University of Würzburg
Institute of Computer Science
Research Report Series

On the Stability of Chord-based P2P Systems

Andreas Binzenhöfer, Dirk Staehle and Robert Henjes

Report No. 347 January 2005

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

On the Stability of Chord-based P2P Systems

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 current generation of P2P networks is intended to provide cost-effective alternatives to the traditional client-server architecture. The main goal is to be able to store and retrieve data in a decentralized manner. The challenge in doing so consists in creating a stable overlay network that allows for fast and efficient searches. In this paper we consider the Chord P2P algorithm in this context and analyze its stability and efficiency with stochastic methods. We present realistic probabilities for the problem of a disconnection and confirm the scalability of key lookups.

1 Introduction

The peer-to-peer (P2P) hype over the last years was mainly driven by legally questionable applications like filesharing. Current P2P networks like gnutella [1], emule [2] and KaZaa [3] are primarily used to share movies, music and the like [4]. Meanwhile, however, the first business models based on P2P architectures start to emerge. An ever increasing number of companies discovers the advantages of decentralized P2P networks. They are no longer dependent on a single central unit nor do they have to invest in server farms to guarantee the scalability of their systems. Thanks to a new generation of P2P systems, based on so called Distributed Hash Tables (DHTs), distributed storage systems or distributed indexes become possible. Together with those new systems, however, new challenges arise as well. The main task of a DHT is to provide a scalable method of associating data with a particular location in a distributed overlay network. Moreover, it has to offer a fast and efficient way of retrieving this information at a later point in time. DHT based P2P algorithms are intended to replace central servers. To do so in a business environment they have to be able to guarantee efficiency and, most important, a consistent global view of the stored data. Since P2P overlay networks are built on top of IP networks, there exists a danger, that the overlay collapses entirely or is split into two networks, that do not know about the existence of each other. In case of such an inconsistent overlay state, successful searches can no longer be guaranteed and it might even not be possible to reestablish a stable overlay network again. An analysis of the evolution of such systems can be found in [5] and [6]. In this paper we concentrate on Chord, a DHT based P2P algorithm, and analyze the way it preserves reachability and stability of its overlay network.

The remainder of this paper is structured as follows. In Section 2 we describe the basic ideas behind Chord that are relevant to our analysis. In particular we summarize how Chord realizes search efficiency and overlay stability. Section 3 validates the scalability of data lookups in the overlay network. In Section 4 the probability for a loss of overlay stability in Chord rings is calculated and fortified by more realistic failure probabilities in Section 5. The results of our analysis are presented in Section 6 and Section 7 finally concludes this paper.

2 Chord basics

The main purpose of 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. The Chord algorithm solves this problems by arranging the participating peers on a ring structure. The position of a peer on this overlay ring is determined by an m-bit identifier using a hash function such as SHA-1 [7] or MD5 [8]. Additionally each document that is to be stored in the P2P network is assigned an m-bit identifier using the same hash function. Based on these identifiers 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.

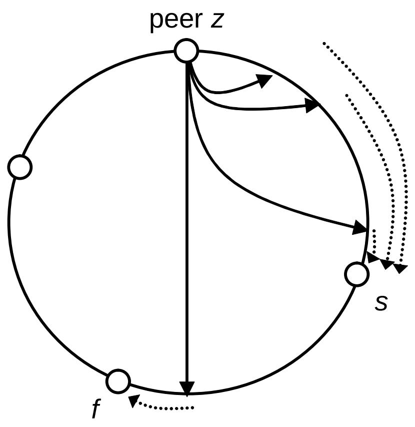

Figure 1: A Chord ring of size $n = 4$, where the first three fingers of peer z coincide with its direct successor s.

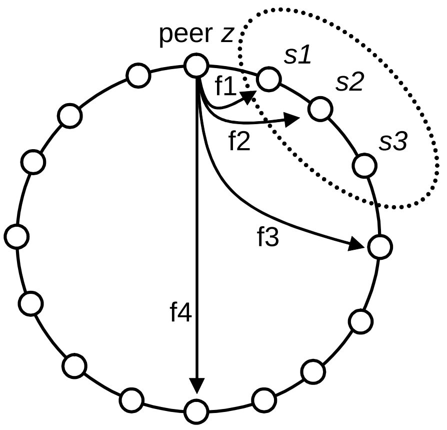

Figure 2: A Chord ring of size $n = 16$, where each peer maintains $r = 3$ successors. The fingers for peer z are also shown.

To maintain the ring structure of the overlay, each peer stores pointers to the first r successors on the ring, i.e. the first r peers that succeed the peer in a clockwise direction on the ring. Thus, if one of the peer's r successors goes offline, the peer will still know the next $r-1$ peers on the ring. If a peer, however loses all its r successors, the ring will be disconnected. According to [9] the connectivity 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 Section 4 we analyze the probability of a disconnection of the ring in detail.

