

# On Obtaining Global Information in a Peer-to-Peer Fully Distributed Environment\*

Márk Jelasity<sup>1</sup> and Mike Preuß<sup>2</sup>

<sup>1</sup> Dept. of AI, Free Univ. of Amsterdam, [jelasity@cs.vu.nl](mailto:jelasity@cs.vu.nl)  
and RGAI, Szeged Univ., Hungary

<sup>2</sup> Dept. of Computer Science, Univ. of Dortmund, [mike.preuss@uni-dortmund.de](mailto:mike.preuss@uni-dortmund.de)

**Abstract.** Networking solutions which do not depend on central services and where the components possess only partial information are robust and scalable but obtaining global information like e.g. the size of the network raises serious problems, especially in the case of very large systems. We consider a specific type of fully distributed peer-to-peer (P2P) environment with many interesting existing and potential applications. We suggest solutions for estimating network size and detecting partitioning, and we give estimations for the time complexity of global search in this environment. Our methods rely only on locally available (but continuously refreshed) partial information. Theoretical analysis and simulation results are also presented.

## 1 Introduction

Peer-to-peer (P2P) systems are becoming more and more popular. The Internet offers an enormous amount of resources which cannot be fully exploited using traditional approaches. Systems that span many different institutions, companies and individuals can be much more effective for certain purposes such as information distribution (e.g. [4, 6, 1]) or large scale computations (e.g. [3, 11, 12]).

Systems exist that go to extremes in the sense of not using central services at all to achieve maximal scalability and minimal vulnerability to possible damages in components. Such an approach was chosen in e.g. [5] for broadcasting. We will focus on another architecture of this kind which we developed as part of the DREAM project [10] (described in more detail in Section 2). In a nutshell, the aim of the DREAM project is to create a complete environment for developing and running distributed evolutionary computation experiments on the Internet in a robust and scalable fashion. It can be thought of as a virtual machine or *distributed resource machine* (DRM) made up of computers anywhere on the Internet. The actual set of machines can (and generally will) constantly change and can grow immensely without any special intervention. Apart from security considerations, anyone having access to the Internet can connect to the DRM and can either run his/her own experiments or simply donate the spare capacity of his or her machine.

\* This work is funded as part of the European Commission Information Society Technologies Programme (Future and Emerging Technologies). The authors have sole responsibility for this work, it does not represent the opinion of the European Community, and the European Community is not responsible for any use that may be made of the data appearing herein.

Although these fully distributed environments can grow to literally astronomical sizes [9] while automatically maintaining their own integrity they have a major drawback: exercising global control and obtaining global information becomes harder and harder as the size increases. Broadcasting or any global search becomes infeasible after a certain point.

This paper discusses methods for obtaining global information based only on *locally available partial information* in the nodes of our environment. These methods scale much better than e.g. broadcasting because their resource requirements are independent of the size of the network. The time complexity of global search based on (continuously refreshed) local information will be addressed in Section 3. In Section 4 a method for estimating the network size is presented. In Section 5 we suggest a way of detecting partitioning.

## 2 The Model

Focusing on the topic of this paper we discuss only a simplified version of our environment, in particular we ignore timestamp handling, and the mechanism of application execution. More information can be found in [7, 8].

The DRM is a network of DRM nodes. Let  $S$  denote that set of all nodes in the DRM, and let  $n = |S|$ . In the DRM every node is completely equivalent. Nodes must be able to know enough about the rest of the network in order to be able to remain connected to it. Spreading information over and about the network is based on epidemic protocols [2].

Every node  $s \in S$  maintains an *incomplete* database containing descriptors of a set  $D(s)$  of nodes ( $|D(s)| = c$ ), where normally  $n \gg c$ . We call these nodes *the neighbours* of the node. The database is refreshed using a push-pull anti-entropy algorithm. Every node  $s$  chooses a node  $s'$  from  $D(s)$  in every time-step. Then any differences between  $s$  and  $s'$  are resolved so that after the communication  $s$  and  $s'$  will both have the descriptors of the nodes from  $D(s) \cup D(s')$ . Besides this,  $s$  will receive a descriptor of  $s'$  and  $s'$  will also receive a descriptor of  $s$ . As mentioned before, the size of the database is limited in  $c$ . This limitation is implemented by removing randomly selected elements.

