

# On the Stability-Scalability Tradeoff of DHT Deployment

Chih-Chiang Wang and Khaled Harfoush

Department of Computer Science

North Carolina State University

E-mail: cwang2, kaharfou@ncsu.edu

**Abstract**—Distributed hash tables (DHTs) provide efficient data naming and location with simple hash-table-like primitives, upon which sophisticated distributed applications can be built. DHT users provide *free* but *unstable* peer-to-peer (P2P) capacity. With stable DHT nodes being relatively scarce, a DHT can either rely on a small set of stable nodes with limited collective capacity, or a larger set of potentially less stable nodes and suffer maintenance and data redundancy overhead. In this paper, we provide an analytical model that captures the tradeoff between the stability and the scalability in DHT-based P2P systems. We use the model to demonstrate that the DHT throughput can be optimized through careful engineering of DHT node selection and data redundancy parameters.

## I. INTRODUCTION

Distributed hash tables (DHTs) typically serve as shared storage infrastructures where stored data objects are mapped to a global identifier space. By partitioning the identifier space across a set of distributed nodes and connecting them into an overlay network, DHTs provide efficient data naming and location with simple hash-table-like primitives, upon which sophisticated distributed applications such as storage [1], content distribution [2] and more can be built.

In order to ensure an adequate level of service availability and performance, a DHT needs to monitor and maintain the connectivity of its overlay and the availability of its stored content. This maintenance requires DHT nodes to constantly probe other nodes, to replicate/code/migrate content, which consumes a large amount of the system resources especially in inherently dynamic P2P systems [3], [4]. Thus, while allowing every member of a P2P system to contribute by serving leads to a wealth of service resources, it comes at the cost of high maintenance overhead consumed to recuperate for unstable nodes that spend a small amount of time in the system. The gain does not always outweigh the cost. On the other hand, relying purely on dedicated, high capacity clusters of servers to accommodate the huge demands of P2P applications is financially costly and is not always, if at all, feasible. As a compromise between pure P2P and pure client/server paradigms, the OpenDHT model relies on a limited set of dedicated, relatively stable distributed nodes under centralized administration, and offers its services to other outsider nodes,

which act merely as clients to the DHT [5]. Still, the cost of bringing up a large set of dedicated distributed nodes to meet the ever-increasing demands of P2P applications is a heavy burden.

In this paper, we advocate the wise deployment of free, potentially unstable resources of P2P users. In other words, P2P members are welcomed to join the DHT infrastructure as long as the gain in service scalability outweighs the maintenance cost incurred due to node instability, whereas members for which the cost outweighs the gain should act only as clients. The service demands that are beyond the DHT's capacity can be handled in an application-dependent manner, be left out to dedicated application servers if available, queued and re-processed, dropped, etc. It is clear that many system attributes are beyond a system administrator's control such as the workload imposed by application users, the capacity of nodes and the time they spend in the system, the popularity of the stored data objects and the diversity of these attributes among the P2P system members. Still, an administrator/designer may be able to control or at least stimulate changes to the set of nodes serving as DHT members, the data objects to be maintained by the DHT, and the appropriate replication/coding strategy to store these objects.<sup>1</sup> In order to maximize the *utility* of the underlying system, the DHT should (1) include the most stable nodes and (2) maintain the most popular data objects. The most stable nodes are indeed needed to increase the reliability of DHT data service and to reduce the overhead associated with overlay and data maintenance, and more popular data objects are more welcomed to the DHT since they are targeted by most of user requests and answering more user requests ensures higher system utility. A careful design allows the DHT to answer as many of the user requests as possible given the uncontrolled system attributes. Figure 1 provides a schematic description of the propose service paradigm.

We cast the problem as a multivariate constrained optimization targeted at optimizing the DHT throughput, and use the model to show how to effectively administer DHT systems by tuning: (1) DHT node selection process, (2) DHT content selection process, (3) content redundancy strategy.

The rest of this paper is organized as follows. In Section

<sup>0</sup>This work was partially supported by NSF grant CAREER ANIR-0347226.

<sup>1</sup>The techniques to monitor and induce changes to the system are out of the scope of this paper.

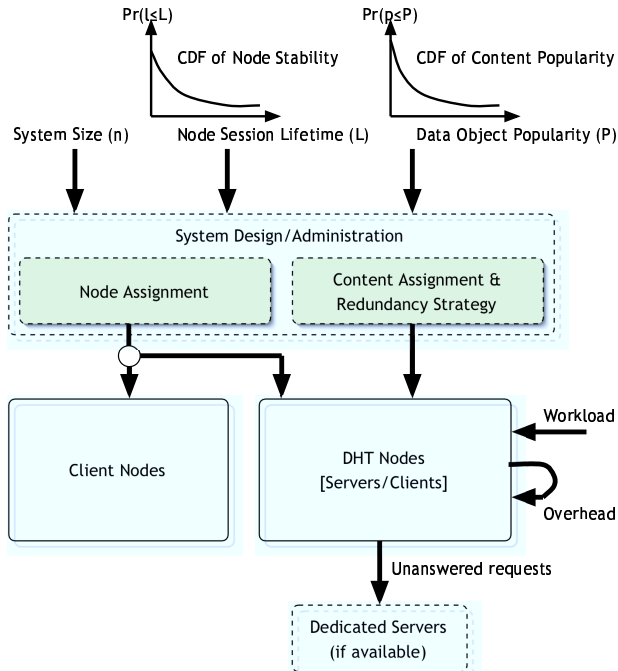


Fig. 1. System Overview.

If we survey existing research on overlay stability and data availability and relate them to our own. In Section III we introduce the basics of our model, the terminology and the assumptions that we make. In Section IV we model the node selection process and its implications on DHT stability and scalability. In Section V we model the content selection process and data redundancy strategies. In Section VI we quantify the overhead associated with DHT deployment. In Section VII we use the results of earlier sections to formulate the optimization function. In Section VIII, we apply the model to different case studies and analyze the results. We finally conclude in Section IX.

## II. RELATED WORK

In order to improve DHT functionality in the presence of high node instability, researchers have taken different approaches, whether by (1) improving node stability through a selection of the most stable nodes as DHT members, or by (2) improving the availability of the maintained data objects through replication and erasure coding.

