A Survey of Chord and its Improvements *

Jacek Cichoń and Przemysław Kobylański

Institute of Mathematics and Computer Science
Wrocław University of Technology
Poland

Abstract. The Chord peer-to-peer protocol was constructed in 2001. Despite of its logical simplicity, scalability, logarithmic in the number of nodes search time it was observed by many authors that it has some weakness.
One of them is a quite nonuniform distribution of spaces controlled by nodes, which implies that some nodes are much more engaged than other. We will show that it is possible to make a very soft modification of Chord, called by us a Binary Chord, which partially eliminates this failure and significantly increases fairness of user workload.
The second weakness concerns the problem of unexpected departures (or failures) of nodes from Chord. Namely, after such departure all pieces of information stored by such a node completely disappears from the system. We discuss three quite natural solutions of this problem. The first one is based on the idea of multiplication of documents. The second one uses the notion of direct sum of Chords and the idea of the third ones relies of slight enlarging of the areas of the Chords virtual space controlled by each node. All proposed modifications can be easily implemented on the basis of the original Chord protocol.

1 Introduction

The Chord peer-to-peer protocol was built by I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek and H. Balakrishnan in 2001 (see [1, 2]). It consists of nodes mapped on a 160-bit identifier circle. Protocol addresses are determined with a common hash function \( h \) with 160-bit output, i.e. of node network address. The same function is used for mapping shared resources and it is usually the file name that is hashed. Resources are assigned to the nodes using a simple rule: the node responsible for file \( x \) is the one with least identifier succeeding \( h(x) \) or equal to \( h(x) \). The succession in this case means a protocol address not smaller than \( h(x) \) modulo \( 2^{160} \).

The network is dynamic and the assignments of resources evolve with node joins and leaves. When a new node \( P \) joins the network it is inserted somewhere inside another node’s (\( Q \)) arc and is handed respective share of \( Q \)’s resources. On the other hand, when a node leaves the network its resources should be passed to some remaining node - otherwise we talk about unexpected departure.

* The paper is partially supported by EU within the 6th Framework Programme under contract 001907 (DELIS)
Finally, let us observe that the size of this arc is a good estimation of node’s participation ratio. It is so since the number of data and requests the node will have to handle is proportional to the size of its subset and the hash function used for mapping has uniform identifier space coverage.

Each node stores a table of 160 fingers (pointers) $\text{finger}[1], \ldots, \text{finger}[160]$ to other nodes. For a given node with identifier $x$, $\text{finger}[i]$ is the first node which succeeds identifier $(x + 2^i - 1) \mod 2^{160}$.

![Fig. 1. An example of finger tables (identifiers on circle 0…15, for $M = 4$).](image)

Notice that most fingers are duplicated. Each node has, in average, $O(\log n)$ fingers. Nodes in classical Chord receive the following messages (operations on nodes):

1. **join**: joins new node as predecessor,
2. **find_successor**: asks node to find the successor of a given identifier,
3. **find_predecessor**: asks node to find the predecessor of a given identifier,
4. **init_finger_table**: initialize table of fingers,
5. **stabilize**: periodically verifies successor and tell the successor about node,
6. **fix_fingers**: refreshes table of fingers.

From formal point of view the Chord protocol can be described as the structure ($\{0, 1\}^{160}, h$), where $h : \Sigma^* \to \{0, 1\}^{160}$ is a hash function. In the rest of this paper we call the Chord protocol the classical Chord protocol. For the statistical analysis, we use a modified model of Chord. Instead of 160-bit identifier circle we consider the interval $[0, 1)$ with node identifiers being now real numbers. The nodes become now points in this interval and the arcs become disjoint subsets of $[0, 1)$. Note however, that the change to continuous identifier space does not impact the correctness of results and their application to actual Chord protocol - it is just a choice that enables and makes easier formal mathematical analysis of the subject.
Fig. 2 shows an example of the classical Chord with four nodes and three documents. \textit{Node} 3 serves documents \textit{doc} 2, \textit{doc} 3 and \textit{node} 4 serves document \textit{doc} 1.

