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

Márk Jelasity[1] and Mike Preuß[2]

[1] Dept. of AI, Free Univ. of Amsterdam, jelasity@cs.vu.nl
and RGAI, Univ. of Szeged, Hungary
[2] Dept. of Computer Science, Univ. of Dortmund, mike.preuss@uni-dortmund.de

Abstract. Networking solutions which do not depend on central services and where the components posses 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.

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. [3, 4]) or large scale computations (e.g. [2, 9]).

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. [7] for broadcasting. We will focus on another architecture of this kind which we developed as part of the DREAM project [8] (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 [7] 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.

For the sake of completeness, let us mention that simulations with network sizes of up to 10000 nodes were performed to support our theoretical considerations here. Unfortunately, due to serious space limitations, we were forced to remove our simulation results to be able to keep most of our theoreical results which might be more appropriate to be published in a research note.

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 [5, 6].

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 [1].

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 [7, 5, 6]. 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.

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 ith 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 $D_1, D_2, \ldots$ are unbiased independent random samples of S.

Let the random variable ξ 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 $i = 1, 2, \ldots$ From the assumption of independence it follows that $P(\xi = i) = (1 - c/n)^{i-1}(c/n)^i$ thus ξ 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 $D_1 \cup \ldots \cup D_{\lceil n/c \rceil} = S$ 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.

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.

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

$$E(d) = |D'|(n - |D|)/n \tag{1}$$

(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 $\overline{d}$. Using (1) and this approximation we can approximate n with the expression

$$n \approx \tilde{n} = \frac{|D'||D|}{|D'| - \overline{d}} \tag{2}$$

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

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

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

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

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

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

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. When the estimated network size decreases suddenly, it probably means that the node that percieves this change is part of a subnetwork that has just been separated from the original larger network.

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. 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.

The possibilities were not fully exploited. Our goal was to give theoretical evidence which suggests 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 [8] as well as the rest of the DREAM staff, Maribel García Arenas, Emin Aydin, Pierre Collet and Daniele Denaro.

References

1. 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.
2. distributed.net. http://distributed.net/.
3. 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.
4. Gnutella. http://gnutella.wego.com/.
5. 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).
6. 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.
7. A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probablistic reliable dissemination in large-scale systems. Submitted for publication, available as http://research.microsoft.com/ camdis/PUBLIS/kermarrec.ps.
8. B. Paechter, T. Bäck, M. Schoenauer, M. Sebag, A. E. Eiben, J. J. Merelo, and T. C. Fogarty. A distributed resoucre evolutionary algorithm machine (DREAM). In *Proceedings of the Congress on Evolutionary Computation 2000 (CEC2000)*, pages 951–958. IEEE, IEEE Press, 2000.
9. SETI@home. http://setiathome.ssl.berkeley.edu/.