OpenDHT [5], [6] runs a DHT on PlanetLab [7] nodes, which offer services to nodes outside the DHT. PlanetLab is a wide-area testbed for networking and distributed systems research, and its nodes are much more stable than typical P2P nodes. In [8], Godfrey et al. investigate several strategies to select stable DHT nodes and explore the performance implications. Also, in [9] Mickens et al. use statistics about nodes' history to predict their future behavior and use more stable nodes for data placement. However, how stable should nodes be to join the DHT? If the process is very selective, then a small number of stable nodes will qualify, which may stifle the system scalability and limit its ability to respond to heavy

workloads. If the process is not selective, then less stable nodes will qualify and the associated overhead will kick-in. None of the previous approaches have studied this tradeoff to determine the optimal stability threshold, which is a motivation of this paper.

In [10], Black et al. show that when nodes are not stable, high data redundancy requires unbearable cross-system bandwidth. In [11], Rodrigues et al. compare redundancy schemes for DHTs, taking overlay node characteristics into account. They conclude that erasure coding favors environments with low node stability but the required maintenance bandwidth for a scalable and highly available system can be unsustainable for home users. In [12], Weatherspoon et al. estimate the amount of data redundancy required for reducing data recovery costs incurred by transient node failures. These studies did not optimize the utility of the system as reflected on the number of answered queries, and did not evaluate the choice of the redundancy schemes under various factors such as the bandwidth and storage constraints, the skew in the popularity of the maintained objects, and the workload imposed on the system, which is a motivation of this paper.

## III. MODEL OVERVIEW

Symbol	Meaning
$n$	number of nodes in the system
$a$	node availability of the system
$a'$	node availability of the DHT set
$q_d$	mean size of data objects
$q_m$	mean size of DHT message
$\sigma_r$	per-node READ rate
$\gamma$	ratio of per-node WRITE rate to READ rate
$c_b$	mean bandwidth budget per DHT node
$c_s$	mean storage budget per DHT node
$W$	number of data objects to be stored in the DHT
$l$	number of data fragments generated per WRITE
$m$	number of data fragments needed per READ
$r$	rate of coding
$A$	availability of DHT data object
$U(x)$	CDF of system nodes' session time
$u'(x)$	CDF of the DHT nodes' session time
$\bar{u}$	mean session time of system nodes
$\bar{d}$	mean downtime of system nodes
$\bar{u}'$	mean session time of the DHT set

TABLE I  
TERMINOLOGY.

Our model is targeted to simple storage systems in which *throughput*, the amount of DHT data that can be retrieved by user requests in some time period, is the performance metric to be optimized. Our objective is mainly to offer qualitative insights into the problem rather than accurate quantitative results. Resources such as bandwidth and storage capacity impose system constraints; other resources such as processing capacity are relatively abundant.

We consider a system of  $n$  nodes in the steady-state, in which short-term node dynamics like transient slowness at DHT nodes [13] have negligible transient effects on the

overall system’s performance. We also dismiss packet loss and retransmissions from our model. Each node cycles between two states: *ON* when the node is in its *session time* (or *uptime*); and *OFF* when in its *downtime*. A node’s session time is a time interval from the moment it joins the system to the moment it subsequently departs, while its downtime is a time interval from its departure to its subsequent re-join. Let  $U(\cdot)$  be the cumulative distribution function (CDF) followed by nodes’ session times and  $\bar{u}$  be the mean session time. The larger  $\bar{u}$ , the more stable the system. Similarly, let  $D(\cdot)$  be the CDF followed by nodes’ downtimes and  $\bar{d}$  be the mean downtime. The actual distributions of these system attributes will be presented in later sections. We define *node availability*,  $a$ , as the probability that a node is in the ON state. Thus,  $a = \bar{u}/(\bar{u} + \bar{d})$ , and the expected number of nodes in the ON state can be expressed as  $a \cdot n$ . It should be clear that a larger  $a$  does not necessarily mean a more stable system.

Nodes that are in the ON state generate *workload* in the form of READ and WRITE requests on data objects. WRITES are used to insert new data objects at the appropriate nodes in the DHT and READs are used to retrieve these objects. Let  $\sigma_r$  be the average number of READ requests per unit time generated by each node in the ON state, and  $\sigma_w = \gamma\sigma_r$ ,  $0 < \gamma \leq 1$ , be the average number of WRITE requests per unit time generated by each node in the ON state. Each READ and each WRITE request refers to a single data object. Let  $q_d$  be the average size in bytes of the data objects and  $q_m$  be the average size in bytes of the control messages exchanged in the system. We assume UDP-based transport protocol is used and, for convenience, assume that each DHT message has  $q_m = 50$  bytes long (28 bytes for the UDP/IP header, 20 bytes for the DHT identifier, and 2 bytes for miscellaneous overhead.) [6], [14]. Our model also accounts for the overhead associated with the transfer of data objects. Assuming 1500 byte packets, an object of size  $q_d$  bytes will fill out the payload of approximately  $\lceil \frac{q_d}{1500 - q_m} \rceil$  packets, each of which consumes  $q_m$  bytes of overhead.

To avoid storing an infinite number of data objects in the DHT, each object is purged out of the DHT after some time period,  $\tau$ , but the purged objects could be re-inserted as nodes wish. In steady state, the average number of unique objects in the system,  $W$ , can be expressed as  $W \propto an\sigma_w\tau$ . We assume objects’ popularity follows some distribution  $P(\cdot)$ , which we investigate in detail in Section V. To keep our analysis manageable, we assume that DHT nodes do not lose references to their assigned data objects if they leave then rejoin the system as long as the objects time-to-live (TTL) value,  $\tau$ , does not expire [10], [11], [12].

Nodes are either assigned to the *DHT set* or the *client set*. Nodes in the DHT set participate in serving client requests whether generated by nodes in the client set or in the DHT set, and nodes in the client set are pure clients. Assuming enough information about nodes’ session times [8], [9], nodes with expected session times above some threshold,  $T$ , are assigned to the DHT when they are in the ON state. DHT nodes that switch to the OFF state and temporarily depart the DHT, do not respond to clients’ requests. In order to avoid

losing content to nodes in the OFF state, data redundancy schemes such as replication and erasure coding are used. Clients’ requests are forwarded to appropriate DHT nodes which maintain references to DHT nodes storing the actual data objects and these references may be outdated due to DHT nodes’ temporary departure. This *transient departure* model of DHT nodes has been used and justified in DHT research [10], [11], [12]. Let  $n' \leq n$  be the number nodes in the ON state that are assigned to the DHT set and  $\eta \equiv n'/(a \cdot n)$  be the fraction of such nodes,  $0 < \eta \leq 1$ . The most stable nodes are assigned to the DHT. Also, let  $w' \leq W$  be the number of data objects (files) managed by the DHT and  $\xi = w'/W$  be the fraction of such objects,  $0 < \xi \leq 1$ . The most popular files are managed by the DHT. The choice of  $\eta$  and  $\xi$  implies stability, availability, capacity DHT characteristics. Let  $\bar{u}'$ ,  $\bar{d}'$ ,  $u'(\cdot)$ ,  $D'(\cdot)$ ,  $a'$  denote the attributes describing the DHT set, corresponding to the  $\bar{u}$ ,  $\bar{d}$ ,  $U(\cdot)$ ,  $D(\cdot)$  and  $a$  attributes describing the pool of  $n$  nodes, respectively. Furthermore, we define  $\check{a}'$  as the probability that a DHT node is in the ON state and stays ON long enough to serve a client request.

