University of Würzburg
Institute of Computer Science
Research Report Series

Estimating the size of a Chord ring

Andreas Binzenhöfer, Dirk Staehle and Robert Henjes

Report No. 348 February 2005

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

Estimating the size of a Chord ring

Andreas Binzenhöfer, Dirk Staehle and Robert Henjes
Department of Distributed Systems
Institute of Computer Science
University of Würzburg, Am Hubland, 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 a peer's neighbor- and fingerlists. To be able to rate the goodness of the estimator we show how to calculate the corresponding confidence intervals.

1 Introduction

The main purpose of peer-to-peer (P2P) networks is to store data in a decentralized overlay network. Other peers will then be able to retrieve this data using some sort of search algorithm. Traditional P2P algorithms like Gnutella [1], Napster or KaZaA [2] use flooding mechanisms or central index servers to realize searches for data stored in the P2P network. Obviously those networks do not introduce any structure to the overlay architecture. The current generation of P2P algorithms, however, is based on structured overlay networks to make searches faster, more reliable and more efficient. Systems like Chord [3], CAN [4] and Kademlia [5] implement so called Distributed Hash Tables (DHT) to organize their overlay network. A DHT assigns each peer in the overlay an m-bit identifier using a hash function such as SHA-1 [6] or MD5 [7]. Additionally each document that is to be stored in the peer-to-peer network is assigned an m-bit identifier using the same hash function. Based on these ids the underlying P2P mechanism decides where to store the documents. That is, the P2P algorithm determines which peers are going to be responsible for which documents. Peers searching for particular documents will then use the same algorithm to retrieve the searched information from the P2P overlay network. As compared to other DHT based P2P algorithms Chord builds a ring topology (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. As can be seen in Figure 1 the peers are ordered in a clockwise ascending manner on the Chord ring. The main advantage of this

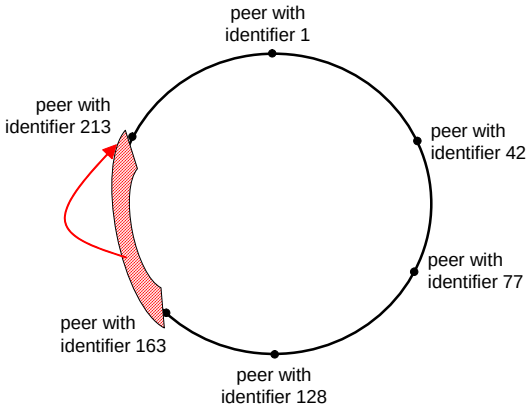

Figure 1: All documents whose identifiers fall into the hatched area will be stored at the peer with identifier 213.

ring structure is that the peer responsible for a particular document is exactly defined. In particular a document is stored at the first peer whose identifier is greater or equal to the documents identifier in the identifier space. All documents whose identifiers fall into the hatched area in Figure 1, e.g., are stored at the peer with identifier 213, since this peer is the first peer that follows this area in a clockwise direction. The advantages that come along with this exact mapping of peers and documents, however, are bought by the overhead needed to maintain the ring structure. In particular the ring structure has to be maintained even under high churn rates. That is, when a great number of peers join or leave the network. To cope with this situations each peer maintains pointers to the first r of its immediate successors on the identifier circle. According to [3] the stability of the Chord ring can be obtained with high probability as long as $r = \Omega^1(\log_2(n))$, where n is the current size 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 adjust r dynamically. 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. 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. In other words a constant value for r is not feasible in practice. Unfortunately a peer is also not able to adjust the size of its neighborlist dynamically to $r = \Omega(\log_2(n))$ since a single peer has no way of knowing the current size n of the Chord ring.

In this paper we therefore introduce an estimator for the current size n of the Chord ring based on the peers current neighborlist. To be able to rate the goodness of the estimator we show how to calculate the corresponding confidence intervals. The estimated value of n can then be used to build a peers neighborlist in actual Chord implementa-

[1]Definition: $T(n) = \Omega(f(n))$ if and only if there are constants c_0 and n_0 such that $T(n) \geq c_0 f(n) \, \forall \, n \geq n_0$

tions. 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. Related work is summarized in Section 3. Section 4 presents the main idea of our estimator. In Section 5 the mathematical framework and all necessary definitions are introduced. The numerical results are then shown in Section 6. Section 7 finally summarizes and concludes the paper.

2 Chord Basics

This section introduces a brief overview of Chord with a focus on aspects relevant to this paper. A more detailed description can be found in [3].