To connect a new node to the DRM one needs only one living address. The database of the new node is initialized with the entry containing the living address only, and the rest is taken care of by the epidemic algorithm described above. Removal of a node does not need any administration at all.

Fortunately, theoretical and empirical results show that limiting the size of the database does not affect the power of the epidemic algorithm, information spreads quickly and the connectivity (thus information flow) is not in danger [5, 7, 8]. For example a database size of 100 is enough to support a DRM of size  $10^{33}$ .

## 3 Global Search

We would like to find nodes in the network which fulfill some criteria. Being able to do so is important in many situations. We might want to find a node that has lots of space or CPU capacity available, or nodes situated in a given geographical area, etc.

Our main purpose is to allow very large networks, in the order of  $n = 10^5$  or more. In such networks any broadcasting approach is infeasible because every node must be able to do global search and for large networks too many broadcasts could be generated resulting in huge traffic. Collecting and storing information about the entire network is not the best solution either because we cannot assume large storage capacity in every node, and on the other hand the network changes constantly: nodes and running applications come and go.

In the following we will examine the limits and possibilities of using only the local database in a node to search the network. The idea is that we listen to the updates and when the appropriate node appears there, we return it. This might seem hopeless but theory and practice show that it is not necessarily the case. Note that this kind of search has practically no costs since we are using the database refreshment mechanism that is applied anyway. The only cost that increases with  $n$  is the waiting time.

### 3.1 Time complexity

Let  $s^* \in S$  be the node we are looking for from node  $s$  ( $s \neq s^*$ ). Let  $D(s) = \emptyset$  at the start of the search. Let the set  $D_i$  denote the nodes in the database  $s$  is updated with during the  $i$ th database-exchange session according to the epidemic algorithm. Note that the elapsed time is not necessarily the same between the updates. In this section we assume that  $D_1, D_2, \dots$  are unbiased independent random samples of  $S$ . We will revisit this assumption in Section 3.2.

Let the random variable  $\xi$  denote the index of the first update in which  $s^*$  can be found. In other words  $s^* \in D_\xi$  and  $\forall i < \xi : s^* \notin D_i$ . From our assumption about the even distribution it follows that  $P(s^* \in D_i) = c/n$  for  $i = 1, 2, \dots$ . From the assumption of independence it follows that  $P(\xi = i) = (1 - c/n)^{i-1}(c/n)^i$  thus  $\xi$  has a geometric distribution with the parameter  $c/n$ . This means that the expected value is  $\mu_\xi = n/c$  and the variance is  $\sigma_\xi^2 = n(n - c)/c^2$ . Note that the optimal case in this framework is when we have  $D_1 \cup \dots \cup D_{\lceil n/c \rceil} = S$  when the information flow speed (the learning speed of  $s$ ) is maximal. The expected value that belongs to this distribution is  $\geq n/2c$ . Compared to this the waiting time in the realistic situation is in the same order of magnitude which is rather surprising.

### 3.2 Simulation Results

The above theoretical distribution of waiting time depends on assumptions on independence and unbiased sampling of the series  $D_1, D_2, \dots$ . It was examined if these assumptions indeed hold using software which can simulate the behavior of the model described in Section 2 for up to 10000 nodes. For each parameter setting the simulation was run until all the nodes showed up in the database of a fixed node  $s$  at least once. In every database exchange session the number of nodes seen for the first time by  $s$  was recorded. This output defines the empirical distribution of waiting time for a fixed node. Statistics were calculated from this empirical distribution.

In the first experiment the effect of network size was examined (see Table 1). The ratio  $n/c = 100$  was fixed so that the effect of network size could be illustrated. The observed values are quite stable and are close to those predicted by the theory. The

1000		2500		5000		7500		10000	
$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
132.904	135.805	125.81	140.028	124.082	133.488	126.171	143.982	127.996	156.549
132.297	134.629	130.777	140.05	123.071	132.324	124.008	138.747	122.507	140.236
134.128	134.109	132.436	142.684	131.564	153.64	130.305	154.89	123.255	141.495

**Table 1.** Results for  $n/c = 100$  to illustrate the effect of different network sizes shown in the columns. The predicted values of waiting time are  $\mu = 100$  and  $\sigma = 99.499$  in each case. The lines belong to 3 independent experiments.