We assume a generic DHT structure in which each DHT node maintains overlay connections to a set of overlay neighbors and  $\log_2(n')$  successor nodes [15], [16]. The expected total DHT bandwidth and storage budget available to the DHT system are  $n' \cdot c_b$  and  $n' \cdot c_s$ , respectively. We assume that the DHT workload is balanced among its nodes by an underlying load balancing mechanism [17], [18]. Typically, the load balancing procedure incurs periodic data transfer in DHTs, but we simply assume that the cost of this traffic is incorporated in the value of  $\sigma_w$ . The symbols used throughout this paper are summarized in Table I.

#### IV. NODE ASSIGNMENT

In this section we rely on P2P measurement results to pinpoint realistic node session time and downtime distributions,  $U(\cdot)$  and  $D(\cdot)$ , respectively. Then show how we select  $n' < n$  nodes in the ON state, a fraction  $\eta = n'/(a \cdot n)$ , as DHT nodes and study the impact of our node selection on the DHT system stability,  $\bar{u}'$ , and node availability,  $a'$  and  $\check{a}'$ .

##### A. Distribution of Node Session Times

We assume apriori knowledge about nodes’ expected session times, which can be obtained by directly questioning the nodes, by monitoring nodes’ session-time history and using it as an indication of their future behavior [8], [9]. Measurement results indicate that nodes’ session times can be well modeled by a long-tailed distribution [3], [12], [19]. We assume that nodes’ expected session times are independently distributed and model them using a shifted Pareto with probability density function (PDF)

$$u(t) = \frac{\alpha}{\beta} \left(1 + \frac{t}{\beta}\right)^{-(\alpha+1)}, \quad \alpha > 1 \quad (1)$$

and cumulative distribution function (CDF)

$$U(t) = 1 - \left(1 + \frac{t}{\beta}\right)^{-\alpha}, \quad \alpha > 1 \quad (2)$$

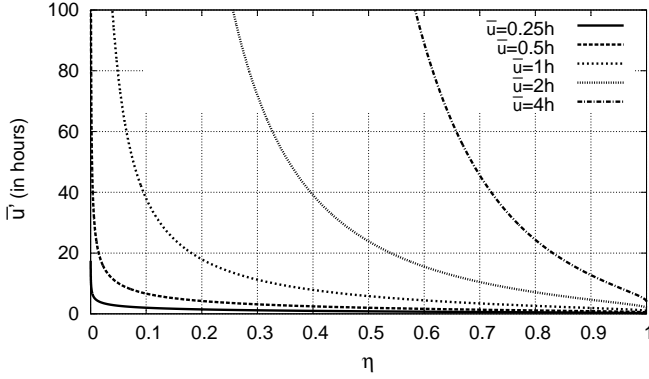


Fig. 2. Scalability ( $\eta$ ) vs. Stability ( $\bar{u}'$ ) for different values of the node selection threshold,  $T$ .

The smaller the value of  $\alpha$  the more stable the system as nodes stay longer in the ON state. We are not aware of any P2P measurement study that reveals the distribution of node downtimes. We assume that the average downtime,  $\bar{d}'$ , is a constant and equal to 5.25 hours according to [20].

### B. DHT Stability

Nodes in the ON state, which are expected to stay alive for at least some time,  $T$ , are included in the DHT set; otherwise, they are included in the client set. The PDF of DHT nodes' session times,  $u'(\cdot)$ , and the CDF of DHT nodes' session times,  $U'(\cdot)$ , can thus be expressed as

$$u'(t) = \begin{cases} 0 & t < T \\ K_T u(t) & t \geq T, \end{cases} \quad (3)$$

where  $K_T$  is a normalization constant and can be expressed as  $K_T = 1 / \int_{t=T}^{\infty} u(t) dt = 1 / (1 - U(T)) = (1 + T/\beta)^\alpha$ , and

$$U'(t) = \begin{cases} 0 & t < T \\ K_T (U(t) - U(T)) & t \geq T, \end{cases} \quad (4)$$

Using the average session time of DHT nodes,  $\bar{u}'$ , as a measure of DHT stability, it can be expressed as:  $\bar{u}' = \int_{t=T}^{\infty} t \cdot u'(t) dt = K_T \int_{t=T}^{\infty} t \cdot u(t) dt$ . Substituting the value of  $K_T$  and  $u(t)$  from Equation 1, and solving the integration, we get

$$\bar{u}' = \frac{\beta + \alpha T}{\alpha - 1} \quad (5)$$

The larger the threshold,  $T$ , the more stable the DHT system becomes (larger  $\bar{u}'$ ) since only the most stable nodes join the DHT.

### C. DHT Scalability

On the other hand, the larger the threshold,  $T$ , the less scalable the DHT system (smaller  $n'$  and  $\eta$ ) since the *bandwidth budget* offered by the small number of DHT nodes is small.