As stated above the Chord algorithm arranges the peers in a clockwise ascending manner on the Chord ring as illustrated in Figure 1. In a very unsophisticated Chord ring each peer would only know the id of its immediate successor in a clockwise direction on the ring. If this successor goes offline the peer does not know the id of the next peer, which would now be its new immediate successor. As a consequence thereof the ring structure is lost and 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 2 shows the successorlist for a peer z for $r = 3$ successors. The list

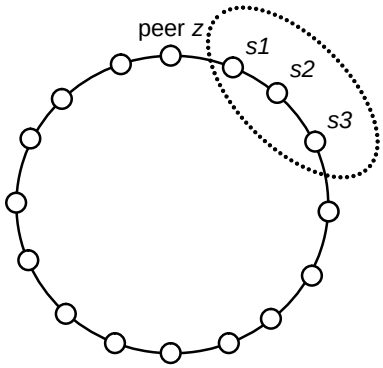

Figure 2: The successor list for peer z consists of the $r = 3$ immediate successors $s1$, $s2$ and $s3$ in a clockwise direction.

consists of $s1$, $s2$ and $s3$ the three immediate successors of peer z. If the immediate successor $s1$ of peer z goes offline now, peer z can still contact the next closest peer $s2$ of its successorlist. As stated in [3] as long as $r = \Omega(\log_2(n))$ a peer is able to know the id of its closest living successor. Later on we will exploit the successorlist of a peer to introduce our estimator.

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 obviously 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. Figure 3 illustrates a search of peer z for another peer y using only the immediate successor. The search has to be forwarded

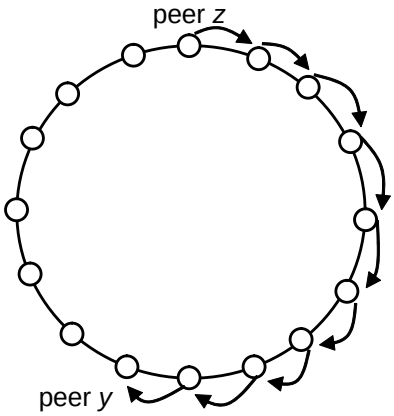

Figure 3: A search of peer z for peer y in a simple Chord ring is passed around the circle using the immediate successors.

half-way around the ring. The average search would require $\frac{n}{2}$ overlay hops, where n is the current size of the Chord ring. 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 4 shows two examplary fingers F_1 and F_2 for the same peer z of the last figure.

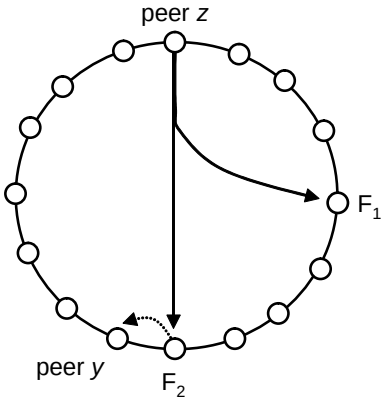

Figure 4: A peers finger pointers can be used to speed up searches. A search of peer z for peer y using finger F_2 only takes two hops as indicated by the dotted line

If peer z would search again for peer y, this time using its finger pointers, the search would only take two overlay hops. For the first hop peer z uses its finger F_2. Peer y can then directly be reached using the successorlist of F_2 as indicated by the dotted line. A detailled mathematical analysis of the search delay in Chord rings can be found in [8].

In the next section we show how to make use of a peers successorlist and fingerlist to estimate the size of the current Chord ring.

3 Related Work

Most works concerning DHT algorithms concentrate on improving the speed and efficiency of queries in such systems, e.g. by taking physical proximity into account. In [9] Proximity Neighbor Selection (PNS) and Proximity Route Selection (PRS) were considered and evaluated. PNS based algorithms chose neighbors in the routing table according to their physical proximity. The PRS algorithms on the other hand deal with the trade-off between the number of hops and the latency of such overlay hops.

Other studies try to make complex queries in DHT based systems possible. In [10] e.g. *n-grams* are used to enable keyword search in DHT based system. Another approach is introduced in [11], where the problem of complex queries is solved by keyword fusion. That is, popular keywords are fused with unpopular keywords once their popularity exceeds a certain threshold.

