Dynamic Lookup Networks*

Dahlia Malkhi

School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel 91904
dalia@cs.huji.ac.il

1 Introduction

Research on scalable lookup services offers to tackle a quintessential problem of distributed systems, the discovery and locating of resources in networked systems. Specifically, it aims to design a distributed lookup service that allows clients to contact local lookup representative(s) from anywhere and find any resource in a reasonable amount of work. Attention to scale and high dynamicity brings challenges outside the scope of classical routing networks research.

In the past two years, researchers have looked at schemes that support dynamic lookup services as a *distributed hash table*, managing the distribution of data among a changing set of servers, and allowing clients to contact any participating server in the network to find any stored resource by name. There are two main challenges that large-scale DHTs must overcome: the first is distributing data in such a way that it remains nearly balanced across the set of active servers, and such that only small changes are necessary when servers join and leave. The second is to maintain a network of connection information between servers so that a lookup for data can be "routed" between servers toward its intended target, and so that servers may join and leave without requiring hash information to be propagated through the entire network.

The first issue is addressed in nearly all recent work using Consistent Hashing [5] and its variations. In this approach, in order to distribute key-value pairs across a set of participating servers, we treat both the servers and the resources as points in some metric where there are simple (and efficient) data structures and algorithms for resolving closest point queries and where the number of neighbors is small. That is, we associate with both resource and server names a value taken from the metric and then ensure that a resource resides on a server whose associated value is *close* (under the metric) to the resource name. For example, Chord [15] maps resources and servers to the unit ring [0..1). For a given set of active servers, a server manages all resources that have names between its counter-clockwise neighbor's name and its own name. Routing is then performed based on a comparison between the values associated with the current server name and the target resource name, using some routing algorithm for the chosen metric. This framework captures several recent proposals, including Consistent

* This work was supported in part by the Israeli Ministry of Science grant #1230-3-01.

Hashing [5] and Chord [15], CAN [11], Plaxton et al. work [10] and OceanStore [7], and Viceroy [9].

The second, remaining problem is to construct an efficient connection graph that routes lookup queries from any starting point to the server closest to the target. Much is known on this domain [3, 14], including networks of low degree, small dilation and even congestion, such as Shuffle-Exchange, Butterfly, and Cube-Connected-Cycle. Deviating from these works, dynamic lookup networks additionally require the following properties: (i) the network size is unknown and dynamically changes as processes join and leave it, and (ii) there should be no central point (or group of processes) that manage the network and hence suffer unproportional amount of load during joins and leaves. Thus, the problem we face is to dynamically construct a routing network that accommodates rapid changes in size and has no central management. In particular, when a server joins the network, it should be possible for it to become a participant in the routing network using a reasonable amount of communication with existing members.

We have already postulated that the network topology provides vicinity links and (efficient) local searching. It is left to determine how to provide long-range links so as to make all searches efficient. Borrowing from the ideas of Kleinberg [6] and Barriere et al. [1], we can augment the underlying metric with *long range contacts* chosen appropriately so that a localized routing strategy produces short paths. The inspiration from [6] is that these few long range contacts should *not* be uniformly distributed, but have a bias toward closer points. [1].

The manner in which long-range link are chosen precisely affects the resulting routing construction. The network construction of [6] emulates a Cube-Connected-Cycle (with poly-logarithmic diameter). The Viceroy network of [9] offers a dynamic emulation of the Butterfly network, yielding a network whose degree is constant yet its dilation is logarithmic. Other lookup networks also emulate known graphs: Chord [15] emulates a Hypercube (logarithmic degree, logarithmic dilation), as do Plaxton et al in [10], Tapestry [16] and Pastry [12]. The CAN algorithm of [11] emulates a multi-dimensional torus, whose degree is a constant d determined as a system parameter, and whose dilation is $O(dn^{(1/d)})$.

Resilience is another important facet of a dynamic and scalable resource location. We envision a two-layer architecture to provide fault tolerance. At the bottom tear, each process is replicated in a small cluster, using any known clustering technology. Each cluster then serves as a *super node* in the second tier, the routing network. A super node may gracefully join or leave the network, but not fail. This is achieved by letting a cluster size vary between a low and a high water mark. When the number of participants in a cluster drops below the low threshold, it seeks to merge with another cluster, thus virtually leaving the network. Likewise, when a cluster size grows above the high mark, it splits and virtually create a new joining node. In order for a full cluster to fail, multiple

[1] Note that Kleinberg's paper is descriptive, trying to explain how a social network allowing small hop routing may develop. We are in the process of developing a constructive algorithm providing a design to a network based directly on this approach.

simultaneous failures must occur, an event that we can rule out with proper tuning. Further issues of atomicity and fault tolerance are exposed in [8].