We next identify the values of  $n'$  and  $\eta$  (scalability measures) as a function of the DHT node selection threshold,  $T$ . Using Little's theorem, the average number of nodes in a stable system is equal to the average arrival rate of these nodes

multiplied by their average lifetime in the system. Applying Little's law to nodes in the ON state in our system, we get

$$an = \lambda \bar{u} \quad (6)$$

where  $\lambda$  is the average arrival rate of nodes to the ON state. Applying Little's law to nodes in the ON state, with expected lifetime larger than  $T$  (the DHT nodes), we get

$$n' = \lambda(1 - U(T))\bar{u}' \quad (7)$$

Solving equations 6 and 7, and substituting for  $U(T)$  from Equation 2, and using  $\bar{u} = \int_{t=0}^{\infty} t \cdot u(t) dt = \beta / (\alpha - 1)$  then

$$\eta \equiv \frac{n'}{an} = \left(1 + \frac{T}{\beta}\right)^{-\alpha} \left(1 + \frac{\alpha}{\beta}T\right) \quad (8)$$

Based on equations 5 and 8, the rate of change of  $\bar{u}'$  as  $T$  changes can be expressed as  $\partial \bar{u}' / \partial T = \alpha / \alpha - 1$ , while the rate of change of  $\eta$  as  $T$  changes, setting  $\beta = 1$  for simplicity, can be expressed as  $\partial \eta / \partial T = -\alpha(\alpha - 1)T / (1 + T)^{\alpha+1}$ . This means that increasing  $T$  degrades the system scalability at a higher pace than the pace at which the system stability is improved. To get some intuition into the degradation in stability,  $\bar{u}'$ , as scalability,  $\eta$ , increases refer to Figure 2. We vary  $\alpha$  between 5, 3, 2, 1.5, and 1.25 to obtain  $\bar{u}$  of 0.25, 0.5, 1, 2, and 4 hours, respectively. Then for each of these cases, we plot the curve of  $\bar{u}'$  as  $\eta$  changes. Clearly, DHT stability drops dramatically as more nodes join the DHT. This calls for accurate tuning of the node selection process as the degradation in DHT stability can easily degrade the system performance.

### D. DHT Node Availability

Recall that a client request is answered by a list of references to nodes storing the requested content. The referenced nodes may have left the system. A reference to a node will be successful if (1) the node is in the ON state, and (2) if it stays long enough in the ON state to serve the requested content to the client.<sup>2</sup> The probability that a DHT node is in the ON state,  $a'$ , can be expressed as

$$a' = \frac{\bar{u}'}{\bar{u}' + \bar{d}'} \quad (9)$$

Let  $\check{a}'$  be the probability that a DHT node is in the ON state *and* stays ON long *enough* to answer a client request. In order to estimate  $\check{a}'$ , we define two measures: (1) a DHT node's *residual lifetime*, which is defined as the amount of time remaining in the node's session time at the moment a client request is intercepted. (2) A request's *sojourn time* is the amount of time that a client's request is expected to spend at a DHT node for the requested content transfer, until the request is fully served. The sojourn time includes the queuing time and the service time of the request. The following distributions are of special interest to the estimation of  $\check{a}'$ : (1) the PDF of DHT

<sup>2</sup>We assume that a client does not leave the system while downloading its requested content.

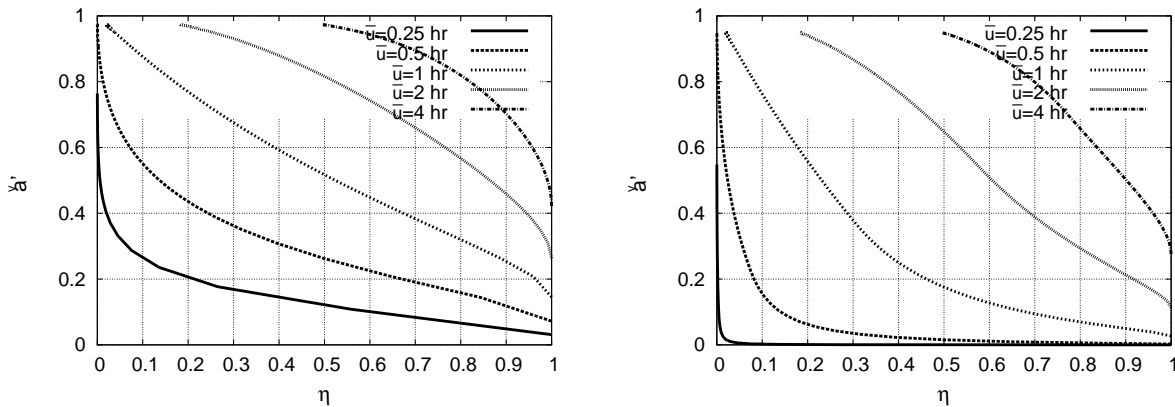


Fig. 3. DHT node availability,  $\check{a}'$ , as  $\eta$  increases when average sojourn time,  $\bar{s}$ , is 0.1 (left) and 5.0 (right).

nodes' residual lifetimes,  $r(t)$ , and (2) the CDF of requests' sojourn times,  $S(t)$ .

Assuming requests arrive at DHT nodes uniformly at random over their session time. Then by renewal theory [21]

$$r(t) = \frac{1 - U'(t)}{\bar{u}'} \quad (10)$$

Also, assuming that data objects are of the same size, and the service (transmission) time of each data object is  $b$  time units, then each DHT node can be modeled as a  $M/D/1$  queue with infinite buffer. Then  $S(t)$  is given by [22]

$$S(t) = \begin{cases} 0 & t < b \\ (1 - \rho) \sum_{k=0}^{\lfloor y \rfloor} \frac{(\rho(k-y))^k}{k!} e^{-\rho(k-y)} & t \geq b \end{cases} \quad (11)$$

where  $y = \frac{t-b}{b}$  and  $\rho$  is the utilization of the DHT node.<sup>3</sup> Given estimates for  $a'$ ,  $r(t)$  and  $S(t)$ ,  $\check{a}'$  can be estimated as

$$\check{a}' = a' \int_{t=0}^{\infty} S(t)r(t) dt \quad (12)$$

To get some intuition into the impact of different system parameters on DHT node availability,  $\check{a}'$ , we fix  $\beta = 1$  and vary  $\alpha$  as in Section IV-C to obtain different node uptime distributions,  $U(\cdot)$ , with  $\bar{u}$  of 0.25, 0.5, 1, 2, and 4 hours. We also fix  $\rho$  and  $b$  values to obtain different sojourn time distributions,  $S(\cdot)$ , with average sojourn times,  $\bar{s}$ , of 0.1 and 5.0 hours. The latter case is intentionally extreme to highlight the difference. In Figure 3 we plot the values of  $\check{a}'$  as  $\eta$  increases for different values of  $\bar{u}$  when  $\bar{s} = 0.1$  (left) and when  $\bar{s} = 5.0$  (right). As expected, a smaller  $\bar{s}$  leads to a better chance at having DHT nodes available in the ON state long enough to serve the requests (larger  $\check{a}'$ ). A small  $\bar{s}$  is also desirable from clients' perspective as it implies better perceived response time to their requests. In order to minimize  $\bar{s}$ , a system designer can either reduce the load,  $\rho$ , on DHT nodes and/or reduce  $b$  by reducing the size of

<sup>3</sup>The computation of  $S(t)$  is very computationally intensive and we rely on reasonable approximations [23], [24]. We omit the description of these approximations for lack of space and intend to include them in an extended version of this paper.

the requested data objects (to reduce their transmission time) through object fragmentation as we describe in Section V. Note that fragmentation slightly increases the load on the DHT system due to the extra messaging overhead. Also, the load on the system can be reduced by increasing the number of DHT nodes. This comes however at the cost of system instability. The multitude of interacting tradeoffs makes the choice of an optimal set of system parameters non-trivial and motivates the optimization that we describe in Section VII.