100		75		50		25		10	
$\mu$	$\mu/\mu^*$	$\mu$	$\mu/\mu^*$	$\mu$	$\mu/\mu^*$	$\mu$	$\mu/\mu^*$	$\mu$	$\mu/\mu^*$
10.311	1.031	14.297	1.072	23.073	1.154	50.689	1.267	142.409	1.424
10.523	1.052	14.971	1.123	23.745	1.187	55.832	1.396	134.804	1.348
10.679	1.068	14.592	1.094	23.889	1.194	53.762	1.344	136.347	1.364

**Table 2.** Results for  $n = 1000$  to illustrate the effect of different database sizes shown in the columns.  $\mu^*$  is the value predicted by the theory. The lines belong to 3 independent experiments.

expected value shows even a slight decreasing tendency. This suggests that our theory can be applied for predicting waiting time in large networks which was our original goal.

In the second experiment the effect of different  $c$  values was examined (see Table 2). The network size  $n = 1000$  was fixed so that the effect of database size could be illustrated. It is clear from the results that the accuracy of the theory increases with the database size.

## 4 Estimating Network Size

Another promising possibility of exploiting the dynamics of the epidemic protocol is network size estimation. Since there is no central service we have no idea about the actual size of the network. It can be estimated however from the characteristics of information flow through the database of a server  $s$ . Intuitively, if there is much new information in the database of a peer then we expect to have a large network.

### 4.1 Theoretically Optimal Estimation

Let us examine a database exchange between  $s$  and  $s'$  (with databases  $D$  and  $D'$  respectively) during the normal functioning of our epidemic protocol. Let  $d = |D' \setminus D|$ , or in words the number of new elements in  $D'$ . If we assume that  $D$  and  $D'$  are independent unbiased samples from  $S$  then  $d$  has a binomial distribution  $B((n - |D|)/n, |D'|)$  with the expected value

$$E(d) = |D'| (n - |D|) / n \quad (1)$$

(In this section we do not assume that  $|D| = |D'| = c$ , the results hold for the general case too.)

Of course we do not know the distribution of  $d$  because its parameters refer to the network size. However we can collect a sample for a fixed  $|D|$  and  $|D'|$  and we can approximate the expected value of  $d$  with the sample average  $\bar{d}$ . Using (1) and this approximation we can approximate  $n$  with the expression

$$n \approx \tilde{n} = \frac{|D'| |D|}{|D'| - \bar{d}} \quad (2)$$

Since  $d$  has a binomial distribution, this approximation is optimal in the following Bayesian sense:

**Proposition 1.** *If  $\xi$  is a random variable from the binomial distribution  $B(p, n)$  and  $\{x_1, \dots, x_k\}$  is an independently drawn sample of  $\xi$  then*

$$\arg \max_p P(x_1, \dots, x_k | B(p, n)) = \frac{x_1 + \dots + x_k}{kn}$$

*Proof.* After substituting the probability values, using the independence assumption and ignoring the binomial coefficients we get

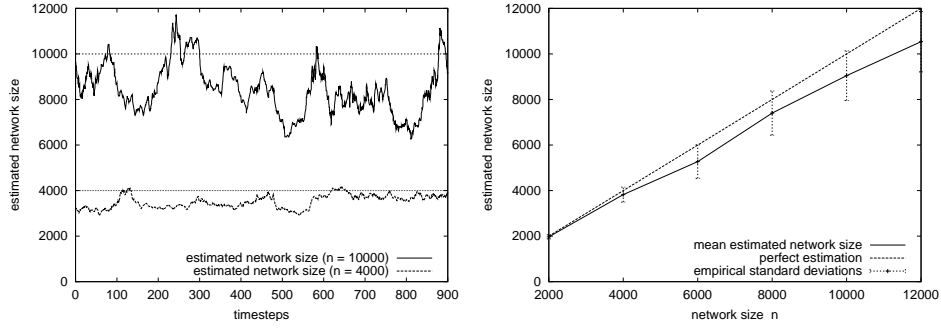
$$\max_p P(x_1, \dots, x_k | B(p, n)) = \max_p p^{x_1 + \dots + x_k} (1 - p)^{kn - (x_1 + \dots + x_k)}$$

