Designing Less-structured P2P Systems for the Expected High Churn

Fabián E. Bustamante and Yi Qiao
Department of Electrical Engineering & Computer Science
Northwestern University, Evanston, IL 60208, USA
Email: {fabianb,yqiao}@cs.northwestern.edu

***Abstract*—We address the problem of highly transient populations in unstructured and loosely-structured peer-to-peer systems. We propose a number of illustrative query-related strategies and organizational protocols that, by taking into consideration the expected session times of peers (their lifespans), yield systems with performance characteristics more resilient to the natural instability of their environments. We first demonstrate the benefits of lifespan-based organizational protocols in terms of end-application performance and in the context of dynamic and heterogeneous Internet environments. We do this using a number of currently adopted and proposed query-related strategies, including methods for query distribution, caching and replication. We then show, through trace-driven simulation and wide-area experimentation, the performance advantages of lifespan-based, query-related strategies when layered over currently employed and lifespan-based organizational protocols. While merely illustrative, the evaluated strategies and protocols clearly demonstrate the advantages of considering peers' session time in designing widely-deployed peer-to-peer systems.**

***Index Terms*—Lifespan, session time, resilience, peer-to-peer.**

I. Introduction

Peer-to-peer (P2P) computing can be defined as the sharing of computer resources and services by direct exchange between the participating nodes. Since Napster's [2] introduction in the late 90s, the area has received increasing attention from the research community and the general public. Peers in P2P systems typically define an overlay network topology by keeping a number of connections to other peers, their "friends," and implementing a maintenance protocol that continuously repairs the overlay as new members join and others leave the system.

Due in part to the autonomous nature of peers, their mutual dependency, and their astoundingly large populations, the transiency of peers (a.k.a. churn) and its implications on the overall system's performance have recently attracted the attention of the research community [3]–[12]. A well-accepted metric of churn is node *session time* – the time from the node's joining to its subsequent leaving from the system. [1] Measurement studies of deployed P2P systems have reported *median session times* varying from one hour to one minute [6], [8], [13].

An earlier version of this paper appeared in IEEE P2P 2005 [1].

[1]We employ *lifespan* and *session time* interchangeably. Another metric of transiency sometimes used, *lifetime*, refers instead to the the time between the node first entering the system and its final departure from it [6].

The implications of such degrees of churn on the system's performance are directly related to the degree of peers' investment in their friends. At the very least, the amount of maintenance-related messages processed by any node would be a function of the degree of stability of the node's neighboring set. Beyond this, and in the context of content distribution P2P systems, the degree of replication, the effectiveness of caches, and the spread and satisfaction level of queries will all be affected by how dynamic the peers' population is.

Our work addresses the problem of highly transient populations in unstructured and loosely-structured P2P systems (collectively, *less-structured P2P systems*). Through active probing of over half-a-million peers in a widely-deployed P2P system, we determined that the session time of peers can be well modeled by a Pareto distribution. In this context, the implication is that the expected remaining session time of a peer is directly proportional to the session's current length, i.e. the peer's *age*. This observation forms the basis for a new set of protocols for peer organization and query-related strategies that, by taking into consideration the expected session times of peers, yield systems with performance characteristics more resilient to the natural instability of their environments.

The lifespan-based approach for organizational protocols was first proposed in our position paper [8], where we show its effectiveness in terms of increased system stability (e.g. over 42% reduction on the ratio of connection breakdowns and their associated costs) through a simulation study. In this article we first demonstrate the advantages of our approach in terms of application performance in a dynamic Internet testbed of 150 widely distributed PlanetLab nodes. We do this using a set of illustrative organizational protocols combined with a number of currently adopted and proposed query-related strategies, including methods for query distribution, caching and replication. Our results show that even simple lifespan-based overlays can significantly boost query performance (with improvements of at least 57% on aggregated query hits and 50% reduction in query resolution time) and increase system scalability by achieving query performance comparable to those of currently deployed protocols with only a third of their load.

We then apply similar ideas to query-related strategies. Through trace-driven simulation and wide-area experimentation we demonstrate the performance advantages of lifespan-based query-related strategies when layered over currently employed organizational protocols as well as when used in combination with their lifespan-based variations. Our results show that lifespan-based strategies can generate over 2-5 times more query hits than alternative strategies. While merely illustrative, the evaluated protocols and strategies clearly demonstrate the advantages of considering peers' session time in designing widely-deployed peer-to-peer systems.

The rest of this paper is structured as follows. Section II provides a brief background on relevant P2P systems, protocols and strategies. Section III and IV present results from our study on peers' lifespans and discuss lifespan-based organizational protocols and query-related strategies. Section V describes our evaluation methodology and presents results from both trace-based simulations and wide-area experiments. We discuss some related work in Section VI and conclude in Section VII.

