An Evaluation of Sampling Algorithms for Estimating the Size of a Chord Network

Gabriele Falace
DEIS, University of Calabria
Rende (CS), Italy
Email: falace@si.deis.unical.it

Paolo Trunfio
DEIS, University of Calabria
Rende (CS), Italy
Email: trunfio@deis.unical.it

***Abstract*—Due to the decentralized nature of structured P2P systems, there is no a direct way for a single node of getting aggregate statistics about the whole network, such as its current size. In this paper we focus on the problem of estimating the size of one of the most popular structured P2P networks, Chord, using a sampling-based approach. With this approach, a node calculates an estimate of the network size after having queried a small number of its successors about some of their properties. We formally define three sampling-based algorithms that exploit well-known structural properties of a Chord network to derive an estimate of its size. An experimental evaluation was carried out through simulations to evaluate the accuracy of the three algorithms in different network scenarios. The evaluation allowed us to identify, among the three algorithms, a *Ring Density Estimation* (*RDE*) technique that was able to estimate the size of all the Chord networks considered with an average error of 2% or less, using only a few tens of sample nodes. Moreover, the simulation results showed that the RDE accuracy is not affected by dynamic network conditions, even in the presence of high nodes failure rates.**

***Keywords*—Structured peer-to-peer networks; distributed hash tables; Chord; sampling algorithms; network size estimation; simulation analysis**

I. INTRODUCTION

Structured P2P systems, like Chord [1], Pastry [2], and Tapestry [3], have attracted wide research interests due to their efficiency and scalability in supporting resource discovery over large-scale distributed networks. Most structured P2P systems use a Distributed Hash Table (DHT) to assign to each node the responsibility for a specific part of the resources in the network. When a node wants to discover a resource identified by a given key, the DHT locates the node responsible for that key in $O(\log n)$ hops using only $O(\log n)$ neighbors per node, where n is the network size.

Due to the decentralized nature of structured P2P systems, there is no a direct way for a single node of getting aggregate statistics about the whole network, such as the total number of nodes currently present in the system. Even if a node may not be interested in knowing the network size for ordinary lookup operations, there are several practical cases in which such information is of great importance. For example, an estimate of the network size can be used for choosing the most appropriate length of the successor-list [4], calculating the popularity of a resource to minimize network traffic [5][6], or tuning the rates at which the network topology is maintained [7].

In this paper we focus on the problem of estimating the size of one of the most popular structured P2P networks, Chord, using a sampling approach in which a node calculates an estimate of the network size after having queried a small number of its successors about some of their properties. This approach is very efficient since the number of messages generated equals the number of nodes in the sample, and avoids unnecessary traffic because the estimations are performed only on-demand. We formally define three sampling-based algorithms that exploit well-known structural properties of a Chord network to derive an estimate of its size.

An experimental evaluation was carried out through simulations to evaluate the accuracy of the three algorithms in different network scenarios. The evaluation allowed us to identify, among the three algorithms, a *Ring Density Estimation* (*RDE*) technique that was able to estimate the size of all the Chord networks considered with an average error of 2% or less, using only a few tens of sample nodes. Moreover, the simulation results showed that the RDE accuracy is not affected by dynamic network conditions, even in the presence of high nodes failure rates.

The remainder of the paper is organized as follows. Section II presents a background on Chord and discusses related work. Section III describes the three sampling algorithms and their performance in static network conditions. Section IV presents an experimental comparison of the algorithms in the presence of nodes failures. Finally, Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

This section provides a short background on the Chord system and discusses some related work aimed at evaluating the size of P2P networks, including Chord networks.

A. Chord Networks Basics

Chord [1] uses a consistent hash function to assign to each node an m-bit identifier, which represents its position in a circular identifier ring ranging from 0 and $2^m - 1$. Each node, N, maintains a *finger table* with m entries, numbered from 1

to m. The i-th entry in the finger table of node N, denoted as *finger*[i], contains two fields: *start*, which is equal to $N+2^{i-1}$ *mod* 2^m, and *node*, which is the first node whose identifier is equal to or follows *start*. Node *finger*[i].*node* is called the i-th *finger* of node N.

N also knows its *successor*, which is the next node on the identifier ring (i.e., *finger*[1].*node*) and its *predecessor*, which is the previous node on the circle. In addition, to cope with possible failures, N maintains a *successor-list* of its r nearest successors on the Chord ring[1]. If N notices that its successor has failed, it replaces the failed successor with the first live node in its successor-list.

