A New Technique with Provable Performance for Congestion Control in DHT Networks: ERT

Palakollu Pavani [#1], Mrs. Syed Shaheen [*2]

[#]M.Tech Scholar, *Assistant Professor
Department of CSE,
Chaitanya Engineering College,
Visakhapatnam, AP, India.

Abstract

Distributed hash table (DHT) networks based on consistent hashing functions have an inherent load balancing problem. The problem becomes more severe due to the heterogeneity of network nodes and the non-uniform and time varying file popularity. Existing DHT load balancing algorithms are mainly focused on the issues caused by node heterogeneity. To deal with skewed lookups, this paper presents an elastic routing table (ERT) mechanism for query load balancing, based on the observation that high degree nodes tend to experience more traffic load. The mechanism allows each node to have a routing table of variable size corresponding to its capacity. The indegree and outdegree of the routing table can also be adjusted dynamically in response to the change of file popularity and network churn. Theoretical analysis proves the routing table degree is bounded. The ERT mechanism facilitates locality-aware randomized query forwarding to further improve lookup efficiency. By relating query forwarding to a supermarket customer service model, we prove a 2-way randomized query forwarding policy leads to an exponential improvement in query processing time over random walking. Simulation results demonstrate the effectiveness of the ERT mechanism and its related query forwarding policy for congestion and query load balancing. In comparison with the existing "virtual-server"-based load balancing algorithm and other routing table control approaches, the ERT-based congestion control protocol yields significant improvements in query lookup efficiency.

Keywords:

Distributed hash table, peer-to-peer, load balancing, congestion control.

1. Introduction

In Structured P2P overlay networks, each node and file key is assigned a unique ID, based on a consistent hashing function. The file keys are mapped on to nodes according to their IDs and a distributed hash table (DHT) definition. The DHT maintains topological relationships between the nodes and supports a routing protocol to locate a node responsible for a required key. Because of their lookup efficiency, robustness, scalability, and deterministic data location, DHT networks have received much attention in recent years. Representatives of the systems include CAN [1], Chord [2], Tapestry [3], Pastry [4], and Cycloid [5].

Because of DHT networks' lookup efficiency, robustness, scalability and deterministic data location, they have received much attention in recent years. DHT networks have an inherent load balancing problem. It is because consistent hashing for file distribution produces a bound of O(log n) key imbalance between the network nodes. The problem becomes even more severe as the nodal heterogeneity increases. What is more, files stored in the system often have different popularities and the access patterns to the same file may vary with time. It is a challenge to design a DHT protocol with the capability of congestion control.

The primary objective of congestion control is to avoid bottleneck in any node (*i.e.* the query load exceeds its capacity).Bottleneck may occur with query overflow in which too many queries received by the node at a time, or with data overflow in which a too high volume of data need to be downloaded and forwarded by the node simultaneously. Although files are often transmitted via a direct connection between source and destination, data forwarding through intermediary nodes in the query routing path is often used for the provisioning of anonymity of file sharing, as in Freenet [6] and Hordes [7].

In the past, many load balancing strategies have been proposed to deal with network heterogeneity; see **[**8, 9] for examples. They assign a key ID space interval to each node based on its capacity. Because the approaches are based merely on key ID assignment, they do not provide any control over congestions caused by the factor of non-uniform and time-varying file popularity. There are other approaches, based on "item-movement", which take into account the effect of file popularity on query load [10, 11]. In these approaches, heavily loaded nodes probe light nodes and reassign excess load between the peers by changing the IDs of related files or nodes. Albeit flexible, the load reassignment process incurs high overhead for changing IDs, especially under churn.

Notice that the existing load balancing approaches assume that each node (or virtual node) should maintain the same number of neighboring relationships, irrespective of its capacity. The principle of power-law networks tells that higher degree nodes tend to experience more query loads [12]. In light of this, in this paper, we present an elastic routing table (ERT) mechanism to cope with node heterogeneity, skewed queries, and churn in DHT networks. Unlike current structured P2P routing tables with a fixed number of outlinks, each ERT has a different number of inlinks/outlinks and the indegree/ outdegree of each node can be adjusted dynamically according to its experienced traffic so as to direct query flow to light nodes. Most recently, Castro *et al.* [13] exploited heterogeneity by modifying the proximity neighbor selection algorithm.

The ERT-based congestion control protocol goes beyond the construction of capacity-aware DHTs. It deals with congestion due to time-varying file popularity by adjusting the indegrees and outdegrees of the routing tables and capacity-aware query forwarding. We summarize the contributions of this paper as follows.