II. Background

The structure of a peer-to-peer system is defined by its organizational protocol. In unstructured (UDP) and loosely-structured or hierarchical P2P (HDP) systems[2], peers define the P2P network overlay through connections with other (mostly randomly) chosen peers. While organizational protocols for unstructured systems, such as Gnutella v0.4 [15], consider all peers as equals, protocols for loosely-structured systems, like Gnutella v0.6 and Kazaa [16]–[18], commonly define a two-level hierarchy distinguishing between common "leaf" peers and well provisioned super-peers.

Connected peers interact with each other by exchanging various types of messages, most of which are broadcasted or back-propagated. Broadcasted messages are sent on to all other peers to which the sender has connections. Back-propagated messages (e.g. replies) are forwarded on the reverse of the path taken by an associated message (e.g. query). In addition to queries and replies, discussed in detail in the following subsection, other types of messages include object transfer and group membership messages such as *ping*, *pong* and *bye*. *Pings* are used to discover hosts on the network. Pings are answered with *pong* messages, containing information (such as contact information and resources shared) about the responding peer and about others than the peer knows about. Information on neighboring peers can be provided either by creating pongs on their behalf or by forwarding the pings and back-propagating their replies. *Byes* are optional messages used to report the closing of connections.

A. Query, Caching and Replication

A key component of resource sharing P2P systems is their search or query mechanism. In highly structured (DHT-based) systems, the search for an object based on its *object identifier* is a relatively easy task, thanks to the strictly controlled placement of objects. In less structured P2P systems, however, the location of an object is independent of the system's topology, leaving peers with only "near blind" query strategies [19]. We review some of these strategies for less structured systems next, including currently used and other proposed techniques for query distribution, caching and replication.

[2]We use the classification proposed by Lv et al. [14]

The earliest and simplest query strategy is *flooding*, where a query is propagated to all neighbors within a certain radius. Addressing flooding's inherent scalability problems, Lv et al. [14] propose *k-random-walks*, where a set of parallel query messages (walkers) are independently forwarded to randomly chosen neighbors at each hop, significantly reducing the number of messages in the network. A number of improvements to the basic strategy have been suggested [5], [20], [21]. Adamic et al. [20] propose using random walk in power-law topologies with walks biased toward high-degree nodes. While this can significantly improve query performance, it could also result in overloaded nodes. Lv et al. [21] and Chawathe et al. [5] suggest taking node capacity into consideration through biased random walks, directed toward high-capacity peers.

To further boost query performance, different strategies for index caching have been proposed, including *Path Caching with eXpiration (PCX)* and *Neighbor Caching with incremental Update (NCU)*. With PCX, each node in the system maintains an index cache, with each entry being a `(key, value)` pair [22]. The `value` in the pair is normally a pointer to the node holding a replica of the object associated with the corresponding `key` [23], [24]. Upon receiving a query message, the node not only checks its own shared objects, but also scans the cache for entries with matching keys. Upon a successful match, the node replies with the associated pair(s). PCX can be used in both DHT-based and less structured systems to improve query results. With NCU [5], [18], [20], each node maintains caches of metadata for all of its neighbors. As with PCX, a node sends a query hit either on its own behalf or on behalf of its neighbors, effectively increasing the reach of a query [25], [26].

Replication is a common approach to improve performance when distributed systems need to scale in numbers of users, objects in the system and geographical area. The most commonly used file replication strategy in P2P systems simply makes replicas of objects on the requesting peer, upon a successfully query/reply. Beyond this, a number of proactive replication strategies that aim at improving query hits while reducing response time have been proposed. Cohen and Shenker [19] modeled various explicit replication strategies; they found square-root replication, which can be efficiently achieved using path replication, to be optimal.

III. Transient Populations and P2P Systems

There has been a number of studies of peers' participation and transiency in P2P systems [8], [13], [25], [27]–[30]. We performed an independent study [8] of peers' lifespans in the widely deployed Gnutella network (v0.6 [18], i.e. with super-peers), collecting over 1 million peer session times for over half-million peers. The next subsections describe our measurement experiment and the characterization of the distribution of peers' session times.

A. Collecting Observed Peers' Lifespans

To actively measure the lifespan of peers in Gnutella, we modified an open source Gnutella client[3] to keep track of

[3]Mutella: *http://mutella.sourceforge.net*

every peer found and periodically check its availability. Our monitoring peer maintains a hash table of peers it has seen so far. Each entry in the hash table includes fields for (1) peer's *IP:port* pair, (2) node type (leaf- or ultra-peer), (3) time of birth (TOB), (4) time when found (TWF), and (5) time of death (TOD).

