On the Fly Estimation of the Peer Population in a Chord-based P2P System

Andreas Binzenhöfer, Dirk Staehle, and Robert Henjes

Dept. of Distributed Systems, Inst. of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
{binzenhoefer,staehle,henjes}@informatik.uni-wuerzburg.de

Abstract. The Chord system is a decentralized peer-to-peer mechanism designed to store and search key/value pairs. The peers in a Chord overlay network are represented on a circle, whereas each peer has to maintain $\log_2(n)$ neighbors to guarantee a stable overlay structure in the presence of high churn rates. A single peer, however, does not know the current size n of the Chord ring. Choosing a constant value for the number of neighbors does not scale to large networks and involves unnecessary overhead in small networks. In this paper we therefore introduce an estimator for the current size of the Chord ring based on local information like a peer's neighbor- and fingerlists. We also show how to calculate the corresponding confidence intervals to minimize the probability to over- or underestimate the current size of the overlay.
Keywords: P2P, Chord, DHT, Overlay Size, Estimation

1 Introduction

In the last couple of years, peer-to-peer (P2P) networks have become widely popular and now build the basis for a wide range of telecommunication applications. P2P overlay networks can, e.g., be used for signaling purposes, distributed document storage, content distribution and distributed indices, like a VoIP phone book. In comparison to the traditional client-server architecture P2P overlay networks do not have a single point of failure and are resistant against distributed denial of service attacks. The current generation of P2P algorithms, is based on structured overlay networks that enable fast and reliable searches. Systems like Chord [1], CAN, and Kademlia provide scalable and robust overlay networks using Distributed Hash Tables (DHT). The advantages that come along with a DHT, however, are bought by the overhead needed to maintain the structure of the overlay network. In particular the ring structure used by Chord has to be maintained even under high churn rates [2]. That is, when a great number of peers joins or leaves the network within a short period of time. To cope with these situations each peer maintains pointers to r other peers in the overlay network. According to [1] the stability of a Chord ring can be obtained with high probability as long as $r = \Omega(\log_2(n))$, where n is the current peer population of the Chord ring. In practice a peer either has to choose the

parameter r large enough to be able to handle the maximum possible ring size or has to adapt r on the fly. The first approach, however, does not scale to large networks, since the maximum possible ring size is not likely to be known a priori. Moreover, choosing the parameter r too large in small networks results in unnecessary overhead, since the participating peers generate more maintenance traffic than actually needed [3]. All in all choosing a large constant value for r results in high maintenance cost in the majority of cases, or insufficient stability in larger than expected overlay networks. A peer, however, is not able to adapt the size of its neighborlist dynamically to $r = \Omega(\log_2(n))$ since a single peer does not know the current size of the overlay network it is participating in.

In this paper we therefore introduce an estimator for the current size n of the Chord ring based on local information like the peer's current neighborlist. In contrast to other approaches [4, 5], our estimator is well adapted to the properties of the Chord algorithm and does not need additional overhead. A participating peer can use the estimator to adjust the size of its neighborlist to the current size of the overlay network. This way, the peer uses the optimal amount of maintenance overhead to guarantee a stable overlay given the current size of the network. However, since there is a certain probability to over- or underestimate the actual size of the overlay network, we also show how to calculate the corresponding confidence intervals for our estimator. The upper and the lower limits of these confidence intervals can also be used as estimates themselves. In a running Chord implementation the estimated value of n can then be used to build a peer's neighborlist. The remainder of this paper is structured as follows. In Section 2 we summarize the most important aspects of the Chord algorithm to provide the basis to understand our estimator. Section 3 presents the main idea of our estimator. In Section 4 the mathematical framework and all necessary definitions are introduced. The numerical results are then shown in Section 5. Section 6 finally summarizes and concludes the paper.

2 Chord Basics

This section gives a brief overview of Chord with a focus on aspects relevant to this paper. A more detailed description can be found in [1].