This paper is a survey of a series of results from other papers. We show no proofs here. They are given in [3], [4] and [5].

2 Modifications of Chord

We discuss in this section four different modifications of the classical Chord protocol. All of them, from the programmer’s point of view, require only very small modifications of the original code of Chord.

2.1 Binary Chord

The Binary Chord protocol differs from Chord only in the strategy of calculating positions on the virtual space \(\{0, 1\}^M\) of new nodes. Namely, suppose that a new node with an identifier \(Id\) is to be placed into the structure; first we calculate the value \(t = h(Id)\); then we find a node \(X\) such that \(t\) is in the interval \([X, succ(X)]\); we calculate the middle point \(t^*\) of the interval \([X, succ(X)]\) and we place the new node at the position \(t^*\).

We see that the Binary Chord is a very simple modification of the classical Chord. However, it was proved in [3], that this simple modification of Chord has better statistical properties than the original one - the distribution of length of nodes is much more uniform than in classical Chord. Let us remark that the binary split method corresponds to the process of adding nodes in the CAN peer-to-peer protocol (see [6]). Therefore, we may consider the Binary Chord as a one-dimensional CAN with Chord routing mechanism. The Binary Chord was also analyzed in [7].

Let us also remark that several other solutions where proposed for balancing the load (i.e. the length of the controlled area) of nodes. For example in [8] authors propose to force short nodes to leave the system and then using them to divide the existing long intervals. Another solution, based on the concept of two-choices was proposed in [9].
2.2 Chord with Replications of Documents

The $r_k$-Chord protocol with $f$ replicas of documents is the structure

$$\left(\{0,1\}^M, h, \{h_1,\ldots,h_k\}\right),$$

where $h, h_1,\ldots,h_k : \Sigma^* \to \{0,1\}^M$ are independent hash functions. The function $h$ is used for placing nodes on the Chords’ virtual space $\{0,1\}^M$. Functions $h_1,\ldots,h_k$ are used for placing documents into $\{0,1\}^M$. If a document $doc$ is to be placed into the system, then it is placed into the positions $h_1(doc),\ldots,h_k(doc)$. This means that we store $k$ copies of each information item put into the system.

![Fig. 3. An example of $r_2$-Chord.]

Fig. 3 presents an example of $r_2$-Chord with four nodes and three documents – each in two copies. Node 1 serves the second copy of document $doc_2$, node 2 serves the second copy of document $doc_1$, node 3 serves the first copy of document $doc_2$ and all copies of document $doc_3$, node 4 serves the first copy of document $doc_1$.

We add one additional procedure to the $r_k$-Chord protocol. Namely, we assume that each node periodically, with some fixed period $\Delta$, checks whether all remaining copies of its documents are in the system. Later, we shall discuss the role of the parameter $\Delta$.

Our strategy of replication of documents in Chord may be called a blind strategy. More advanced methods were investigated by many authors (see e.g. [10]), but we analyze in this paper only the simplest, i.e. the blind one.

2.3 Direct unions of Chords

The $u_k$-Chord (direct union of $k$ copies of Chord) is the structure

$$\left(\{0,1\}^M \times \{1,\ldots,k\}, \{h_1,\ldots,h_k\}, H\right),$$

where $\{h_1,\ldots,h_k\} \cup \{H\}$ are independent hash functions. Each new node with an identifier $Id$ uses the value $h_i(Id)$ to calculate its position in the $i$-th copy $C_i = \{0,1\}^S \times \{i\}$ of the Chord. Each document $doc$ is placed at point $H(doc)$.
in each copy $C_i$ of the Chord. Notice that we use the same function $H$ to place new documents in distinct copies of the Chord. Notice also that $u_1$—Chord is the classical Chord.

Fig. 4 presents an example of $u_2$—Chord with four nodes and three documents. Document $doc_1$ is served by node 4 in the first Chord and node 3 in the second Chord, document $doc_2$ is served by node 3 in the first Chord and by node 1 in the second Chord, document $doc_3$ is served by node 3 in the first Chord and node 2 in the second Chord.

