ClosestNode.com: An Open Access, Scalable, Shared Geocast Service for Distributed Systems

Bernard Wong   Emin Gün Sirer
Dept. of Computer Science, Cornell University, Ithaca, NY 14853
{bwong, egs}@cs.cornell.edu

ABSTRACT

ClosestNode.com is an accurate, scalable, and backwards-compatible service for mapping clients to a nearby server. It provides a DNS interface by which unmodified clients can look up a service name, and get the IP address of the closest server. A shared system for performing such a mapping amortizes the administration and implementation costs of proximity-based server selection. It is aimed at minimizing the amount of effort required for system developers to make new and existing infrastructure services proximity-aware.

1. INTRODUCTION

There has been much recent activity on overlays, peer-to-peer systems, web services, distributed online games, content distribution, and many other distributed systems where service components are distributed geographically. This work, fueled by recently emerging techniques for self-organization and enabled by the emergence of the PlanetLab testbed for deployment, has three salient features. First, these systems rely critically on service components that are distributed widely. Second, the scale of distribution is so large that simple techniques, such as keeping track of all components, are unlikely to work well. Third, and most importantly, clients of these systems benefit significantly from being directed to the nearest suitable service component. Selecting a close-by server is critical in order to ensure that the services provide low latency, achieve high sustainable throughput, and scale well by reducing aggregate load on the network.

Directing clients to the nearest suitable server requires an accurate and scalable mechanism for locating the closest server. Ideally, this mechanism would support existing applications without changes to client software. The problem is trivial if any of these three critical properties are ignored. If accuracy in finding the closest node is not an issue, a client can be directed to any server. If scale is not an issue, a client can simply measure its latency to all servers regardless of system size and pick the one that offers the lowest latency. And if support for legacy clients is not an issue, every client can be forced to download complex client-side software for performing extensive measurements and finding the mapping to the nearest server. Yet many realistic systems require all three properties at once. For instance, clients of an Akamai-like content distribution network [18, 4, 21] typically need to be routed to the nearest caching server from a URL accessed through an unmodified browser. A client wishing to download a software package through an automated software installer needs to be routed to the nearest server containing the package without having to modify the installer software. Clients of overlays and peer-to-peer services [14] typically need to be routed to the nearest overlay node transparently.

Past work on directing clients to closest server replicas can be classified into systems based on virtual coordinates [9, 10, 3, 7, 19, 17, 11, 2, 6, 8] and active probing [22, 5, 13, 20, 16, 1]. While these systems provide the basic algorithms for determining the approximate position of a node in the network and finding a nearby server, there is a significant gap between the interface they provide and the interface required to support legacy clients. The legacy clients benefit most from a simple nameserver interface through which they can do a lookup for the closest desired server, whereas past systems typically provide a complex interface for reasoning about node positions that is not tied into a naming service.

The work required to create a new system that maps clients to nearby servers is non-trivial. The system must be able to return accurate results even for unreachable clients that are behind network address translators or firewalls. Its manner of probing clients must be sufficiently benign to avoid triggering sensitive intrusion detection systems. This includes rate-limiting probes, aggressive caching of latency data to avoid repeated probes, and user-specified blacklists to exclude probing specific clients. Currently, this effort must be repeated for each new service.

In this paper, we outline **ClosestNode.com**, a shared service for providing proximity server selection for distributed systems that amortizes the cost of development over many services. To ensure transparent support for legacy clients, we use DNS as the interface for initiating closest node selection lookups. Consequently, any current systems that rely on DNS for resolving service names to IP addresses can easily transition to ClosestNode.com with no changes to the clients other than providing them with an updated domain name. The system works by assigning a service, say *CobWeb*, to the domain *cobweb.closestnode.com*, where DNS lookups to the domain will trigger a closest node selection lookup to the requesting client. ClosestNode.com uses Meridian [22], an accurate, lightweight, and scalable node selection framework, as its underlying node selection mechanism, essentially acting as a DNS to Meridian gateway. Past work has shown that Meridian achieves high-accuracy and scalability; coupling it with DNS provides backwards compatibility.