In addition to its list of successors each peer also maintains a list of pointers to more distant peers. The entries of this list are called fingers. They are used as shortcuts through the ring when searching for other peers or documents. The *i-th* entry of the fingerlist of peer z contains the identity of the first peer that succeeds z's own m-bit identifier by at least 2^{i-1} on the Chord ring. That is, the *i-th* finger of peer z with hash value id_z is pointing to the direct successor of $id_z + 2^{i-1}$ in a clockwise direction. Fig. 1 shows a simplified example using a Chord ring of size $n = 4$. The first three fingers of peer z are all pointing to peer s, since peer s is the first peer that succeeds the corresponding theoretical finger positions in a clockwise direction on the ring. The fourth finger is pointing to peer f accordingly. In theory a peer has to maintain a list of m fingers, where 2^m is the size of the hash functions co-domain. According to [9], however, there are only $O(\log_2(n))$ distinct fingers in this list, since some of them coincide as illustrated in Fig. 1. An in-depth analysis of searches in Chord rings can be found in [10].

Finally, Fig. 2 shows a snapshot of a Chord ring consisting of 16 overlay peers. Peer z maintains a successorlist of size 3 consisting of peers s1, s2 and s3. The distinct fingers of peer z are pointing to peers f1, f2, f3, and f4 respectively. In the following section we will validate that a peer only has to store $O(\log_2(n))$ different finger pointers and even render this statement more precisely.

3 Validation of Chords Search Efficiency

One of the main features of a Chord ring is the ability to search for other peers and resources. To speed up such searches, a peer can use its finger pointers as shortcuts in the overlay network. When a peer first joins the overlay network it initializes its fingers by using the fingerlist given to it by its direct successor. To keep these shortcuts up to date a peer has to update its fingers in some periodic way. Since the most distant finger represents the best shortcut through the Chord ring it is advisable to do so starting with the more distant fingers and moving to the closer ones step by step. Once the updated finger equals the direct successor, the peer should restart with the most distant finger, since up from this point all closer fingers will coincide with the direct successor, just as the first three fingers shown in Fig. 1.

As mentioned above, the *i-th* entry in the fingerlist of peer z contains the identity of the first peer that succeeds z's own hash-value by at least 2^{i-1} on the Chord ring. Assuming an identifier space of size 2^m, peer z with hash value id_z has its fingers at

$$id_z + 2^{i-1} \text{ for } i = 1 \text{ to } m.$$

Typically, the size of the identifier space is set to $m = 160$ bits. That is, each peer on the Chord ring has $m = 160$ possible fingers to update. However, according to [9]

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

only $O(\log_2(n))$ of them are actually different, where n is the size of the Chord ring. Assuming equally distributed peer IDs we are able to state the above more precisely.

Theorem 3.1 *In a Chord ring of size n with equally distributed peer IDs each peer has exactly $\lceil log_2(n) \rceil$ different fingers.*

Proof To prove the theorem, we calculate the index i_s of the most distant finger that is still pointing to the direct successor of the peer. Since we are assuming an identifier space of size 2^m and an overlay network of size n with equally spaced peer IDs, the distance to the direct successor of the peer is

$$\frac{2^m}{n}.$$

Furthermore the fingers of the peer are pointing to

$$id + 2^{i-1}.$$

The index i_s of the most distant finger still pointing to the direct successor can thus be calculated as

$$\begin{aligned} i_s &= max\left\{ i : 2^{i-1} \leq \frac{2^m}{n} \right\} \\ &= max\left\{ i : i \leq m - \log_2(n) + 1 \right\} \\ &= \lfloor m - \log_2(n) + 1 \rfloor \end{aligned}$$

Since there are m theoretical fingerlist entries and starting with index i_s all of them are actually different, a peer has a total of exactly:

$$m - (i - 1) = m - (\lfloor m - \log_2(n) + 1 \rfloor - 1) = \lceil \log_2(n) \rceil$$

different fingers.

If we abandon the assumption that the peer IDs are distributed equally, we can still calculate $p_{coin}(x)$, the probability that at least the first x fingers coincide with the direct successor of the peer. Therefor, we need $p_{area}(s)$, the probability that the hash ID of a peer does not lie within a specific area of size s in the identifier space. Obviously,

$$p_{area}(s) = 1 - \frac{s}{2^m}.$$

Considering that $p_{coin}(x)$ is the probability, that none of the remaining $n - 1$ overlay peers lies between a peer and the theoretical position of its $x - th$ finger, it follows that

$$p_{coin}(x) = p_{area}\left(2^{x-1}\right)^{n-1} = \left(1 - \frac{2^{x-1}}{2^m}\right)^{n-1}.$$

Fig. 3 shows how this probability affects the number of different fingers in regular Chord rings. The figure shows $p_{coin}(x)$ plotted against x, the number of fingers that coincide