Figure 1 shows an example of Chord network with m = 4 bits and n = 5 nodes. For each of the five nodes in the network, with identifiers 0, 3, 6, 10 and 13, the corresponding finger table is shown. Since m = 4, each finger table includes four entries. For instance, let us consider the finger table of node 3. The *start* fields point at exponentially increasing distance from 3 (4=3+1, 5=3+2, 7=3+4, and 11=3+8), while the *node* fields contain the first node found in clockwise direction along the ring starting from the corresponding *start* (6, 6, 10, and 13).

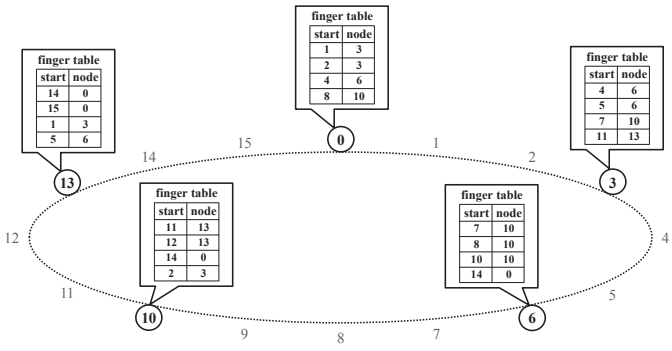


Figure 1. An example of Chord network with m = 4 bits and n = 5 nodes.

B. Estimating the Size of a P2P Network

There are a several works in literature that studied how to estimate the size of a fully-decentralized P2P network. Some of them [8][9][10] focused on finding solutions for unstructured P2P networks, like Gnutella [11]. Basically, these works use gossip-based techniques in which each node periodically exchanges some information with its neighbors, in order to derive a global estimate of the network size. The most important problem of gossip-based techniques on unstructured networks is the large amount of messages needed to obtain an accurate estimate, as highlighted in the experimental study proposed by Le Merrer et al. [12].

In the context of structured P2P networks, two significant studies related to our work are those proposed by Shafaat et al. [7], and by Binzenhöfer et al. [4]. Both of them estimate the network size starting from an evaluation of the density of nodes on the identifier circle. This approach is also at the basis of the LEA and RDE algorithms described in this paper. However, the sampling approach differs from those used in [7] and [4] as outlined in the following.

In [7], each node propagates its estimate using a gossip-based algorithm inspired from the approach in [10], but adapted to the structured overlay. After a certain number of rounds, the estimated size converges to the real size with the desired level of accuracy. Differently from the sampling approach, this algorithm requires that all nodes in the network are involved in the procedure, which may be too costly if just a single node is interested in estimating the network size. On the other hand, this algorithm fits well and is very effective in those scenarios where all nodes need an estimate of the network size.

Using the technique proposed in [4], a node estimates the density of nodes on the identifier circle starting from some statistics that can be deducted from its finger table. Given such density estimate, the overall network size estimate can be easily calculated. Differently from the sampling approach, this technique has the advantage of not producing messages at all, but the accuracy of the estimates can be low for some classes of applications, since the estimated sizes generally lie in between $0.5n$ and $2n$, where n is the actual ring size.

III. SAMPLING-BASED SIZE ESTIMATION

The three algorithms described in the following share the same sampling approach. The node wishing to estimate the network size, hereafter referred to as the *requesting node*, submits a request that is forwarded in clockwise direction along the ring, from each node to its successor, until k nodes are reached (i.e., k is the sample size). Each node receiving the request, referred to as a *responding node*, performs some algorithm-specific computations and then forwards the request to its successor until the k-th node is reached. The k-th node replies directly to the requesting node with some summary information. This information is then used by the requesting node to derive an estimate of the network size, on the basis of the specific algorithm employed.

For each of the three algorithms defined in this paper, we will discuss: i) the property of Chord networks exploited by the algorithm; ii) the algorithm pseudo-code; iii) an experimental evaluation of the accuracy achieved by the algorithm.

The experimental evaluation was performed using a custom network simulator written in Java. For each algorithm, simulations have been conducted for network sizes (n) of 1000, 2000, 4000, 8000 and 16000 nodes, and sample sizes (k) ranging from 10 to 100 nodes, with steps of 10 nodes. According to [1], we used m = 160 bits for the node identifiers, which are uniformly distributed across the identifier space. For each combination $\langle n,k \rangle$, 100 independent simulation runs were executed; at each run, the algorithm was started by a different requesting node.

[1]According to Stoica et al. [1], it is safe to choose r = O($\log_2 n$).

