Elastic Routing Table with Probable Performance for Congestion Control in DHT Networks

Haiying Shen and Cheng-Zhong Xu
Department of Electrical & Computer Engineering
Wayne State University, Detroit, MI 48202
{shy,czxu}@ece.eng.wayne.edu

Abstract

Structured P2P (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 receive more traffic load. The mechanism allows each node to have a routing table of variable size corresponding to node capacities. The indegree and outdegree of the routing table are also adjusted dynamically in response to the change of file popularity and network churn. Theoretical analysis proves the routing table degree is bounded. The elastic routing table 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 elastic routing table 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 improvement in query lookup efficiency.

1. Introduction

In structured P2P overlay networks, each peer (or 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 variable-degree DHTs like CAN [28], Chord [33], Pastry [29], Tapestry [34], Kademlia [21], and constant-degree DHTs like Cycloid [32], Koorde [12] and Vicery [19].

DHT networks have an inherent load balancing problem. It is because consistent hashing produces a bound of O(log n) imbalance degree of keys 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 occurs with query overflow in which too many queries received by the node at a time or data overflow in which a too high volume of data needed to be downloaded and forwarded by the node simultaneously. Although files are often transmitted via a direct connection between source and destination, data forwarding using intermediary nodes in the query routing path is often used for the provisioning of anonymity of file sharing, as in Freenet [8], Mantis [5], Mutis [1], and Hordes [14].

In the past, many load balancing strategies have been proposed to deal with the network heterogeneity; see [33, 10, 4] for examples. It is known that a long key ID space interval has a high probability of being contacted than a short interval. The approaches share a common idea of "virtual server", in which a physical node simulates a number of virtual overlay servers so that each node is assigned ID space interval of a different length according to its capacity. The simplicity of the approaches comes at a high cost to maintain the relationship between a node's responsible interval

and its capacity. Moreover, 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", to take into account the effect of file popularity on query load [3, 31]. In the approaches, heavily load nodes probe light nodes and re-assigns 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) has the same and constant DHT size. In other words, each node maintains 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 [2]. 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 and direct query flow to light nodes.

Most recently, Li, *et al.* [16] pursued similar ideas of using non-uniform DHTs in different nodes for making a good tradeoff between lookup efficiency and maintenance cost. Castro, *et al.* [6] 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 the construction of capacity-aware DHTs. 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 DHT networks. 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.

The rest of this paper is structured as follows. Section 2 presents a concise review of representative congestion control approaches for unstructured and structured P2P systems. Section 3 shows the ERT-based protocol, focusing on initial indegree assignment and periodic adjustment. Section 4 gives the details of topology-aware randomized query forwarding policy. Section 5 shows the performance of the protocol with comparison of a variety of metrics. Section 6 concludes this paper with remarks on possible future work.

2. Related Work

There have been many load balancing algorithms to deal with node heterogeneity and network churn [33, 10, 4]. "Virtual server" [33] 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.*[10] addressed the problem by arranging a real server for virtual Id space of consecutive virtual Ids. When a node selects its virtual servers, it first picks a random starting point and then selects one random ID within each of $\Theta(\log n)$ consecutive intervals of size $\Theta(1/n)$. In [4], 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 a optimal balance with high probability.

Since a node with a longer interval has a higher probability of being contacted, the load balancing algorithms based on ID space interval assignment control traffic congestion due to the node heterogeneity in capacity. Initial key ID space partitioning is insufficient to guarantee load balance, especially in DHTs with churn. It is often complemented by dynamic load reassignment. Godfrey and Rao *et al.* proposed schemes to rearrange load between heavy nodes and light ones according to their capacities so as to avoid bottleneck [9, 27]. They assumed that query load was uniformly distributed in the ID space.

Bharambe *et al.* [3] proposed a load balancing algorithm to deal with the congestion caused by biased lookups. They defined a node's load as the number of messages routed or matched per unit time. The load balancing algorithm proceeds in a way that heavily loaded nodes probe a number of sample nodes and requests lightly loaded nodes to leave from their current locations and rejoin at the location of the heavily loaded nodes. To get the load distribution information, each node periodically samples nodes within a certain distance and maintains approximate histograms. This requires much communication and maintenance cost, especially in churn. In addition, node ID changes due to the node leave/rejoin incurs high overhead.

The idea of using irregular routing tables with respect to the node capacity has been recently pursued by Hu *et al.* [11], and Li *et al.* [16]. Their foci were on a trade-off between maintenance overhead and lookup efficiency. Hu *et al.* proposed to deploy large routing tables in high capacity nodes to exploit node heterogeneity and improve lookup efficiency. Li *et al.* designed an Accordian mechanism on Chord to vary the table size in different network scales and churn rates without compromising lookup efficiency. Castro *et al.* [6] proposed a neighbor selection algorithm to construct routing tables based on node different capacities. Its basic idea of using the node indegree 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 non-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 [26, 17, 18, 7] for recent examples. Like congestion control in DHT networks, their solutions are based on the principle of power-law networks that high degree nodes play an important role in communication. 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

In this section, we will introduce a congestion control protocol based on ERT in DHT overlays. 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 proportional 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. Node i has a capacity that it is willing to devote or able to process queries. We assume that node i's capacity c_i 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 T. We refer to node with traffic load $l_i \leq c_i$ as a *light node*; otherwise a heavy node or a 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 a 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 s_i should be kept close to 1. One way to achieve this is to measure the traffic load l_i 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 [18], 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 d_i, $1 \leq i \leq n$, as the number of inlinks of node i. With the assumption that nodes and file queries are uniformly distributed in a DHT network without churn, l_i is directly related to d_i. 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 without churn. To deal with skewed queries caused by non-uniform and time-varying file popularity and network churn, we propose an integral indegree adaptation component to dynamically adjust node indegree periodically, which results in DHT routing table change. We refer to such an elastic routing table 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, Pastry and Tapestry, and is a constant number in Cycloid, Koorde and Viceroy. The outlinks determine node indegree. 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 technical questions:

1. How to determine a node's indegree in order to make full use of its capacity and keep it light loaded at the same time?

2. How to construct ERTs with different indegrees for nodes of different capacities, and meanwhile retaining the original DHT neighbor selection functions in lookup routing?

We normalize capacity to c so that the average of c is 1; that is, $\sum_i c_i = n$. Recall that in a uniform system without churn, l_i is related to d_i directly. It follows that $s_i \approx \frac{d_i/\sum_i d_i}{c_i/\sum_i c_i}$, and $d_i \approx c_i \frac{\sum_i d_i}{n}$. Taking $\frac{\sum_i d_i}{n}$ as a constant α, we define α as *indegree per unit capacity*. It is a system parameter and is determined as a function of different metrics in system experience such as inlink query forwarding rate, query initiation rate, etc. in non-uniform system with churn. We first set node i's maximum indegree d_i^∞ as $\lfloor 0.5 + \alpha c_i \rfloor$ based on the fact that the maximum number of queries a node can process at a time depends on its capacity. The initial indegree of node i is βd_i^∞, where β is a pre-defined percentage for reservation purpose. There is a tradeoff in α determination: if α is too small, high capacity nodes cannot be fully utilized because of low indegree, while a large α makes it very possible that low capacity nodes, even high capacity nodes become heavy nodes. What's more, large α leads to extra maintenance cost for corresponding overlay connections. Therefore, it is very important to determine a suitable α.

In the following, we will explain how to build ERTs with initial setting of the indegree for each node. Like bidirectional links in Gnutela, we let each DHT node i maintain a backward outlink (*backward finger*) for each of its inlink, in order to know the nodes which forward queries to it. Consequently, a double link is maintained for each routing table neighbor. Once node i joins the system, it needs to build its routing table based on DHT protocols. In order to control each d_i below d_i^∞, we set a restriction that only nodes with available capacity $d_i^\infty - d_i \geq 1$ can be the joining node's neighbors. Each neighbor in node i's routing table creates a backward finger to node i. After building a basic routing table, node i then probes other nodes who can take it as their neighbor to achieve its initial indegree βd_i^∞. As a result, high capacity nodes produce high indegrees while low capacity nodes lead to low indegrees.