Let us consider for a while the structure $u_2$—Chord. Suppose that one node leaves a system in an unexpected way, i.e. without sending its resources to its predecessor. When the number of nodes is large, then with high probability the both areas controlled by the leaving node in two copies of Chord are disjoint, so no information item stored in the system is lost. Therefore there exists a possibility of restoring lost information items in the first copy of the Chord from the second copy. Similar remark holds for a structure $u_k$—Chord, where $k \geq 2$.

We assume that the structure $u_k$—Chord is equipped with the procedure for retrieving partially lost information. It should be used by a node when it loose the connection with its immediate successor on each of its copies of the virtual space.

2.4 Folded Chord

Let $succ(X)$ be the successor of a node in Chord. In the classical Chord protocol each node controls the subinterval $[X, succ(X))$ of the space $\{0,1\}^L$. The $k$-folded Chord, denoted as $f_k$—Chord, is the modification of the Chord protocol in which each node controls the space $[X, succ^k(X))$, where $succ^1(X) = succ(X)$ and $succ^{k+1}(X) = succ(succ^k(X))$.

Notice that $f_1$—Chord is the classical Chord. Let us consider for a while the structure $f_2$—Chord and suppose that a number of nodes in the system is large. Suppose that the small group $B$ of nodes leaves the system in an unexpected way. Then the probability of the event $(\exists b \in B)(succ(b) \in B)$ is negligible. Consider a node $y$ such that $succ(y) \in B$. Let $x$ be its predecessor and let $z = succ(b)$ and $u = succ(y)$. Note that when the node $b$ leaves the system then $z$ is a new successor of $y$. Then the node $y$ may send a copy of all information items from the
interval \([b, z)\) and may ask the node \(z\) for all information item from the interval \([z, u)\). This way we can rebuild the original structure and no information item will be lost. Of course, we can do it if the leaving group \(B\) satisfies the property \((\forall b \in B)(\text{succ}(b) \notin B)\). A similar procedure may be build for \(f_k\)-Chord for every \(k \geq 2\).

3 Distribution of Nodes

Let us recall that the coefficient of variation of the random variable \(X\) is defined by \(cv[X] = \frac{\text{std}[X]}{\text{E}[X]}\). This coefficient is a measure of concentration of the random variable with positive mean. Distributions with \(cv < 1\) are considered low-variance.

The following theorem summarizes a series of results on the distribution of length of areas controlled by nodes in considered structures.

**Theorem 1.** Let \(L\) denote the length of the areas controlled by a given node in the structure with \(n\)-nodes. Then
1. in classical Chord and in Chord with replicas of documents: \(cv[L] \approx 1\);
2. in the Binary Chord: \(cv[L] \approx \sqrt{\frac{1}{\ln 2}} - 1 \approx 0.665\);
3. in \(u_k\)-Chord and \(f_k\)-Chord: \(cv[L] \approx \frac{1}{\sqrt{k}}\).

The first result partially explains the big variance of distributions of length of areas controlled by nodes in the classical Chord observed by many authors. It can be proved that in this structure w.h.p. there exists a node with length less than \(1/n^2\) and also that there exists a node with length greater than \(\log n/n\).

The second result (see [3]) holds only in the Binary Chord only after adding to them \(n\) nodes without removing. In the dynamic scenario numerical experiments show that we have \(cv[L] \approx 0.85\). However, this behavior requires and still awaits precise theoretical explanation.

4 Failure Tolerance

4.1 Chord with Multiplied Documents

Let the number of nodes in a \(r_L\)-Chord be \(n + 1\). Let \(A\) be a subset of nodes. Suppose that we remove these nodes from the system. We are interested in the probability of the event „no information item put into the system is lost”.

**Theorem 2.** Let the number on nodes in \(r_L\)-Chord be \(n + 1\), let \(d\) denote the number of documents put into this structure. Let \(Z_{d:L:n}\) denote the number of nodes which must be removed in order to lose some information from the system. Then

\[
\text{E}[Z_{d:L:n}] = 1 + n \cdot \frac{\Gamma(d + 1)\Gamma(1 + \frac{1}{L})}{\Gamma(d + 1 + \frac{1}{L})}.
\]
We will discuss the limit properties of the variable $Z_{d,L:n}$ when $d = \delta n$, where $\delta$ is a fixed number. Notice that $\delta$ is the mean number of documents put into the system by each node. In real application, when the number of nodes is large, we may suspect that $1 \leq \delta \leq 100$.