While all those algorithms concentrate on the efficiency of DHT system, we believe that the real issue concerns their redundancy and stability. As mentioned in Section 1 there is a trade-off between maintenance cost and stability. This trade-off can be controlled by the size r of a peer's successor-list. If r is chosen too small the stability of Chord will be less than desired. In [12], e.g., a survey of different studies regarding session times in various peer-to-peer systems is done. The performance of existing DHT implementations under the observed churn rates is evaluated. The conclusion of the work states that performance is less than desirable at the higher end of these churn rates. We believe that this is at least in part due to the fact that r was set to a constant value of 10, underestimating the size of the regarded Chord rings.

Overestimating the parameter r on the other hand leads to unnecessary overhead. In [13], e.g., it was shown that the real scalability problem can be found in the service bandwidth needed to maintain redundancy and stability in dynamic overlay networks. However, especially in these dynamic networks it is most important to know the size of the current network to be able to adjust the maintenance cost needed to obtain redundancy and stability.

Therefore in [14] the trade-off between high maintenance cost and poor stability in dynamic networks is investigated. The results show that it is crucial to adapt parameters dynamically. However, to be able to adapt the parameters to current conditions, estimates for the current size of the network are needed. The authors introduce an estimator for the size of Pastry based networks, that can in some way be extended to other networks like CAN or Chord. However, there was no mathematical statement about the goodness of the estimator nor any confidence intervals were given. Moreover since the estimator was primarily designed for Pastry networks, it does not exploit additional characteristics that are typical for Chord rings.

A rough estimator for the size of butterfly based P2P overlay networks is introduced in [15]. Other approaches rely on active probing of the overlay network. A distributed way of estimating the size of overlay networks is presented in [16], where an additional

logical ring among existing nodes is maintained and nodes exchange their estimators upon arrival and departure. In [17] the size of general overlay networks is estimated by actively sending samples to other nodes and evaluating the answer statistics.

Current estimators either lack a mathematical description, need additional overhead to actively probe the network or do not exploit all properties of a Chord ring. In this work we therefore present a mathematical substantiated estimator, which is well adapted to the properties of the Chord algorithm. Instead of solely relying on a peer's successorlist it also takes the peers fingerlist into account. We calculate confidence intervals for our estimator and show how it performs in different network sizes and when exposed to different churn rates.

4 Analytical Model

In this section we present the analytical framework of our model based on a peers successor- and fingerlists. At first we have a closer look at the identifier space itself. 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 5.

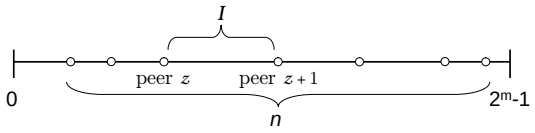

Figure 5: A total of n peers share the identifier space of length 2^m. The random variable I describes the length of the interval between two peers.

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

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

$$\cdots\left(1 - \frac{n-1}{N-i+1}\right)\cdot\left(\frac{n-1}{N-i}\right) \tag{2}$$

$$\approx (1-p)^{i-1}p \approx (1-p)^i p \tag{3}$$

The approximation is justified as $n >> 1$ and consequently $N >> i$. Thus, we can conclude that the Interval $I(z)$ between a peer and its direct neighbor is approximately

geometric with parameter p:

$$I(z) \sim geom(p) \text{ where } p = \frac{n}{2^m} \tag{4}$$

We validate this approximation by generating 10000 snapshots of a chord ring 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 6 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.