## V. CONTENT ASSIGNMENT & REDUNDANCY STRATEGY

Selecting a subset,  $w' \leq W$ , of the most popular data objects to be managed by the DHT, instead of managing all  $W$  data objects, can help the system answer more of READ requests especially when *stable* DHT resources are limited. Data redundancy schemes are useful in offering more download points (service capacity) for popular content that would otherwise be rarely used to serve unpopular content. Data fragmentation is useful to improve the system response time and DHT nodes' availability as motivated in Section IV-D. In this section we rely on P2P measurement results to pinpoint realistic object popularity distributions,  $P(\cdot)$ , and select a fraction  $\xi = \frac{w'}{W}$  of the most popular objects to be managed by the DHT. We also survey well-known data redundancy strategies and analyze the relation between  $\xi$  and the probability that a requested data object is available for clients to download from the DHT.

### A. Distribution of Object Popularity

Measurement studies indicate that the popularity of files in P2P systems exhibit a pronounced skew, with most of the file references targeted at a small number of files, and a small number of references to the majority of the files [4]. We use the widely accepted Zipf distribution with Zipf-exponent of  $\alpha = 1$  to model this skew in data-object popularity, which has a CDF of the form:

$$P(w) = \left( \frac{\sum_{i=1}^w i^{-\alpha}}{\sum_{i=1}^W i^{-\alpha}} \right) \quad 1 \leq w \leq W \quad (13)$$

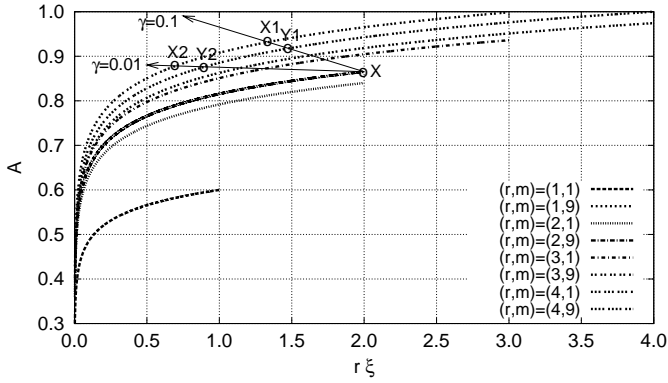


Fig. 4. Plot of DHT throughput,  $A$ , when DHT node availability,  $\tilde{a}' = 0.6$ .

### B. Data Redundancy Strategies

Past work on DHT-based storage systems [10], [25] generalize commonly adopted *replication* and *erasure coding* data redundancy schemes into a choice of coding parameters,  $l$  and  $m$ , where  $l$  is the number of data fragments generated per stored data object and  $m \leq l$  is the number of fragments that need to be retrieved from the  $l$  fragments, in order to successfully reconstruct the requested data object. Replication is the special case in which  $m = 1$ , and  $l$  denotes the number of replicas of the data object. The *rate of coding*,  $r = \frac{l}{m}$ , hence expresses the amount of redundancy.

We argue that managing only a fraction  $\xi = w'/W$  of the most popular data objects by the DHT, while ignoring the rest as they are beyond the capacity of the available DHT resources, and replicating the managed DHT objects can help the DHT system answer more requests. To make the case for this argument notice that (1) at most a fraction  $P(w')$  of the client requests will be answered by the DHT, and (2) redundancy strategies are aimed at providing enough DHT service capacity to answer the fraction  $P(w')$  of the requests. One can generalize our measure of node availability,  $\tilde{a}'$ , to incorporate the selected fraction of DHT objects,  $\xi$ , and the redundancy strategy,  $(r, m)$ , as follows. Let  $A$  be the probability that a requested data object is maintained by the DHT and that  $m$  out of  $l$  fragments of the stored object are downloadable from the DHT.  $A$  can be expressed as:

$$A = P(w') \sum_{i=m}^l \binom{l}{i} (\tilde{a}')^i (1 - \tilde{a}')^{l-i} \quad (14)$$

A larger  $A$  results in better DHT *throughput*. We will show that using  $\xi < 1$  and  $r > 1$  may improve  $A$ . In Figure 4, we pick  $\tilde{a}' = 0.6$  and plot  $A$  as we vary the *storage cost*,  $r\xi$ , for different redundancy strategies,  $(r, m)$ . Note that different  $A$  values imply different system utilization. In order to fairly compare different cases, we need to mitigate the difference in system utilization and maintain similar workload on the system. Given  $\gamma$  the ratio of per-node WRITE rate over READ rate, the workload posed on the DHT system is roughly a constant if we fix  $A + \gamma r\xi$ . In Figure 4 we plot a line labeled

$\gamma = 0.1$  on which  $A + \gamma r\xi$  is some fixed constant, and we pick three points  $X$ ,  $X_1$  and  $Y_1$  along the line that use the same object fragmentation  $m = 9$ . At point  $X$ , we have  $\{r = 2, m = 9, \xi = 1.0, A = 0.865\}$ . At point  $X_1$ , we have  $\{r = 3, m = 9, \xi = 0.447, A = 0.931\}$ . Comparing  $X$  and  $X_1$  confirms our intuition that a larger  $r$  can lead to a better  $A$ . However, at point  $Y_1$  we have  $\{r = 4, m = 9, \xi = 0.37, A = 0.917\}$ . Comparing points  $X_1$  and  $Y_1$  reveals that a larger  $r$  does not always lead to better  $A$  and thus an optimal value of  $r$  needs to be identified. The same conclusions hold for points  $X$ ,  $X_2$ , and  $Y_2$  along the line labeled  $\gamma = 0.01$ . Figure 4 also reveals that tuning object fragmentation,  $m$ , can improve  $A$  but marginally compared tuning the redundancy,  $r$ . A larger  $m$  typically improves the clients' perceived performance as it reduces their requests' sojourn times, but this improvement is offset by messaging overhead as we elaborate in the next sections.

