remainder of this work is organized as follows. In chapter 2 we begin with a short introduction to the basic Chord finger and routing algorithms. We continue by presenting our *freebie finger* concept followed by an analysis of its applicability and advantages. Chapter 5 concludes this paper with some discussion and further research interest.

2. CHORD FINGERS

To provide scalable routing within the Chord ring each Chord node maintains connections to m nodes, called *fingers*, whereas $0...2^m$ is the identifier space for the unique IDs of the nodes. Each finger entry is updated periodically by the *fix_finger* procedure to avoid storing links to nodes that are no longer part of the Chord network. Content items stored in the Chord network are assigned and transferred to the nodes that succeed the IDs of the

content items. Upon receiving a query for a certain id each node first checks whether it hosts the content or not. In the first case it answers the query directly. Otherwise, it forwards the query to the neighbor, with the largest ID in its *finger table* that does not exceed (using modulo function) the ID of the queried content.

In basic Chord a node's k^{th} finger points to the first node succeeding the node id by at least 2^{k-1} (mod 2^m). Fingers are initialized by running a *find_successor* query for the theoretical finger position $(n.id + 2^{k-1}) \bmod 2^m$, resulting in the first node succeeding it. Due to this routing and key assignment algorithm any content available in the network is found in at most *m* hops, with high probability.

Improved finger algorithms do not stick to such a completely determined finger selection but offer the possibility to chose fingers based on different criteria. In our point of view, the most promising idea is to select fingers on proximity criteria (*i.e.*, latencies) from a set of nodes next to the theoretical finger positions.

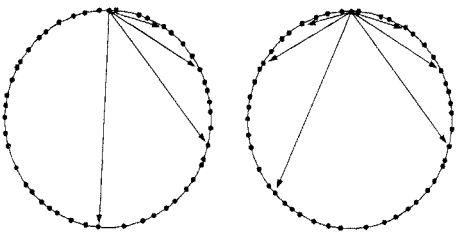

Figure 1. Sample finger distribution in a Chord ring (left image) and in S-Chord (right image) for the node on top of the ring.

The Chord finger geometry provides the nodes with a lot of routing information in close neighborhood and with little information about farther nodes (see Figure 1 left). As can be seen in the figure, the right part of the Chord ring (from the node in top's point of view) is well covered with fingers, whereas the left part is blank. Therefore, the idea of extending the *finger table* to the complete ring comes to mind. S-CHORD [Mesaros, 2003] for example proposes to keep the same number of fingers to be comparable in performance with Chord, but organizes the *finger table* in two approximately symmetric sides (Figure 1 right).

3. FREEBIE FINGERS

We propose extending the routing information each Chord node stores by a list of all nodes that have a finger table entry pointing to the node. This

freebie_finger_list can be built-up by caching the originators of all *fix_finger* requests a node receives. The nodes in the resulting list are distributed counterclockwise in approximately the same way as the nodes in the finger table are distributed clockwise, i.e. more nodes in close neighborhood and only a few farther away. Figure 2 shows the finger distribution for a node in a sample Chord network (solid lines) and the finger table entries from other nodes that point to the node.

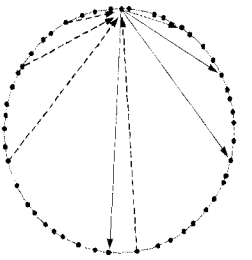

Figure 2. Sample finger distribution in a Chord ring (solid lines) and additional routing information from nodes that store a finger table entry pointing to the node (dashed lines).

Unlike S-Chord we do not have to reduce the number of fingers in the first half of the ring to be comparable to Chord in means of overhead traffic needed for updating the finger information, but can stick to m fingers and up to approximately m freebie fingers. Considering the fact that in a Chord network with N nodes each node knows only about $log_2(N)$ different fingers (Figure 3) [Binzenhöfer et al., 2004], a list with freebie fingers also consists of only about $log_2(N)$ entries (Figure 4). To avoid storing nodes in the *freebie_finger_list* that are no longer pointing to the node or even worse are no longer participating in the network, each entry is marked with a timestamp and is removed from the list if the entry has not been updated within a certain time interval, e.g. 1,5 * *fix_finger* period.

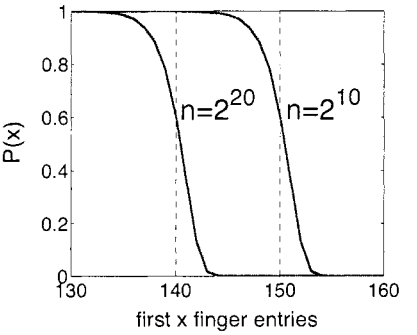

Figure 3. The distribution shows the probability P(x) of the first x finger entries being identical in a Chord network with 210 and 220 participating nodes.