The simulation results discussed in the remainder of this section are obtained in static network conditions. Static conditions means that, once created, the network does not change its composition, i.e., there are no nodes that leave or join the Chord ring over the time. Under static conditions the finger tables of all nodes are always in a consistent state. An evaluation of the algorithms in the presence of node failures that produce finger table inconsistencies will be presented in Section IV.

A. Distinct Fingers Averaging

A well known property of Chord networks is that, if the identifier space is not fully populated (i.e., the number of nodes, n, is lower than 2^m), the finger table contains redundant fingers. In a network of n nodes, the number u of unique (i.e., distinct) fingers of a generic node is likely to be $\log_2 n$ [1]. Based on this basic property, we define a simple *Distinct Fingers Averaging* (*DFA*) size estimation algorithm that first calculates the average number of distinct fingers $\bar{u}$ from a sample of k nodes, and then estimates the number of nodes in the network, $\hat{n}$, as $2^{\bar{u}}$.

The pseudo-code of the DFA algorithm is shown in Figure 2. First, the requesting node N sends a DFA_REQUEST message to itself. The request includes three fields: i) the identifier of the requesting node (N); ii) the number of nodes to which the request must still be forwarded to (initially equal to k); iii) the sum u_{sum} of all the values of u of the nodes that have already processed the request (initially equal to 0).

Distinct Fingers Averaging (DFA) Algorithm
Input: sample size $k > 0$
Output: estimated network size $\hat{n}$

▷ Requesting node N:
send DFA_REQUEST[N,k,0] to N;
receive DFA_RESPONSE[u_{sum}] from S;
$\bar{u} := u_{sum} \div k$;
$\hat{n} := 2^{\bar{u}}$;

▷ Responding node R:
receive DFA_REQUEST[N,k,u_{sum}] from S;
$u_{sum} := u_{sum}$ + *distinct_fingers*();
if $k > 1$ **then**
  send DFA_REQUEST[N,k-1,u_{sum}] to *successor*(R);
else
  send DFA_RESPONSE[u_{sum}] to N;
end if

▷ Function *distinct_fingers*():
$u := 0$;
$F := -1$;
for $i := 1$ to m **do**
  if *finger*[i].*node* $\neq F$ **then**
    $u := u+1$;
    $F :=$ *finger*[i].*node*;
  end if
end for
return u;

Figure 2. Distinct Fingers Averaging (DFA) Algorithm.

Each responding node R, after having received a DFA_REQUEST, calculates the new value of u_{sum} by adding the number of distinct fingers in its finger table, calculated through the *distinct_fingers* function, to the old value of u_{sum} carried by the request. Then, if $k > 1$, the request is forwarded to the next node along the ring, with the new value of u_{sum} and with k decreased by 1. Otherwise, a DFA_RESPONSE, containing the new value of u_{sum}, is sent directly to the requesting node N.

As soon as N receives the DFA_RESPONSE, it calculates the average number of unique fingers $\bar{u}$ and then estimates the network size $\hat{n}$ as discussed earlier. For example, let us consider again the small Chord network in Figure 1. Assume that node 3 wants to estimate the network size on a sample of three nodes (k = 3) using the DFA algorithm. Nodes 3, 6 and 10 compute their values of u, which are respectively equal to 3, 2 and 3, and so u_{sum} = 8. Therefore, node 3 calculates $\bar{u}$ = 2.67 and finally $\hat{n}$ = 6.35.

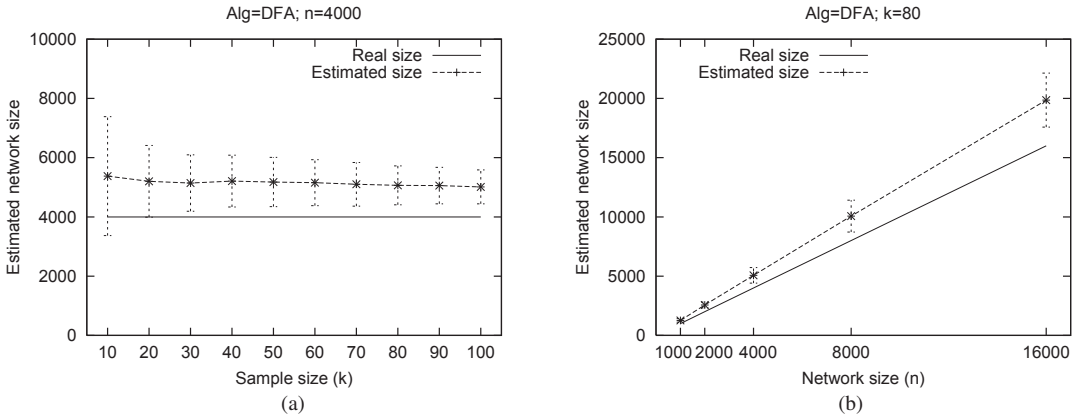
As mentioned before, an evaluation was carried out using a custom network simulator for different combinations of n and k values. Figure 3 presents an extract of the simulation results for the DFA algorithm. In particular, Figure 3a shows the estimated network size when n = 4000 and k passes from 10 to 100, while Figure 3b shows the estimated network size when k is fixed to 80 and n ranges from 1000 to 16000. The values are obtained by averaging the results of 100 simulation runs. Error bars represent the standard deviations from the means.