### VI. DHT RESOURCE CONSUMPTION

The maintenance of the DHT overlay structure and data, and the READ and WRITE requests all consume DHT bandwidth. The DHT data objects also consume disk storage. Our choice of  $\eta$ ,  $\xi$ ,  $r$ , and  $m$  affect the amount of consumed resources. In this section we quantify the average resource expenditure.

The bandwidth consumed in downloading data objects by READs per online node is estimated as

$$B_1 = \sigma_r q_d A \quad (15)$$

$B_1$  provides a measure of the DHT throughput, and is the metric that our model is optimizing. The bandwidth consumed in uploading new data objects by WRITEs per online node is

$$B_2 = \gamma \sigma_r q_d r \xi \quad (16)$$

Each READ request and WRITE request message also consumes bandwidth. The bandwidth consumed by READ messages per online node can be expressed as

$$B_3 = \sigma_r q_m \left( H + m + \left\lceil \frac{q_d/m}{1500 - q_m} \right\rceil m P(w') \right) \quad (17)$$

where  $H$  is the average number of DHT hops to reach an arbitrary DHT node. The bandwidth consumed by WRITE messages per online node is

$$B_4 = \gamma \sigma_r q_m \left( H + r m + \left\lceil \frac{q_d/m}{1500 - q_m} \right\rceil r m \xi \right) \quad (18)$$

For maintenance traffic, we assume that each DHT node communicates periodic heartbeat messages at a rate  $\frac{h}{T}$  to a set of  $N$  DHT nodes, which includes (1) the node's overlay neighbors, (2) its  $\log_2(n')$  successor nodes, and (3) nodes which contain data fragments that the DHT node is referencing. We set  $h$  to 1 heartbeat message every 10 seconds, which is large enough to maintain accurate node-state information according to previous DHT studies [26], [27]. Each message incurs  $\log_2(n')$  *stabilization* messages to ensure the correctness of the routing overlay [15], [16]. For  $H = 1$ , the DHT overlay is a full mesh so  $N$  is  $n'$ ; for  $H > 2$ ,  $N$  is the maximum of

$rm$  and  $\log_2(n') + n'^{\frac{1}{\alpha}}$ . The overall maintenance traffic per online DHT node is,

$$B_5 = \frac{q_m N h \log_2(n')}{T} \quad (19)$$

The disk space consumed by all the DHT data objects can be expressed as

$$D = r w' q_d \quad (20)$$

Let the residual bandwidth capacity per DHT node,  $c'_b$ , be the bandwidth remaining from  $c_b$  after excluding the bandwidth consumed in DHT lookup and maintenance overhead.

$$c'_b = c_b - B_5 - \frac{\sigma_r q_m (H + m) + \gamma \sigma_r q_m (H + r m)}{\eta} \quad (21)$$

The service time per fragment is just a fragment's payload plus its header overhead divided by the residual bandwidth.

$$b = (q_d/m + q_m \lceil \frac{q_d/m}{1500 - q_m} \rceil) / c'_b \quad (22)$$

The utilization  $\rho$  is computed as the load of accepted READ/WRITE requests over the residual bandwidth ( $c'_b$ ) per DHT node, which can be expressed in terms of  $b$ .

$$\rho = \frac{\sigma_r (P(w') + \gamma r \xi) m b}{\eta} \quad (23)$$

## VII. SYSTEM OPTIMIZATION

Our objective is to maximize the DHT throughput,  $B_1$ , subject to constraints on the available bandwidth and storage resources. We next formulate these constraints. The overall DHT bandwidth consumption is  $an \cdot (B_1 + B_2 + B_3 + B_4) + n' \cdot B_5$  and the overall DHT bandwidth budget is  $n' c_b$ , which leads to the following bandwidth constraint:

$$B_1 + B_2 + B_3 + B_4 \leq \eta \cdot (c_b - B_5)$$

Also, the storage consumed by all the DHT data objects,  $r w' q_d$ , must be constrained by the storage budget of the DHT,  $n' c_s / a'$ , which leads to the following storage constraint:

$$a' \frac{r \cdot w' \cdot q_d}{n'} \leq c_s$$

Additional constraints are needed to make our constrained optimization more realistic. First, because fragments of an object need to be placed at different DHT nodes, the constraint  $rm \leq n'$  is necessary. Second, we assume that no matter how small a fragment is,  $b \geq 100$  millisecond. Last, the expected sojourn time for downloading an object should be less than the average session time of nodes, so we assume  $\bar{a}' = 0$  for  $\bar{u} < m \cdot \bar{s}$ .

## VIII. MODEL RESULTS

The optimization problem formulated in the previous section is not amenable to closed form solution since it consists of interacting nonlinear terms, and many variables, such as  $n'$ ,  $w'$  and  $m$ , are bounded, positive integers. Since our objective is to offer qualitative insights and not efficient solutions, we rely on an exhaustive search of the state space  $(\eta, \xi, r, m)$  for the optimal combination, leading to the maximum  $B_1$  (or  $A$ ).<sup>4</sup>

In this section we study the evolution of the optimal DHT tuning parameters,  $\eta$ ,  $\xi$ ,  $r$ , and  $m$ , as the *load* on the system increases. The load exercised by each node in the ON state can be expressed as  $\sigma_r q_d$  in MBytes/hour. Furthermore, we study the optimal tuning when the object size  $q_d$  is small compared to a large  $q_d$ , in order to highlight the difference between DHT applications with different object sizes. Towards this end, we fix a test case with a pool of  $n = 10,000$  nodes. We set the WRITE to READ ratio  $\gamma$  to 0.01, the average node session time  $\bar{u} = 0.5$  hours, the average node downtime  $\bar{d} = 5.25$  hours, the node bandwidth budget  $c_b = 100MB$  per hour, the node storage budget  $c_s = 10GB$ , and the number of data objects  $W = 10,000$ .