- An initial indegree assignment for capacity-aware DHTs construction. The indegrees are provably bounded.
- A policy for periodic indegree adaptation to deal with the non-uniform and time-varying file popularity. It is proved that the indegree bounds remain bounded.
- A topology-aware randomized query forwarding policy on the elastic DHTs. It is proved that the ERT-enabled query forwarding leads to an exponential improvement in query processing time over random walking.
- Comprehensive simulations demonstrate the superiority of the elastic congestion control protocol, in comparison with the "virtual-server"-based load balancing policy and other routing table control approaches

2. Related Work

There have been many load balancing algorithms to deal with node heterogeneity and network churn **[**8, 9]. "Virtual server" [14] is a popular approach, in which each real node runs O(log n) virtual servers and the keys are mapped onto virtual servers so that each real node is responsible for the key ID space of different length proportional to its capacity. It is simple in concept, but the virtual server abstraction incurs large maintenance overhead and compromises lookup efficiency. Godfrey *et al.*[8] addressed the problem by arranging a real server for virtual ID space of consecutive IDs. In [9], Bienkowski *et al.* proposed a distributed randomized scheme to let a linear number of nodes with short ID space interval to divide the existing long ID space interval, resulting

in an optimal balance with high probability.Initial key ID space partitioning is insufficient to guarantee load balance, especially in churn. It is often complemented by dynamic load reassignment.

Godfrey *et al.* proposed schemes to rearrange load between heavy nodes and light ones so as to avoid bottleneck [7]. They assumed that query load was uniformly distributed in the ID space. Bharambe *et al.* [10] proposed a load balancing algorithm to deal with the congestion caused by biased lookups. It proceeds in a way that heavily loaded nodes requests lightly loaded nodes to leave from their current locations and rejoin at the location of the heavily loaded nodes.

TABLE 1
Notations

Notation	Description
n	the number of nodes
$\tilde{n}$	estimated n
c	node capacity
$\tilde{c}$	estimated c
l	node load
s	query distribution share
d	node indegree
α	indegree per unit capacity
d^{∞}	node maximum indegree
β/μ	a pre-defined percentage
$\gamma_c/\gamma_n/\gamma_l$	a factor
T	a unit time period
ν_{max}	the maximum incoming query rate
ν_{min}	the minimum incoming query rate
g	congestion rate

Castro *et al.* [13] proposed a neighbor selection algorithm to construct routing tables based on node capacities. Its basic idea of using node indegrees to exploit node heterogeneity is similar to our initial indegree assignment algorithm. Their algorithm directs most traffic to high capacity nodes because it does not choose low capacity nodes as neighbors unless the indegree bounds of high capacity nodes are reached. In contrast,ERT mechanism would distribute the traffic between the neighbors proportional to their capacities so as to make full utilization of both high and low capacity nodes. More importantly, ERT mechanism deals with more than node heterogeneity. It deals with biased lookups and network churn by adjusting the table indegree and outdegree dynamically and query forwarding.

Finally, we note that the problem of congestion control is not unique in structured P2P networks. It has been a crucial performance issue in unstructured P2P networks, as well. Many studies have been devoted to flow control in unstructured networks; see for recent examples. Like congestion control in DHT networks, their solutions are based on the principle of power-law networks as well. Since they were designed for flooding or random walk query routing networks, the flow control algorithms on unstructured P2P networks cannot be applied for load balancing and congestion control in DHT networks.

3. Elastic DHT

ERT is designed based on the principle of power-law networks that a higher degree node tends to receive more query load. The ERT-based protocol constructs routing tables of different number of outlinks so as to distribute query load among the nodes in proportion to their capacities. Each node dynamically adjusts its indegree according to its actual query load.

3.1 Query Load Balancing and ERT

We assume a DHT network with n physical nodes, labelled as an integer from 1 to n. Node i, $1 \leq i \leq n$, has a capacity that it is willing to devote or able to process queries.We assume that node i's capacity ci is a quantity that represents the number of queries that node i can handle in a given time interval T. In practice, the capacity should be determined as a function of a node's access bandwidth, processing power, disk speed, etc. We define the load of node i, l_i, as the number of queries it receives and transmits to its neighbors over time T. We refer to node with traffic load $li \leq ci$ as a *light node*; otherwise a heavy or overloaded node.