On each iteration, the monitoring peer updates the existing entries and inserts newly found peers. To avoid potential measurement errors, each of our probes tries to establish an application-level connection checking for specific Gnutella packet headers. Since it knows with certainty only the TOB of previously known and reborn peers, live peers found for the first time are included with only the TWF field set to the current time. A peer is considered dead when a connection attempt fails (i.e. a third try times out; we use the default timeout value of 10 seconds) or an unexpected response is received. Table I summarizes the strategy used for updating peer lifespan information. For example in the fourth case in the table (in italics), if a peer was found dead in the previous scan and alive in the current one, its TOB is set to the current time.

Last Scan	**Current Scan**	**Action**
Unknown	Dead	None
Unknown	Alive	TWF = *T*
Dead	Dead	None
Dead	*Alive*	*TOB* = T
Alive	Dead	TOD = *T*
Alive	Alive	None

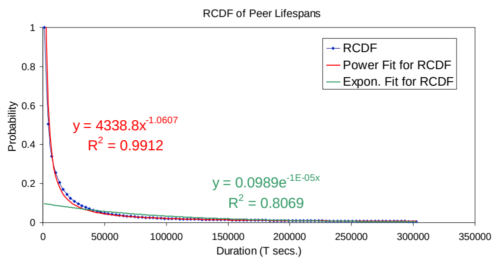
TABLE I
STRATEGY USED FOR UPDATING THE PEER TABLE IN EACH ITERATION (*T*: CURRENT TIME).

To scale monitoring and generate lifespan measurements with sufficient granularity, we evenly distribute the peer table (based on the hash values of peers) over 20 monitoring peers running across 17 hosts. This approach allowed us to achieve a granularity of 1,300 seconds (about 21 minutes) when scanning over 30k to 40k entries per client.

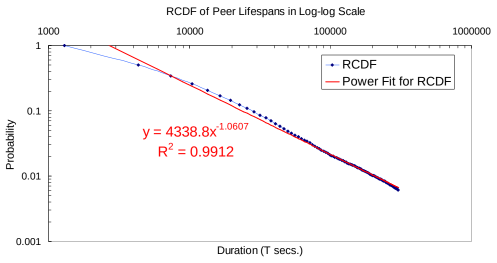
B. *Peer Lifespan Distribution*

We measured the lifespans of more than 500,000 peers continuously between March 1st and 8th, 2003. To account for the fact that sessions may be active (or inactive) for times longer than our sampling duration, we resort to the *create-based method* [13]: we divide the captured trace into two halves and report lifespans only for sessions started in the first half. If a session ended during either the first or second half, we can obtain its lifespan by subtracting the starting time from the ending time; if a session was still active at the very end of the trace, we get a lower bound for its lifespan, which is larger than half the trace length, i.e. 3.5 days. This method provides accurate information about the distribution of lifespans for sessions that are shorter than half the trace, as well as percentage of sessions that are even longer.

Figure 1 presents the Reverse Cumulative Distribution Function (RCDF) of peers' observed lifespans shorter than 3.5 days (and longer than 1,300 seconds). The lifespan distribution is presented in both normal axes (a) and log-log scale (b). The distribution in the log-log scale plot can be approximated by a straight line, indicating that the peer lifespan distribution can be modeled by a Pareto distribution of the form λT^k ($k < 0$). More precisely, the probability of a session exceeding T seconds is λT^k. The R^2 value higher than 0.99 verifies the very high goodness of fit of the model. In contrast, the exponential curve fails to model the observed data with a R^2 value of only 0.80.



(a) Normal plot.



(b) Log-Log plot.

Fig. 1. Distribution of lifetimes in Gnutella over a period of 7 days. The two additional lines in subfigure (a) show two attempts (a Pareto distribution and an exponential curve) to fit a curve to these data. Subfigure (b) shows the same distribution on a log-log scale; the straight line here indicates that the distribution can be modeled by λT^k, where the constant k is less than zero and proportional to the slope of the line.

The Pareto distribution belongs to the UBNE (Used-Better-than-New-in-Expectation) class of distributions. In this context, this means that the expected remaining session time of a peer is directly proportional to the session's current length. Our work [8] was the first one characterize the distribution of peer session times. The next section presents a number of lifespan-based organizational protocols and query-related strategies for loosely-structured P2P systems based on this observation.