We compare two scenarios with object sizes,  $q_d$  of  $1MB$  and  $1.45KB$ , respectively. The choice of  $1MB$  is to emulate the large chunk size used in file-sharing systems like bitTorrent [28]. The choice of the small  $1.45KB$  object size is to contrast the  $1MB$  case [6] and is such that each object perfectly fits in one packet to factor out the overhead caused by packet fragmentation. In Figure 5 we plot the optimal system parameters,  $\eta$ ,  $\xi$ ,  $r$ ,  $m$ , the resulting DHT performance as reflected on the system throughput,  $B_1$ , the average sojourn time,  $\bar{s}$ , and the incurred messaging and maintenance bandwidth overhead,  $B_2 + B_3 + B_4 + \eta \cdot B_5$ , and the disk storage overhead,  $D$ , as we vary the load exercised by each node in the ON state on the system,  $\sigma_r q_d$ , between 1 to 200 MB per hour, both when  $q_d = 1$  MB and when  $q_d = 1.45$  KB scenarios. The curves in Figure 5 reveal the following conclusions: (1) As the load on the system increases, the model reacts by incrementally including more nodes (from the most stable ones) into the DHT set, increasing  $\eta$ , and improving the system throughput,  $B_1$ . The act of including more nodes (improving scalability) into the DHT set continues only up to a load value of about 60 MB/hour, when the bandwidth budget ( $\eta c_b$ ) is saturated, and adding more nodes does not help as the nodes that would be added are less and less stable, and would waste system bandwidth in maintenance traffic without improving  $B_1$ .  $\eta$  thus converges to a value (of 0.86 in this example) even for larger load values. (2) The fraction of objects stored in the DHT,  $\xi$ , starts to decrease once  $\eta$  stabilizes and the load further increases; in other words, if the system cannot gain more service capacity by adding more DHT nodes, then it needs to cut down on the number of maintained data objects, keeping only a fraction of the most

<sup>4</sup>We limit the range of some variables to realistic values, such as  $m \in [1, 100]$ , and also *quantize* non-integer variables to improve the search efficiency.

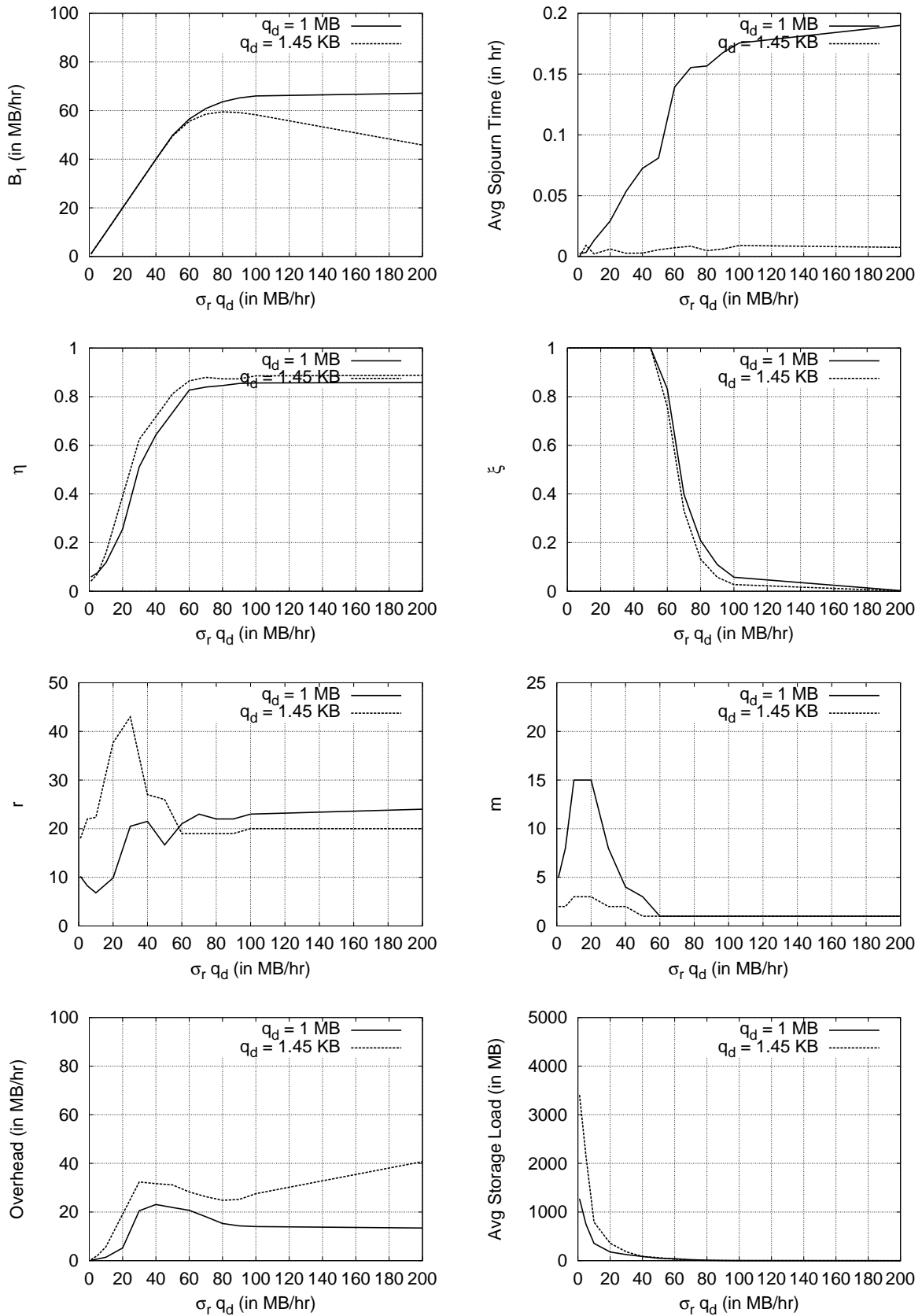


Fig. 5. Results of DHT model as we vary the workload on the system.