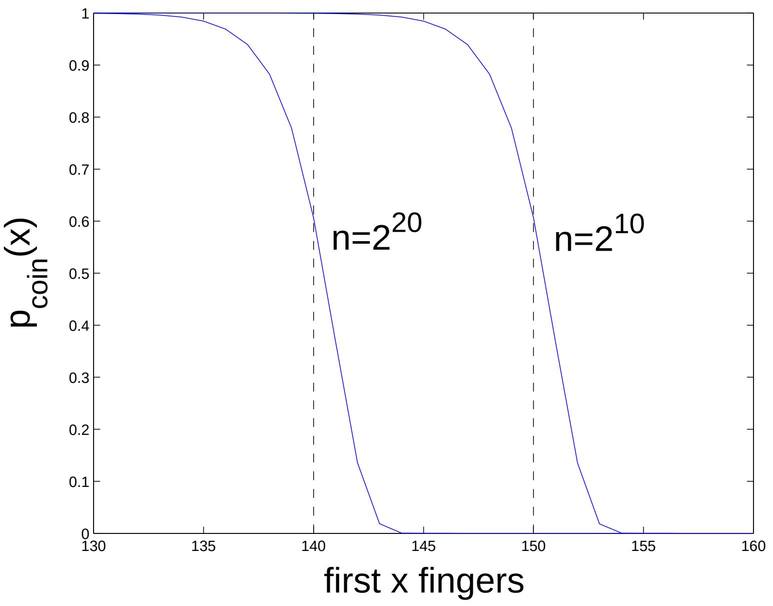

Figure 3: Probability that the first x fingers coincide with the peers direct successor

with the direct neighbor of the peer, for two different overlay sizes. For $n = 2^j$ we expect to obtain j different fingers. That is, using an identifier space of size $m = 160$ the first $160 - j$ fingers should coincide with the peers direct successor. As can be seen in the figure it is very unlikely that a peer has significantly more or less distinct fingers than expected. So as long as the hash function distributes the peer IDs uniformly to the identifier space a peer maintains approximately $\log_2(n)$ different fingers. In the next section we shift our focus to the peers successorlist and calculate the probability that at Chord ring loses its overlay structure.

4 Validation of Chords Stability

A P2P overlay network is connected if there exists a route from every peer to every other peer. Running the Chord algorithm each peer maintains a list of $O(\log_2(n))$ successors to keep the overlay connected. A peer gets disconnected from the network if all of its successors fail. According to [9] a Chord overlay network stays connected with high probability, even if every peer fails with probability $\frac{1}{2}$. The proof relies on the fact that even so every individual peer fails with probability $\frac{1}{2}$ it is very unlikely that all $O(\log_2(n))$ successors of a peer fail at the same time. The conclusion that thus all peers stay connected with high probability, however, misses a subtle point. That is, even so a local disconnection (one specific peer loses all its successors) might be very unlikely one can not draw the conclusion that a global disconnection (at least one peer in the overlay loses all its successors) is very unlikely as well. To gain a better understanding of this subtle but important point, we introduce some definitions:

- p_{fail}: probability that a node fails
- $p_{ld}(r)$: probability that a specific node loses all its r successors and gets locally disconnected
- $p_{gd}(n, r, p_{fail})$: probability of a global disconnection, i.e. the probability that at least one peer gets locally disconnected in a network of size n, where each node knows r successors, and each node fails with probability p_{fail}

The probability for a local disconnection can then easily be calculated as

$$p_{ld}(r) = p_{fail}^r.$$

Obviously, the more successors a peer has, the less likely it gets locally disconnected. Since peers usually maintain a succesorlist of size[2] $r = O(\log_2(n))$, a local disconnection is less likely in larger networks. However, based on this observation alone, we can not conclude that the probability of a global disconnection is comparably small as well. That is, the more nodes there are in the overlay network, the higher the probability that at least one of them gets locally disconnected. In other words, there is a trade-off between these two mechanisms. On the one hand, the larger the Chord ring becomes, the more successors are maintained by a peer, resulting in a smaller probability for a local disconnection. On the other hand, the larger the Chord ring becomes, the more peers run the risk of getting locally disconnected, resulting in a higher probability for a global disconnection.

To estimate the stability of a Chord ring, we need to calculate the probability $p_{gd}(n, r, p_{fail})$ of a global disconnection. That is, we need to calculate the probability, that at least once r or more contiguous peers fail on the Chord ring. As an approximation we neglect the ring structure of the overlay network and imagine the overlay peers arranged in an ascending row as shown in Fig. 4. We regard the probability $p_{rd}(x, r, p_{fail})$, that at least

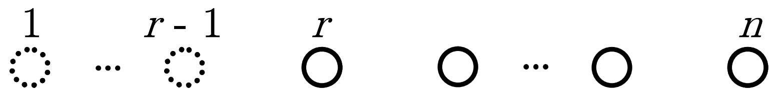

Figure 4: The n peers of a Chord ring arranged in an ascending row.