As shown in Figure 3a, when n = 4000, the average value of $\hat{n}$ passes from 5378 using k = 10, to 5014 using k = 100. Therefore, the average relative error passes from 34% with k = 10, to 25% with k = 100. Even if the average relative error decreases only a little as the value of k increases, we note that the standard deviations decrease significantly by increasing the sample size, thus motivating the use of larger samples (e.g., k = 80-100).

Finally, Figure 3b shows that, fixed the sample size (k = 80), on average DFA overestimates the network size of 24-28%, independently from the value of n.

B. Local Estimates Averaging

The second algorithm is based on the approach proposed by Binzenhöfer et al. [4], already mentioned in Section II-B. Based on this approach, a node can locally estimate the network size by observing the difference between the *start* and *node* values of each entry in its finger table. For instance, node 3 in the Chord network of Figure 1 can derive the following information from its finger table: from *finger*[1], there is only one node (i.e., node 6) in the interval [4,6]; from *finger*[3], there is one node (10) in [7,10]; from *finger*[4], there is one node (13) in [11,13]. Therefore, it is possible to derive three estimates of the ring density: ρ_1 = 1 ÷ (6-4+1) = 0.33, ρ_2 = 1 ÷ (10-7+1) = 0.25, ρ_3 = 1 ÷ (13-11+1) = 0.33. Given the average of these values, $\bar{\rho}$ = 0.31, a local



(a) (b)

Figure 3. Network size estimated by the DFA algorithm: (a) n = 4000, k = from 10 to 100; (b) k = 80, n = from 1000 to 16000.

estimate of the network size is $\hat{n}_{local} = \bar{\rho} \cdot 2^m = 4.89$. The *Local Estimates Averaging* (*LEA*) algorithm aims at improving the accuracy that can be achieved by a single local estimation, by calculating the average of the local estimates computed on a sample of k nodes.

Figure 4 shows the pseudo-code of the LEA algorithm. Similarly to the DFA algorithm, a LEA_REQUEST is forwarded k times along the ring starting from the requesting node N. The request includes three fields: i) the identifier of the requesting node; ii) the forwarding counter (initially set to k); iii) the sum $\hat{n}_{sum}$ of all the local size estimates made by the nodes that have already processed the request (initially 0).

Each responding node calculates the new value of $\hat{n}_{sum}$ by adding the network size estimate calculated on its finger table through the *local_estimation* function, to the old value of $\hat{n}_{sum}$ carried by the request message received. Then, either the request is forwarded to the next node along the ring (if k is still greater than 1), or a LEA_RESPONSE, containing the overall $\hat{n}_{sum}$, is sent to the requesting node N.

When N receives the response, it can estimate the network size, $\hat{n}$, as $\hat{n}_{sum} \div k$. As an example, let us assume that node 3 of Figure 1 wants to estimate the network size on a sample of three nodes using the LEA algorithm. Nodes 3, 6 and 10 compute their $\hat{n}_{local}$, which are respectively equal to 4.89, 4.67 and 6.22, and so $\hat{n}_{sum}$ = 15.78. Finally, node 3 calculates $\hat{n}$ = 5.26.

Figure 5 presents an extract of the simulation results for the LEA algorithm: Figure 5a shows the estimated network size with n = 4000 and k ranging from 10 to 100; Figure 5b shows the estimated network size with k = 80 and n ranging from 1000 to 16000.

The simulation results show that the average estimates obtained with LEA are significantly better than those obtained using DFA. For example, with n = 4000 (see Figure 5a), the average relative error was under 14% even for the lower values of k. However, the standard deviations with LEA resulted to be much higher, which means that is very likely for a node to obtain a highly inaccurate network size estimate using this algorithm, in particular with $k < 70$.