To probe nodes for indegree expansion, the IDs of those nodes should be firstly decided. The ID set can be determined in the opposite way of the original DHT neighbor selection algorithm. We will explain it later in DHT examples. After the determination of the ID set, node i sends requests targeting to some of the nodes in the set that are not in the list of its backward fingers. On receiving such a request, the node which is responsible for the ID checks if it can take node i as its routing table neighbor. If can, it adds node i into its corresponding entry in its routing table and sends back a positive reply. Once node i receives positive reply from a node, say node j, it builds a backward finger to node j. In the following, we take Cycloid, Pastry and Tapestry as DHT examples to explain indegree expansion algorithm. The algorithm can be applied to Chord and other structured overlays that have little flexibility in the selection of neighbors by relaxing their routing table neighbor constraints. Readers are referred to[30] for the details.

In Cycloid with dimension d, a node $(k, a_{d-1}a_{d-2}\ldots a_k \ldots a_0)$ $(k \neq 0)$ has one cubical neighbor $(k-1, a_{d-1}a_{d-2}\ldots \overline{a}_k xx...x)$, and two cyclic neighbors $(k-1, b_{d-1}b_{d-2}\ldots b_0)$ and $(k-1, c_{d-1}c_{d-2}\ldots c_0)$ in its routing table. By the opposite way of neighbor selection, a node $(k-1, a_{d-1}a_{d-2}\ldots a_k \ldots a_0)$ can send requests targeting to $(k+1, a_{d-1}a_{d-2}\ldots \overline{a}_k xx...x)$ to ask nodes to take it as their cubical neighbors, and also it can send requests targeting to $(k+1, a_{d-1}a_{d-2}\ldots a_k xx...x)$ to ask nodes to take it as their cyclic neighbors. For instance, node i (3,101-0-0000) can probe $(4, 101-1-xxxx)$ to increase its indegree. Let's say, node i first sends a request targeting (4,101-1-0000). Assuming node j receives the request, if $j \in (4, 101-1-xxxx)$, j adds i as its cubical neighbor and node i build a backward finger to j. Let's assume that node i needs to increase its indegree to 10, but it is only 6 after cubical backward finger probing. Then node i probe cyclic backward finger for the rest 4 indegree. Algorithm 1 shows the pseudocode of indegree expansion algorithm in Cycloid. In the pseudocode, we represent represent $a_{d-1}a_{d-2}\ldots a_k \ldots a_0$ in node ID $(k, a_{d-1}a_{d-2}\ldots a_k \ldots a_0)$ as a_{id}.

Pastry's routing table is organized into $\lceil \log_{c^b} n \rceil$ (b is a configuration parameter) row with $2^b - 1$ entries each. An entry at row m of node i's routing table refers to a node whose ID shares the node i's ID in the first m digits, but whose $(m+1)^{th}$ digit is not the $(m+1)^{th}$ digit in node i's ID. For example, node (10233102) can have nodes with ID $(10xxxxxx)$ at its row 2. Tapestry's routing table neighbor selection algorithm is similar to Pastry's. Since each entry has multiple choices, node i $(a_{d-1}a_{d-2}\ldots a_{k-1}a_k \ldots a_0)$ can send request targeting to $(a_{d-1}a_{d-2}\ldots a_{k-1}\overline{a}_k x \ldots x)$ to ask nodes to take it as their k^{th} row entry.

The initial indegree assignment algorithm proceeds recursively. We prove the ERT indegree resulted from the algorithm is bounded. We assume a DHT that manages a unit-size ID space, i.e., $[0, 1) \subseteq \mathbb{R}$ employing arithmetic modulo 1, and the DHT uses consisting hash [13] 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 c_i and the network scale n within a factor of γ_c and γ_n, respectively, of the true values, with high probability[1]; readers are referred to [20, 25] for details of such an estimation process. We denote $\tilde{n}$ as estimated n and $\tilde{c_i}$ as estimated c.

1 An event happens with high probability (w.h.p.) when it occurs with probability $1 - O(n^{-1})$.

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

1: *//probe backward fingers of cubical neighbor*
2: figure out a set of cubical neighbor inlinks $ID = (k+1, a_{d-1}a_{d-2}\dots\overline{a}_k xx\cdots x)$
3: $id = (k+1, a_{d-1}a_{d-2}\dots\overline{a}_k 00\cdots 0)$
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 $ID = (k+1, a_{d-1}a_{d-2}\dots a_k xx\cdots x)$
13: $id = (k+1, a_{d-1}a_{d-2}\dots a_k 00\cdots 0)$
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**

Theorem 3.1 *The initial indegree assigned to a node i is between $\alpha c_i/\gamma_c - O(1)$ and $\alpha c_i\gamma_c + O(1)$ w.h.p.*

Proof With γ_c and γ_n as the maximum error factor of a node's estimated capacity and n, $\tilde{c}_i$ is within the factor $\gamma_c\gamma_u$ of c_i and $\tilde{n}$ is within the factor γ_n of n w.h.p. As a result, the indegree first assigned to node i is at most $\lfloor 0.5 + \tilde{c}_i\alpha(\tilde{n})\rfloor \leq \tilde{c}_i\alpha(\tilde{n}) + O(1) \leq \gamma_c c_i\alpha(\gamma_n n) + O(1) \leq \gamma_c c_i\alpha(n) + O(1)$. The indegree first assigned to node i is at least $\lfloor \tilde{c}_i\gamma(\tilde{n}) - 0.5\rfloor \geq \tilde{c}_i\alpha(\tilde{n}) - O(1) \geq c_i/\gamma_c\alpha(n/\gamma_n) - O(1) \geq c_i/\gamma_c\alpha(n) - O(1)$. ∎

3.3. Periodic Indegree Adaptation

In practice, nodes join and leave DHT overlays continuously and the files in the system may have non-uniform and time-varying popularity. Considering the fact that query load often vary with time, the initial indegree assignment is not robust enough to limit a node's query load under its capacity. To ensure that queries flow toward nodes with sufficient 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 indegree periodic 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 l_i over T periodically and checks whether it is overloaded or lightly loaded by a factor of γ_l; *i.e.* whether $g_i = l_i/c_i > \gamma_l$ or $< 1/\gamma_l$. In the former case, it decreases $\mu(l_i - c_i)$ 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 case with the same logical distance, it chooses the one with longest physical distance. In the latter case, it increases $\mu(c_i - l_i)$ 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 theorem shows that the ERT indegree in the process of adapation remains bounded.

Theorem 3.2 *With indegree assignment and periodic adaptation algorithm, a node i has an inegree between $\frac{c_i}{\gamma_c\gamma_l\nu_{max}}$ and $\frac{c_i\gamma_c\gamma_l}{\nu_{min}}$ where ν_{max} and ν_{min} represent the maximum and minimum incoming query rate per inlink in the system, respectively. And its indegree change is bounded in each adaptation.*

Proof Node i does not need to update its indegree when $\tilde{c}_i\gamma_l \geq l_i \geq \tilde{c}_i/\gamma_l$, where $\frac{c_i}{\gamma_c} \leq \tilde{c}_i \leq \gamma_c c_i$. Assume during a certain time period T, the average incoming query rate per inlink of node i is ν_i; that is, on average, there are ν_i queries coming from each inlink during time T. Assume that node i has degree d_i at a certain time point during T. Such that, $l_i = \nu_i d_i$.