once r or more contiguous peers fail in such a row of x peers. Moreover, we can assume a random distribution of failures, since the hash function distributes peers equally in the identifier space and physical proximity does thus not reflect overlay proximity. For the sake of simplicity we use the short notation $p_{rd}(x)$ instead of $p_{rd}(x, r, p_{fail})$ where appropriate. Obviously, the probability that r or more peers fail in a row of less than r peers is zero, as indicated by the dotted peers in Fig. 4. If we consider the same probability in a row of exactly r peers, all peers have to fail accordingly. The corresponding

[2]See [11] for a discussion of how to estimate the size n of the current overlay network

formulae are:

$$
\begin{aligned}
p_{rd}(x, r, p_{fail}) &= 0 \quad \text{if } x < r & (1)\\
p_{rd}(x, r, p_{fail}) &= p_{ld}(r) = p_{fail}^{r} \quad \text{if } x = r & (2)
\end{aligned}
$$

In case of $x > r$ we obtain:

$$p_{rd}(x) = p_{rd}(x-1) + (1 - p_{rd}(x-r-1)) \cdot (1 - p_{fail}) \cdot p_{ld}(r) \quad (3)$$

The formula is build recursively. To calculate $p_{rd}(x)$, we take the probability $p_{rd}(x-1)$, that there was at least one local disconnection in the first $x-1$ peers and add the probability, that the first local disconnection occurs at peer x. The second addend of this sum is best explained using Fig. 5. There are two requirements in order that the

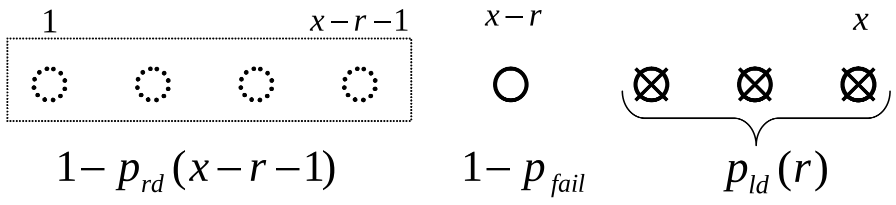

Figure 5: Probability, that the first local disconnection occurs at peer x.

first local disconnection occurs exactly at peer x. First of all there must not be a local disconnection in the first $x-r-1$ peers as indicated by the dotted box in Fig. 5. Secondly, peer $x-r$ must not fail, while all of the last r peers have to fail to cause the disconnection at peer x.

According to the above formula, the first local disconnection can occur at peer r. That is, there are still $r-1$ peers, that could experience a local disconnection but are not accounted for in our equation. To improve the accuracy of our approximation, we add $r-1$ peers at the end of the row as shown in Fig 6. Thus, there are n peers in a row of

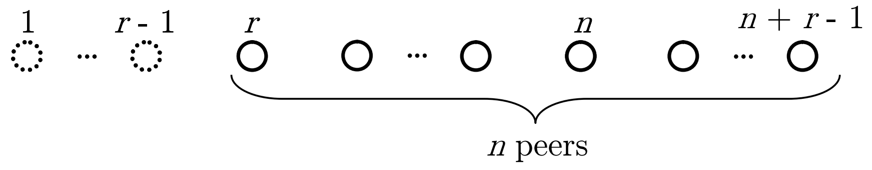

Figure 6: To incorporate the first $r-1$ peers, they are added at the end of the row.

$n+r-1$ peers that can experience a local disconnection. The resulting approximation for the probability of a global disconnection in a Chord ring of size n is:

$$p_{gd}(n, r, p_{fail}) \approx p_{rd}(n+r-1, r, p_{fail}) \quad (4)$$

The remaining approximation arises as we neglect the ring structure of the overlay network. In fact the probability is slightly to high, since the $r-1$ peers we added at the end of the row are obviously correlated with the first $r-1$ peers in the row. That is, there are some failure patterns in the last $r-1$ recursion steps, that have already been taken into account before and are thus counted twice.

In Section 6 we validate Eq. 4 by simulation and present realistic probabilities of a global disconnection. Note that the formula is not limited to the special case of

$$r = \lceil \log_2(n) \rceil.$$

In fact we are able to evaluate the consequences of using too large or to small values for r, i.e. of using more or less than $log_2(n)$ successors.

5 Realistic Failure Probabilities

In the last section we were simply assuming values for p_{fail}, the probability that a node fails. In practice, however, there is not much sense in saying a node fails with a certain probability, without specifying a corresponding time frame. To guarantee overlay stability a Chord peer refreshes its successorlist every t_{stab} seconds by periodically calling a *stabilize()* procedure. This *stabilize()* function takes care that a peers successorlist is up to date by merging its list with the list of its closest successor alive. Thus, a peer gets locally disconnected if all of its known successors go offline between two *stabilize()* calls. Therefore, one should consider the probability, that a peer fails within this periodic update interval instead of assuming some arbitrary values for p_{fail}.