From Theorem 2 and classical approximations of the Gamma function we deduce the following result:

**Corollary 1.** $E[Z_{\delta n:L:n}] = \frac{\Gamma(1+\frac{1}{L})}{\delta \pi}n^{1-\frac{1}{L}} + O\left(\frac{1}{n^{\frac{1}{L}}}\right)$.

Notice that in the classical case, when $L = 1$, we have $E[Z_{\delta n:1:n}] \approx 1 + \frac{1}{\delta}$ and this means that if $\delta \geq 1$ (which is a very probable value in P2P systems) then after each unexpected departure some information will disappear from the system with high probability. The situation changes when we keep in the Chord two copies of each document, namely when $L = 2$ then $E[Z_{\delta n:2:n}] \approx \Gamma\left(\frac{3}{2}\right)\sqrt{\pi} \approx 0.89\sqrt{\frac{\pi}{\delta}}$.

Observe that the expected value of the random variable $Z_{\delta n:L:n}$ depends on the number of documents. Moreover, from formula (1) we get $cv[Z_{\delta n:L:n}] \sim \sqrt{\Gamma(1+\frac{2}{L})/\Gamma(1+\frac{1}{L})}$. From this approximation we get $cv[Z_{\delta n:1:n}] \sim \sqrt{2} \approx 1.41421$, $cv[Z_{\delta n:2:n}] \sim 2/\sqrt{\pi} \approx 1.2838$, $cv[Z_{\delta n:3:n}] \sim 1.064$ and so on. Therefore we see that the random variables $Z_{\delta n:L:n}$ are very poorly concentrated near their mean value, so we need some additional results.

We say that a subset $A$ of nodes is **safe** if after removing nodes from the set $A$ from Chord no information will completely disappear from the system. We say that a subset $A$ of nodes is **unsafe** if after removing nodes from $A$ some information will completely disappear from system is.

**Theorem 3.** Let $n$ be the number of nodes in $r_2$–Chord with $\delta n$ documents. Suppose that $1 \leq \delta \leq 1000$. Let $A$ be a subset of the set of nodes.

1. If $|A| \leq \frac{1}{\ln n} \sqrt{\frac{\pi}{\delta}}$ then $A$ is safe with high probability;
2. If $|A| \geq \sqrt{3\ln n} \sqrt{\frac{\pi}{\delta}}$ then $A$ is unsafe with high probability.

By „with high probability” we understand that the probability tends to 1 when $n$ tends to infinity.

### 4.2 Direct Union of Chords

Let $A = \{n_1, \ldots, n_k\} \subseteq \{1, \ldots, n+1\}$ be a random subset of nodes from the structure $u_2$–Chord with $n+1$ nodes. We denote by $K_{A,1}$ the unions of intervals controlled by nodes from $A$ in the first circle and we denote by $K_{A,2}$ the unions of intervals controlled by nodes from $A$ in the second circle. We say that the set $A$ is **safe** if $K_{A,1} \cap K_{A,2} = \emptyset$. Notice that if the set $A$ is safe then no information disappears from the system after simultaneous unexpected departure of all nodes from $A$. Moreover, a proper mechanism may be used to recover lost information from the first circle by information stored in the second circle and conversely. We say that the set $A$ is **unsafe** if $A$ is not safe.
Theorem 4. Let $n + 1$ be the number of nodes in the structure $u_2$—Chord and let $A$ be a random set of nodes with cardinality $k$.

1. If $k \leq \sqrt{\frac{n}{\ln(n)}}$ then $A$ is safe with high probability;
2. If $k \geq \sqrt{n \ln n}$ then $A$ is unsafe with high probability.

Notice that the bounds in Theorem 4 do not depend on the number $d$ of documents put into the system.