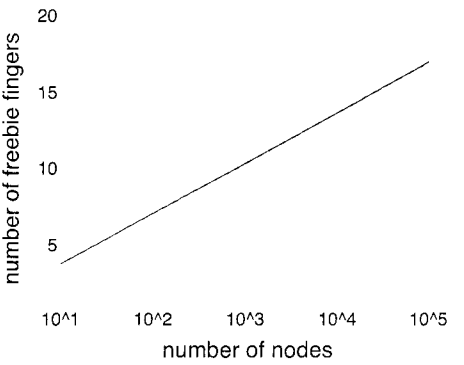

Figure 4. The chart shows the average number of freebie fingers for different network sizes as well as the 90% confidence interval.

Storing this additional routing information does not require any additional network traffic and hardly any processing power and storage. Additionally, using the freebie fingers is useful for routing queries through the network especially if symmetrical routing as proposed in S-Chord is applied. In the following chapter we describe how to use the freebie fingers to provide faster and more efficient routing through the network.

4. USING FREEBIE FINGERS

Since the basic Chord *find_successor* algorithm already searches through the finger table for the closest finger preceding the queried id, beginning with the most distant finger, extending the algorithm for using the freebie fingers is very simple. Both, expanding the existing finger table with the freebie fingers or modifying the algorithm so that it searches through both lists, is possible. In the first case you have to pay attention that *fix_fingers* still only updates the first m fingers.

Even faster routing can be achieved by using symmetrical routing that is naturally predestined for a symmetrical finger concept. As a symmetrical routing algorithm is described in detail in [Mesaros, 2003], we try to focus on analyzing the benefits such an algorithm provides.

Doubling the number of fingers (by using the freebie fingers) as well as using symmetric routing algorithms reduces the average number of hops needed for querying ids. Figure 5 shows the correlation between the number of nodes participating in a Chord network (N) and the resulting average path length (number of hops) for the three different mentioned routing algorithms basic Chord, S-Chord and Chord using Freebie Fingers. To simplify matters

we assume a fully populated network, i.e. the size of the ID space equals the network's size. For the basic Chord algorithm the maximum number of hops is limited to $log_2(N)$ in an error-free environment and the average path length is $\frac{1}{2}\ log_2(N)$ hops. Using S-Chord already reduces both maximum and average path length by about 25% and 10% respectively. An additional gain of about 20% with respect to average path lengths can be achieved by additionally using our *freebie finger* concept. That is an improvement of 25% compared to basic Chord fingers. The maximum path length using freebie fingers is $\frac{1}{4}\ log_2(N)$.

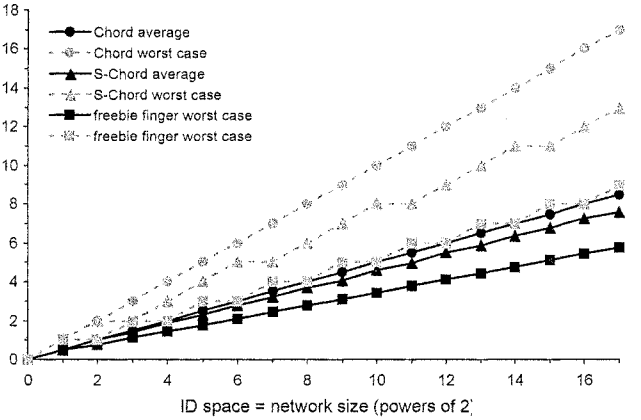

Figure 5. Correlation between the network size (fully populated network) and the resulting average path length (number of hops).

The authors of SChord claim that "the denser the network, the longer the lookup path length", and therefore, they compare only fully populated networks. We argue that this is not true and that the lookup path is longer in poorly populated networks than in fully populated networks with the same number of participating nodes. The maximum number of required hops may be more than Chord's $log(N)$ hops in networks with an ID space greater than the network size. Anyway, most lookups could be resolved within less than $log(N)$ hops.

Figure 6 shows the results from our simulations for Chord, S-Chord and our freebie finger approach. We ran all simulations in Matlab, assuming a 128bit ID space and up to 65.536 nodes participating in the network. For each network size and each Chord variant we ran 100.000 simulations. We plotted the average lookup path length and the 99%-quantile for each variant. Compared to a fully populated network (Figure 5), we come to the following conclusions: Both average and maximum path lengths in basic Chord increase, because the IDs are not evenly distributed in the ID space. S-Chord and our freebie fingers improve their average path length compared

to a fully populated network, as they can route in both directions, but the maximum path lengths increase, too. Still, more than 99% of all lookups can be answered in less than *log(N)* hops. Regarding average values, S-Chord achieves 20% shorter lookup paths compared to Chord, and our freebie finger approach can reduce lookup times by about 40%.