On account of this, we regard the average online time of a peer E_{on} in seconds. Assuming that the online time is exponentially distributed with $\lambda_{on} = \frac{1}{E_{on}}$ it follows that

$$A(t) = 1 - e^{-\lambda_{on} t} \tag{5}$$

is the distribution function of the online time of a single peer. Due to the memoryless property of the exponential distribution the probability that a peer goes offline within t_{stab} seconds is[3]:

$$p_{fail} = A(t_{stab}) = P(A \leq t_{stab}). \tag{6}$$

We can then use this p_{fail} in Eq. 4 to calculate the probability of a global disconnection within t_{stab} seconds. The probability of a global disconnection increases with the number of *stabilize()* calls. That is, the longer the Chord ring exists, the greater the probability of a global disconnection within its lifetime becomes. The probability $p_{it}(n,i)$ that a Chord ring of size n gets globally disconnected sometime within i *stabilize()* calls can be calculated as follows:

$$p_{it}(n,i) = 1 - (1 - p_{gd}(n, r, p_{fail}))^i. \tag{7}$$

In Section 6 we present a parameter study, covering reasonable values for t_{stab}, E_{on} and r, the current size of the successorlist. We also show the impact of realistic failure probabilities on the probability of a global disconnection and analyze how this probability increases over time.

[3]The probability that a peer goes offline and online again within t_{stab} seconds is neglected in this context.

6 Numerical Results

In this section we concentrate on results regarding the problem of a disconnection. At first we have a closer look at the probability of a local disconnection. Figure 7 plots the probability of a local disconnection against the overlay size for three different failure probabilities of a peer. The number of successors is thereby set to $\lceil \log_2(n) \rceil$. As expected the probability of a local disconnection strongly decreases with the size of the overlay network. This is obviously due to the fact that a peer maintains more successors in larger networks and is thus less likely to be disconnected. Note that in a ring of size $n = 10^6$ and a failure probability of $p_{fail} = \frac{1}{2}$ we have a very low probability of a local disconnection of about 10^{-6}.

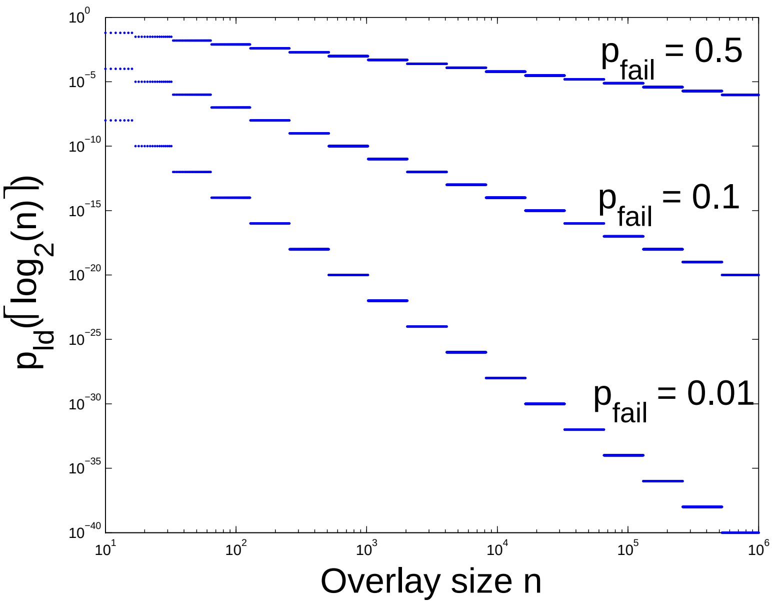

Figure 7: Probability of a local disconnection for different values of p_{fail}

To show, that based on these facts alone, we can not derive a very low probability for a global disconnection as well, we calculate the probability of a global disconnection for $p_{fail} = \frac{1}{2}$. Figure 8 shows this probability for networks of sizes $n = 2^k$, where each peer maintains a successorlist of size $r = \log_2(n) = k$. The probability of a global disconnection does indeed decrease with the size of the overlay network. However, it does not run towards zero but asymptotically reaches a probability of about 40 percent. So when every node fails with probability $p_{fail} = \frac{1}{2}$ and every peer maintains a successorlist of size $r = \log_2(n)$ Chord does not stay connected with very high probability but gets disconnected with a probability of roughly 40 percent.