When $l_i > \tilde{c}_i\gamma_l$, node i updates its indegree to $d_i - \mu(l_i - \tilde{c}_i)$. Since $l_i > \tilde{c}_i\gamma_l$, $\nu_i d_i > \tilde{c}_i\gamma_l$, the indegree is at most $d_i - \mu\tilde{c}_i(\gamma_l - 1)$, $d_i - \mu\frac{c_i}{\gamma_c}(\gamma_l - 1)$. It is $d_i - \mu(\nu_i d_i - \tilde{c}_i) \geq (1-\mu)\nu_i d_i + \frac{\mu c_i}{\gamma_c}$. Consequently, the indegree is decreased to between $(1-\mu)\nu_i d_i + \frac{\mu c_i}{\gamma_c}$and $d_i - \mu\frac{c_i}{\gamma_c}(\gamma_l - 1)$. On the other hand, when $l_i < \frac{c_i}{\gamma_l}$, node i updates its indegree to $d_i + \mu(\tilde{c}_i - l_i)$. Since $l_i < \frac{\tilde{c}_i}{\gamma_l}$, $\nu_i d_i < \frac{\tilde{c}_i}{\gamma_l}$, the indegree is at least $d_i + \mu\tilde{c}_i(1 - \frac{1}{\gamma_l})$, $d_i + \mu\frac{c_i}{\gamma_c}(1 - \frac{1}{\gamma_l})$. It is $d_i + \mu(\tilde{c}_i - \nu_i d_i) \leq (1 - \mu\nu_i)d_i + \mu c_i\gamma_c$. Therefore, the indegree is increaseed to between $d_i + \mu\frac{c_i}{\gamma_c}(1 - \frac{1}{\gamma_l})$ and $(1 - \mu\nu_i)d_i + \nu c_i\gamma_c$. The indegree is changed until it reach a status that $\tilde{c}_i\gamma_l \geq \nu_i d_i \geq \tilde{c}_i/\gamma_l$, $\frac{c_i\gamma_c\gamma_l}{\nu_{min}} \geq d_i \geq \frac{c_i}{\gamma_c\gamma_l\nu_{max}}$. ∎

For example, in a network of size 2048, if a node's capacity is 50 and its average incoming query rate is 0.5, its indegree bounded by 100 in the case $\gamma_l = 1$.

The following theorem shows the outdegree of an ERT is bounded as well. We leave its proof in Appendix.

Theorem 3.3 *A Cycloid node has an outdegree of at most $\frac{2\gamma_c\gamma_l c_{max}}{\nu_{min}} - O(\frac{2^d}{d}) + O(1)$ w.h.p. where d is the DHT dimension.*

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, one 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. For example, node i receives a query with key (2,1010-0011) and the query should be forwarded to a cubical neighbor according to Cycloid routing algorithm. It first randomly choose two options (3,1010-0010), (3,1010-0001) among the three cubical neighbors if $b = 2$.

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 [31] for a landmarking 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. Is there any method that can help 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 [23] 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.* [24] improved the performance of two-choice method 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 node 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 query 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. For example, a lookup node has three options for forwarding a query: i_1, i_2 and i_3. Using the above method, it forwards the query to i_2 with information of overloaded node i_1. In the second step, node i_2 has three options: i_1, j_1 and j_2. Because it knows that i_1 is overloaded, it will not select i_1 as a option for query forwarding. 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 γ_l, another specific 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 [22, 23] if we take γ_l as the threshold T in the latter. In the STSM, customers arrive at a Poison stream of rate λn ($\lambda < 1$) at n FIFO servers. Each customer chooses a server independently and uniformly at random and only makes