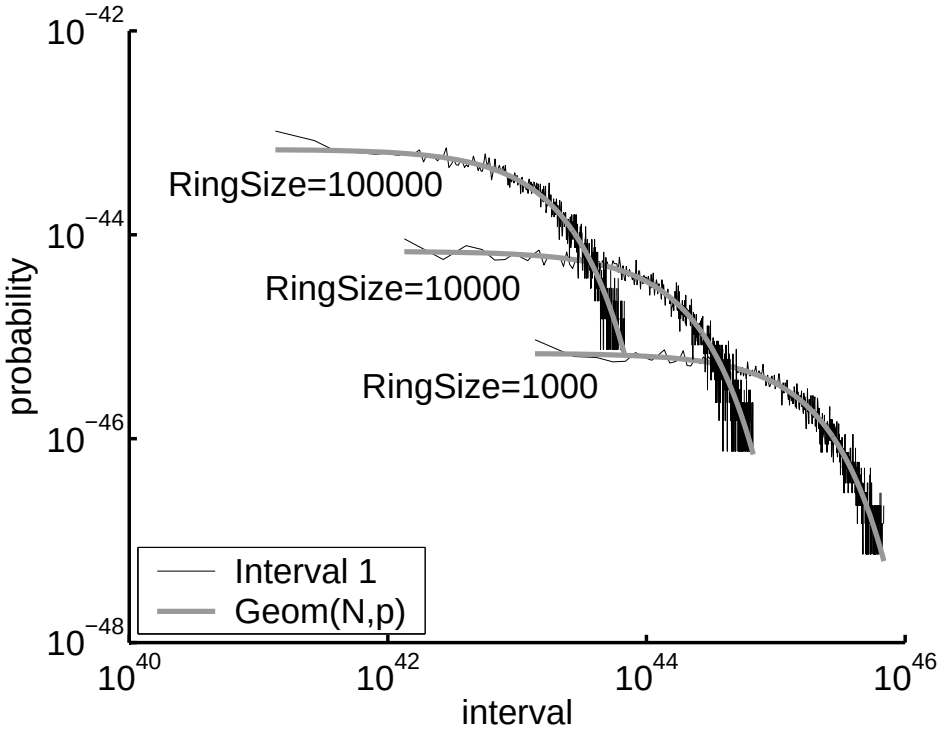

Figure 6: Interval 1 is well-approximated by the geometric distribution

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. The probability that the location of peer $z+2$ is directly after peer $z+1$ is

$$\frac{n-2}{N - (S(z+1) - S(z))}$$

as there are $n-2$ unknown peers and

$$(N - (S(z+1) - S(z)))$$

free identifiers remaining. Consequently, 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

$$\frac{n-2}{N-(S(z+1)-S(z))} \approx p$$

and thus 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 7 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.

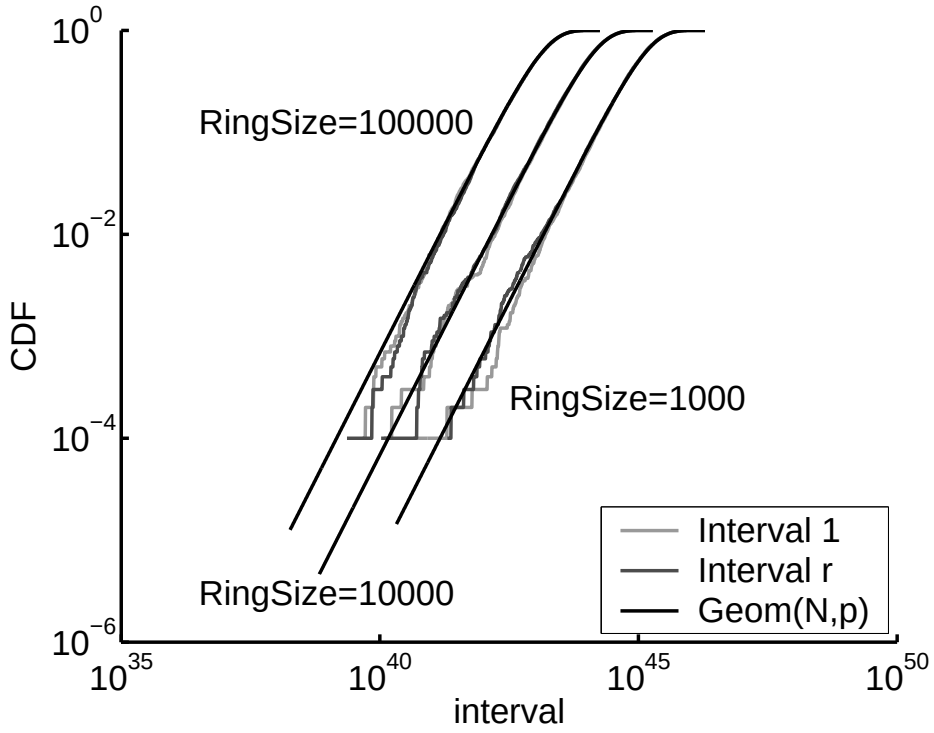

Figure 7: Interval 1 and Interval r follow a geometric distribution.