The motivation for this work is apparent. In order to share resources and access services over large, dynamic networks, users require means for locating them in an efficient manner. The fundamental service that is required is a lookup directory that maps names to values. The Domain Name System (DNS) is a known example of such a lookup service, but one that is static and furthermore, is tailored to the DNS hierarchical namespace and suffers from increased load and congestion at nodes close to the root of the DNS tree. In contrast, we aim to build an on-the-fly dynamic lookup service, and pave the way to deployment in peer-to-peer networks, where the participating servers are particularly dynamic and no central control or information is maintained. Good solutions may find application in many peer-to-peer systems like music-sharing applications (e.g., Gnutella [4]), file sharing and anonymous publishing systems (e.g., Freenet [2]) and distributed engine that take advantage of CPU sharing (e.g., Seti@home [13]).

2 Research goals

The research and development of dynamic, scalable lookup networks is intensively going on in academic and industrial groups. At the Hebrew University, we have recently launched a project called Viceroy to build a dynamic, scalable lookup service. Bringing algorithmic ideas into practice will require further attention to various components, and surface additional challenges. We list some of our immediate goals here.

Given the decentralization and scalability of the problem at hand, randomization is a natural design choice. However, previous works derive their performance properties from the goodness of random choices made initially in the construction. We envision that in a long lived system, the quality of such initial random choices will necessarily degrade as the system evolves, many processes depart and so on. Therefore, one of our first goals is to re-balance the network dynamically against an adversary that adaptively impacts the randomness in the system. We need to further devise load shifting mechanisms that will not change the basic structure of the lookup network, and will be completely localized so as to preserve the decentralized nature of our approach. Moreover, we look for solutions that maintain these good measures over long range, and are not based on good randomization of initial choices made by processes when joining. Thus, we consider a powerful adversarial situation, where departures and joining are controlled by a malicious, partially adaptive adversary. This problem is further intensified by the need to deal with multiple concurrent changes.

Second, in each one of the works cited above, one particular network is emulated. In contrast, we wish to form a generic strategy for dynamic routing network emulation, that many graph topologies fit.

The next topic that concerns deployment of lookup methods in practice is adaptation and caching in order to cope with variable access load. In reality,

there may be situations where the load becomes unbalanced due to transient hot spots. For example, in a music-sharing application, many users might access Madonna's recent album shortly after release. To accommodate such situations, the basic efficient design should be accompanied by a good caching strategy. A particular challenge is the lack of inherent hierarchy that would naturally support cache-flushing.

A final issue that deserves our immediate attention is locality of placement. In practice, it might be preferable to take into consideration proximity and network connectivity in the decision of how to place servers and interconnect them. Locality is considered in [12], leading to encouraging results. These ideas and others need to be explored further for general lookup networks.

Acknowledgments

The author collaborates with several people on various aspects of the dynamic name service problem, including Ittay Abraham, Baruch Awerbuch, Yossi Azar, Yair Bartal, Moni Naor, Elan Pavlov and David Ratajczak. Members of the Viceroy team include Danny Bickson, Oren Dobzinski, Keren Horowitz, Anat Talmy, and Roy Werber.

References

1. L. Barriere, P. Fraigniaud, E. Kranakis and D. Krizanc. "Efficient routing in networks with long range contacts". In the *15th International Symposium on Distributed Computing (DISC '01)*, October 2001.
2. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. "Freenet: A distributed anonymous information storage and retrieval system". In *Proceedings the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.
3. T. H. Cormen, C. E. Leiserson and R. L. Rivest. "Introduction to algorithms". MIT Press, 1990.
4. `http://gnutella.wego.com`.
5. D. Karger, E. Lehman, F. T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web". In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 654–663, May 1997.
6. J. Kleinberg. "The small world phenomenon: An algorithmic perspective". Cornell Computer Science Technical Report 99-1776, October 1999. (Published, shorter version available as "Navigation in a Small World", *Nature* 406(2000).)
7. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. "OceanStore: An architecture for global-scale persistent storage", Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.
8. N. Lynch, D. Malkhi and D. Ratajczak. "Atomic data access in Content Addressable Networks: A position paper". Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02).
9. D. Malkhi, M. Naor and D. Ratajczak. "Viceroy: A scalable and dynamic emulation of the Butterfly". In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC'02)*, July 2002. To appear.
10. C. Plaxton, R. Rajaram, and A. Richa. "Accessing nearby copies of replicated objects in a distributed environment". In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures* (SPAA 97), pages 311–320, June 1997.
11. S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. "A scalable content-addressable network". In *Proceedings of the ACM SIGCOMM 2001 Technical Conference*. August 2001.
12. A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329-350, November 2001.
13. `http://www.setiathome.ssl.berkeley.edu`.
14. H. J. Siegel. "Interconnection networks for SIMD machines". *Computer* 12(6):57–65, 1979.
15. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for Internet applications". In *Proceedings of the SIGCOMM 2001*, August 2001.
16. B. Y. Zhao, J. D. Kubiatowicz and A. D. Joseph. "Tapestry: An infrastructure for fault-tolerant wide-area location and routing". U. C. Berkeley Technical Report UCB/CSD-01-1141, April, 2001.