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
```

additional choices if the previous choice is beyond a predetermined 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 1. A key difference between the query forwarding model and the supermarket model is that the servers are homogeneous in the supermarket model but heterogeneous in the query forwarding model.

Along the line of analytical approach in [22, 23], 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. Readers are referred to appendix for the theorem proof.

Theorem 4.1 *For any fixed time spot T, the time a query waits before being forwarded during the time interval $[0, T]$ is bounded and b-way $(b \geq 2)$ forwarding yields an exponential improvement in the expected time for a query queuing in a server.*

5. Performance Evaluation

This section demonstrates the distinguished peoperties of the ERT-based congestion control protocol through simulation built on an $O(1)$-degree Cycloid network. Simulations on other $O(\log n)$-degree networks are expected to produce better results. We assumed a bounded Pareto distribution for the capacity of nodes. This distribution reflects real world situations where machines' capacities vary by different orders of magnitude. The queries are consecutively generated with a random source node and a random target key, unless otherwise noted. Table 1 lists the parameters of the simulation and their default values.

Table 1: Simulated environment and algorithm parameters.

Environment Parameter	Default value
Cycloid dimension d	8
Number of nodes n	Fixed at 2048
Node capacity c	Bounded Pareto: shape 2 lower bound: 500 upper bound: 50000
Query/lookup number	3000
Overload threshold γ_l	1
Indegree adaption constant μ	$1/2$
Indegree adaption period	1 second
Indegree per normalized capacity α	dimension d+3
Query process time in light nodes	0.2 second
Query process time in heavy nodes	1 second

We evaluate the effectiveness of the congestion control protocol in the following metrics:

- Congestion rate of a node i, as defined by $g_i = l_i/c_i$. Ideally, the congestion rate should be kept around 1, implying the node is neither overloaded nor under utilized, and its capacity is fully utilized. We use the metric of *the 99th percentile maximum congestion* to measure the network congestion, and use the metric of *the 99th percentile congestion of minimum capacity node* to reflect the node utilization.
- Query distribution share s_i. Recall that share $s_i = \frac{l_i/\sum l_i}{c_i/\sum c_i}$. It represents the performance of fair load distribution, *i.e.* the total system load is distributed among nodes based on their capacity. The objective of fair sharing is hard to achieve in DHT networks because of a number of reasons. First, it is hard to collect the load and capacity of other nodes. Second, DHT is a dynamic system with continuous node joins and departures, as well as continuous query initialization. It is hard to control instant share of each node in such a dynamic situation. Third, since query load is not uniformly distributed among the nodes, and it changes with file popularity and churn. Although fair sharing is not the objective of congestion control, we use the metric of *the 99th percentile share* to show how it can be approximated by the control of indegrees.
- Query processing time. It is determined by two factors: lookup path length and the number of heavy nodes encountered in each path. The metric of path length reflects the performance of the query forwarding algorithm, and the metric of number of heavy nodes in each path shows how the congestion control protocol avoids

heavy nodes in direct traffic flow in order to reduce lookup latency.

We conducted experiments on Cycloid networks without congest control (Base) and with ERT-based congestion control (ERT). For comparison, we also include the results due to a "virtual server" load balancing method [10] (VS) and a neighbor selection algorithm for indegree control (NS) [6]. The NS algorithm bears resemblance to the ERT initial indegree assignment as to select neighbor based on node indegree bound. However, NS always selects high capacity nodes as neighbors. It may over-compromise the needs of low capacity nodes. ERT makes full use of node capacity by letting nodes reach their indegree bounds. Moreover, ERT allows dynamic indegree adaption (A) and facilitates query forwarding (F) to deal with network churn and skewed lookups. We represent the congestion control in different combinations by ERT/A, ERT/F, and ERT/AF, respectively.

We measured their performance as functions of total lookup number and query processing speed at each node. We varied lookup number from 1000 to 5000, with 1000 increase in each step. We also varied the processing time of a query in a light node from 0.1 second to 2.1 second and 5 times of that in a heavy node. The total query load increases in both cases and we observed similar results in simulation. Because of space constraints, we present the results due to the change of total query number.

5.1. Congestion Control Efficiency

We measured each node's maximum congestion during all test cases and calculated the 99th percentile maximum node congestion. Figure 1(a) shows the congestion rate due to each method increases as more lookup queries arrive. The NS protocol produces a higher 99th percentile maximum congestion rate than Base. It implies that a heavy node in NS has much more load corresponding to its capacity than a heavy node in Base. This is expected because NS strongly biases high capacity nodes as routing table neighbors. The high capacity nodes may turn out to be overloaded.

In contrast, VS and ERT/AF lead to much lower congestion rates. That is, they demonstrate effectiveness in controlling the load of each node based on its capacity. Although ERT/A and ERA/F do not perform as well as VS, the combined effect of adaptation and forwarding makes ERT/AF outperform VS.

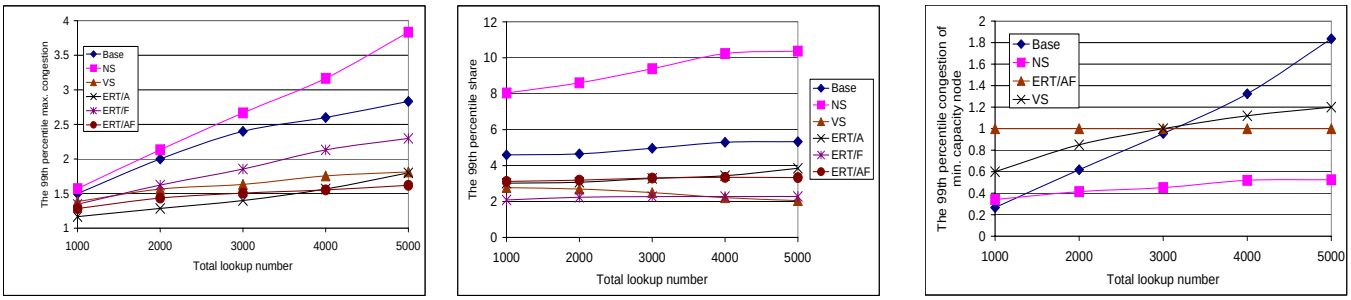
The relative performance between NS, VS, and ERT/AF with respect to Figure 1(a) can be verified by the 99th percentile congestion rate of minimum capacity node in Figure 1(c). It is expected to see that the low capacity node becomes congested as the query load increases. Without congestion control (Base), the congestion rate increases sharply. The congestion control protocols delay the occurrence of congestion. In particular, the NS protocol over-protected low capacity nodes due to its high capacity-biased neighbor selection policy. In comparison, ERT/AF keeps low capacity nodes fully utilized, without driving them into overloaded states.

Figure 1(b) shows the 99th percentile node share. We can see that NS generates a much higher share rate, in comparison with the other protocols for the same reason of the observations in Figure 1(a) and (c). That is, NS heavily relies on high capacity nodes for query routing. Excluding low capacity nodes in neighbor selection may lead to a waste of system resources because their capacities can be used to ease the burden of high capacity nodes in certain situations. In contrast, VS and ERT/AF do not have this preference in neighbor selection and they achieve good query load sharing between heterogeneous nodes. The small gain of VS is due to its fine grained ID space partition between virtual servers. An ideal share in DHT is difficult to achieve because of the DHT strict controlled topology, routing algorithm, non-uniform and variable file popularity, and churn. VS approximates fair sharing by static ID space assignment. However, it is at the cost of more maintenance overhead and lookup cost. It cannot handle skewed lookups either.

5.2. Lookup Efficiency

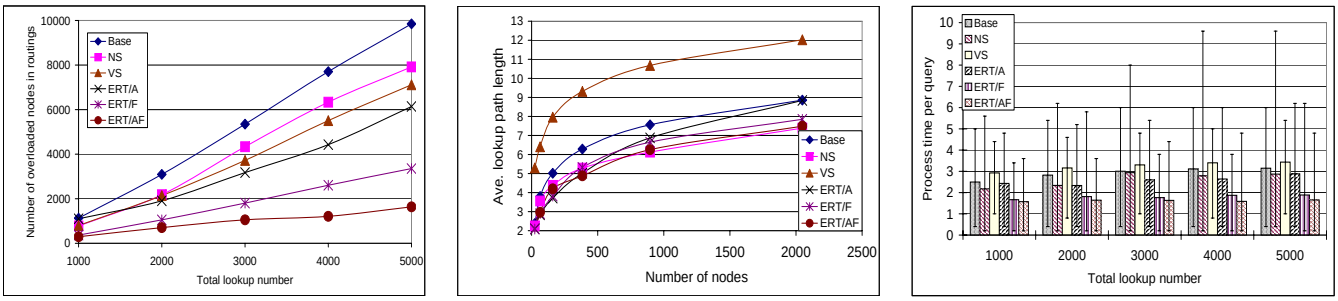