Numerical experiments show that the expected number of nodes needed to be removed from $u_2$—Chord to obtain an unsafe configuration equals approximately $0.603 \sqrt{n}$. We do not have a mathematical proof of this fact. But we can prove the following generalization of Theorem 4: if $A$ is a random subset of nodes from the structure $u_k$—Chord such that $|A| \leq \left(\frac{n}{\ln n}\right)^{1-1/k}$ then $A$ is safe with high probability.

4.3 Folded Chord

Let $A = \{n_1, \ldots, n_k\} \subseteq \{1, \ldots, n\}$ be a random subset of nodes from the structure $f_d$—Chord with $n$ nodes. We say that the set $A$ is safe if a distance of every node from $A$ to the next point from $A$ (in the fixed orientation of Chord) is at least $d$. Notice that if the set $A$ is safe then no information disappears from the system after simultaneous unexpected departure of all nodes from $A$. Moreover, a proper mechanism may be used to recover information lost from the system. We say that the set $A$ is unsafe if $A$ is not safe.

Theorem 5. Let $n$ be the number of nodes in the structure $f_2$—Chord and let $A$ be a random set of nodes with cardinality $k$.

1. If $k \leq \sqrt{\frac{n}{\ln(n)}}$ then $A$ is safe with high probability;
2. If $k \geq \sqrt{n \ln n}$ then $A$ is unsafe with high probability.

5 The Repairing Process

We shall discuss in this section the problem of repairing (complementing) partially removed information from the system. First we fix following notation:

1. $\mu$ - the average number of departures from system in a second
2. $T$ - the average time a node spends in a system (during one session)
3. $N$ - the average number of nodes in the system
4. $u$ - the proportion of unexpected departures among all departures
5. $T_r$ - the average time needed for complementing partially lost information
6. $N_r$ - medium number of nodes waiting for complementing
7. $\delta$ - the average numbers of documents in the system per one node

Using the Little’s Law from Queueing Theory (see e.g. [11]) we get the following two relations $N = \mu \cdot T$, $N_r = u\mu T_r$, so $N_r = uN \frac{T_r}{T}$.
5.1 Safety Bounds

For each modification of the Chord protocol discussed in this paper we found a safety bound \( S_N \). For the \( r_2 \)-Chord we have \( S_N = \frac{1}{\ln N} \sqrt{\frac{N}{\delta}} \), for \( u_2 \)-Chord and \( f_2 \)-Chord we have \( S_N = \sqrt{\frac{N}{uN}} \). If we want to keep our system in a safe configuration with high probabilities, then the following inequality must be satisfied:

\[
T_r \leq \frac{TS_n}{uN}.
\]

(2)

Suppose that \( N = 10^5 \), \( T = 30 \) minutes and \( u = 0.1 \). From equations which we derived above from the Little's theorem we get \( \mu = 10^5/1800 \approx 55.55 \), so approximately 5.55 nodes unexpectedly leaves the system in one second. We shall use these parameters to calculate other parameters of considered systems.

\textbf{r} \(_2\)-\textbf{Chord} Suppose that each server stores in its local base the list with items \((doc, NoC, Info)\) for each information item put into the system. The field \( doc \) is the description of document which allows to calculate the positions of the document in the Chord structure, \( NoC \) is the number of copies (equals 0 or 1 in the Chord with two copies of documents) and \( Info \) is additional information about the document, which usually contains the URL of a station which contains the source of the document.

We assume that each node periodically, with period \( \Delta \), sends to the structure a command \( \text{Update}(doc, 1-NoC, Info) \) which is a simple modification of original AddDoc command with one additional flag, marking this is not the AddDoc command but only the gossiping of stored information.

Let us recall that the classical Chord (see [1]) check its neighbors periodically with period 2 seconds. Therefore we must assume that the first 2 seconds are wasted - during this time the node does not now its neighbor left the system. Therefore, there are only \( T_r - 2 \) seconds for information recovery. Moreover, during this time each node must send approximately \( \delta \) Update's messages. Therefore this operation must be done periodically with period \( \tau = \frac{T_r-2}{\delta} \).