The main goal of ClosestNode.com is to make the transition of a traditional distributed service to one that takes advantage of network proximity as simple and painless as possible. By supporting oblivious clients and requiring only minimal changes to the infrastructure service itself, the majority of infrastructure services running on PlanetLab today can be quickly made proximity-aware by using ClosestNode.com. We have recently built a content distribution network, called CobWeb [18], that uses ClosestNode.com for directing clients to the nearest content cache. Using ClosestNode.com was easy to perform on the server side and provided a transparent interface on the client side.

2. FRAMEWORK

Our basic framework consists of a small number of DNS servers acting as authoritative name servers for services built on top of ClosestNode.com. A service, say *CobWeb*, first registers with ClosestNode.com on our website and is provided with a domain *cobweb.closestnode.com*. The only changes required to the service are to execute a daemon process on each of its servers, and to provide the ClosestNode.com database with a small number of servers that act as entry points to the service. The daemon, based on the Meridian [22] system for geographic query routing, builds and maintains an overlay that is used for routing queries; the maintenance overhead is modest, rarely exceeding 1 kB/sec. When a DNS request comes in for *cobweb.closestnode.com*, the ClosestNode.com DNS server receiving the request will issue a Meridian closest node discovery query to one of the entry point servers of *CobWeb*. Since the client may reside in an unroutable portion of the IP space, and since we do not require any changes to the client, Meridian uses the client's upstream recursive DNS server as the query target, and follows the Meridian closest-node protocol to locate a nearby server. Once the closest node discovery query completes, the ClosestNode.com DNS server returns the IP address of the closest node in response to the original DNS query request. Figure 1 illustrates the protocol.

The Meridian closest-node protocol, described fully in [22], organizes the servers into a lightweight, loosely structured overlay. The overlay is structured to provide each server with near-authoratative information about nodes in its own vicinity, as well as knowledge of a sufficient number of diverse and distant nodes to facilitate efficient overlay traversal. A server receiving a Meridian closest node discovery request first determines its own latency to the target, and then requests its neighbors that can potentially be closer to the target than itself by some threshold to determine their respective latency to the target. If any of the neighbors are closer to the target by the required threshold, the query is forwarded to the neighbor closest to the target and query processing repeats on that server. Otherwise, the IP address of the closest known server to the target is returned to the requester.

The Meridian daemon process issues both ICMP ECHO packets as well as DNS requests for localhost to determine a server's latency to the client's DNS server. Using different, equivalent probing methods is necessary for gathering accurate latency data for cases where the clients issuing queries are behind firewalls. For clients that can not be successfully probed, longest prefix match of the DNS server IP address to the entry point nodes are used for server selection. Meridian uses rendezvous hosts to relay control messages and queries to servers behind NATs and firewalls, and thus enables deployment of servers anywhere in the network.

Although Meridian has been previously shown to achieve high accuracy for closest node queries, the accuracy of the ClosestNode.com framework is also dependent on the client being close to its upstream recursive DNS server. This is the typical configuration for both corporate and home users, as it is advantageous for ISPs and institutions to use an upstream recursive DNS server for caching purposes and to locate it near it clients to reduce DNS query latency.

The shared authoritative name servers are unlikely to pose bottlenecks to scalability for realistic workloads. Should the query load on ClosestNode.com increase to the point where authoritative name servers pose impediments to scalability, load on individual DNS servers can be reduced by simply adding additional servers. The effort of administering the service, however, is constant and shared among all the different services using ClosestNode.com.