Lookup latency is determined by two factors: lookup path length and query processing time in each node along the path. Figure 2(a) shows the total number of overloaded nodes encountered in query routings grows with the query load. It also shows that ERT/AF leads to much high lookup efficiency in comparison with the others. Though NS and VS improve over Base to a certain extent, there remain a large percentage of congested nodes in the systems in comparison with ERT/AF. NS biases high capacity nodes for query load, which may make them more likely overloaded as the system query load increases. Due to DHT's strictly controlled topology and precise lookup algorithm, the assumption of uniformly distributed load of VS does not hold true. The fixed outdegree of nodes in NS and VS prevents each node from adapting traffic load on nodes elastically. In contrast, ERT/AF enables each node match its indegree to its capacity, and adapts its indegree in response to the change of its experienced query load. Furthermore, the query forwarding operation helps avoid overloaded nodes during query routing. Less overloaded nodes in routing leads to more efficient lookup.

Figure 2(b) shows the path lengths due to different congestion protocols as the network size increases. It is expected that VS leads to a much longer query path length than Base because of the additional virtual server layer in routing. This is consistent with the observation in [10] that VS achieves the objective of load balancing at the cost of lookup efficiency and the path length increases by at most an additive



(a) Maximum congestion (b) Share (c) Congestion of minimum capacity node

Figure 1: Effectiveness of congestion controls.



(a) Heavy nodes in routings (b) Lookup path length (c) Lookup time

Figure 2: Effectiveness of congestion control protocols on lookup efficiency.

constant. ERT/AF takes distance into account in query forwarding. Likewise, NS considers node distance in neighbor selection. Both of them reduce path length of Base significantly.

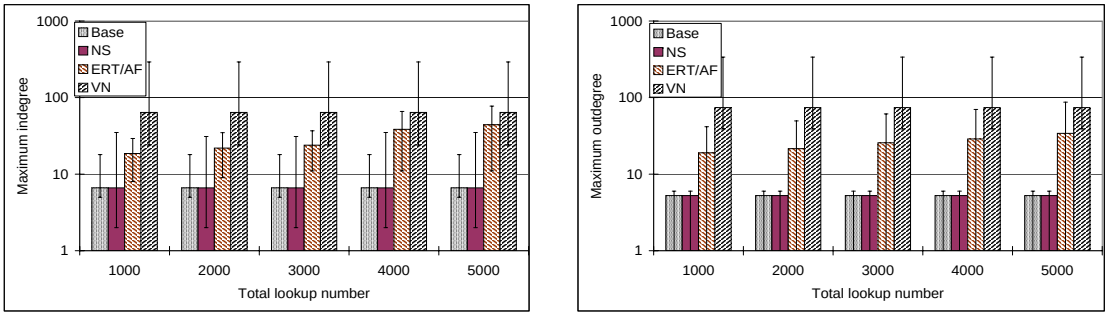
Figure 2(c) shows the average, 1st and 99th percentiles of processing time per query as combined effect of reduced congested nodes and lookup path length on the overall query processing time. Although VS reduces the number of congested nodes of Base, its benefits may be outweighed by its extended path length. Without dynamic congestion control, Base and NS may forward queries to congested nodes. Static indegree assignment by NS only results in marginal processing time reduction. On the contrary, ERT/AF tends to direct queries to light nodes, which have sufficient capacity to handle them promptly.

5.3. Maintenance Cost

Recall that ERT-based congestion control protocol achieves its goal using elastic routing tables to adapt each node indegree to its load. In addition to maintain the tables, each node needs to maintain a list of backward fingers (the same number of its indegree). We measured the maximum indegree and outdegree of each node, and calculated the average, 1st and 99th percentiles of those values in each method. As we mentioned that the routing table size in NS and VS is fixed, but the node degree in ERT is variable. We use maximum indegree and outdegree instead of average for the evaluation of the management overhead of ERT in the worse case. Figure 3(a) and (b) plot the results. Though inlinks in Base and VS don't need to be maintained, we include their degrees for comparison. As expected, the figures show that the indegree and ourdegree rates of Base, NS and VS do not change, while the rates of ERT/AF change as total query load changes. Because ERT tunes node indegees to adapt to different query load accordingly, some light loaded node indegrees reach a high value to request more load. Indegree change leads to outdegree change. The indegree and outdegree of VS are much higher than others. The reason is virtual node usage leads to larger overlay size. Our results turn out that the combination of the average, 1st and 99th percentiles indegree and outdegree of ERT in the worse case is much less than the outdegree rates of VS, respectively. Thus, to achieve congestion control, VS needs much higher cost for maintenance, while ERT only needs a little extra maintenance cost.

5.4. Effect of Skewed Lookup

Besides node heterogeneity in capacity, query load imbalance occurs with non-uniform and time-varying file popularity and peer interest variation. In this section we consider the effect of skewed lookups.



(a) Indegree

(b) Outdegree

Figure 3: Degrees of routing tables in different congestion control protocols.

We consider an "impulse" of 100 nodes whose IDs are distributed over a contiguous interval of the ID space, and whose interest are in the same 50 keys randomly chosen from the ID space. We varied the query process rate from 0.1 to 2.2 second per query on a light node, with 0.5 second increase in each step. Figure 4(a) and (b) plot the query processing time of each method. It is surprising to see that the overloaded node number and the process time per query of VS is much more than Base. As claimed by the authors in [10] that a good balance of VS is guaranteed only under the uniform load assumption; this explains why VS has poor performance in skewed lookups. In VS, a real node selects IDs of its virtual nodes randomly within consecutive intervals. When query load concentrates on a certain ID space interval, the load is allocated to consecutive virtual servers. Since most of the virtual servers may reside on the same real node, the node more likely becomes overloaded. In contrast, by assigning and adjusting node indegree based on load dynamically, combined with topology-aware randomized forwarding algorithm, ERT/FA can handle skewed lookups caused by the change of file popularity and node interests. NS yields a similar lookup latency to Base on average, but exhibits a large variance.

Figure 4(c) plots the 99th share of each method. By comparing it with Figure 1(b), we can observe that the share rate of each method is higher in skewed lookups. It is expected because the query load concentrates on certain ID space part, then certain nodes. The share rate of NS is still much higher than others in skewed lookups because of its strong bias toward high capacity notes in neighbor selection. It is a resource waste to let low capacity nodes idle.

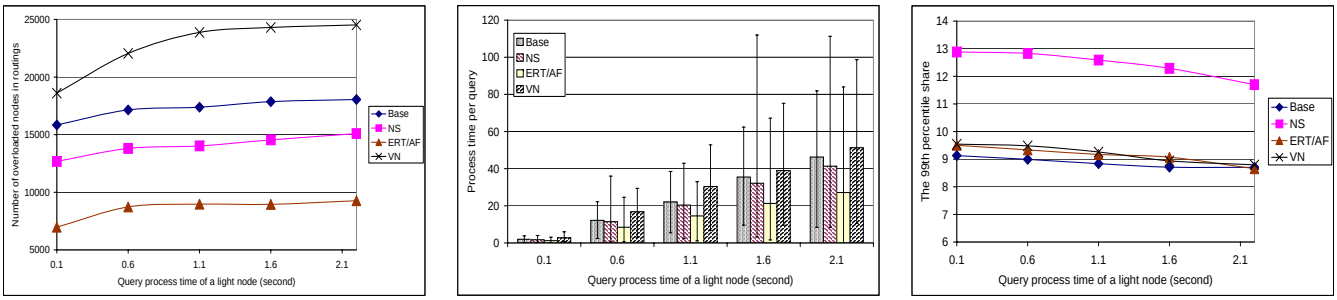
5.5. Effect of Churn

In DHT networks with churn, a great number of nodes join, leave and fail continually and rapidly, leading to continuous change of node indegree and outdegree, and overlay topology. This gives another challenge to congestion control. If a node leaves, its query load will be allocated to some other nodes. It is necessary to ensure that these nodes are light loaded. When a node joins the system, it is important to make full use of its capacity.

This section evaluates ERT/FA's adaptability to different levels of churn. In this experiment, the lookup rate was modelled by a Poisson process with a rate of 1; that is, there was a lookup every 1 second. The node join/departure rate was also modelled by a Poisson process. We ranged node interarrival/interdeparture time from 0.1 to 0.9 seconds, with 0.1 second increment in each step. Lower time corresponds to higher churn. Our results are collected from all node including the current nodes in the system when all lookups finish and the nodes departed.