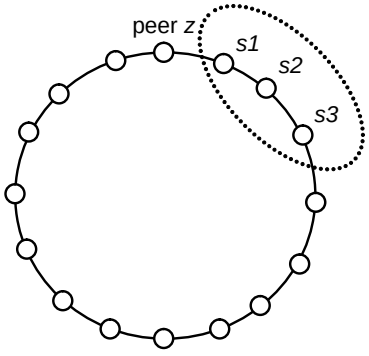

Fig. 1. The successorlist for peer z.

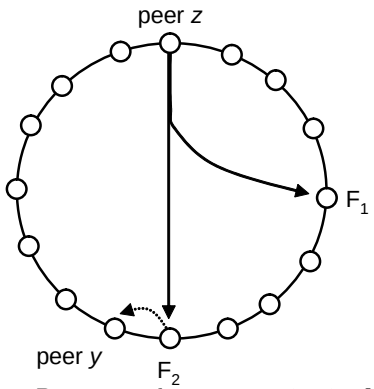

Fig. 2. Peer z searches peer y using its fingers.

In general a DHT assigns each peer in the overlay an m-bit identifier using a hash function such as SHA-1. Chord builds a ring topology (clockwise marked with numbers from 0 to 2^m), where the position of a peer on this ring is determined by a peers m-bit identifier. If the ring structure is lost, the functionality of the Chord algorithm can no

longer be guaranteed. Therefore a peer stores information about its r immediate successors on the ring. Figure 1 shows the successorlist for a peer z and $r = 3$ successors. It consists of $s1$, $s2$, and $s3$, the three immediate successors of peer z. If the immediate successor $s1$ of peer z goes offline, peer z can still contact the next closest peer $s2$ of its successorlist. As stated in [1] $r = \Omega(\log_2(n))$ peers are sufficient to ensure that each peer knows the id of its closest living successor.

However, if a peer would only maintain pointers to the peers in its successorlist as mentioned above, searches for resources stored in the P2P network would take very long. A peer looking up another peer or a resource would have to pass the query around the circle using its successor pointers. To speed up searches a peer z in a Chord ring also maintains pointers to other peers, which are used as shortcuts through the ring. Those pointers are called fingers, whereby the *i-th* finger in a peers finger table contains the identity of the first peer that succeeds z's own id by at least 2^{i-1} on the Chord ring. That is, peer z with hash value id_z has its fingers pointing to the first peers that succeed $id_z + 2^{i-1}$ for $i = 1$ to $\log_2(m)$, where 2^m is the size of the identifier space.

Figure 2 shows two examplary fingers F_1 and F_2 for peer z. Using this finger pointers, a search for peer y does only take two overlay hops. A detailed mathematical analysis of the search delay in Chord rings can be found in [6]. In Section 3 we show how to use the successorlist and the fingerlist of a peer to estimate the current size of the Chord ring.

3 Analytical Model

In this section we present the analytical framework of our model based on a peer's successor- and fingerlists. As stated above a total of n peers share the identifier space of length $N = 2^m$. We furthermore assume that, by the hash function, the position $S(z)$ of every peer z is distributed uniformly in the identifier space. Accordingly, every identifier is occupied by a peer with probability $p = n/N$. Let $I(z) = S(z+1) - S(z)$ be the random variable describing the length of the interval between peer z and peer $z+1$, i.e. the distance between two peers as illustrated in Figure 3. We assume a collision-free hash

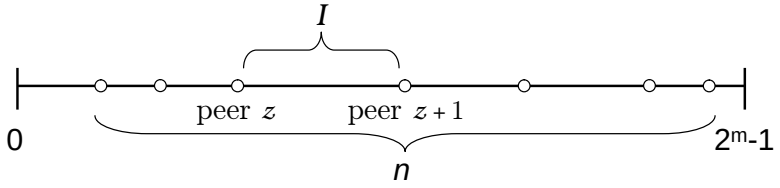

Fig. 3. The random variable I describes the length of the interval between two peers.