The main idea of our algorithm is to estimate the parameter p of the geometric distribution of I by $\widehat{p}$. From this we can then conclude

$$\widehat{n} = \widehat{p} \cdot 2^m$$

To be able to estimate $\widehat{p}$ we need to obtain realizations of I. As stated above, those can be gathered by looking at our neighborlist. As shown in Figure 8 the intervals between

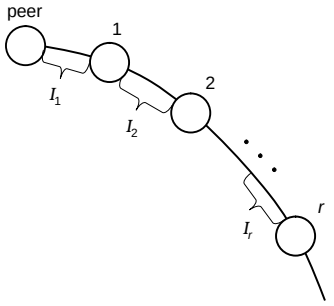

Figure 8: The intervals between a peers r successors can be regarded as realizations of the random variable I.

a peers r immediate successors can be regarded as r different realizations of the random variable I. However, the more realizations of I we obtain the better our estimator is going to be. More realizations of I can be found if we have a closer look at a peers fingerlist as presented in Section 2. As has been shown in [3] 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 peers successor list. 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 9 illustrates this issue in detail. The Figure shows three exemplary

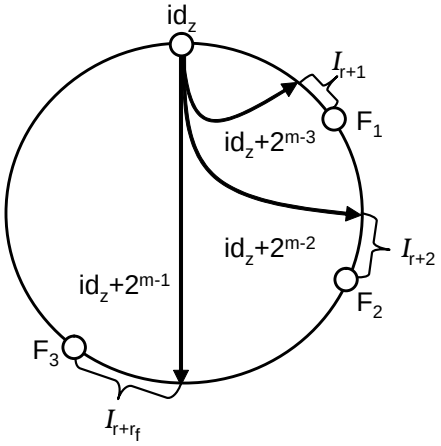

Figure 9: The distance between the theoretical and actual position of a peers *i-th* finger can be regarded as another realization of the random variable I.

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. The explanation can be found in Figure 10. 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

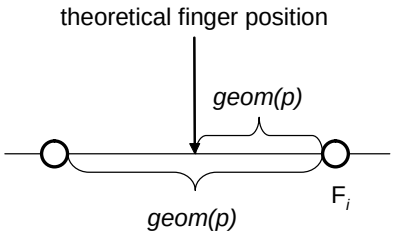

Figure 10: Due to the memoryless property of the geometric distribution the interval between a fingers theoretical and actual position is also geometrically distributed and can thus be regarded as another realization of I.

point in this interval, due to the memoryless property of the geometric distribution, we can then conclude that the interval between the theoretical position of the finger and the actual finger is as well geometrically distributed with the same parameter p. Again, we validate this assumption by means of the snapshots we used above. Figure 11 compares the distances of the theoretical and actual finger positions with the geometric position. We consider only those fingers that don't coincide with the successorlist. The Figure shows the geometric distribution with regard to three different ring sizes. Each of these distribution is compared to the simulated distributions of each finger. Note that there are more simulated curves for the ring with 10000 peers than with 1000 peers, as there are more distinct fingers in larger rings as stated above. Again the plot is presented on a log-log scale, since the curves match exactly on a linear scale. 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. The next Section describes how we are going to estimate the parameter p from this set of realizations.

5 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 peers successorlist. Since the estimator is based on a peers successor- and fingerlist and those lists in turn are adjusted according to the estimator, we assume that to get started, a peer is notified about the current size of the Chord ring by its immediate successor when first entering the network. In this section we show how to estimate the parameter p of the geometric distribution of I using a maximum-likelihood estimator (MLE). The MLE is used since we already know that the random variable I is geometrically distributed but the parameter p is still unknown. The basis for the MLE is a likelihood function $L(p)$ which is defined as follows:

$$L(p) = f_p(I_1) f_p(I_2) \cdots f_p(I_i)$$

where $f_p(I)$ is the probability mass function with parameter p. The MLE $\widehat{p}$ of the unknown value of p is then defined to be the value that maximizes the likelihood function