Figure 5(a) shows the 99th percentile maximum congestion of each method. Comparing it with Figure 1(a), we find that the rate of each method in churn is lower than without churn at the point of 3000 lookups. It is because with continuous nodes join, the same query load is distributed among more nodes than in static DHT. The rates of NS and Base grow inversely proportional to node interarrival time, and the rates of VS and ERT/FA maintain constant. When node interarrival/interdeparture time is 0.1, the rate of NS is higher than Base's, and it decreases slightly blow the Base's when node interarrival time is 0.3-0.5. This implies that NS has difficulty to cope with high churn. Recall that in NS, high capacity nodes have denser inlinks. In high churn, some high capacity nodes may don't have enough capacity for a sudden query flow, which originally should be responsible by nodes departed. In a modest churn, the flow is not so intense for nodes to handle. In high churn, VS has marginally less rate than Base, which implies that VS can deal with churn to a certain extend. We can also see that ERT/FA keeps the rate close to 1 in different levels of churn. in controlling node congestion in churn.

Figure 5(b) shows the 99th percentile share of each method. It demonstrates that like in static DHT, NS performs not so well as others in fair load balance in churn. The 99th percentile share of ERT/FA in churn is higher than that without churn. It is because continuous node joins and departures induce more load on some nodes relative to their capacity. On the other hand, because of churn, NS's 99th



(a) Heavy nodes in routings (b) Lookup time (c) Share

Figure 4: Effectiveness of congestion control protocols in skewed lookups.

percentile share is higher than Base, and VS has more balanced query load distribution than others.

Figure 6(a) shows the heavy node number in routings of each method in churn. We can see that the number of NS is much higher than Base in high churn, and the number decreases as the node interarrival time increases; both of them are larger than the result of ERT/AF. This observation is consistent to the findings in Figure 5(a). It confirms that ERT/FA performs the best in reducing heavy nodes processing query. Figure 6(b) shows the lookup path length of each method. Comparing it with Figure 2(b), we can detect that there's no big difference, except that ERT/FA has less path length in churn. We also recorded average timeout for each method. A timeout occurs when a node tries to contact a departed node. The number of timeouts experienced by a lookup is equal to the number of departed nodes encountered. The average timeout of ERT/FA is 0, and less than 0.06 in other approaches. The reason for shorter path lengths and less timeout of ERT/FA is that its ERT avoids timeouts by letting each node have multiple neighbors in each table entry. Consequently, when a entry neighbor left, others can be used as a substitute instead of making a detour routing. Figure 6(c) shows the average, 1st and 99th percentiles of query processing time per node of each method. They are consistent to those without churn in Figure 2(c), except that NS yields higher latency than Base in high churn. It validates the conclusion that NS is not efficient in coping with high churn.

5.6. Effect of Adaption and Query Forwarding

To evaluate the quality of the indegree adaption and topology-aware randomized query forwarding, we compare ERT/AF with algorithms without indegree adaption (ERT/F) or without query forwarding algorithm (ERT/A).

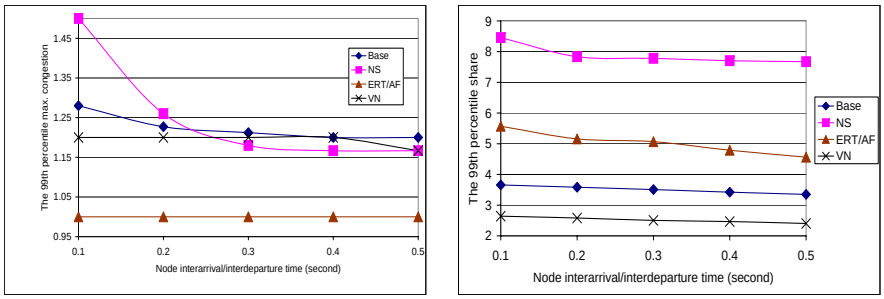
Figure 1(a) plots the 99th percentile maximum congestion rates of different versions. From the figure, we can observe that the topology-aware 2-way randomized forwarding algorithm is effective in reducing the congestion rate of Base when query load is not high, but becomes not so effective when the system is highly loaded. For this reason, it is imperative to have a complementary method to guarantee low node congestion. The figure shows that our indegree adaptation algorithm reduces the congestion rate significantly in various load conditions.

The 99th percentile shares in Figure 1(b) confirm the superior performance of forwarding. The query forwarding algorithm controls query flow to light nodes, ensuring that queries are forwarded only to nodes with sufficient capacity to handle them. Indegree adaptation also helps for fair load balancing, though the improvement is not so much as forwarding.

Figure 2(a) shows the heavy node number encountered in each lookup path. From the figure, we can observe that both forwarding and indegree adaptation greatly help eliminate heavy nodes. Their combination demonstrates an accumulated effect. Figure 2(b) plots the path length of each version. Recall that topology-aware randomized query forwarding algorithm takes node logical distance and physically distance into account in routing. It is expected that forwarding leads to a short lookup path. In contrast, indegree adaptation has no effect in path length. Oveall, the combined effect on overloaded nodes reduction and lookup path length shortening results in a great saving of lookup latency, as shown in Figure 2(c).

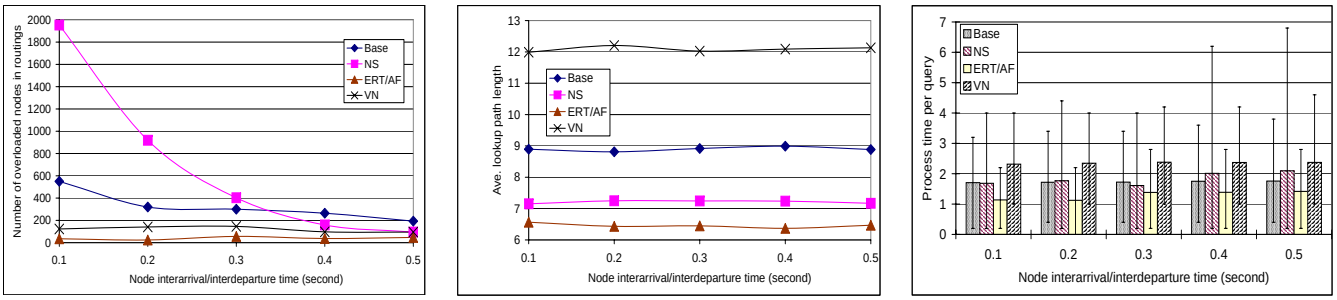
6. Conclusions

DHT networks have an inherent congestion problem caused by query load due to the nature of heterogeneity and dynamism of network nodes. There are DHT networks that use virtual servers to partition the ID space among nodes proportional to node capacities to achieve congestion control. However, their assumption of equal file popularity does not hold true in practice, and their overlay maintenance cost is high in churn. Other methods achieve congestion control by static mapping between node indegree and capacity, as well as biasing high capacity nodes for overlay neighbors. Static mapping cannot deal with non-uniform and time-varying file popularity and churn. Such bias makes it most likely that high capacity nodes become bottle-



(a) Maximum congestion (b) Share

Figure 5: Effectiveness of congestion control protocols in networks with churn.



(a) Heavy nodes in routings (b) Lookup path length (c) Lookup time

Figure 6: Effectiveness of congestion control protocols on lookup efficiency in churn.

necks. 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.

Simulation results show the superiority of the congestion control protocol compared with other methods in static network, skewed lookups and in churn, and show the effectiveness of each algorithm in the protocol. It makes full use of each node's capacity while control each node' load below its capacity. It improves the lookup efficiency in DHT network by reducing lookup latency.

References

[1] Mute. http://mute-net.sourceforge.net/.

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

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

[4] M. Bienkowski, M. Korzeniowski, and F. M. auf der Heide. Dynamic load balancing in distributed hash tables. In *Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[5] S. Bono, C. Soghoian, and F. Monrose. Mantis: A lightweight, server-anonymity preserving, searchable P2P network. Technical report, TR-2004-01-B, Information Security Institute, Johns Hopkins University, 2004.

[6] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proc. of the 2nd Symposium on Networked Systems Desing and Implementation (NSDI)*, 2005.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *Proc. of ACM SIGCOMM*, 2003.

[8] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[9] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2004.

[10] P. Brighten Godfrey and I. Stoica. Heterogeneity and load balance in distributed hash tables. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2005.

[11] J. Hu, M. Li, W. Zheng, D. Wang, N. Ning, and H. Dong. Smartboa: Constructing p2p overlay network in the heterogeneous internet using irregular routing tables. In *Proc. of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2004.

[12] M. F. Kaashoek and R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[13] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and Panigrahy R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 1997.

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