function, i.e. each peer has a unique identifier. Further, let us assume that without loss of generality peer z has identifier 0, i.e. $S(z) = 0$. Then, the probability that another peer sits on position 1 is $(n-1)/(N-1)$ as there remain $n-1$ peers for $N-1$ free identifiers. The probability $P(z+1, i)$ that $S(z+1) = i$ is

$$P(z+1,i) = \left(1-\frac{n-1}{N-1}\right)\cdot\left(1-\frac{n-1}{N-2}\right)\cdots\left(1-\frac{n-1}{N-i+1}\right)\cdot\left(\frac{n-1}{N-i}\right) \quad (1)$$

$$\approx \left(1-\frac{n}{2^m}\right)^{i-1}\cdot\frac{n}{2^m} \approx \left(1-\frac{n}{2^m}\right)^{i}\cdot\frac{n}{2^m}. \quad (2)$$

The approximation is justified as $n >> 1$ and $N >> i$. Thus, we can conclude that the Interval $I(z)$ between a peer and its direct neighbor is approximately geometric with parameter p: $I(z) \sim geom(p)$ where $p = \frac{n}{2^m}$. We validate this approximation by generating 10000 snapshots of chord rings with 1000, 10000, and 100000 peers in an identifier space of size 2^{160}. Peer z has identifier 0. We evaluate the distance to peer $z+1$ and refer to this distance as Interval 1, which is equal to $S(z+1) - S(z)$. Figure 4 compares the simulated distribution with the theoretical geometric distribution. Since the curves match exactly when plotted on a linear scale we use a log-log scale. Considering the magnitude of the interval sizes and probabilities, the geometric distribution and the simulated distribution are almost identical. The dithering in the simulated curve comes from the limited amount of values that we gain from the simulations.

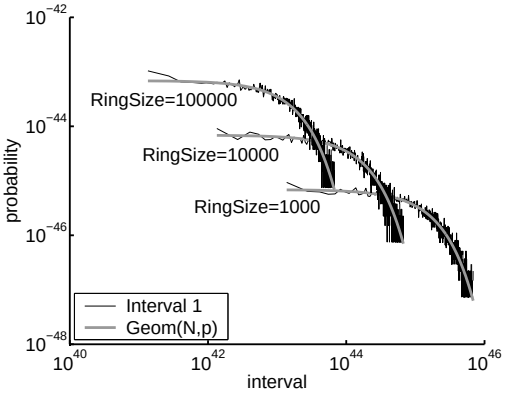

Fig. 4. Interval 1 is well-approximated by the geometric distribution.

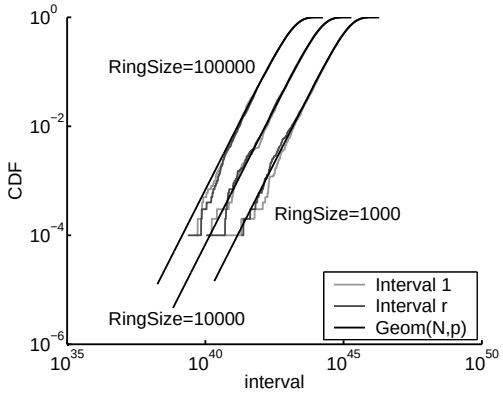

Fig. 5. Interval 1 and Interval r follow a geometric distribution.

Ideally, peer z does not only know its direct neighbor but the next $r = \lceil \log_2(n) \rceil$ neighbors and can calculate the distances between them. From peer z's point of view Interval $I(z+1)$ depends on Interval $I(z)$. However, we can argue again that due to the enormous size of the identifier space the intervals between all r neighbors of Peer z are iid and we introduce the random variable I for an arbitrary interval between two neighbored peers.

In Figure 5 we validate this approximation by means of the cumulative distribution function (CDF) of Interval 1 and Interval r, i.e. the interval between the last two successors. We can see that the curves for both intervals match very well with the geometric distribution independent of the ring size. The simulated curves start with a probability of $1e-4$ as we generated 10000 snapshots. Note that the distribution of 99% of the intervals (CDF$\geq 1e-2$) coincides with the geometric distribution.