To confirm this result we simulated the probability of a global disconnection by generating snapshots of rings of a specific size and counting the percentage of those rings that did not get disconnected after 50 percent of all peers failed. The simulations were repeated until the confidence intervals became smaller than 0.001. For smaller values

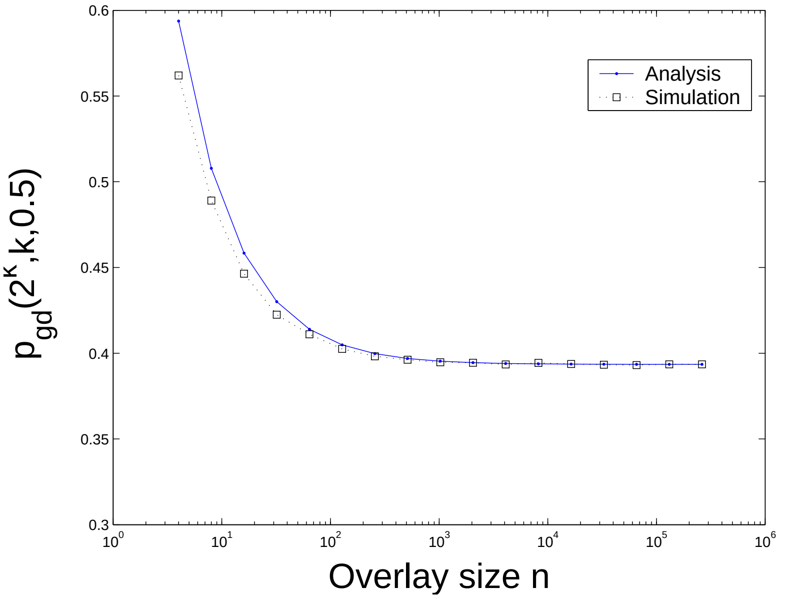

Figure 8: Probability of a global disconnection in the special case of $p_{fail} = \frac{1}{2}$.

of n the results obtained by our analysis are slightly above the simulated values as the analysis neglects the ring structure. The error becomes negligible for overlay sizes above $n = 100$.

In practice, however, a failure probability of $p_{fail} = \frac{1}{2}$ is obviously too pessimistic. To obtain realistic values for p_{fail} we evaluate Eq. 6 for different average online times of a peer and different values of t_{stab}. Figure 9 shows that even if the average peer only stays online for 10 minutes and successorlists are only refreshed every 60 seconds, the probability that a peer fails within this frame of time is still less than 10 percent.

In the following analysis we therefore concentrate on p_{fail}= 0.1, 0.05 and 0.01. Figure 10 illustrates that a global disconnection is very unlikely for these values of p_{fail}. Even for a peer failure probability of 10 percent a Chord ring of size 10^5 will be globally disconnected with a probability of less than 10^{-12}. The staircase form of the curve arises from the fact, that the plot is done for arbitrary n and corresponding successorlists of size $r = \lceil \log_2(n) \rceil$. So whenever the overlay size n crosses a power of two, each peer starts to maintain one additional successor in its successorlist. Therefore the probability of a disconnection abruptly decreases whenever a power of two is exceeded. It then slightly increases until the next power of two, since the probability of a local disconnection stays the same, but there are more peers that can get disconnected and cause a global disconnection.

So far, the results relied on a dynamic adaptation of the size of a peers successorlist. In practice, however, it is more common to choose a fixed succesorlist size a priori. Figure 11 illustrates the probability of a global disconnection for fixed successorlist sizes

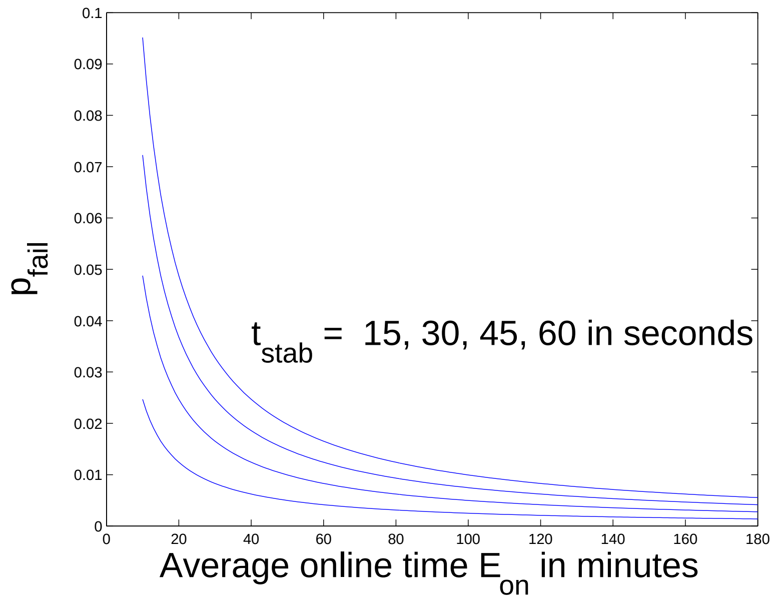

Figure 9: Failure probabilities in dependency of the average online time of a peer