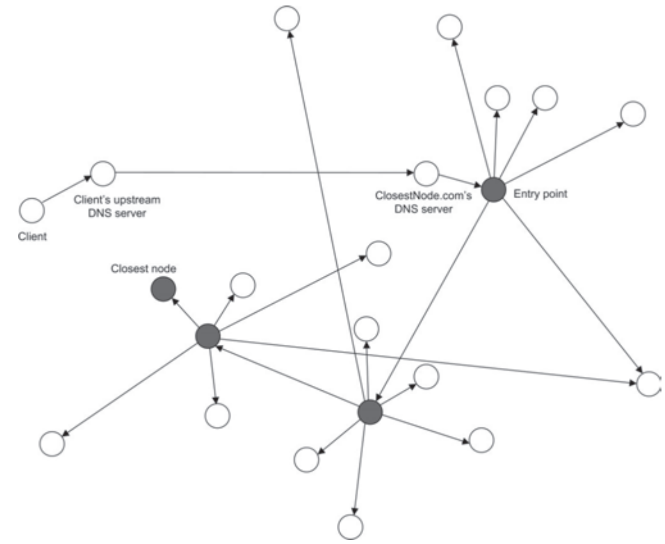

Figure 1: The ClosestNode.com's DNS server initiates a closest node discovery protocol in response to a DNS lookup. The closest node discovery protocol selectively probes the client's upstream DNS server, and routes the query to closer servers in exponentially-closer steps. The IP address of the closest server is returned to the client to complete the query.

For high churn systems, the ClosestNode.com DNS servers use a heartbeat counter to the entry point nodes to ensure the liveness of the entry points and avoid long timeouts waiting for a reply from a offline entry point node. ClosestNode.com enables additional entry point nodes to be added dynamically in order to compensate for failures of entry-point nodes. The ClosestNode.com servers also enable the services to custom-tailor the caching policy for latency measurements. Each service can specify its desired DNS time-to-live value, which trades-off lookup overhead against the maximum time limit for a stale mapping.

ClosestNode.com performs several optimizations to reduce lookup latencies perceived by clients. First, each ClosestNode.com DNS server selects the closest entry point node for each service as its primary entry point, reducing the round trip time between the ClosestNode.com DNS server and the entry point node. Second, the number of hops taken by the closest node discovery protocol can be capped to bound the maximum latency, trading off server selection accuracy for latency. Finally, to totally remove additional delays for the most sensitive services, the initial DNS lookup is resolved immediately to the IP address of the entry point node that shares the longest IP prefix with the client's DNS server. This response is returned with a very low time-to-live, and the ClosestNode.com DNS server performs a normal closest node discovery query in the background that is cached and propagated across all of the ClosestNode.com DNS servers. When the same client DNS server requests for the same domain after the initial answer expires, the ClosestNode.com DNS server can return the more accurate cached result.

3. NODE SELECTION ALTERNATIVES

The main design decision for the node selection mechanism is between schemes based on virtual coordinates and active probing. In networks where nodes are relatively stable and long-lived, virtual coordinates typically incur a lower average node selection latency than active node selection schemes. Virtual coordinates in such networks can be computed once and used many times. However, in infrastructure services with transient clients, the cost to assign virtual coordinates to a given client can not be amortized as the typical client only performs one selection. Since the network overhead for latency probes required to perform node selection using active probing can be made comparable to the overhead incurred by virtual coordinate systems to localize a client [22], existing virtual coordinate systems do not have a special advantage over active probing.

However, by mapping Internet latencies into a low dimensional Euclidean space, virtual coordinate schemes incur inherent embedding errors that result in inaccurate node selection. Virtual coordinates are also insufficient to solve the the node selection problem itself. Coordinate systems must be paired with either a central database with the coordinates of every node, or a CAN [12] like structured overlay that would significantly increase the complexity of the system. Selection schemes based on databases of IP address to geographical region [15] mappings can reduce the closest node discovery latency to that of a database lookup, but are fragile to IP assignment changes and are typically less accurate than passive or active selection schemes.

4. FUTURE WORK

Although minimizing latency is the primary goal of server selection for most systems, we plan on incorporating other metrics in determining the most appropriate server to select. This includes system load, available bandwidth, and price of each link. Services can provide their own utility functions based on these metrics, and a scheme similar to ClosestNode.com, perhaps better titled BestFitNode.com, can determine a client to server mapping that maximizes utility.