The main idea of our algorithm is to estimate the parameter p of the geometric distribution of I. We denote the estimated value of p as $\widehat{p}$. From this we can then conclude that

$\widehat{n} = \widehat{p} \cdot 2^m$. To be able to estimate p we need to obtain realizations of I, which can be gathered by looking at our neighborlist. As shown in Figure 6 the intervals between a peer's

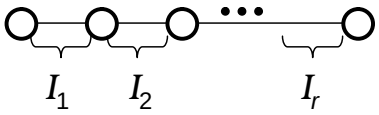

Fig. 6. Realizations of the random variable I.

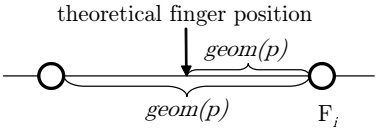

Fig. 7. Memoryless property of I.

r immediate successors can be regarded as r different realizations of the random variable I. More realizations of I can be found if we have a closer look at a peer's fingerlist as presented in Section 2. As has been shown in [1] only $O(\log_2(n))$ of those $\log_2(m)$ fingers are actually different, i.e. are actually pointing to different peers. The explanation lies within the fact, that especially the first fingers tend to coincide with a peer's successorlist. The interesting fact concerning our estimator, however, is that the actual position of the *i-th* finger on the ring is different from its theoretical position $id_z + 2^{i-1}$. Figure 8 illustrates

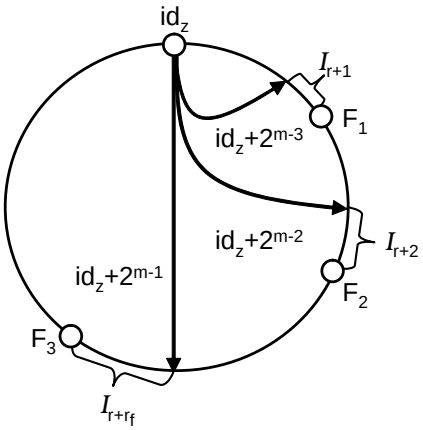

Fig. 8. The distance between the theoretical and actual position of a peer's *i-th* finger.

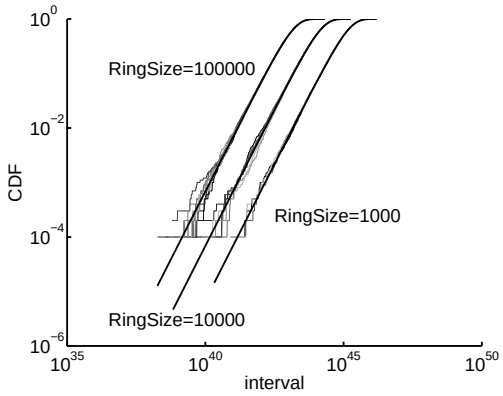

Fig. 9. The interval between actual and theoretical finger position is geometric.

this issue in detail. The Figure shows three exemplary fingers for a peer z pointing to $id_z + 2^{m-3}, id_z + 2^{m-2}$ and $id_z + 2^{m-1}$ respectively. As we can see the actual positions of the finger peers F_1, F_2 and F_3 are different to the fingers theoretical positions. This distance, however, can be interpreted as another realization of the geometrically distributed random variable I.

As stated above we already know that the length of the interval between a finger F_i and the previous peer on the ring is geometrically distributed. If we now chose a random point in this interval, due to the memoryless property of the geometric distribution, the interval between the theoretical position of the finger and the actual finger is as well geometrically distributed with the same parameter p as illustrated in Figure 7. Again, we validate this assumption by means of the snapshots we used above. Figure 9 compares the distances of the theoretical and actual finger positions with the geometric distribution. Again the curves match very well. By means of the geometric distribution of the finger intervals, we obtain another $r_f \approx \log_2(n)$ realizations of I from a peer's fingertable, leaving us with a total of $r + r_f$ different realizations of the random variable I.