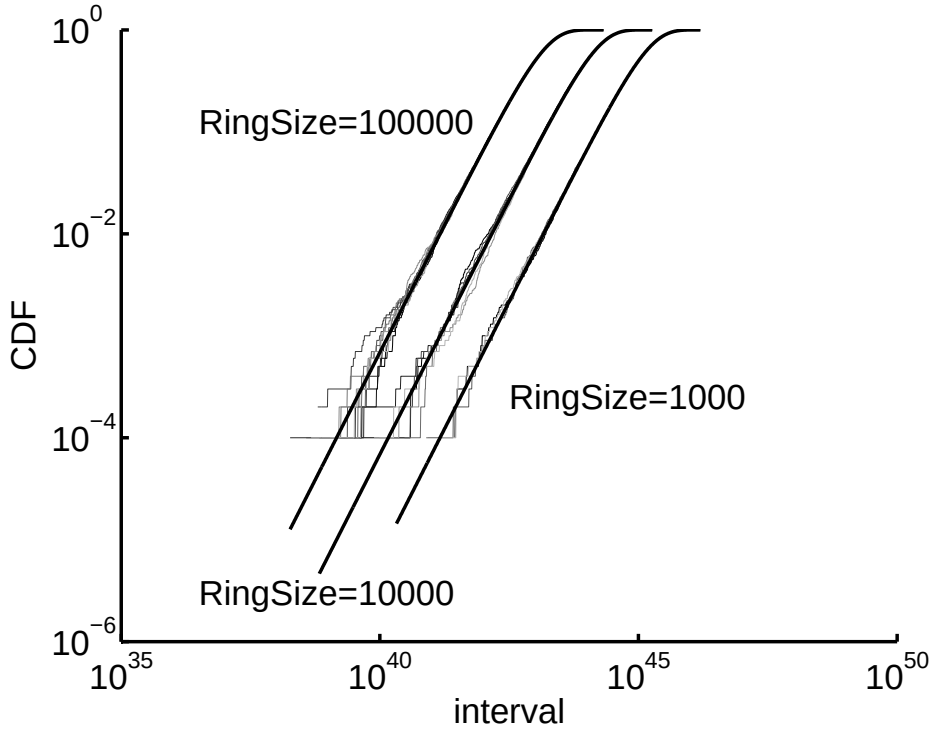

Figure 11: The interval between actual finger position and theoretical finger position is geometric.

$L(p)$. That is,

$$L(\widehat{p}) \geq L(p)$$

for all possible values of p. In our case we have

$$f_p(I) = (1-p)^I p$$

and

$$L(p) = (1-p)^{\sum_{i=1}^{r+r_f} I_i} p$$

As has been shown in [18] the MLE in this case can be computed as

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

where $\overline{I}(2r)$ is the sample mean. With $\widehat{p}$ we can then 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 successor-list will be set to

$$\widehat{r} = \lceil log_2(\widehat{n}) \rceil$$

An obvious advantage of this approach is that the size of the successor-list 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 successor list is logarithmically dependent on the size of the Chord ring. In practice a peer is going to use this estimator to set the size of its successor-list as follows. When first entering the Chord ring, a peer learns the current size of the Chord ring from its direct neighbors and adjusts the size of its successor-list accordingly. Afterwards it periodically uses the MLE $\widehat{p}$ to estimate the current size of the Chord ring and dynamically adapts the size of its successor-list. 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 [18] 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 consequences of overestimating the real value of p. That is due to the fact that a successor-list, that is too small has a negative effect on the stability of the Chord ring. A successor-list 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 successor-list as

$$\widehat{r}_+ = \lceil log_2(\widehat{n_+}) \rceil$$

Again, we round up to minimize the probability of underestimating the real value of r. The next section summarizes how the estimator performs in an actual Chord implementation.

6 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 length 2^m. Then the distances between the first r consecutive peers are calculated and given as input to our estimator. We regard different ring sizes to see how the estimator scales to larger networks. Furthermore, we evaluate the difference between the upper and lower limit of the estimator and study the influence of the corresponding confidence levels. Additionally, we investigate how accurate the estimator and its upper bound are able to estimate the actually required number of successors. Finally we study the influence of churn by varying the number of successors a peer maintains.

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 12 shows the results of our simulations for a given ring size of 10000 and a successorlist of size 14. As can be seen in the Figure, our estimator $\widehat{n}$ is well in the right order

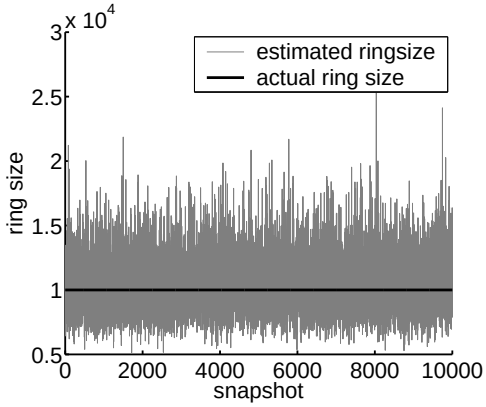

Figure 12: 10000 estimates of the ring size as compared to the actual ringsize

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 13, 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

