Latency Model of a Distributed Hash Table with Big Routing Table

Daishi Kato
Internet Systems Research Laboratories, NEC Corporation daishi@cb.jp.nec.com

Abstract

In peer-to-peer research, one of the most popular areas is Distributed Hash Table (DHT). Among many topics in the DHT area, this paper focuses on DHT latencies, which are mainly caused by its basic multi-hop lookup function. Some DHTs already have the capability of building big routing tables to reduce hop counts. However, none of them are explicitly trying to enlarge the routing table and lower the hop count. This paper provides a simple latency model of DHT and discusses how big routing tables help reduce latency.

1. Introduction

Even though the popularity of peer-to-peer systems continues to increase, questions remain about their scalability. To increase the scalability of peer-to-peer systems, some research such as Chord [1], Pastry [2], and Kademlia [3] have proposed distributed hash table (DHT) techniques. In a DHT system, one "puts" a (*key*, *value*) pair of data, which others can "get" by a *key* from the DHT. "Lookup," the basic function of DHT, takes a *key* as an input and returns the "closest" peer(s) to the *key*. Each system has a different definition of "close"; for example, in Pastry the definition is based on a numerical distance between an initially assigned peer identifier value and a hash value of the key by some consistent hash function.

Along with different definitions of "close," each system also uses a different routing algorithm, with different average sizes of routing tables (denoted as k), and average hop counts (denoted as h). The typical algorithm can achieve:

$$k = 2^b \log_{2^b} N \quad h = \log_{2^b} N \tag{1}$$

where N is the total number of peers in a network, and b is a configurable variable. Although we refer these formulas in [2] and [3], Chord can also achieve the same (or better) performance [4]. There are two major choices for b: one is to make b small to keep k low, which we call a "small routing table," and the other is to enlarge b to keep h low, which we call a "big routing table."

The latencies in DHTs are mainly caused by the latencies in the lookup function. We focus on lowering the lookup latencies by reducing the hop count with big routing tables; however, actually keeping a big routing table updated tends to be expensive. Kademlia and other DHT algorithms have a "parallel lookup" capability to send simultaneously requests to multiple peers. With parallel lookup, the routing table does not have to be completely updated, allowing it to contain an entry of a peer that has already exited the network.

For simplicity in this paper we assume the following: a) The latency of the lookup function is the latency of DHTs, because other factors that increase DHT latencies does not basically depend on network size. b) The underlying network is uniform. Methods that deal with the underlying network topology are left for other research such as [5]. c) A method to build routing tables that allow parallel lookups is given. We believe such a method can easily be assembled based on the "accelerated lookups" in [3].

2. Latency Model

This section describes the parameters of a simple DHT model. In section 1, we defined N, k, and h. Since we are ignoring the underlying network topology, we define a constant D as a latency of networks, which is similar to the Internet Control Message Protocol (ICMP) ping round-trip time. λ_C is defined as the average rate at which each peer joins and exits (or fails) the network. In other words, in every $1/\lambda_C$ period of time, one peer probably exits the network, and another peer joins the network. A typical DHT algorithm uses a ping mechanism to periodically send a small packet to detect peer exits (or failures), and λ_p is defined as the expected rate at which a peer sends a ping packet to each peer in its routing table. Here w is called a lookup width, which denotes how many peers a peer sends a request to in each parallel lookup step.

We then calculate the expected lookup latency d and the probability of lookup failure $P_{lookupfailure}$. Our definition of "good algorithm" for DHT latencies is the achievement of low d with acceptably (depending on the application of the DHT) high $P_{lookupfailure}$, under the certain bandwidth.

Recalling that D is a constant, we get $d = Dh$. According to [6], assuming that the time that a peer exists in a network is exponentially distributed, the probability that a

peer in a routing table is failed is $1 - e^{-\lambda_C/\lambda_p}$. We also assume that the parallel lookup works perfectly, meaning that the lookup function returns the correct result even if all but one peer has failed at each lookup step. With these assumptions, we get the following equation:

$$P_{lookupfailure} = 1 - \{1 - (1 - e^{-\lambda_C/\lambda_p})^w\}^h \qquad (2)$$

We do not describe the bandwidth here, but our approach is to reduce λ_p to control the bandwidth with a big routing table. Increasing w might help to reduce λ_p. Further bandwidth discussion remains as future research.