4 Estimating the size of the Chord ring

The main goal of this section is to introduce an estimator $\widehat{n}$ for the size of the current Chord ring. This estimator can then be used to dynamically adjust the estimated necessary size $\widehat{r} = \log_2(\widehat{n})$ of a peer's successorlist. We estimate the parameter p of the geometric distribution of I using the maximum-likelihood estimator (MLE) [7]

$$\widehat{p} = \frac{1}{\overline{I}(r + r_f) + 1}$$

where $\overline{I}(r+r_f)$ is the sample mean. With $\widehat{p}$ we calculate the estimated size $\widehat{n} = \widehat{p} \cdot 2^m$ of the current Chord ring. Finally $\widehat{n}$ will be used to determine the number of successors the peer is going to maintain. The size of the successorlist will be set to $\widehat{r} = \lceil log_2(\widehat{n}) \rceil$. An obvious advantage of this approach is that the size of the successorlist is not as sensitive to errors as the estimated size of the Chord ring itself. That is due to the fact that the size of the successorlist is logarithmically dependent on the size of the Chord ring. The disadvantage is that so far we cannot make any statement of how good the MLE $\widehat{p}$ estimates the actual size of the ring. Therefore we build confidence intervals for $\widehat{p}$. According to [7] the $100(1-\alpha)$ confidence interval for p is given by

$$\widehat{p} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{\widehat{p}^2(1-\widehat{p})}{r + r_f}}$$

where $z_{1-\frac{\alpha}{2}}$ (for $0 < \alpha < 1$) is the upper $1-\frac{\alpha}{2}$ critical point for a standard normal random variable.

However, the consequences of underestimating the real value of p are by far more severe than the consequences of overestimating the real value of p. That is due to the fact that a successorlist, that is too small has a negative effect on the stability of the Chord ring. A successorlist that is too large on the other hand only results in some additional overhead. To minimize the danger of underestimating n we use the upper limit of the confidence interval to estimate n:

$$\widehat{n}_+ = \left(\widehat{p} + z_{1-\frac{\alpha}{2}} \sqrt{\frac{\widehat{p}^2(1-\widehat{p})}{r + r_f}} \right) 2^m$$

This $\widehat{n}_+$ is then used to calculate the size $\widehat{r}_+$ of the successorlist as $\widehat{r}_+ = \lceil log_2(\widehat{n_+}) \rceil$. Again, we round to the next biggest integer to minimize the probability of underestimating the real value of r. The next section summarizes how the estimator performs in an actual Chord implementation.

5 Numerical Results

In this section we show the results obtained by our simulations. If not stated otherwise, each snapshot of our simulations is done by uniformly placing n peers into the identifier space of size 2^m. Then, the distances between the first r consecutive peers are calculated and given as input to our estimator. We study different ring sizes to investigate how the estimator scales to larger networks.

To see how accurate our estimator $\widehat{n}$ estimates the current ring size we generated 10000 snapshots of a specific ring size n. We then set the number of successors to the ideal value $r = \lceil \log_2(n) \rceil$ and compared the estimated ring sizes to the actual ring size. Figure 10 shows the results of our simulations for a given ring size of 10000 and a successorlist of size 14.

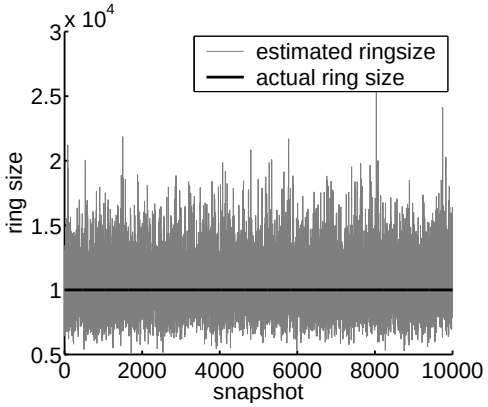

Fig. 10. 10000 estimates of the ring size as compared to the actual ringsize.

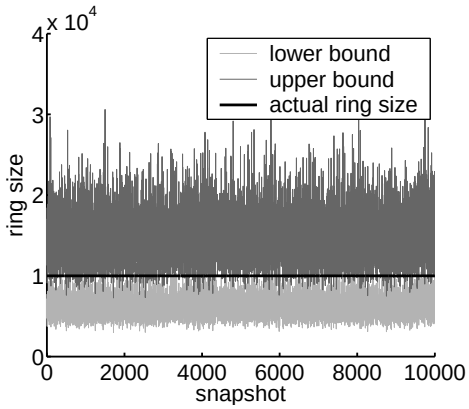

Fig. 11. The lower and upper bound of the estimator with a confidence level of 95%.