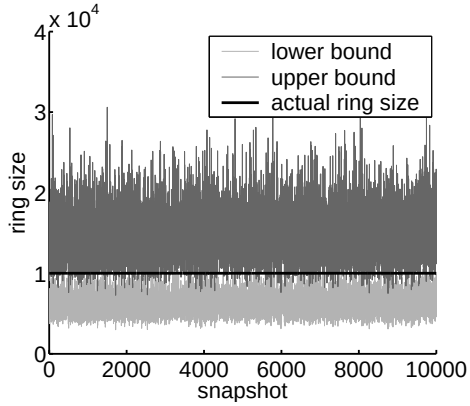

Figure 13: The lower and upper bound of the estimator with a confidence level of 95% as compared to the actual ring size

stays beneath the actual size of the ring with high probability. Whereas the upper bound ranges between n and $2n$ to $3n$, underestimating the real value of n at times.

To analyze the number of times the lower bound overestimates and the upper bound underestimates the actual ring size we plotted the sorted snapshots in Figure 14. The Figure shows the normalized 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 times the upper bound underestimates the actual ring size, the part of the lower bound above the black line the number of times the lower bound overestimates the actual ring size respectively. Note

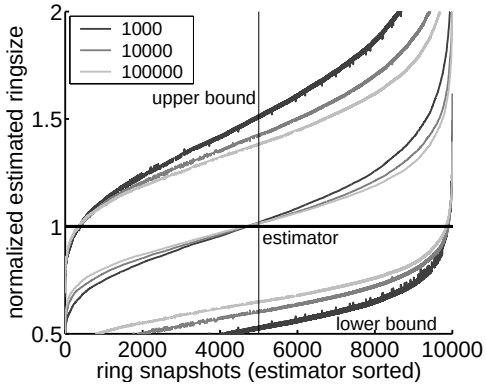

Figure 14: Sorted estimates gained by the estimator and its lower and upper bounds

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 highly 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.

As can be seen in Figure 15 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

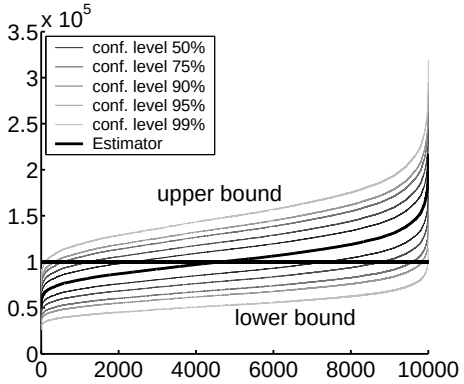

Figure 15: Influence of the confidence level on the upper and lower bound in a network of size 10^5

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 the frequency of under- and overestimating the actual size and the precision of the estimate has to be adapted to the application of the estimator.

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 16 and 17 show the estimated number of required neighbors in a network of size 10^4 and 10^5. In Figure 16 the actually required number of neighbors is $14 = \lceil \log_2(10^4) \rceil$. The regular estimator provides the 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

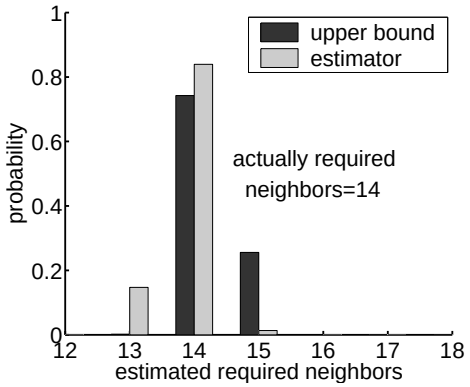

Figure 16: The upper bound does never underestimate the actually required number of neighbors, as does the regular estimator

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 17) 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

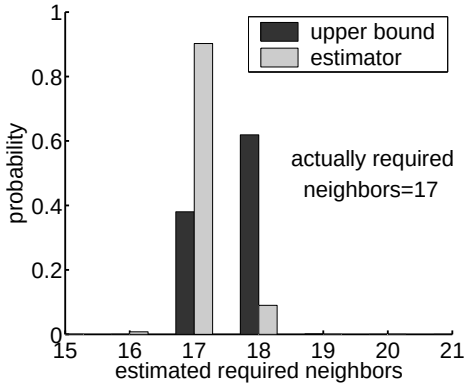

Figure 17: The regular estimator reaches the actually required number of neighbors in about 90% of all cases, thereby scarcely underestimating the actual value.