The purpose of a congestion control protocol is to avoid heavy nodes in query routings and distribute query load among nodes corresponding to their capacities. From the view

point of an entire system, fair load distribution is achieved by letting each node's load share proportional to its "fair load share", as defined by

$$s_i = \frac{l_i / \sum_i l_i}{c_i / \sum_i c_i}.$$

Ideally, the fair share si should be kept close to 1. One way to achieve this is to measure the traffic load li of every node periodically and forward queries according to collected global traffic load information. Obviously, this method is too costly to be used in any scalable overlay networks. In [10], Lv*et al.* showed that a high degree node in Gnutella network would most likely experience high query load. We apply the same principle to the design of congestion control in DHT networks. We define *indegree*, denoted by di, $1 \leq i \leq n$, as the number of inlinks of node i. Under the assumption that nodes and file queries are uniformly distributed in a DHT network without churn, li is directly related to di. For that reason, we propose to set the initial indegree of each node to an appropriate value, according to its capacity. As a result, heterogenous nodes would have routing tables of different sizes (outdegree). It is expected that the higher the indegree of a node, the higher traffic load the node would experience in the uniform system. To deal with skewed queries caused by nonuniform and time-varying file popularity and network churn,we propose an integral indegree adaptation component to dynamically adjust node indegree periodically. We refer to such an elastic routing table as ERT.

3.2 Initial Indegree Assignment

In current DHT networks, each routing table has a fixed number of outlinks for a given network size n. For example, it is log n in Chord and a constant number in Cycloid [16]. The outlinks determine node indegrees. Since we want to have a node's indegree to be proportion to its capacity, we reverse the relationship to determine node outdegree by setting an appropriate value of indegree. To the end, we need to address two questions:

1. How to determine a node's indegree to make full use of its capacity and keep it lightly loaded meanwhile?
2. How to construct ERTs with different indegrees for nodes of different capacities, and meanwhile retain the original DHT neighbor selection functions in lookup ?

Algorithm 1 Pseudo-code for indegree expansion algorithm of Cycloid node $i\ (k, a_{d-1}a_{d-2}\ldots a_k\ldots a_0)$.

1: *//probe backward fingers of cubical neighbor*
2: figure out a set of cubical neighbor inlinks $ID = (k+1, a_{d-1}a_{d-2}\ldots\overline{a}_k xx\cdots x)$
3: $id = (k+1, a_{d-1}a_{d-2}\ldots\overline{a}_k 00\cdots 0)$
4: **while** not finish probing all IDs in ID $\wedge((d_i^{\infty} - d_i) \geq \beta d_i^{\infty})$ **do**
5: **while** id is in backward fingers **do**
6: $id = (k+1, a_{id}++) \in ID$
7: **end while**
8: probe id for cubical neighbor inlink
9: $id = (k+1, a_{id}++) \in ID$
10: **end while**
11: *//probe backward fingers of cyclic neighbor*
12: figure out ID of cyclic neighbor inlinks $ID = (k+1, a_{d-1}a_{d-2}\ldots a_k xx\cdots x)$
13: $id = (k+1, a_{d-1}a_{d-2}\ldots a_k 00\cdots 0)$
14: **while** not finish probing all IDs in ID $\wedge((d_i^{\infty} - d_i) \geq \beta d_i^{\infty})$ **do**
15: **while** id is in backward fingers **do**
16: $id = (k+1, a_{id}++) \in ID$
17: **end while**
18: probe id for cyclic neighbor inlink
19: $id = (k+1, a_{id}++) \in ID$
20: **end while**

The initial indegree assignment algorithm proceeds recursively. We prove the ERT indegree resulted from the algorithm is bounded. Readers are referred to for the proof of all theorems. We assume a DHT that manages a unit-size ID space [0, 1), and the DHT uses consisting hash to partition the ID space among the nodes. Thus, the responsible ID space imbalance is log n. We assume that each node i can estimate its capacity ci and the network scale n within a factor of c and n, respectively, of the true values, with high probability1; readers are referred to for details of such an estimation process.

3.3 Periodic Indegree Adaptation

Considering the fact that query load often varies with time, the initial indegree assignment is not robust enough to limit a node's query load under its capacity. The congestion control protocol should adapt to the change of query rate and lookup skewness caused by non-uniform and time-varying file popularity, as well as network churn.