The following table contains the upper bounds for the time \( T_r \) (with the fixed parameters \( N = 10^5 \), \( T = 30 \) min, \( u = 0.1 \)) and the parameter \( \tau \), for different parameter \( \delta \):

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_r )</td>
<td>4.9441</td>
<td>3.496</td>
<td>2.8547</td>
<td>2.47205</td>
<td>2.21107</td>
<td>2.01842</td>
<td>1.86896</td>
<td>1.748</td>
<td>1.64803</td>
</tr>
<tr>
<td>( \tau )</td>
<td>2.9441</td>
<td>0.748</td>
<td>0.28486</td>
<td>0.11801</td>
<td>0.04221</td>
<td>0.00307</td>
<td>&lt; 0</td>
<td>&lt; 0</td>
<td>&lt; 0</td>
</tr>
</tbody>
</table>

We see that we are unable to stay in safe states when \( \delta \geq 7 \). Moreover, if \( \delta = 4 \) then each node should send Update messages with frequency 0.12 seconds, which is smaller than a medium time of the lookup process (which is approximately 0.2 seconds). Hence, we see, that the \( r_2 \)-Chord can stay in the safe configuration only when the total number of documents stored in the system is relatively small.
**u₂−Chord and f₂−Chord** In this case the upper bound for the time $T_u$ is $T_u = \sqrt{\frac{1}{n \ln n}}$. Using the same parameters as above we get 16.7757 seconds, so subtracting 2 seconds we obtain 14.7 seconds for the repairing process.

Let us consider the structure $u₂−Chord$. During this time the node $a$ should send a request to each node from the second copy of the Chord which can contain information from the gap controlled now by $a$. The following theorem shows that there are few such nodes:

**Theorem 6.** Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_{n+1}$ be independent and uniformly distributed random variable in $[0, 1]$. Let $L_n = |\{i : Y_i < X_{1:n}\}|$. Then $E[L_n] = 1$.

Therefore, we see that the expected number of intervals from the second copy of Chord which has nonempty intersection with a given interval from the first copy is precisely 2 - hence the expected number of nodes which should be asked for its copy of information is 2. Hence the node which wants to recover partially lost information must

1. send few messages to the second copy; the time required for localization of nodes in the second copy equals approximately $\frac{1}{2} \log_2 n \cdot 0.2$ sec
2. wait for receiving necessary information; the time required for fulfilling this operation equals approximately $2 \cdot \delta \cdot \frac{d_s}{t_s}$ sec where $d_s$ is the size of the information item in the system and $t_s$ is the transmission speed.

So the total repairing time is $T_c = \frac{1}{10} \log_2 n + \frac{2 \delta \cdot d_s}{t_s}$. If we assume that each information item stored in the system is of size 0.5 kB and assuming that the transmission speed $t_s = 500$ kB/s then even if $\delta = 1000$, then we need about 5.33 seconds, so we have enough time for finishing this operation within the upper bound, i.e. within 14 seconds.

In the structure $f₂−Chord$ the procedure of localization of copies of documents is more simple: the given node $a$ should ask its actual successor for a portion of information and send some portion of information to its predecessor. The rest of analysis is the same. Obviously, in the $f₂−Chord$ structure each node $X$ should remember fingers to $\text{succ}^1(X)$ and $\text{succ}^2(X)$ to make the recovery process as short as possible. Using the same parameters as for the $u₂−Chord$ we calculate that the time needed for finishing the repairing process is about 2.2 seconds.

### 5.2 Information Lifetime

It is clear that keeping the system in a stable configuration is not sufficient to ensure that the process of recovery will finish successfully. We assume that considered systems evolved sufficiently long which allow us to use asymptotic results from the renewal theory. This assumption is usually satisfied in practice since P2P protocols evolve for hundreds of days while the average lifetime of nodes is negligible to the age of the system.