Figure 5b shows that, fixed the sample size, the average accuracy of the estimate is basically independent from the network size, with a relative error ranging between 10 and 17%. We note, however, that the standard deviations significantly increase with the network size.

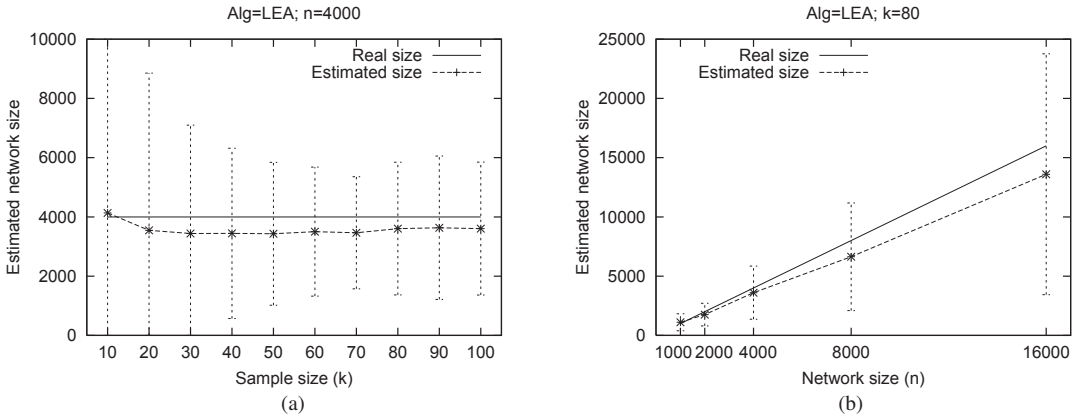
Local Estimates Averaging (LEA) Algorithm
Input: sample size $k > 0$
Output: estimated network size $\hat{n}$

▷ Requesting node N:
send LEA_REQUEST[N,k,0] to N;
receive LEA_RESPONSE[$\hat{n}_{sum}$] from S;
$\hat{n} := \hat{n}_{sum} \div k$;

▷ Responding node R:
receive LEA_REQUEST[N,k,$\hat{n}_{sum}$] from S;
$\hat{n}_{sum} := \hat{n}_{sum}$ + *local_estimation*();
if $k > 1$ **then**
  send LEA_REQUEST[N,k-1,$\hat{n}_{sum}$] to *successor*(R);
else
  send LEA_RESPONSE[$\hat{n}_{sum}$] to N;
end if

▷ Function *local_estimation*():
$\rho_{sum} := 0$;
$u := 0$;
F := -1;
for $i := 1$ to m **do**
  if *finger*[i].*node* $\neq F$ **then**
    $u := u$+1;
    F := *finger*[i].*node*;
    S := *finger*[i].*start*;
    $l := [(F - S)$ *mod* $2^m] + 1$;
    $\rho := 1 \div l$;
    $\rho_{sum} := \rho_{sum} + \rho$;
  end if
end for
$\bar{\rho} := \rho_{sum} \div u$;
$\hat{n}_{local} := \bar{\rho} \cdot 2^m$;
return $\hat{n}_{local}$;

Figure 4. Local Estimates Averaging (LEA) Algorithm.



(a) (b)

Figure 5. Network size estimated by the LEA algorithm: (a) n = 4000, k = from 10 to 100; (b) k = 80, n = from 1000 to 16000.

C. Ring Density Estimation

The third algorithm, called *Ring Density Estimation* (*RDE*), shares with LEA the idea of calculating the ring density to derive an estimate of the network size. In LEA, each node in the sample estimates a ring density value for each distinct entry in its finger table. In RDE, the ring density ρ is calculated only once, at the end of the sampling procedure, by dividing the number of nodes in the sample, k, by the length l of the Chord ring arc that begins at the requesting node and ends at the k-th node in the sample. The value of l is therefore calculated as [(S - N) *mod* 2^m] + 1, where N is the identifier of the requesting node and S is the identifier of the k-th node in the sample.

The pseudo-code of the RDE algorithm is shown in Figure 6. In this case, the minimum value of k is 2, because at least two nodes are required to get an arc length greater than one. The RDE_REQUEST includes only two fields: i) the identifier of the requesting node (N); ii) the forwarding counter (initially k).