[15] D. Lewin. Consistent hashing and random trees: Algorithms for caching in distributed networks. masters thesis. Technical report, Department of EECS, MIT, 1998. Available at the MIT Library, http://thesis.mit.edu/.

[16] J. Li, J. Stribling, R. Morris, and F. Kaashoek. Bandwidth efficient management of dht routing tables. In *Proc. of the 2nd Symposium on Networked Systems Desing and Implementation (NSDI)*, 2005.

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of ACM International Conference on Supercomputing (ICS)*, 2001.

[18] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make gnutella scalable? In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[19] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *Proc. of the 21st ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.

[20] G. Manku. Balanced binary trees for id management and load balance in distributed hash tables. In *Proc. of the 23th ACM Symposium on Principles of Distributed Computing (PODC)*, 2004.

[21] P. Maymounkov and D. Mazires. Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[22] M. Mitzenmacher. The power of two choices in randomized load balancing, 1996. PhD thesis, University of California, Berkeley.

[23] M. Mitzenmacher. On the analysis of randomized load balancing schemes. In *Proc. of the 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 292–301, 1997.

[24] M. Mitzenmacher, B. Prabhakar, and D. Shah. Load balancing with memory. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[25] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004.

[26] S. Osokine. The flow control algorithm for the distributed broadcast-route networks with reliable transport links. Technical report, 2001. http://www.grouter.net/gnutella/flowcntl.htm/.

[27] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in structured p2p systems. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 329–350, 2001.

[29] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[30] H. Shen and C. Xu. Elastic routing table for congestion control in chord. Technical report, Electrical and Computer Engineering Department, Wayne State University, 2005. Under prepare.

[31] H. Shen and C. Xu. Locality-aware randomized load balancing algorithms for structured p2p networks. In *Proc. of international conference on Parallel Processing*, 2005.

[32] H. Shen, C. Xu, and G. Chen. Cycloid: A scalable constant-degree p2p overlay network. *Performance Evaluation*, 2005. An early version appeared in Proc. of International Parallel and Distributed Processing Symposium (IPDPS), 2004.

[33] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 1(1):17–32, 2003.

[34] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: An Infrastructure for Fault-tolerant wide-area location and routing. *IEEE Journal on Selected Areas in Communications*, 12(1):41–53, 2004.

A. Proof of a Bounded Outdegree of ERTs

Theorem A.1 *A node has an outdegree of at most* $\frac{2\gamma_c\gamma_l c_{max}}{\nu_{min}} - O(\frac{2^{d'}}{d'}) + O(1)$ *w.h.p. where* d' *is the DHT dimension.*

Proof We use Cycloid DHT as an example. Same proving process can be applied to any other DHT. A node i with cyclic index k_i can have at most 2^{k_i} IDs for routing table entry selection, we use $\mathcal{O}_i$ to represent a set those IDs, and it can have at most 2^{k_i} IDs for routing table entry backward finger selection, we use $\mathcal{I}_i$ to represent a set of those IDs. Because there are totally $d' \cdot 2^{d'}$ IDs and n nodes in a Cycloid system with dimension d', the probability of node number for a ID is $\frac{n}{d'2^{d'}}$, denoted by η. Consequently, the node number in 2^{k_i} IDs is proximately $\eta 2^{k_i}$, denoted by J_i. We define average ID space responsible by a node as I, and use I_i to represent the I of node i.

Let X_i be the indicator variable for the event that node i probes node j for indegree expansion. If j is chosen by i, then i has a backward finger to node j, which increments j's outdegree with a outlink to i. Our purpose is to find out the upper bound of $\sum_i X$ represented by X. We assume that Cycloid nodes also probe their successors and pre-

decessors for indegree expansion. Recall that no node appears more than once in another node's routing table or successor and predecessor lists in DHT networks. The probability that node j is chosen to have an outlink pointing to i is $\frac{1}{J_i}$. The probability that node j is within $a/2$ number of i's successors or predecessors is $aI + 2I_j$. Let's assume that node i probes m_i IDs. Then, the probability is expressed as $P(|ID_j - ID_i| \leq \max\{0, m_i - J_i\}I) = \max\{0, m_i - J_i\}I + 2I_j \cdot P(m_i > J_i)$, where P represents probability.

In the case that $i \in \mathcal{O}_j$, when $J_i \geq m_i$, $E[X_i] = m_i\frac{1}{J_i}$; when $J_i < m_i$, $E[X_i] = J_i\frac{1}{J_i} + (m_i - J_i)I + 2I_j$. In the case that $i \notin \mathcal{O}_j$, when $J_i \geq m_i$, $E[X_i] = 0$ and when $J_i < m_i$, $E[X_i] = (m_i - J_i)I + 2I_j$.

Therefore, $E[X_i] =$

$$\begin{cases} min\{J_i, m_i\}\frac{1}{J_i} + \max\{0, m_i - J_i\}I + 2I_jP(m_i > J_i) & i \in \mathcal{O}_j; \\ \max\{0, m_i - J_i\}I + 2I_jP(m_i > J_i) & i \notin \mathcal{O}_j \wedge i \neq j. \end{cases}$$

Thus,

$E[X] = \sum_{i\in R} E[X_i]$

$= \sum_{i\in\mathcal{O}_j}(min\{J_i, m_i\}\frac{1}{J_i} + \max\{0, m_i - J_i\}I + 2I_jP(m_i > J_i)) + \sum_{i\notin\mathcal{O}_j\wedge i\neq j}(\max\{0, m_i - J_i\}I + 2I_jP(m_i > J_i))$

$= \sum_{i\in\mathcal{O}_j}(min\{J_i, m_i\}\frac{1}{J_i})$ (1)

$+ \sum_{i\in R\wedge i\neq j}(\max\{0, m_i - J_i\}I + 2I_j \cdot P(m_i > J_i))$ (2).

In the following, we will determine the upper bound for equation (1) and (2) respectively.

$\sum_{i\in\mathcal{O}_j}(min\{J_i, m_i\}\frac{1}{J_i})$ (1)

$= \sum_{i\in\mathcal{O}_j}(min\{J_i, m_i\}\frac{1}{\eta 2^{k_i}})$,$(J_i = \eta 2^{k_i})$

$= \sum_{i\in\mathcal{O}_j}(min\{J_i, m_i\}\frac{1}{\eta 2^{k_{j-1}}})$, $(k_i = k_{j-1})$

$= \frac{1}{\eta 2^{k_j-1}}\sum_{i\in\mathcal{O}_j} min\{J_i, m_i\}$.

By theorem 3.2, $\max_{i\in\mathcal{O}_j}\mathcal{I}_i = \max_{i\in\mathcal{O}_j}\frac{c_i\gamma_c\gamma_l}{\nu_{min}}$ for nodes $i \in \mathcal{O}_j$, Hence the above term:

$\leq \frac{1}{\eta 2^{k_j-1}}\eta 2^{k_j}\max_{i\in\mathcal{O}_j}\mathcal{I}_i$

$= 2\max_{i\in\mathcal{O}_j}\frac{c_i\gamma_c\gamma_l}{\nu_{min}}$

$\leq \frac{2\gamma_c\gamma_l c_{max}}{\nu_{min}}$

Next we will find out the upper bond for the equation (2). We define I_{max} as the longest ID space interval responsible by a node. Average spacing for a node is $1/n$ and the bound of imbalance of space interval is $2^{-O(\log n)} = O(1/n)$ [13, 15], thus the longest space interval of a node is $1/n + O(1/n)$. Considering the estimation error factor, $I_{max} = \gamma_n/n + \gamma_n O(1/n)$ w.h.p.

$\sum_{i\in R\wedge i\neq j}(max\{0, m_i - J_i\}I + 2I_j \cdot P(m_i > J_i))$ (2)

$= \sum_{i\in R\wedge i\neq j}(max\{0, m_i - \eta 2^{k_i}\}I + 2I_j \cdot P(m_i > J_i))$

$\leq I_{max}\sum_{i\in R\wedge i\neq j}(m_i - \eta 2^{k_i} + 2)$

$= I_{max}[2(n-1) + \sum_{i\in R\wedge i\neq j}(m_i - \eta 2^{k_i})]$

$\leq I_{max}[2(n-1) + \frac{n\gamma_c\gamma_l}{\nu_{min}} - \eta\sum_{i\in R\wedge i\neq j} 2^{k_i}]$

$= I_{max}[2(n-1) + \frac{n\gamma_c\gamma_l}{\nu_{min}} - \frac{\eta n(2^{d'-1})}{d'}]$

$\leq [\gamma_n/n + \gamma_n O(1/n)][O(n) - \frac{n(2^{d'-1})}{d'}]$, $(I_{max} = \gamma_n/n + \gamma_n O(1/n))$

$= \gamma_n O(\frac{1}{n})[O(n) - \frac{n(2^{d'-1})}{d'}]$

$= \gamma_n[O(1) - O(\frac{2^{d'}}{d'})]$

$= O(1) - O(\frac{2^{d'}}{d'})$

Combing the above results, we get the upper bound of a node i's outdegree is:

$\frac{2\gamma_c\gamma_l c_{max}}{\nu_{min}} - O(\frac{2^{d'}}{d'}) + O(1)$

B. Proof of Exponential Improvement in Lookup Efficiency