Let us recall that a random variable $X$ has the shifted Pareto distribution with parameters $\alpha$ and $\beta$ ($X \sim Pa(\alpha, \beta)$) if $\Pr[X > x] = (\frac{\beta}{\beta + x})^\alpha$. It is easy
to check that if $X \sim \text{Pa}(\alpha, \beta)$ and $\alpha > 1$ then $\mathbb{E}[X] = \frac{\beta}{\alpha - 1}$. Notice that an expected value of $X$ is finite if and only if $\alpha > 1$.

It was observed by many authors that the distribution of lifetime $L$ of nodes in P2P system (i.e. session duration) has a Pareto distribution (see e.g. [12]) with the slope parameter $\alpha \approx 2.06$. We model the lifetime as a random variable $L$ with shifted Pareto distribution. Since $\mathbb{E}[L] = \frac{\beta}{\alpha - 1}$ and $\mathbb{E}[L] = T$, we get $\beta = (\alpha - 1) \cdot T$. Therefore we will assume that

$$\Pr[L > x] = \frac{1}{(1 + \frac{x}{(\alpha - 1)T})^{\alpha - 1}}.$$ 

The residual lifetime is the conditional random variable of the lifetime conditioned by the condition that the modeled object is alive. The cumulative distribution function of the residual lifetime of a random variable $L$ with cumulative distribution $F$ is (see [12])

$$F_R(x) = \frac{1}{\mathbb{E}[L]} \int_0^x (1 - F(x)) dx.$$ 

From this equation we easily deduce the following result:

**Lemma 1.** Suppose that $X \sim \text{Pa}(\alpha, \beta)$. Let $X^*$ be the residual lifetime of the variable $X$. Then $X^* \sim \text{Pa}(\alpha - 1, \beta)$.

Therefore, if $\beta = (\alpha - 1)T$ and $x \ll T$ then the probability that a given (alive) node will stay in the system for a time longer than $x$ is $(1 + \frac{x}{(\alpha - 1)T})^{\alpha - 1} \approx 1 - \frac{x}{T}$.

The following result also follows almost directly from the definition of the Pareto distribution:

**Lemma 2.** Suppose that $L_1$ and $L_2$ are two independent random variables such that $L_1, L_2 \sim \text{Pa}(\alpha, \beta)$. Let $L = \min\{L_1, L_2\}$ and let $L^*$ be the residual lifetime of the variable $L$. Then $L^* \sim \text{Pa}(2\alpha - 1, \beta)$.

Let us look at the system from the point of view of one information item put into the system. Therefore let us fix an information item $\xi$ put into the system and let $T_\xi$ be time of survival of this item in the system.

**Theorem 7.** Suppose that the parameters of the system guarantee that the system is in the safe configuration for a long period of time. Then

$$\mathbb{E}[T_\xi] \geq \frac{T}{u} \frac{p}{1 - p},$$

where $p = 1/(1 + \frac{T_\xi}{(\alpha - 1)T})^{2\alpha - 1}$. 
5.3 Summary

Let us consider the structure with \( N = 10^5 \) nodes. Let us assume that the average life-time of a node in this system is \( T = 30 \) minutes. Suppose that the system is in the stable configuration. Then, as we have seen above, the parameter \( \mu \approx 55.55 \), so we see that approximately 55 nodes leave and other 55 nodes join the system in a second.

Suppose that the parameter \( u = 0.1 \). This means that approximately 5.55 nodes unexpectedly leave the system in one second. In the classical Chord system a given information item can survive no dangerous moment, which happens in average in time \( T/u \), so the average life-time of a given information item in the classical Chord equals about 300 minutes, i.e. 5 hours.

For both systems \( u_2 - \text{Chord} \) and \( f_2 - \text{Chord} \) we have \( T_r = T/(u\sqrt{n \ln n}) \), so, from Theorem 7 we get \( \mathbb{E}[T^e_r] \geq 0.339 \cdot T \cdot \sqrt{n \log n} \approx 10834.9 \) min, i.e. \( \mathbb{E}[T^e_r] \geq 180.5 \) hours (which is approximately 1 week). Hence, using the structure \( u_2 - \text{Chord} \) or \( f_2 - \text{Chord} \) we can increase the information lifetime in the system from 5 hours to 180 hours, i.e. 36 times.