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 18 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

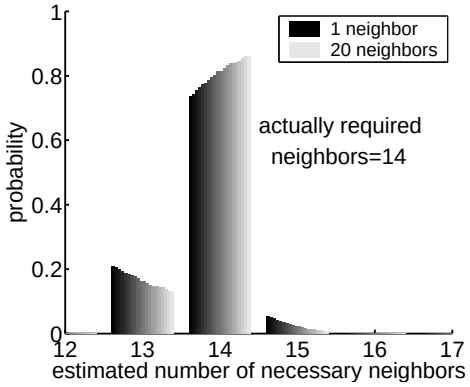

Figure 18: The bars represent the results obtained by using 1 to 20 neighbors as input for the regular estimator in a ring of size 10^4

the estimator can rely on, the 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 19. The bars increase and decrease more rapidly 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. Moreover since $13 = \log_2(8192)$ all estimated values of $n < 8913$ will result in an underestimation of r. Thus, the estimator can not fully take advantage of the mathematical round step. 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.

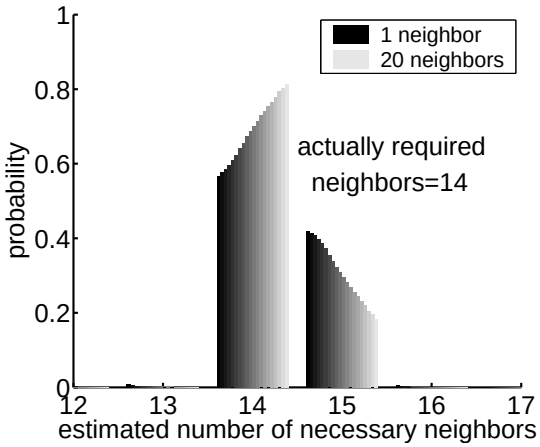

Figure 19: The upper bound is more sensitive to the number of neighbors than the regular estimator

7 Conclusions

In a P2P network a peer has no way of knowing 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 peers current successorlist but also exploit information gained by the peers fingerlist. Unlike the peers successors the finger pointers are note correlated to each other and still deliver a good 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 number of times 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. The most obvious application is the dynamical adaptation of a peers successorlist. The upper bound of our estimator is especially suited for this task, since it practically never underestimates the required number of neighbors. How exactly the estimator and its upper and lower bounds can be applied in practice and what other areas of application there are is a matter of future work.

References

[1] Gnutella website, “http://www.gnutelliums.com.”

[2] KaZaA website, “http://www.kazaa.com/us/index.htm.”

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM 2001*, (San Diego, CA), August 2001.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *ACM SIGCOMM 2001*, (San Diego, CA, USA), 2001.

[5] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS 2002*, (MIT Faculty Club, Cambridge, MA, USA), March 2002.

[6] FIPS PUB 180-1, "Secure hash standard." Federal Information Processing Standards Publication 180-1, April 1995.

[7] R. Rivest, "RFC 1321 - The MD5 Message-Digest Algorithm," April 1992.

[8] A. Binzenhöfer and P. Tran-Gia, "Delay Analysis of a Chord-based Peer-to-Peer File-Sharing System," in *ATNAC 2004*, (Sydney, Australia), December 2004.

[9] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of dht routing geometry on resilience and proximity," in *ACM SIGCOMM 2003*, (Karlsruhe, Germany), August 2003.

[10] M. H. et al., "Complex queries in dht-based peer-to-peer networks," in *IPTPS 2002*, (MIT Faculty Club, Cambridge, MA, USA), March 2002.

[11] L. Liu, K. D. Ryu, and K.-W. Lee, "Supporting efficient keyword-based file search in peer-to-peer file sharing systems," in *Globecom 2004*, (Dallas, TX), November 2004.

[12] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *2004 USENIX Annual Technical Conference*, (Boston, MA), June 2004.

[13] C. Blake and R. Rodrigues, "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), May 2003.

[14] R. M. et al., "Controlling the cost of reliability in peer-to-peer overlays," in *IPTPS 2003*, (Berkeley, CA, USA), February 2003.

[15] D. M. et al., "Viceroy: A scalable and dynamic emulation of the butterfly," in *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing*, 2002.

[16] K. Horowitz and D. Malkhi, "Estimating network size from local information," *Inf. Process. Lett.*, vol. 88, no. 5, pp. 237–243, 2003.

[17] M. B. et al., "Estimating aggregates on a peer-to-peer network." Technical Report, Dept. of Computer Science, Stanford University, 2003.

[18] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 3rd ed., 1999.