We define $b_i(t)$ as the number of servers with i spare capacities at time t; $m_i(t)$ as the number of servers with at most i spare capacities at time t; $p_i(t) = d_i(t)/d$ as the fraction of servers of i spare capacity; and $s_i(t) = m_i(t)/d$ as the fraction of servers with at most i spare capacities. Such that, $p_i = s_i - s_{i-1}$. In an *empty system*, which corresponds to one with no customers, $s_c = 1$, and $s_i = 0$ for $i < c$. A *fixed point* π is a point p in which $\frac{ds_i}{dt} = 0$.

The rate of spare capacity change in a node depends on whether it has more or fewer than T spare capacities. In the following, we calculate $\frac{ds_i}{dt}$ in the case $i \geq T-1$ and $i < T-1$ respectively. An arriving query occupies the i^{th} capacity of a server if one of b events happen: first, its first choice has $i+1$ spare capacities; second, its first choice has $\leq T-1$ spare capacities and its second choice has $i+1$ spare capacities; $\cdots$, its first b-1 choices have $\leq T-1$ spare capacities and its b^{th} choice has $i+1$ spare capacities. So that, there are $\lambda n(p_{i+1} + s_{T-1}p_{i+1} + s_{T-1}^2 p_{i+1} + \cdots + s_{T-1}^{b-1}p_{i+1})$ servers whose spare capacities change from $i+1$ to i during dt. Meanwhile, dp_i servers change their spare capacities from i to $i+1$. As a result, we get:

$\frac{ds_i}{dt} = \lambda(p_{i+1} + s_{T-1}p_{i+1} + s_{T-1}^2 p_{i+1} + \cdots + s_{T-1}^{b-1}p_{i+1}) - p_i, i \geq T-1,$

$\frac{ds_i}{dt} = \lambda[(s_{i+1} - s_i)(1 + s_{T-1} + s_{T-1}^2 + \cdots + s_{T-1}^{b-1})] - (s_i - s_{i-1}), i \geq T-1, p_i = s_i - s_{i-1},$

$\frac{ds_i}{dt} = \lambda(s_{i+1} - s_i)\frac{s_{T-1}^b - 1}{s_{T-1} - 1} - (s_i - s_{i-1}), i \geq T-1.$

When $i < T-1$, the number of queries arriving over dt is $\lambda b dt$, and that for an query being forwarded to a server with $i+1$ spare capacity is $b_i dt = d(s_i - s_{i-1})dt$. Consequently, $\frac{ds_i}{dt} = \frac{1}{b}\cdots\frac{dm_i}{dt} = \lambda(s_{i+1}^b - s_i^b) - (s_i - s_{i-1})$

The differential equations for the QFM when $i < T = 1$ considering a node with b specific neighbors is:

$\frac{ds_i}{dt} = \lambda(s_{i+1}^b - s_i^b) - (s_i - s_{i-1}), i < T-1.$

We get the differential equations for QFM:

$$\frac{ds_i}{dt} = \begin{cases} \lambda(s_{i+1} - s_i)\frac{s_{T-1}^b - 1}{s_{T-1} - 1} - (s_i - s_{i-1}) \\ \qquad\qquad\qquad\qquad c \geq i \geq T-1 \quad (3); \\ \lambda(s_{i+1}^b - s_i^b) - (s_i - s_{i-1}) \\ \qquad\qquad\qquad\qquad i < T-1 \quad (4). \end{cases}$$

Lemma B.1 *The QFM with $d \geq 2$ has a unique fixed point with $\sum_{i=c-1}^{-\infty} s_i < \infty$ given by*

$$\begin{cases} s_i = (\lambda - A)\frac{A^{c-i}-1}{A-1} + A^{c-i}, A = \lambda\frac{s_{T-1}^b - 1}{s_{T-1} - 1} \\ \qquad\qquad\qquad\qquad T-1 \leq i \leq c; \\ s_i = \lambda^{\frac{b^{T-i-1}-1}{b-1}} \cdot s_{T-1}^{b^{T-i-1}-1} \\ \qquad\qquad\qquad\qquad i < T-1. \end{cases}$$

Proof With the condition $\frac{ds_i}{dt} = 0$ for all i, we derive the value of s_i including s_{T-1} when $c \geq i \geq T-1$ with $s_c = 1$.

We summer the equation (3) over all if $c \geq i \geq T-1$, and get $s_{c-1} = \lambda - A + As_c$, assuming $A = \lambda\frac{s_{T-1}^b - 1}{s_{T-1} - 1}$. By induction: $s_{c-2} = (\lambda - A)(1 + A) + A^2 \cdots$, we get $s_i = (\lambda - A)\frac{A^{c-i}-1}{A-1} + A^{c-i}, A = \lambda\frac{s_{T-1}^b - 1}{s_{T-1} - 1}, c \geq i \geq T-1$.

By summing the equation (4) over all $i \leq T-1$ with $s_{-\infty} = 0$, we can derive that $s_{T-2} = \lambda s_{T-1}^b$. By induction: $s_{T-3} = \lambda s_{T-2}^b = \lambda(\lambda s_{T-1}^b)^b = \lambda^{\frac{b^2-1}{b-1}} s_{T-1}^{b^2} \cdots$

we get $s_i = \lambda^{\frac{b^{T-i-1}-1}{b-1}} \cdot s_{T-1}^{b^{T-i-1}-1}$ ($i < T-1$).

We use $\sum_{i=c-1}^{-\infty} s_i < \infty$ to ensure that the sum converges absolutely. ∎

Theorem B.1 *The fixed point for the QFM decreases doubly exponentially.*

Proof For a small threshold, the behavior of (4) is very similar to that of the STSM, which was proved doubly exponentially decrease in [22, 23].

To prove π_i decreases doubly exponentially we need to prove that $\pi_{T-j-1} = \lambda(\pi_{T-j}^b)$ for all $j \leq T-1$. To prove the latter, we need to prove that $\pi_{T-2} = \lambda(\pi_{T-1}^b)$, which was proved for equation (4) in [22, 23]. With $\frac{ds_i}{dt} = 0$ at the fixed point, we get

$\lambda\pi_{i+1}^b - \pi_i = \lambda\pi_i^b - \pi_{i-1}$ for $i \leq T-2$.

Such that, the theorem is proved if $\pi_{T-2} = \lambda(\pi_{T-1}^d)$.

From equation (3) we have

$\lambda[(\pi_T - \pi_{T-1})(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1})] - (\pi_{T-1} - \pi_{T-2}) = 0$

$\pi_{T-2} - \lambda\pi_{T-1}^b = (1+\lambda)\pi_{T-1} - \lambda\pi_T(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1}) + \lambda(\pi_{T-1}^2 + \pi_{T-1}^3 + \cdots + \pi_{T-1}^{b-1})$ (5)

Hence, if the right hand side of equation (5) is 0, the theorem is proved.

At the fixed point in (3) for $c - 2 \leq i \leq T-1$, we get

$\lambda[(\pi_{i+2} - \pi_{i+1})(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1})] = \pi_{i+1} - \pi_i$

We summing the left and right hand sides of the above equations for all values of i and get

$\lambda\pi_c(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1}) - \lambda\pi_T(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1}) = \pi_{c-1} - \pi_{T-1}$

With $\pi_c = 1$ and $\pi_{c-1} = \lambda$, we get $\lambda(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{d-1}) - \lambda\pi_T(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{b-1}) = \lambda - \pi_{T-1}$,

$(1+\lambda)\pi_{T-1} = \lambda\pi_T(1 + \pi_{T-1} + \pi_{T-1}^2 + \cdots + \pi_{T-1}^{d-1}) + \lambda(\pi_{T-1}^2 + \pi_{T-1}^3 + \cdots + \pi_{T-1}^{b-1})$

As a result, the right hand of equation (5) is 0 and the theorem is proved. ∎

Theorem B.2 *For any fixed time spot T, the time a query waits before being forwarded during the time interval $[0, T]$ is bounded and b-way ($b \geq 2$) forwarding yields an exponential improvement in the expected time for a query queuing in a server.*

Proof By equation (4), we can get that in the case when $i < T-1$, an incoming query arriving on a node at time t lets the node has i spare capacity with probability $s_{i+1}(t)^b - s_i(t)^b$, and this query becomes the $(c-i)^{th}$ query in the process waiting queue of the server. So that the expected waiting time of the query is $\sum_{i=T-2}^{-\infty}(c-i)(s_{i+1}(t)^b - s_i(t)^b) = (c - T + 2)s_{T-1}^b + \sum_{i=T-2}^{-\infty}(s_i^b(t))$

By equation (3), we can get that in the case when $i \geq T-1$, the expected waiting time of a query is $\sum_{c-1}^{T-1}(c - i)(A/\lambda)(s_{i+1}(t) - s_i(t)) = (A/\lambda)(\sum_{i=c}^{T} s_i - (c - T + 1)s_{T-1})$, $A = \lambda\frac{s_{T-1}^b - 1}{s_{T-1} - 1}$. By Lemma B.1, at $t \to \infty$, the QFM converges to the fixed point. So that the expected waiting time for a query in a server can be made $(A/\lambda)(\sum_{i=c}^{T}((\lambda - A)\frac{A^{c-i}-1}{A-1} + A^{c-i}) - (c - T + 1)s_{T-1}) + (c - T + 2)s_{T-1}^b + \sum_{i=T-2}^{-\infty}(\lambda^{\frac{b^{T-i}-b}{b-1}} s_{T-1}^{b^{T-i}-b}) + o(1)$.

In QFM, the time a query waits on a node when $i \geq T-1$ is less than the time when $i < T-1$ because in the former case the query is processed by a light node. The above bound can be enlarged to $\sum_{i=c}^{-\infty} s_i^b = \sum_{i=c-1}^{-\infty} \lambda^{\frac{b^{c-i}-b}{b-1}}$. Then we can apply the proved result of exponential time improvement in [22, 23] to QFM. ∎

Theorem B.3 *For any fixed time spot T, the maximum congestion of a node in an QFM over the interval $[0, T]$ is exponentially improved w.h.p.*

Proof It was proved in [22] that the most load in an initially empty STSM of over $[0, T]$ is exponentially improved. Since maximum congestion is the most load divided by the node's fixed load, it results in exponentially improved maximum congestion of a node. ∎