Elementary calculus shows that the maximum of this polinom of  $p$  is at  $p = (x_1 + \dots + x_k) / (kn)$  which proves the proposition.  $\square$

## 4.2 Obtaining Two Independent Samples

The correctness of the approximation depends on our assumptions over the independence of the two samples  $D$  and  $D'$ . Unfortunately (having in mind the mechanisms described in section 2) we can see that after the information exchange between  $s$  and  $s'$  each have an entry pointing to the other with a probability of at least  $1/2$ . This— together with the fact that the databases of  $s$  and  $s'$  are very similar at this time point— may result in the violation of the independence assumption if  $s$  and  $s'$  communicate again within a short time.

In Section 3 this effect did not prove to be strong enough to prevent us from applying theoretical predictions. According to our preliminary experiments network size prediction is more sensitive to dependence of the sample sets. The most promising approach proved to be using the *long term memory* invented in [8] as one of the two needed samples. The long term memory of  $s$  stores a random selection of peers  $s$  communicated with in the past. It changes slowly because its update cycle is longer than that of the local database. This reduces the influence of the last communications dramatically, giving us a very stable basis for measuring  $\bar{d}$  and thus estimating the network size  $n$ . Note that with using the long term memory as  $D$  and the local database as  $D'$  (instead of the database of the peer) every direct influence of a remote node can be eliminated.



**Fig. 1.** left: Two single simulation runs estimating stable networks of 4000 and 10000 nodes. right: Network size estimation compared to the real network size.

### 4.3 Simulation Results

In all our simulations  $|D'| = 100$  and  $|D| = 75$  ( $D$  being the long term memory instead of the database). Furthermore,  $\bar{d} = \frac{d_1 + \dots + d_k}{k}$  where  $d_1, \dots, d_k$  are the last  $k$  observed values of  $d$ , in other words  $\bar{d}$  is the moving average of the observations during the last  $k$  database exchanges. The value  $k = 150$  was used in each case.

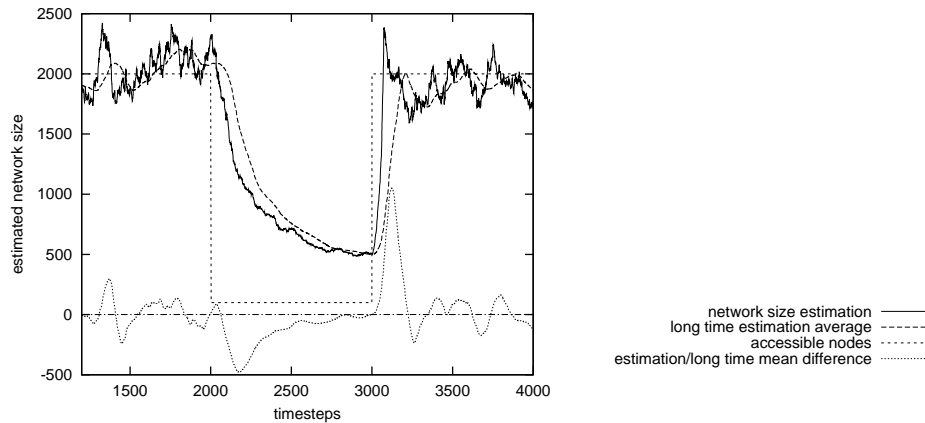
Figure 1 (left) shows two typical simulation runs. The estimated size is slightly lower than the real size in both cases indicating that we could not achieve complete independence, but it provides an acceptable approximation. Another observation is that for the larger network size we have higher fluctuations. The reason is that in that case equation (2) is more sensitive to changes in  $\bar{d}$  provided that all other constants are the same. To prevent instability of the estimation for large networks,  $k$  can be increased. However, this slows down the adaption of  $\bar{d}$  to new conditions. The choice of  $k$  is a tradeoff between estimation quality and flexibility w.r.t. changing network properties.

Figure 1 (right) shows the aggregated results of 3 runs for each network size. The empirical standard deviation increases steadily for larger networks as expected whereas estimation quality is decreasing. Although it would be hard to extrapolate to even larger networks from this data we can make the conservative conclusion that the proposed method gives a relatively close lower bound on network size. Furthermore, the size of 12000 is already large enough to be considered practically relevant.