Ring Density Estimation (RDE) Algorithm
Input: sample size $k > 1$
Output: estimated network size $\hat{n}$

▷ Requesting node N:
  send RDE_REQUEST[N,k-1] to *successor*(N); // $k > 1$
  receive RDE_RESPONSE[S] from S;
  l := [(S - N) *mod* 2^m] + 1;
  $\rho := k \div l$;
  $\hat{n} := \rho \cdot 2^m$;

▷ Responding node R:
  receive RDE_REQUEST[N,k] from S;
  if $k > 1$ **then**
    send RDE_REQUEST[N,k-1] to *successor*(R);
  else
    send RDE_RESPONSE[R] to N;
  end if

Figure 6. Ring Density Estimation (RDE) Algorithm.

Differently from the DFA and LEA algorithms, the responding nodes do not perform local computations. In fact, when a node receives a request, it either forwards the request to the next node along the ring (if $k > 1$), or sends a RDE_RESPONSE with its identifier to the requesting node.

As soon as N receives the response, which includes the identifier of the last responding node S, it calculates: i) the length l of the arc from N to S; ii) the ring density ρ; iii) the estimated number of nodes, $\hat{n}$, as $\rho \cdot 2^m$. As an example, let us assume that node 3 of Figure 1 wants to estimate $\hat{n}$ using the RDE algorithm with k = 3. The request reaches node 10, which responds to node 3. Node 3 calculates l = 10-3+1 = 8, ρ = 3 ÷ 8 = 0.375, and finally $\hat{n}$ = 0.375 · 16 = 6.

Figure 7 presents an extract of the simulation results for the RDE algorithm. As for DFA and LEA, Figure 7a shows the estimated network size with n = 4000 and k ranging from 10 to 100, while Figure 7b shows the estimated network size with k = 80 and n ranging from 1000 to 16000.

The simulation results show that the average estimates obtained with the RDE algorithm are much more accurate than those produced by DFA and LEA. For example, with n = 4000 (see Figure 7a), the average value of $\hat{n}$ passes from 4464 using k = 10, to 3999 using k = 100. Thus, the average relative error passes from 12% with k = 10, to almost 0% with k = 100. Also the standard deviations decrease significantly by increasing the sample size.

Finally, Figure 7b shows that, with k = 80, the estimates obtained with RDE are always very accurate and improve as the network size increases. For example, the relative error is about 2% with n = 1000, 1% with n = 4000, and less than 0.5% with n = 16000.

IV. EVALUATION IN THE PRESENCE OF FAILURES

The experimental results discussed throughout the previous section showed that, in static network conditions, the RDE algorithm outperforms both DFA and LEA. In particular, RDE was able to estimate the size of all the Chord networks considered with an average error of 2% or less, using only a few tens of sample nodes.

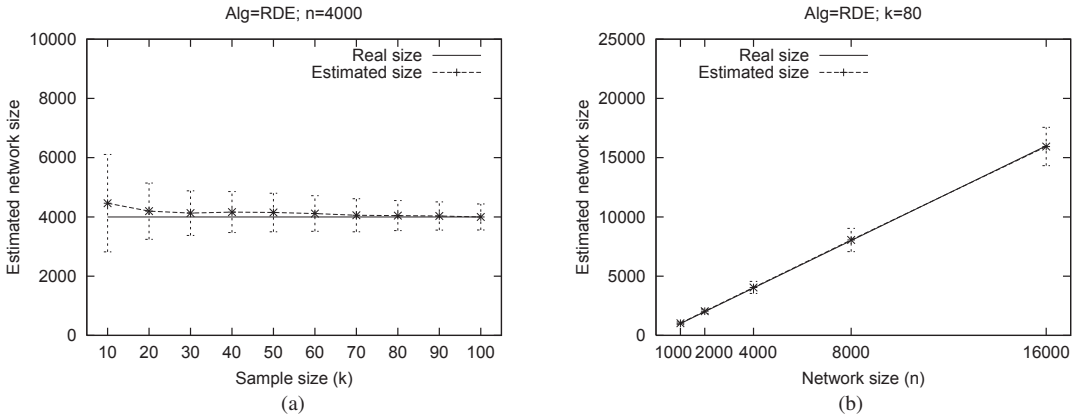


Figure 7. Network size estimated by the RDE algorithm: (a) n = 4000, k = from 10 to 100; (b) k = 80, n = from 1000 to 16000.