popular ones in order to allocate the DHT resources more efficiently. (3) As  $\eta$  grows, the DHT uses high redundancy parameters  $r$  or  $m$  to counter the node instability and to maintain the availability of stored DHT objects close to 1.0. Eventually  $r$  and  $m$  stabilize as well because the the number of DHT nodes is limited and maintaining a large number of replicas,  $r$ , and fragments,  $m$ , is costly. When the load stresses the bandwidth budget, because  $m$ 's improvement on  $A$  is marginal compared to  $r$ ,  $m$  is reduced to 1 and the system adapts the replication strategy in order to save querying and maintenance overhead. (4) Before the workload stresses the bandwidth budget, the case of  $q_d = 1.45KB$  favors the use of a higher degree of replication and therefore consumes more bandwidth and storage resources than the case of  $q_d = 1MB$  without offering a better throughput,  $B_1$ . As  $\eta$  saturates, the use of  $q_d = 1.45KB$  degrades the system throughput, even though it leads to smaller sojourn time. The reason for the high degree of data replication in the case of  $q_d = 1.45KB$  is that  $q_d = 1.45KB$  cannot exploit fragmentation effectively, since  $q_d = 1.45KB$  fits in one IP packet of 1500 bytes after including the IP header. Fragmenting this packet will introduce header and messaging overhead. (5) The average sojourn time per data service in both settings is far below the sojourn constraint of 0.5 hours, but as  $q_d$  is increased or the sojourn time constraint is lowered, the system responds by reducing  $\xi$  or increasing  $m$  to lower the average sojourn, either of which causes performance degradation. Also the storage load of the DHT at the point when  $\xi$  is about to drop below 1.0 is about  $50MB$  and the corresponding  $W$  is about  $10GB$  corresponding to  $10K$  objects of  $1MB$  each, and this implies that we can use a system storage budget of roughly  $W = (10GB/50MB)(10GB) = 2TB$  when the bandwidth budget is  $100MB$  per hour and the per-node storage budget is  $10GB$  in this setup. In general, the observed trends persist for different setups.

## IX. CONCLUSIONS AND FUTURE WORK

We made the case that careful design of DHT-based P2P systems can optimize their utility even in the presence of unstable members. The combination of node selection, content selection, and data redundancy strategies are major factors in DHT administration. While we have shown the evolution trends of the system parameters leading to the optimal DHT throughput, verifying these results in realistic P2P setups and applying them to administer DHT systems is a natural extension. Also, extending the current model beyond average case analysis, by studying appropriate distribution of replication parameters among nodes, and accounting for the heterogeneity in node capacity and storage, is another possible extension.

## X. ACKNOWLEDGEMENTS

We wish to thank Dr. George Rouskas, Dr. Douglas Reeves, and Dr. Rudra Dutta, and Mr. Shu Huang for their helpful discussions. We would also like to thank the INFOCOM '07 reviewers for their constructive input.

## REFERENCES

- [1] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area Cooperative Storage with CFS," in *Proc. of ACM SOSP*, Chateau, Banff, Canada, 2001.
- [2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "High-bandwidth Content Distribution in a Cooperative Environment," in *Proc. of IPTPS*, Berkeley, CA, USA, 2003.
- [3] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. of ACM MMCN*, San Jose, CA, USA, 2002.
- [4] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-peer File-Sharing Workload," in *Proc. of ACM SOSP*, New York, NY, USA, 2003.
- [5] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Uses," in *Proc. of ACM SIGCOMM*, Philadelphia, PA, USA, 2005.
- [6] "OpenDHT," <http://opendht.org/>.
- [7] "Planetlab," <http://www.planet-lab.org/>.
- [8] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," in *Proc. of ACM SIGCOMM*, Pisa, Italy, 2006.
- [9] J. Mickens and B. Noble, "Predicting Node Availability in Peer-to-peer Networks," in *Proc. of SIGMETRICS POSTER*, Banff, Alberta, Canada, 2005.
- [10] C. Blake and R. Rodrigues, "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two," in *Proc. of HotOS*, Lihue, Hawaii, USA, 2003.
- [11] R. Rodrigues and B. Liskov, "High Availability in DHTs: Erasure Coding vs. Replication," in *Proc. of IPTPS*, Ithaca, NY, USA, 2005.
- [12] H. Weatherspoon, B. G. Chun, C. W. So, and J. Wubiatowicz, "Long-term Data Maintenance: A Quantitative Approach," in *Technical Report UCB/CSD-05-1404*, Berkeley, CA, USA, 2005.
- [13] S. Rhea, B. Chun, J. Kubiatowicz, and S. Shenker, "Fixing the Embarrassing Slowness of OpenDHT on PlanetLab," in *Proc. of USENIX WORLDS*, San Francisco, CA, USA, 2005.
- [14] R. Rodrigues and C. Blake, "When Multi-hop Peer-to-peer Lookup Matters," in *Proc. of IPTPS*, La Jolla, CA, USA, 2004.
- [15] D. Nowell, H. Balakrishnan, and D. Karger, "Observations on the Dynamic Evolution of Peer-to-Peer Networks," in *Proc. of IPTPS*, Cambridge, MA, June 2001.
- [16] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *IEEE Transactions on Networking*, vol. 11, 2003.
- [17] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," in *Proc. of IEEE INFOCOM*, Hong Kong, China, 2004.
- [18] P. Godfrey and I. Stoica, "Heterogeneity and Load Balance in Distributed Hash Tables," in *Proc. of IEEE INFOCOM*, Miami, FL, USA, 2005.
- [19] D. Leonard, V. Rai, and D. Loguinov, "On Lifetime-based Node Failure and Stochastic Resilience of Decentralized Peer-to-peer Networks," in *Proc. of SIGMETRICS*, Banff, Alberta, Canada, 2005.
- [20] R. Bhagwan, S. Savage, and G. Voelker, "Understanding Availability," in *Proc. of IPTPS*, Berkeley, CA, USA, 2003.
- [21] L. Kleinrock, *Queueing Systems*. John Wiley and Sons, 1976, vol. 2.
- [22] J. Roberts, U. Mocchi, and J. Virtamo, *Broadband Network Teletraffic, Final Report of Action COST 242*. Springer, 1996.
- [23] R. M. Oliver, "Table of the Waiting Time Distribution for the Constant Service Queue (M/D/1)," *International Journal of Computer Mathematics*, vol. 2, pp. 35–56, 1968.
- [24] J. T. Virtamo, "Numerical Evaluation of the Distribution of Unfinished Work in an M/D/1 System," *Electronics Letters*, vol. 31, pp. 531–532, 1995.
- [25] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris, "Designing a DHT for Low Latency and High Throughput," in *Proc. of NSDI*, Berkeley, CA, USA, 2003.
- [26] S. Q. Zhuang, D. Geels, I. Stoica, and R. H. Katz, "On Failure Detection Algorithms in Overlay Networks," in *Proc. of IEEE INFOCOM*, Miami, Florida, USA, 2005.
- [27] S. Krishnamurthy, S. El-Ansary, E. Aurell, and S. Haridi, "A Statistical Theory of Chord under Churn," in *Proc. of IPTPS*, Ithaca, NY, USA, 2005.
- [28] "Bittorrent," <http://www.bittorrent.org/>.