As can be seen in Figure 10, our estimator $\widehat{n}$ is well in the right order of magnitude and roughly oscillates between $0.5n$ and $2n$. Depending on the range of application, however, under- or overestimating might be crucial to the performance of the application on top of the estimator.

In Figure 11, we therefore compare the lower bound $\widehat{n}_-$ and the upper bound $\widehat{n}_+$ of our estimator to the actual ring size, again using 10000 snapshots of a ring of size 10000. The confidence level in this example is set to 95%. The lower bound $\widehat{n}_-$ of the estimator stays beneath the actual size of the ring with high probability. Whereas the upper bound stays above the actual size of the ring with high probability.

To analyze the probability that the lower bound overestimates and the upper bound underestimates the actual ring size we plotted the sorted snapshots in Figure 12. The Figure shows the results obtained for the estimator and its lower and upper bounds for three different ring sizes. Again a confidence level of 95% is used. The part of the upper bound beneath the black line represents the number of snapshots where the upper bound underestimates the actual ring size, the part of the lower bound above the black line the number of snapshots where the lower bound overestimates the actual ring size respectively. Note that the median of the estimator itself approximately intersects with the actual ring size as indicated by the two straight black lines. This justifies our assumption that the random variable I is approximately geometric since the median of an estimator based on exactly geometric intervals would exactly intersect with the actual ring size.

Another important fact that can be derived from the figure is that we over- and underestimate the actual ring size less significantly in larger networks. This is of course due to the fact that we use more neighbors in larger networks. The primary reason, however, lies in the fact, that a peer also has more distinct fingers and thus more uncorrelated realizations of I in larger networks. Note that the tiny spikes in the graphs of the lower and upper bound arise since we only sorted the estimator itself and plotted the corresponding upper and lower bounds.

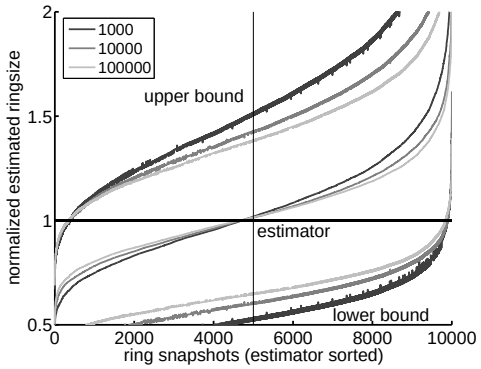

Fig. 12. Sorted estimates gained by the estimator and its lower and upper bounds.

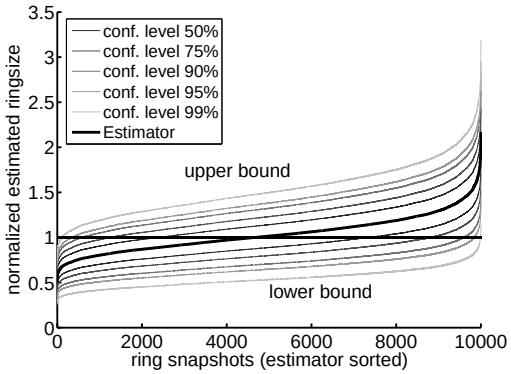

Fig. 13. The upper and lower bound of the estimator in a network of size 10^5.

As can be seen in Figure 13 the lower and upper bound of the estimator can be fine tuned by adjusting the confidence level. The confidence level in this example was varied between 50% and 99%. The higher we set the confidence level, the more the curves of the upper and lower bound drift away from the estimator. This means that the higher we choose the confidence level, the less frequently we will under- and overestimate the actual ring size. However, the drawback of a high confidence level is that the estimates of the upper and lower bound get less precise. The trade-off between overlay stability and maintenance overhead can thus be fine tuned by the confidence level.