The goal of this section is evaluating the performance of the three algorithms in dynamic network conditions. In a dynamic network, nodes can leave and join the network at any time. As the network evolves, finger table inconsistencies eventually arise due to node failures (or graceless disconnections) or in the face of concurrent joins. In Chord networks, inconsistencies are fixed periodically through the execution of a stabilization procedure that refreshes finger table entries and corrects successor/predecessor identifiers if they are found to be uncorrect.

To evaluate the performance of the three algorithms in the presence of finger table inconsistencies for a given value of n, we proceed as follows. First, using the simulator, we initialize a random Chord network of size n. After initialization, the network is in stable conditions and all the finger tables are in consistent state. Then, we remove a given percentage p of nodes from the network so as to simulate their failures, without executing the periodical stabilization procedure. In this way, a certain level of finger table inconsistency arises, proportionally to the value of p.

Given n and p, we execute the three algorithms with the desired sample size and compare the network size estimates obtained, as well as the resulting relative error. Figure 8 presents an extract of the comparison between the three algorithms in dynamic conditions, with a percentage p of failed nodes ranging from 0 to 30%. As for the results discussed in the previous section, the results presented here are obtained by averaging the results of 100 simulation runs.

Figures 8a and 8b compare size estimates and corresponding relative errors obtained by DFA, LEA and RDE using k = 80, in a network having an initial size of 4000 nodes, with increasing percentages of failed nodes. Figure 8a shows that DFA and LEA produce the same estimates (about 5000 and 3600 nodes, respectively) independently from the fact that an increasing number of nodes leaves the network. This is because DFA and LEA perform their estimations based on finger tables information. Since finger tables were last updated when the network was in stable conditions (i.e., with p = 0%), such algorithms continue to produce always the same estimate. As a consequence, the relative errors of DFA and LEA tend to increase as the percentage of failed nodes increase, as shown in Figure 8b. Note that the relative error of LEA has a minimum around p = 10%, when the actual network size is 3600, since this value equals the network size initially estimated by LEA.

While DFA and LEA suffer from finger table inconsistencies, the RDE algorithm confirms its accuracy even in presence of node failures. In fact, differently from the other two algorithms, RDE does not rely on finger tables to derive its estimates, and therefore does not suffer from the inaccurate information they may contain. We recall that each node needs to know only its actual successor to run the RDE algorithm. Using the Chord successor-list (see Section II-A), it is guaranteed with high probability that a node knows its actual successor even in the presence of high failures rates. As shown by Figures 8a and 8b, the values of $\hat{n}$ estimated by RDE follow the actual values of n, with an average relative error of 1% or less.

Similar results were obtained by increasing the network size from 4000 to 16000 nodes, as shown by Figures 8c and 8d. In fact, also in this case DFA and LEA produce the same estimates (about 19900 and 13600 nodes, respectively) independently from the percentage of nodes failed, while the estimates generated by RDE follow the real network sizes with a constant relative error of about 0.5%. On the basis of the simulation results presented above, we conclude that the RDE accuracy is not influenced by dynamic network conditions, even in the presence of high nodes failure rates.

V. CONCLUSIONS

In this paper we focused on the problem of estimating the size of a Chord network using a sampling-based approach. Sampling algorithms are efficient since the number of messages generated is equal to the number of nodes in the sample, and avoid unnecessary traffic because the estimations are performed only on-demand. We formally defined three algorithms based on well-known structural properties of Chord