We design an periodic indegree adaptation algorithm to help each node adjust its indegree periodically according to the maximum load it experienced. Specifically, every node i records its query load li over T periodically and checks whether it is overloaded or lightly loaded by a factor of l; *i.e.* whether gi = li/ci > γl or < 1/γl. In the former case, it decreases μ(li − ci) indegree by asking some of its backward fingers to delete it from their routing table, then deletes corresponding backward fingers, and decreases its maximum indegree d_i^{∞} correspondingly. To choose backward figures for removing, it chooses the backward figure with longest logical distance. In the latter case, it increases μ(ci − li) indegree by probing other nodes to take it as their neighbors using the inlink expansion algorithm discussed in Section 3.2 and increases its d_i^{∞} correspondingly. The following theorems show that the ERT indegree and outdegree in the process of adaption remain bounded.

4. Topology-aware Randomized Query Forwarding

Periodic indegree adaptation is not sufficient to deal with query load imbalance caused by churn and skewed lookups. In this section, we present a complementary topology-aware randomized query forwarding algorithm to help forward queries towards light nodes, and meanwhile reducing lookup latency.

4.1 Query Forwarding

With the initial indegree assignment and periodic adaptation algorithms, each node's routing table has a variable size. With a high probability, each ERT has a set of outlinks in each of its routing table entries. For example, a Cycloid node i (4,101-1-1010) has cubical outlinks pointing to nodes (3, 1010-0000), (3, 1010-0001) and (3, 1010-0010). If it receives a query and decides that the query be forwarded to its cubical neighbors based on its original routing algorithm, there would be three candidates to take the query.

A simple forwarding policy is random walk, in which one of the outlinks is selected randomly. Another one is gradient based walk that forwards the query to the "best" candidate in terms of their workload. Instead of probing all of the neighbors to find out the best candidate, we restrict the search space to a small set of size b. That is, once receiving a query, node i first randomly selects b neighbors (outlinks) and then probes the nodes in the set sequentially, until a light node is found. In the case that all candidates are overloaded, the query is forwarded to the least heavily loaded one.

The b-way randomized query forwarding is further enhanced by taking into account the underlying topology information in the candidate selection. In the topology-aware forwarding policy, a node selects the best candidate among b neighbors by two extra criteria: close to the target ID by the logical distance (hops) in the DHT network and close to the node by the physical distance on the Internet; Readers are referred to [11] for a land marking method to measuring physical distance between two nodes on DHT networks. In the case that the two candidates are both lightly loaded, the closer node in logical distance is selected. Their physical distance is used to break the tie of logical distance.

Probing b neighbors is a costly process. How to find a good candidate from b neighbors at a relatively low cost? Query forwarding in this context can be regarded as a supermarket customer service model. The supermarket model is to allocate each incoming task (a customer) to a lightly loaded server with the objective of minimizing the time each customer spends in the system. Mitzenmacher [12] proved that granting a task with two server choices and dispatching it to one of the servers with less workload lead to an exponential improvement over the single choice in the expected execution time of each task. But a poll size larger than two gains much less substantial extra improvement. Furthermore, Mitzenmacher *et al.* [13] improved the performance of two choice methods dramatically by the use of

memory. In this method, each time a task is allocated; the least loaded of that task's choices after allocation is remembered and used as one of the possible choices for the next task. We tailored this memory-based randomized task dispatching method with modifications to topology-aware randomized query forwarding.

We set b = 2. A node first randomly selects two options, say node i and j. It then selects the better one, say node i, and the least loaded node between i and j is remembered after node i increases by one load unit. We assume the node is still i, which is used for the next forwarding. Later, when the node needs to forward a query to the same routing table entry, it only needs to randomly choose one neighbor, instead of two. With the remembered i, it starts the process again.

To further reduce the heavy nodes in query routing, a query flows by the use of the information of overloaded nodes encountered before, to avoid overloaded node in the succeeding routing. Algorithm 2 shows the pseudocode of the topology-aware randomized query forwarding algorithm.

4.2 Analysis of Query Forwarding

The simple query forwarding model (QFM) can be rephrased as the following: after a node receives a query, it forwards the query to one of its neighbor options. If the chosen neighbor is heavily loaded by a factor l, another specific neighbor is turned to. This process is repeated until the node finds a light neighbor. In the case that all neighbor options are heavy, the query is forwarded to the least heavily loaded option. We assume that the query forwarding time for a query is constant and incoming query is Poison distributed.