3. An Example

In this section, we ascribe values to the parameters described in section 2.

We let $N = 1,000,000$, $b = 1$ for the small routing table, and $b = 10$ for the big routing table. Equation 1 shows that $k = 39.86$ and $h = 19.93$ for small routing tables, and $k = 2040$ and $h = 1.993$ for big routing tables.

According to [7], 80% of peers have latencies of at least 70 ms, hence $D = 0.07$ (seconds). [7] has also mentioned that the median session duration is approximately 60 minutes, and in this case, $\lambda_C = 1/60/60 = 0.0002777$ (event/second). We simply set $w = 3$ from the α value described in [3].

As expected, $d = 1.395$ (seconds) with the small routing table, and $d = 0.1395$ (seconds) with the big routing table. The difference of about 1 second is rather long, especially if it is the waiting time for an end user. A big routing table reduces latencies well, but its major drawback is an increase of the maintenance cost.

Figure 1 shows the relationship between λ_p and $P_{lookupfailure}$. Note that the x-axis is log scale. The graph indicates that to achieve the same $P_{lookupfailure}$, a big routing table requires a lower λ_p than a small routing table. By reducing λ_p of big routing tables, the maintenance cost can also be lowered to that of small routing tables.

4. Related Work

J. Xu et al. [8] discuss the tradeoffs between routing table size and hop count. One of their goals is to prove the optimal cases, whereas our approach is to abandon routing table size and concentrate on low hop counts.

R. Mahajan et al. [9] introduce a dynamic ping rate to control the network bandwidth. Although our approach lowers the ping rate statically, this technique is also adaptable.

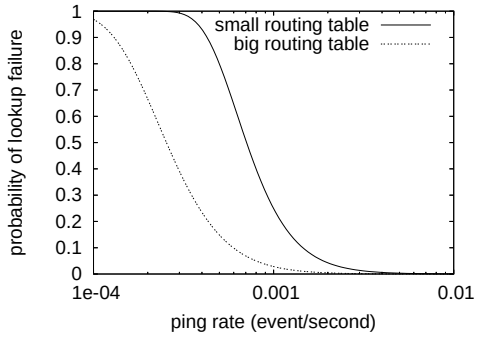

Figure 1. ping rate – probability of lookup failure

5. Summary

This paper primarily compared "big routing tables" and "small routing tables." DHT latencies, caused by hop count, can be reduced with big routing tables. We showed a simple latency model with parameters and illustrated that big routing tables help reduce latencies. The possibility of reducing maintenance costs of a big routing table is presented, and future research should give a complete method.

References

[1] I. Stoica et al. *Chord: A scalable peer-to-peer lookup service for internet applications*. In Proc. of ACM SIGCOMM (Aug. 2001).

[2] A. Rowstron et al. *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*. In Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), (Nov. 2001).

[3] P. Maymounkov et al. *Kademlia: A Peer-to-peer Information System Based on the XOR Metric*. In Proc. of the 1st International Workshop on Peer-to-Peer Systems (Mar. 2002).

[4] J. Li et al. *Comparing the performance of distributed hash tables under churn*. In Proc. of the 3rd International Workshop on Peer-to-Peer Systems (Feb. 2004).

[5] M. Freedman et al. *Sloppy hashing and self-organizing clusters*. In Proc. of the 2nd International Workshop on Peer-to-Peer Systems (Feb. 2003).

[6] S. Zhuang. *Exploring Tradeoffs in Failure Detection in P2P Networks*. In Proc. of the 1st IRIS Student Workshop (Aug. 2003).

[7] S. Saroiu et al. *A Measurement Study of Peer-to-Peer File Sharing Systems*. In Proc. of Multimedia Computing and Networking 2002 (Jan. 2002).

[8] J. Xu et al. *On the Fundamental Tradeoffs Between Routing Table Size and Network Diameter in Peer-to-Peer Networks*. IEEE Journal on Selected Areas in Communications. (Vol. 22, No. 1, Jan. 2004)

[9] R. Mahajan et al. *Controlling the Cost of Reliability in Peer-to-Peer Overlays*. In Proc. of the 2nd International Workshop on Peer-to-Peer Systems (Feb. 2003).