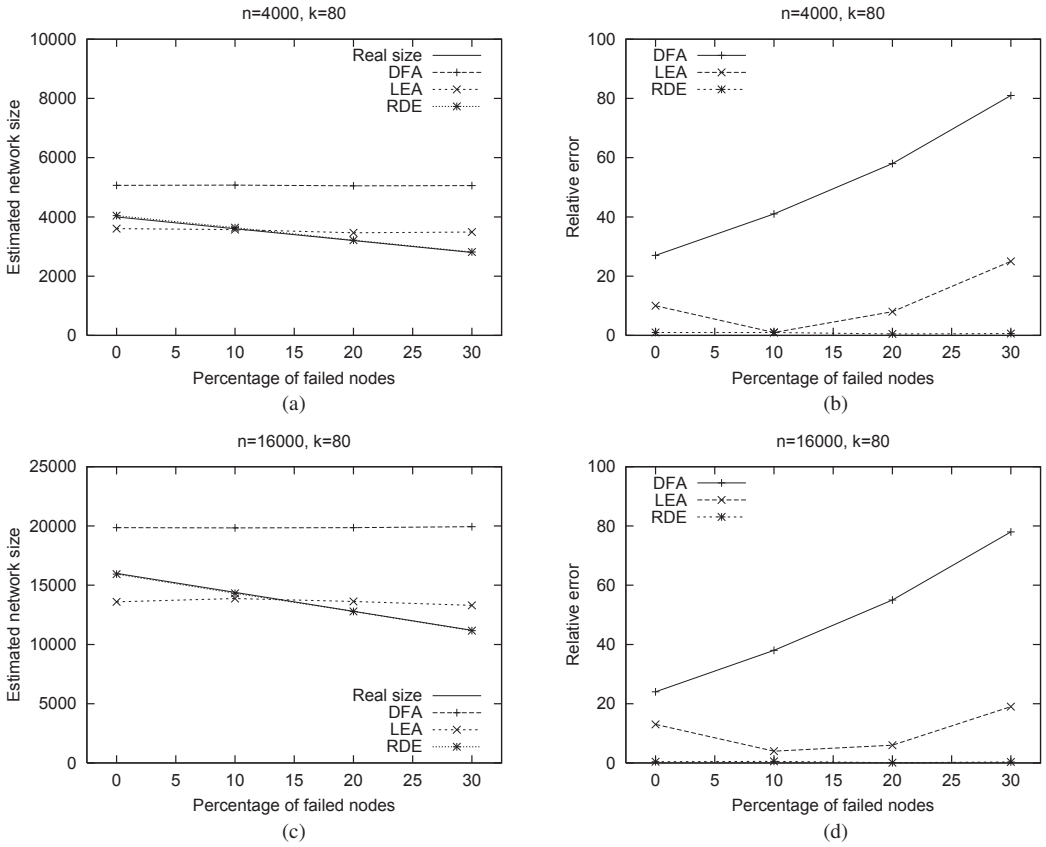


Figure 8. Comparison of size estimates and corresponding relative errors obtained by DFA, LEA and RDE using k = 80, with a percentage of failed nodes ranging from 0 to 30%): (a) estimated size with n=4000; (b) relative error with n=4000; (c) estimated size with n=16000; (d) relative error with n=16000. Error bars are omitted for the sake of readability.

networks: Distinct Fingers Averaging (DFA), Local Estimates Averaging (LEA), and Ring Density Estimation (RDE).

A simulation analysis was carried out to evaluate and compare the accuracy of the three algorithms in different network scenarios. The simulation results demonstrated that DFA and LEA tend respectively to overestimate and underestimate the network size, and are sensitive to finger tables inconsistencies arising from dynamic network conditions. On the contrary, the RDE algorithm was able to estimate the size of all the Chord networks considered with an average error of 2% or less, using only a few tens of sample nodes.

Finally, the simulation results demonstrated that the accuracy of the RDE algorithm is not affected by dynamic network conditions, even in the presence of high nodes failure rates. Based on these results, we conclude that RDE technique can be practically used as a simple but accurate strategy for estimating the size of a Chord ring in a wide range of network scenarios.

REFERENCES

[1] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. of ACM SIGCOMM 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," Proc. of Middleware 2001.

[3] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," IEEE Journal on Selected Areas in Communications, vol. 22, no. 1, 2004.

[4] A. Binzenhöfer, D. Staehle, and R. Henjes, "Estimating the size of a Chord ring," Tech. Rep. 348, Institute of Computer Science, University of Würzburg, 2005.

[5] D. Talia and P. Trunfio, "Enabling dynamic querying over distributed hash tables," Journal of Parallel and Distributed Computing, vol. 70, no. 12, 2010.

[6] C. Comito, D. Talia, and P. Trunfio, "Selectivity-based XML query processing in structured peer-to-peer networks," Proc. of the 14th Int. Database Engineering and Applications Symposium (IDEAS), 2010.

[7] T. M. Shafaat, A. Ghodsi, and S. Haridi. "A practical approach to network size estimation for structured overlays," Proc. of the 3rd Int. Workshop on Self-Organizing Systems (IWSOS), 2008.

[8] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," Tech. Rep., Stanford University, 2003.

[9] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," Proc. of the 44th Annual Symposium on Foundation of Computer Science, 2003.

[10] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Trans. on Computer Systems, vol. 23, no. 3, 2005.

[11] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," IEEE Internet Computing, vol. 6, no. 1, 2002.

[12] E. Le Merrer, A-M. Kermarrec, and L. Massoulie, "Peer to peer size estimation in large and dynamic networks: a comparative study," Proc. of the 15th IEEE Int. Symposium on High Performance Distributed Computing, 2006.