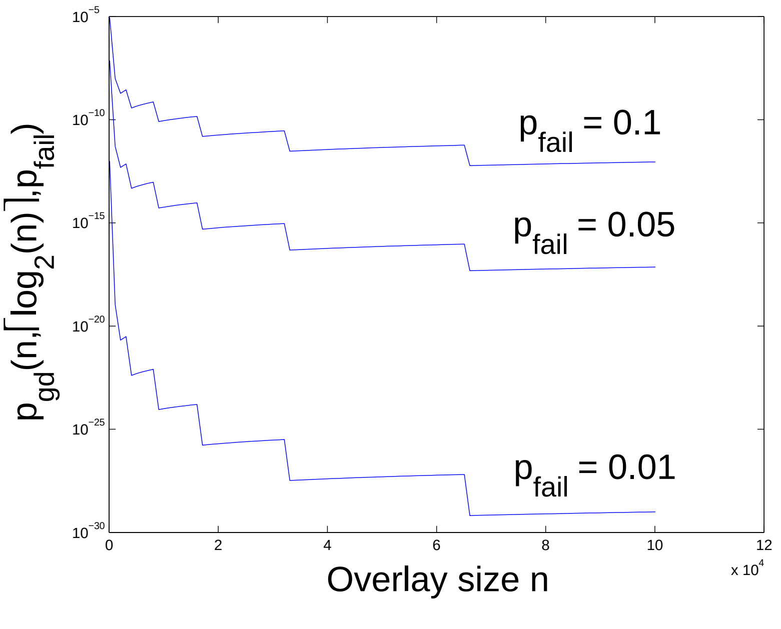

Figure 10: Probability of a global disconnection maintaining a successorlist of $\lceil \log_2(n) \rceil$.

of 3, 6 and 9. The failure probability of a peer is set to $p_{fail} = 0.01$. As we can see, the probability of a disconnection increases with the overlay size but scales very well to larger networks. Moreover, the order of magnitude of the probability of a global disconnection can be adjusted by choosing the corresponding successorlist size. Obviously, less than $\lceil \log_2(n) \rceil$ neighbors are sufficient to guarantee a stable Chord ring when we assume a realistic failure probability of $p_{fail} = 0.01$.

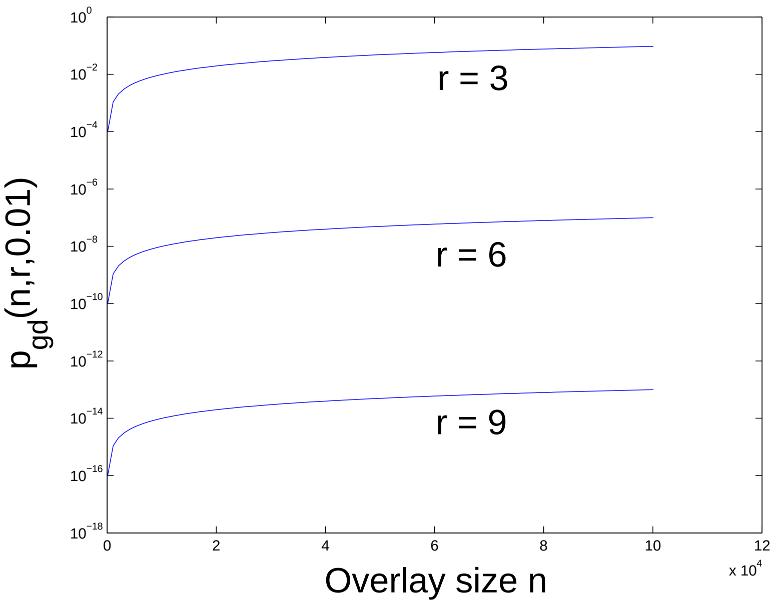

Figure 11: Impact of a fixed number of successors on the global disconnection.

To illustrate the effects of extremely high failure probabilities we plot the probability of a global disconnection against the number of successors r. In Figure 12 we show the results for a peer failure probability $p_{fail} = \frac{1}{2}$ and three different ring sizes $n = 2^5$, 2^{10} and 2^{15}. The black dotted lines represent the suggested successorlist size $\lceil \log_2(n) \rceil$. Again the suggested number of successors results in a disconnection probability of about 40 percent. To guarantee a global disconnection probability close to zero, in this example a peer has to maintain a successorlist of size $\lceil \log_2(n) \rceil + 7$ or more.

Note that so far we calculated disconnection probabilities within one single *stabilize()* period. As mentioned in Section 5 the probability of a global disconnection increases over time. That is, the longer the Chord ring exists, the greater the probability that it gets disconnected within its lifetime. Figure 13 plots the probability that a Chord ring gets disconnected sometime within i *stabilize()* calls against the number of *stabilize()* calls for different global disconnection probabilities. Assuming a *stabilize()* period of length $t_{stab} = 30$ seconds, $8 \cdot 10^4$ *stabilize()* calls roughly correspond to one month. Thus, the results demonstrate that the probability that a Chord ring gets disconnected sometime within the first month of its lifetime is by magnitudes greater than the same probability within one single *stabilize()* period.