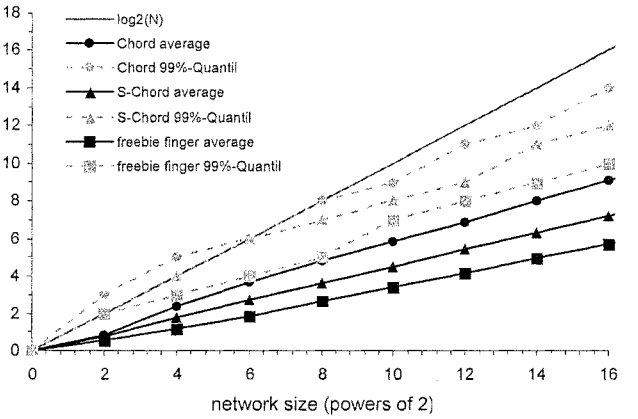

Figure 6. Correlation between the network size (poorly populated network, 128 bit ID space) and the resulting average path length (number of hops).

Reducing the average hop count similarly reduces the required traffic for lookups. It also results in shorter average query times for two reasons. First, in the ideal case where all finger entries are up-to-date and no absent nodes disturb the lookup path, the lookup time is only reduced by the transmission time the additional hops would have required. Second and more importantly, the probability of running into timeouts due to wrong finger entries that point to absent nodes is reduced with every avoided hop. As timer values are set to clearly higher values than the average transmission time, the average lookup time can be reduced clearly if fewer timeouts occur.

Another advantage of holding more fingers is a higher flexibility in choosing the next hop [Gummadi et al., 2003]. Especially under high churn rates, *i.e.* nodes joining and leaving the network frequently and therefore fingers pointing to absent nodes with high probability having more alternatives may be very valuable.

5. CONCLUSION

Unstructured P2P networks try to resolve their lookups by flooding the network, but it is not guaranteed that the queried content can be found. Due to TTL counters in the query messages queries can only be resolved in close

neighborhood. That is why content has to be replicated evenly over the network to achieve a high query hit rate. The bigger the network, the more often content has to be replicated. The main advantage is that unstructured P2P networks require no higher signaling traffic in scenarios with high churn rates. Structured P2P networks on the other hand can always tell whether a certain object is available or not, but a lot of signaling traffic is required to keep the network structure stable. This is why high churn rates lead to an increase of overhead traffic in structured P2P networks.

Using our freebie finger concept is one possible step towards reducing signaling traffic and lookup latencies. In our simulation and testbed we additionally update our finger tables by exploiting existing traffic in the network. Each time a packet from a node in the finger list is received we know that this node is still participating in the network and we can skip this finger entry in the next *fix_finger()* invocation. The more traffic per node the more often a packet from one of our fingers is received. Concerning this matter we are going to present some analysis in a future paper.

In our point of view, future structured P2P networks will be applicable in scenarios with high churn rates, e.g. ad hoc and file-sharing applications.

6. REFERENCES

Binzenhöfer, A., Staehle, D., and Henjes, R. (2004)."On the Stability of Chord-based P2P Systems," University of Würzburg 347, Nov 2004.

BitTorrent (2004). Incentives to Build Robustness in BitTorrent, bitconjurer.org/BitTorrent/bittorrentecon.pdf, 2004

Cordasco, G., Gargano, L., Hammar, M., Negro, A., and Scarano, V. (2004). "F-Chord: Improved Uniform Routing on Chord," presented at Structural Information and Communication Complexity: 11th International Colloquium, SIROCCO 2004.

Gnutella (2002) Gnutella 0.6 RFC, http://rfc-gnutella.sourceforge.net/draft.txt, 2002

Gummadi, K. et al. (2003). "The Impact of DHT Routing Geometry on Resilience and Proximity," presented at ACM SIGCOMM 2003.

Kazaa (2003) Kazaa homepage, http://www.kazaa.com/us/index.htm

Mesaros, V.A., Carton, B., and Roy P.V. (2003). "S-Chord: Using Symmetry to Improve Lookup Efficiency in Chord," presented at 2003 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03).

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., (2001). "A Scalable Content-Addressable Network," presented at ACM SIGCOMM Conference.

Schollmeier, R. and Dumanois, A. (2003)."Peer-to-Peer Traffic Characteristics" UNICE2003

Skype (2004) homepage, http://www.skype.com/. 2004

Stoica, I., Morris, R. , Karger, D., Kaashoek, M. and Balakrishnan, H. (2001). "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," presented at ACM SIGCOMM Conference.