The most obvious application of the estimator is the dynamic adaptation of a peers successorlist. Since a peer ideally maintains a list of at least $r = \lceil \log_2(n) \rceil$ neighbors the estimate in this case does only depend logarithmically on the estimate of n. As it is more critical to underestimate than to overestimate the required number of successors, we will only compare the estimator and its upper bound in the following. Since we additionally round the estimate for the upper bound $\widehat{r}_+ = \lceil \log_2(\widehat{n_+}) \rceil$ we set the confidence level to moderate 95% in the remainder of this section. Figures 14 and 15 show the estimated number of required neighbors in a network of size 10^4 and 10^5. In Figure 14 the actually required number of neighbors is $14 = \lceil \log_2(10^4) \rceil$. The regular estimator provides the

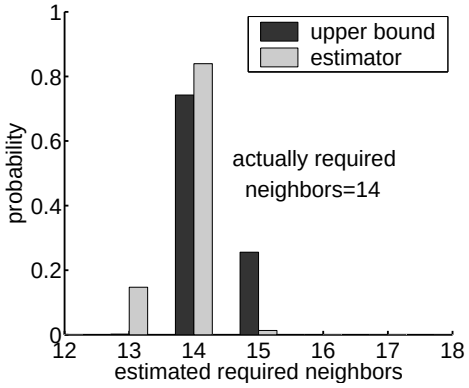

Fig. 14. The upper bound does never underestimate the actually required number of neighbors.

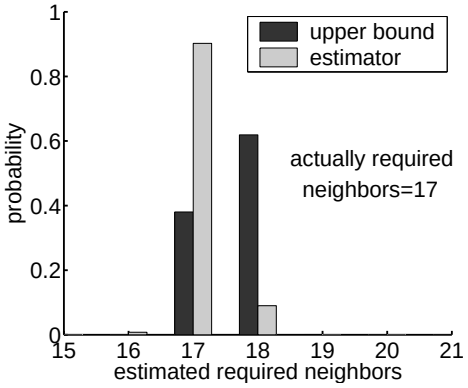

Fig. 15. The regular estimator reaches the actually required number of neighbors in about 90% of all cases.

correct number of neighbors in over 80% of all cases. However, in almost 20% of the snapshots the estimator would set the size of the successorlist to 13, one peer less than needed. In order to minimize the danger of underestimating the required number of successors, one should therefore use the number of neighbors estimated by the upper bound. While the upper bound does almost never underestimate in the current example, it tends to overestimate more frequently than the regular estimator.

In a ring of size 10^5 (see Figure 15) the upper bound overestimates the required number of neighbors by 1 in over 60% of all cases. In return it never understimates the actually required number of successors. The regular estimator on the other hand again underestimates the actual value, even though only in very few cases. Note that in about 90% of all cases the regular estimator meets the actually required number of neighbors. Given the fact that the upper bound only slightly overestimates the desired number of neighbors, we suggest to prefer the upper bound to the regular estimator in critical applications.

So far the results presented in this section were based on the ideal number of neighbors in the given networks. To see how the estimator performs when relying on an unideal number of neighbors, we again simulate 10000 snapshots for a ring of size 10^4 and evaluate the estimator and its upper bound based on successorlists of different size. Thereby the number of successors used as input to the estimator ranges between 1 and 20 successors. The actually required number of neighbors in this example is again 14. Figure 16 shows the results corresponding to the regular estimator. The bars represent the results obtained by using 1 to 20 neighbors. The brighter the color, the more neighbors have been used as input to the estimator. Obviously, the more neighbors the estimator can rely on, the

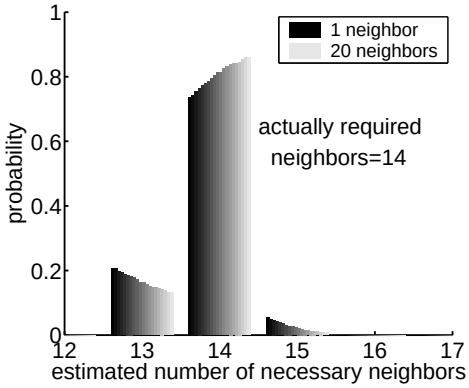

Fig. 16. The bars represent the results obtained by using 1 to 20 neighbors in a ring of size 10^4.