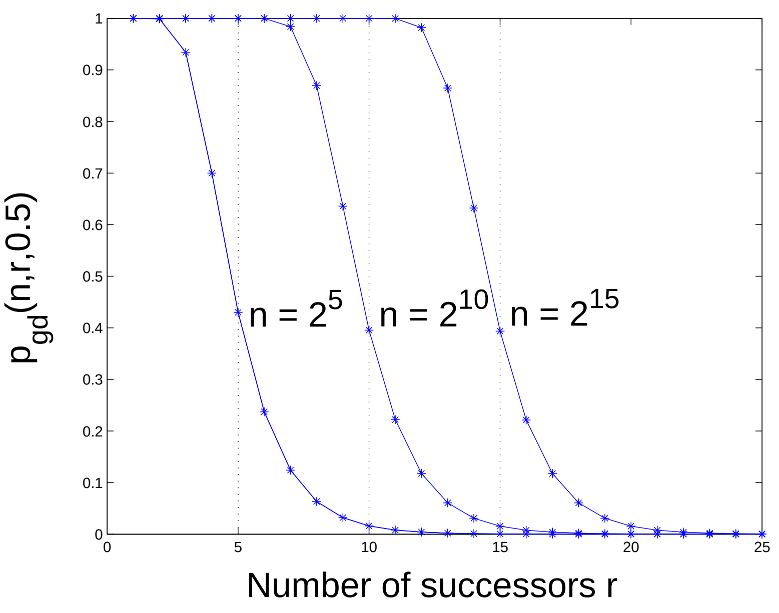

Figure 12: Probability of a global disconnection for different successorlist sizes.

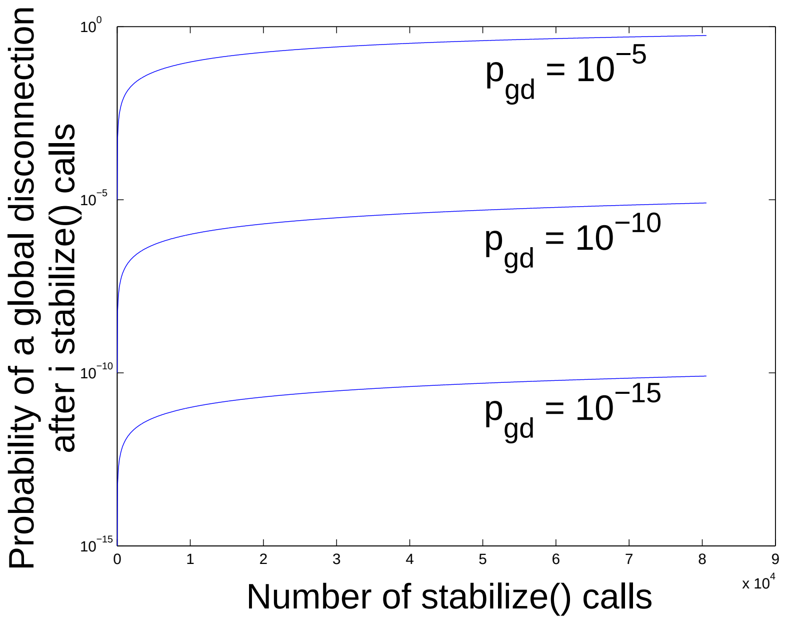

Figure 13: Probability of a global disconnection after i stabilize() calls.

7 Conclusions

In this paper we studied the efficiency and stability of a P2P system based on the Chord algorithm. We confirmed that in a Chord ring of size n each peer maintains approximately $\lceil \log_2(n) \rceil$ different fingers. The main focus of this paper is set on the problem of a disconnection of the Chord ring. It was shown, that when every peer fails with probability $\frac{1}{2}$ a successorlist of size $r = \Omega(\log_2(n))$ is not sufficient to guarantee a stable Chord ring with high probability. In fact the probability of a global disconnection is approximately 40 percent in this case.

For realistic use cases we derived an equation to calculate failure probabilities in dependency of the average online time of a peer and showed, that subject to these circumstances Chord can still guarantee a stable overlay network with high probability. The formulas presented in this paper can thus be used for system dimensioning purposes.

Acknowledgements

The authors would like to thank Prof. Tran-Gia, Gerald Kunzmann, and Steffen Rusitschka for the helps and discussions during the course of this work.

References

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

[2] Emule website, "http://www.emule-project.net."

[3] KaZaA website, "http://www.kazaa.com/us/index.htm."

[4] T. Hoßfeld, K. Leibnitz, R. Pries, K. Tutschku, P. Tran-Gia, and K. Pawlikowski, "Information Diffusion in eDonkey Filesharing Networks," in *ATNAC 2004*, (Sydney, Australia), December 2004.

[5] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the Evolution of PeertoPeer Systems," in *ACM PODC*, (Monterey, CA, USA), July 2002.

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

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

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

[9] 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.

[10] 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.

[11] A. Binzenhöfer, D. Staehle, and R. Henjes, "Estimating the size of a chord ring.".