The forwarding model can be regarded as a variation of *strong threshold supermarket model* (STSM) proposed in [12] if we take γl as the threshold T in the latter. In the STSM, customers arrive at a Poison stream of rate λn (λ < 1) at n FIFO servers. Each customer chooses a server independently and uniformly at random and only makes additional choices if the previous choice is beyond a pre-determined threshold. If both choices are over the threshold, the customer queues at the shorter of its two choices. The service time for a customer is exponentially distributed with mean. A key difference between the QFM and the STSM is that the servers are homogeneous in the supermarket model but heterogeneous in the query forwarding model.

Algorithm 2 Pseudo-code for topology-aware randomized query forwarding algorithm executed by node i.

```
receive query Q with overloaded node information A
determine the set of outlinks for the query forwarding based on DHT routing algorithm.
choose options J = {j_1, j_2 ...} from the outlink set excluding overloaded node in A
if memory has a node j_a then
    randomly choose a node j_b from J
else
    randomly choose two nodes j_a and j_b from J
end if
//choose the better node from j_a and j_b
probe node j_a and j_b for load status
if j_a and j_b are heavy then
    add j_a and j_b to A
    forward Q and A to the least heavily loaded node
else
    if one node is light and one node is heavy in j_a and j_b then
        add the heavy node to A
        forward Q and A to the light node
    end if
else
    choose nodes J_log logically nearest to target ID from j_a and j_b
    choose nodes J_phy physically nearest to node i from J_log
    forward Q and A to a node in J_phy
end if
```

Along the line of analytical approach in, we analyze the performance of the query forwarding algorithm. The following theorem shows that the 2-way randomized query forwarding improves lookup efficiency exponentially over random walking.

5. Conclusion

DHT networks have an inherent congestion problem caused by query load due to the

nature of heterogeneity and dynamism of nodes. Non uniform and time-varying file popularity makes the problem more severe. This paper presents a ERT-based congestion control protocol for DHT networks, which consists of three components: indegree assignment, periodic indegree adaptation, and topology aware query forwarding. Theoretical analysis establishes the bounds of the indegree and outdegree, and proves the performance of the protocol in general in terms of both query load balance factor and query processing time.

6. References

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. ACM SIGCOMM, pp. 329-350, 2001.

[2] I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. on Networking, vol. 11, no. 1, pp. 17-32, Feb. 2003.

[3] B.Y. Zhao et al., "Tapestry: An Infrastructure for Fault-Tolerant Wide Area Location and Routing," Technical Report No. UCB/ CSD-01-1141, 2001.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. Middleware Conf., 2001.

[5] H. Shen, C. Xu, and G. Chen, "Cycloid: A Scalable Constant-Degree P2P Overlay Network," Performance Evaluation, vol. 63, no. 3, pp. 195-216, 2006.

[6] I. Clarke and O. Sandberg et al. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[7] B. Levine and C. Shields. Hordes: A multicast-based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[8] B. Godfrey and I. Stoica. Heterogeneity and load balance in distributed hash tables. In *Proc. of INFOCOM*, 2005.

[9] M. Bienkowski, M. Korzeniowski, and F. M. auf der Heide. Dynamic load balancing in distributed hash tables. In *Proc. of IPTPS*, 2005.

[10] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *Proc. of ACM SIGCOMM*, 2004.

[11] H. Shen and C. Xu. Locality-aware randomized load balancing algorithms for structured p2p networks. In *Proc. of ICPP*, pages 529–536, 2005.

[12] L. A. Adamic, B. A. Huberman, R. M. Lukose, and A. R. Puniyani. Search in power law networks. In *Physical Review E*, volume 64, pages 46135–46143, 2001.

[13] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proc. of NSDI*, 2005.

[14] I. Stoica and R. Morris et al. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 2003.

[15] H. Shen and C. Xu. Locality-aware randomized load balancing algorithms for structured p2p networks. In *Proc. of ICPP*, pages 529–536, 2005.

[16] H. Shen, C. Xu, and G. Chen. Cycloid: A scalable constant-degree p2p overlay network. *Performance Evaluation*, 63(3):195–216, 2006.

7. About the Authors

Palakollu Pavani is currently pursuing her M.Tech in Computer Science and Engineering Dept at Chaitanya Engineering College Kommadi, Visakhapatnam. Her research interests include Networking.

Mrs. Syed Shaheen is currently working as an Assistant Professor in Computer Science and Engineering Dept at Chaitanya Engineering College Kommadi, Visakhapatnam. Her research interests include Data Mining, Networking.