We would like to recite that these results correspond to a fixed setting of the crucial parameters. It is easy to predict from the theoretical results that using a larger long term memory ( $D$ ) would result in more stable predictions and increasing the size of  $D'$  has beneficial effects as well, as Section 3 suggests.

## 5 Detecting Partitioning

During the operation of a DRM the underlying physical network may get partitioned. For a user it may be valuable to detect this partitioning because this could mean a serious degradation of his or her available computational resources. The approach we are presenting in this paper offers a potential solution for detecting such sudden changes.



**Fig. 2.** Size estimation during partitioning. The lower curve is generated from the difference between the estimation and its long time average and may be used to detect partitioning.

We simulated partitioning by disabling all communication between a small group and the rest of the network. Figure 2 depicts a run with a total of 2000 nodes and an isolated cluster of  $5\% = 100$  nodes which includes our observation point. The isolation begins at time-step 2000 and ends at time-step 3000. The estimation is computed like in Section 4.3. As expected, the estimation changes suddenly as well.

Due to lack of space we can only indicate a possible approach here to use the obviously available information for actually predicting change. Figure 2 shows a smoothed curve which is the average of the last 300 estimation values. Due to the slower change of the smoothed version the difference of the smoothed and plain estimation together with the actual estimation value can provide a reliable prediction. The lower curve shows this (smoothed) difference.

## 6 Conclusions

In this paper techniques were presented that are able to provide global information in a distributed networking environment where no central services are available. The techniques are based on the dynamics of the epidemic protocol which is run in an environment where each node knows only a tiny bit about the whole network but where this knowledge is continuously updated by an epidemic protocol.

Possibilities of performing global search in this environment were analyzed both theoretically and empirically. It was shown that the underlying epidemic algorithm pumps the complete system-state through every local node very quickly. It is notable that the design goals of our epidemic protocol did not include this requirement, it was an unexpected but useful side-effect. Depending on the waiting time available this makes global search feasible in many cases.

It was also suggested that the dynamics of information flow through a node can be exploited in many ways. One of these is estimating network size, another is predicting

partitioning. It was demonstrated that network size can be predicted with acceptable accuracy even for large networks. The possibility of developing a partitioning detector was also argued for.

The possibilities were not fully exploited. Our goal was to give theoretical and empirical evidence which suggest that it is worth doing research in the direction of possible exploitations of information sources which are naturally present in certain types of distributed environments. We believe that techniques like the ones suggested in this work can many times offer a cheap yet effective alternative to implementing expensive additional protocols and services or introducing additional restrictions in the design.

## Acknowledgments

The authors would like to thank the other members of the DREAM project for fruitful discussions, the early pioneers [10] as well as the rest of the DREAM staff, Maribel García Arenas, Emin Aydin, Pierre Collet and Daniele Denaro.

## References

1. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 46–66, 2000.
2. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database management. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, Aug. 1987. ACM.
3. distributed.net. <http://distributed.net/>.
4. P. Druschel and A. Rowstron. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, Banff, Canada, 2001. ACM.
5. P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2001)*, Göteborg, Sweden, 2001.
6. Gnutella. <http://gnutella.wego.com/>.
7. M. Jelasity, M. Preuß, and B. Paechter. A scalable and robust framework for distributed applications. Accepted for publication in the Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002).
8. M. Jelasity, M. Preuß, M. van Steen, and B. Paechter. Maintaining connectivity in a scalable and robust distributed environment. Accepted for publication in the Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), May 21–24, 2002, Berlin, Germany.
9. A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. Submitted for publication, available as <http://research.microsoft.com/camdis/PUBLIS/kermarrec.ps>.
10. B. Paechter, T. Bäck, M. Schoenauer, M. Sebag, A. E. Eiben, J. J. Merelo, and T. C. Fogarty. A distributed resource evolutionary algorithm machine (DREAM). In *Proceedings of the Congress on Evolutionary Computation 2000 (CEC2000)*, pages 951–958. IEEE, IEEE Press, 2000.
11. SETI@home. <http://setiathome.ssl.berkeley.edu/>.
12. United Devices<sup>tm</sup>. <http://ud.com/>.