Numerical simulations confirm our theoretical estimations. They even show that the average information lifetime in the structure \( f_2 - \text{Chord} \) (with the parameters as above) is slightly longer that 180 hours and that the standard deviation of this parameter is about 30 hours.

6 Statistics

Let us consider the following experiment: we are randomly and independently choosing \( k \) nodes from the Chord structure with \( n \) nodes and measuring the length \( x_1, \ldots, x_k \) of the spaces under their control (recall that we normalize the total length of the Chord space to 1). Let

\[
\hat{n}(x_1, \ldots, x_k) = \frac{k}{x_1 + \cdots + x_k}.
\]

Then \( \hat{n} \) is an estimator of the total number of nodes in the system. The following theorem holds:

**Theorem 8.** The function \( \hat{n} \) is a biased estimator of \( n \), namely \( \mathbb{E}[\hat{n}] = \frac{k}{k-1} n \). Moreover, if \( k \geq 20 \), then

\[
\Pr\left[ \frac{1}{2} \hat{n} \leq n \leq 2\hat{n} \right] \geq 0.99637.
\]

Hence this estimator may be used for practical estimation of number of nodes in the Chord and its variants. Therefore, if some node would like to estimate the number of nodes in the system, it should draw a random sample of 20 nodes, ask them about length of their areas (i.e. the address of its successor) and use the estimator \( \hat{n} \).

Each node has a table of fingers of size approximately \( \log_2 n \), so we may try to use these nodes as a random sample. However, this solution has a systematic
mistake. We explain this fact. Notice that a Chord with \( n \) nodes splits the Chord’s space into \( n \) disjoint intervals \( I_1, \ldots, I_n \). Let us consider the following experiment: we draw randomly a point \( \zeta \) from the interval \([0, 1)\), find \( i \) such that \( \zeta \in I_i \) and as a result we return the length \(|I_i|\) of the interval \( I_i \). Let as call the random variable constructed this way „the length of randomly chosen interval” and denote it by \( L \). It can be proved (see \([5]\)) that \( \mathbb{E}[L] = \frac{2}{n+1} \). Notice that the mean length of each interval \( I_1, \ldots, I_n \) is precisely \( \frac{1}{n} \), so the expected length of randomly chosen interval is twice larger that the expected length of any interval. This phenomenon is easy to be explained: random points prefer longer intervals.

We cannot use the length of arcs controlled by the finger because we would make the systematic mistake discussed above. We use instead the predecessors and successors of fingers. This is a procedure NoN for establishing the number of nodes of a node \( A \):

```plaintext
function NoN(A : node):integer
begin
    S:= len(pred(A))
    for all \( x \in \text{Fingers} \) do
        S:= S + len(succ(x))) + len(pred(x)))
    NoN:= (2 \(|\text{Fingers}| + 1)/S - 1;
end;
```

(where \( \text{len}(X) \) is the normalized length of the area controlled by the node). Let us observe that all necessary information for this procedure can be gathered during the standard routine of checking fingers. The two modifications are the following:

1. each node must store information about sizes of arcs controlled by its successor and predecessor
2. during the check-up procedure each node should send to requesting node not only its identifier but also the sizes of successor and predecessor

This modification does not decrease the speed of protocol nor change any of its important properties.

Numerical experiments confirm the good properties of this estimator. Let us notice that we have also checked the behavior of this estimator in the Binary Chord and we observed that in this situation it behaves even better than in the Chord. Finally, let us observe that this estimator is more precise in the direct unions of a Chords, since each node has \( a \)-times more fingers that in the classical Chord, so may it use more local observations.

A small modification of the above procedure may be used for estimation of the numbers of documents in the system. Namely, assume that we are not only measuring the numbers \( x_1, \ldots x_k \) but that we get also numbers \( d_1, \ldots, d_k \) of documents stored by each node. Then

\[
\hat{D}(x, d) = \frac{d_1 + \cdots + d_k}{x_1 + \cdots + x_k}
\]
is an estimator of the total number of documents in the system. Numerical simulations confirm good statistical properties of this estimator. Its precision is better than the precision of the estimator $\hat{n}$.

References