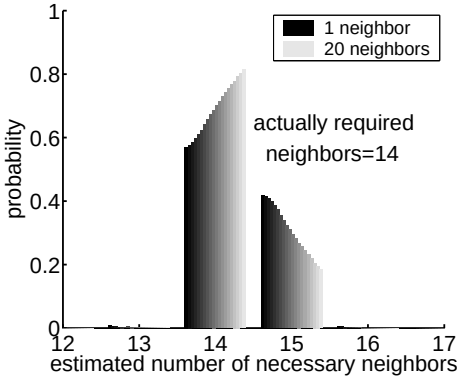

Fig. 17. The upper bound is more sensitive to the number of neighbors.

better the obtained results become. That is, the more realizations of I we can give as an input to the estimator, the more precisely it calculates the actually required number of neighbors and the less often it over- and underestimates this value. Still the estimator underestimates the actual value, even in the case of 20 neighbors.

For comparison, the results obtained by the upper bound are summarized in Figure 17. The bars increase and decrease more intensely than the bars in the last figure. That is due to the fact, that the more realizations of I we obtain, the smaller the confidence interval is going to be. Thus the upper bound will converge to the estimator. Having a closer look

at the figure, we also notice that the probability that the upper bound underestimates the required number of neighbors is negligible but not entirely zero. Obviously, this is especially noticeable for small successorlists, since a small successorlist also means fewer realizations of I. Note, that independent on the size of the successorlist the upper bound is able to rely on the realizations of I gained by its fingerlist. Thus, it supplies an applicable estimate of the required number of neighbors independent on the number of successors used as input.

6 Conclusions

In a P2P network a peer does not exactly know the current size of the overlay network it is participating in. In this paper we utilize the overlay structure established by the Chord P2P algorithm to estimate the current size of the overlay network as seen by a single peer. Thereby we do not only rely on the density of a peer's current successorlist but also exploit information gained by the peer's fingerlist. Unlike the peer's successors the finger pointers are not correlated to each other and therefor deliver a better estimate when the peer maintains a small successorlist. In general, the estimated sizes lie in between $0.5n$ and $2n$, where n is the actual ring size.

We are furthermore able to calculate confidence intervals for our estimator, whereby the upper and lower bounds can be used as estimators themselves. The lower bound underestimates the actual ring size with very high probability, while the upper bound lies well above the actual overlay size. The probability that the lower and upper bound over- or underestimate the target value respectively can be minimized by increasing the confidence level.

Knowing the size of the overlay network is a frequently required feature in P2P networks. Chord, e.g., requires the estimate to maintain a list of $log_2(n)$ successors. The estimator can be used to dynamically adapt the size of a peer's successorlist. The upper bound of our estimator is especially suited for this task, since it practically never underestimates the required number of neighbors.

References

1. Stoica, I. et al.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: ACM SIGCOMM 2001, San Diego, CA (2001)
2. Blake, C., Rodrigues, R.: High availability, scalable storage, dynamic peer networks: Pick two. In: Proceedings of HotOS IX: The 9th Workshop on Hot Topics in Operating Systems, Lihue, Hawaii, USA (2003)
3. Mahajan, R. et al.: Controlling the cost of reliability in peer-to-peer overlays. In: IPTPS 2003, Berkeley, CA, USA (2003)
4. Horowitz, K., Malkhi, D.: Estimating network size from local information. Inf. Process. Lett. **88** (2003) 237–243
5. Bawa, M., et al.: Estimating aggregates on a peer-to-peer network. Technical Report, Dept. of Computer Science, Stanford University (2003)
6. Binzenhöfer, A., Tran-Gia, P.: Delay Analysis of a Chord-based Peer-to-Peer File-Sharing System. In: ATNAC 2004, Sydney, Australia (2004)
7. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. 3rd edn. McGraw-Hill Higher Education (1999)