To simplify the deployment of ClosestNode.com, we plan on deploying a single Meridian overlay across PlanetLab that can be shared across multiple services. Each registered service would provide its entire list of servers, and a mapping from each PlanetLab node to its closest server in the list can be determined periodically and stored in a database that is available to the ClosestNode.com DNS servers. When a closest node request arrives for a registered service, the closest PlanetLab node can be determined, and the server closest to the PlanetLab node can then be returned as the closest node. This makes adopting ClosestNode.com trivial and enables caching and cross-service sharing of latency data. However, the accuracy of the server selection will likely suffer compare to the current system, and this deployment mode is targeted only for services that can not readily deploy their own Meridian overlay.

5. SUMMARY

ClosestNode.com is a scalable, shared geocast service for distributed systems. The system amortizes the cost of providing proximity-based server selection by sharing infrastructure across multiple services. ClosestNode.com uses Meridian as its node selection mechanism, and supports oblivious clients by acting as a DNS to Meridian gateway, enabling server selection via DNS lookups. The goal of ClosestNode.com is to minimize the amount of work for distributed system builders to provide proximity-based server selection. The system is free and open to access, and has been in continuous use since July 2005 by CobWeb, a high-volume content distribution network running on PlanetLab. We encourage developers of other large scale distributed systems to try ClosestNode.com and get the benefit of an accurate, scalable and backwards-compatible system for directing clients to nearby servers.

6. REFERENCES

[1] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable Peer Finding on the Internet. In *Global Internet Symposium*, Taipei, Taiwan, Nov. 2002.

[2] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Intl. Conference on Distributed Computing Systems*, Tokyo, Japan, Mar. 2004.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *SIGCOMM*, Portland, OR, Aug. 2004.

[4] M. Freedman, E. Freudenthal, and D. Mazires. Democratizing Content Publication with Coral. In *Symposium on Networked Systems Design and Implementation*, San Francisco, CA, Mar. 2004.

[5] C. Kommareddy, N. Shankar, and B. Bhattacharjee. Finding Close Friends on the Internet. In *Intl. Conference on Network Protocols*, Riverside, CA, Nov. 2001.

[6] L. Lehman and S. Lerman. PCoord: Network Position Estimation Using Peer-to-Peer Measurements. In *Intl. Symposium on Network Computing and Applications*, Cambridge, MA, Aug. 2004.

[7] H. Lim, J. Hou, and C. Choi. Constructing Internet Coordinate System Based on Delay Measurement. In *Internet Measurement Conference*, Miami, Florida, Oct. 2003.

[8] Y. Mao and L. Saul. Modeling Distances in Large-Scale Networks by Matrix Factorization. In *Internet Measurement Conference*, Taormina, Sicily, Italy, Oct. 2004.

[9] T. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *INFOCOM*, New York, NY, June 2002.

[10] T. Ng and H. Zhang. A Network Positioning System for the Internet. In *USENIX*, Boston, MA, June 2004.

[11] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for Scalable Distributed Location. In *Intl. Workshop on Peer-To-Peer Systems*, Berkeley, CA, Feb. 2003.

[12] S. Ratnasamy, P. Francis, M. Hadley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *SIGCOMM*, San Diego, CA, Aug. 2001.

[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *INFOCOM*, New York, NY, June 2002.

[14] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *SIGCOMM*, Philadelphia, PA, Aug. 2005.

[15] M. Schiffman. ipgeo. http://www.packetfactory.net/projects/ipgeo, 2005.

[16] K. Shanahan and M. Freedman. Locality Prediction for Oblivious Clients. In *Intl. Workshop on Peer-To-Peer Systems*, Ithaca, NY, Feb. 2005.

[17] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In *INFOCOM*, San Francisco, CA, Apr. 2003.

[18] Y. Song, V. Ramasubramanian, and E. Sirer. Optimal Resource Utilization in Content Distribution Networks. In *Computing and Information Science Technical Report TR2005-2004*, Cornell University, Nov. 2005.

[19] L. Tang and M. Crovella. Virtual Landmarks for the Internet. In *Internet Measurement Conference*, Miami, Florida, Oct. 2003.

[20] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. In *Hot Topics in Networks*, Princeton, NJ, Oct. 2002.

[21] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec. 2002.

[22] B. Wong, A. Slivkins, and E. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *SIGCOMM*, Philadelphia, PA, Aug. 2005.