IV. LIFESPAN-BASED PROTOCOLS AND STRATEGIES

We first provide an overview of lifespan-based organizational protocols [8]. The basic idea behind these protocols is to dynamically increase the system's dependency on a node as the node's long-term commitment to the community becomes clear. This can be achieved by simply giving preference to peers with longer expected sessions times. Given the UBNE nature of lifespan distribution [8], a peer's current age is a fair estimate of its session time.

Similarly, existing query-related strategies can be easily modified to incorporate lifespan-based ideas. We then present a number of query-related strategies that rely on lifespan to steer query walkers and decide on replica placement and cache eviction.

We close the section by outlining a lightweight distributed protocol for peers' age discovery [1].

A. Peer Lifespan and P2P Protocols

In most P2P protocols, there are at least two instances where peers need to choose among "acquaintances": (1) when deciding who to befriend and (2) when needing to respond to a third-party's request for references. In Gnutella-like protocols, for example, the first group would be contacted for connection requests and the second one would be included in *pong* replies to *ping* messages.

Peers normally keep a number of other peers as neighbors by accepting an upper-bounded number of incoming connections and trying to maintain a lower-bounded number of outgoing ones. To cope with the dynamic changes in P2P user population, most systems implement some kind of maintenance or recovery protocol that continuously repairs the overlay as nodes join and leave the network. Nodes joining the network use a number of control messages to let others know of their arrival. The departure of a node is noticed by its neighbors through periodic monitoring.

Our lifespan-based approach can be equally applied to the organization of unstructured and loosely structured protocols. In our position paper [8] we described its use in a set of illustrative unstructured protocols: a basic protocol that employs estimated lifetime when selecting new neighbors (*LSPAN-1*), a slightly more refined protocol that relies on lifespan ideas for both recommendation and selection (*LSPAN-2*), and a third protocol that takes the estimated number of available connections into consideration to better distribute connection attempts among candidate peers. The following paragraphs describe each of these protocols and Table II highlights their main aspects.

Lifespan-Based Friend Selection (LSPAN-1): Our first protocol takes peers' observed lifespans into consideration only when deciding with whom to open a connection. Peers piggy-back their own birth time in their ping messages and propagate other peers' birth times with their replies. When a peer needs to open a new connection, e.g. after the departure of a friend, it simply selects the oldest known peer as its new partner. Notice that the selection process incorporates some degree of randomness. While a peer chooses the oldest peer(s) from among those it knows of, this group is made from the random set of recommendations forwarded by other peers in the network.[4]

Lifespan-Based Friend Selection and Recommendation (LSPAN-2): LSPAN-2 uses lifespan in both opportunities: when selecting peers for connecting and when generating a response to a third-party's request for references. From the perspective of the peer trying to open a new connection, this ensures that the set of potential friends is made of long-lived peers. The two protocols introduced so far would blindly favor older peers and will naturally result in an increase in the number of connection attempts made to them. Since the actual number of incoming connections that a peer can accept is typically bounded by its maximum number of incoming connections, our last protocol considers the estimated number of available incoming connections of a peer when selecting who to connect to.

Taking Available Connection into Consideration (LSPAN-3): LSPAN-3 uses a weighted credit selection scheme that incorporates both criteria: the peer's current age and the estimated number of available incoming connections. The estimated number of available incoming connections is the difference between the optimal and current number. The optimal number of incoming connections is upper-bounded, and its value at a given point in time lies between a half and three-fourths of the maximum number of incoming connections, depending on the peer's age (the older it is, the larger the optimal incoming connection number). In deployed P2P systems we expect to find a positive correlation between the lifespan of a peer and its maximum number of connections: peers behind a modem can only support very limited connections to others, and tend to remain online for very short times, while peers using T1/T3 connections will have a larger maximum number of connections and often stay active for several days [13]. Correspondingly, the number of maximum connections allowed by a given peer in our protocols is related to the peer's current session time. For our experiments this number ranged between 5 and 50, with an average value of 20.[5]

Protocol	**Connect?**	**Recommend?**
LSPAN-1	Oldest	Random
LSPAN-2	Oldest	Oldest
LSPAN-3	Oldest & more avail. connections	Random

TABLE II
LIFESPAN-BASED PROTOCOLS AND HOW THEY SELECT AMONG CANDIDATE NEIGHBORS AND RECOMMENDATIONS.

We first demonstrated the advantages of a lifespan approach to organizational protocols, showing its effectiveness in terms of increased system stability through a simulation-based study [8]. Using one of the 20 traces collected,[6] we evaluated our three illustrative protocols and compared them with two decentralized protocols. The alternative protocols used for comparison are closely based on Gnutella- and Kazaa-like protocols: *Unstructured Decentralized Protocol (UDP)* is based on an improved version of Gnutella v0.4 [15] and *Hierarchical Decentralized Protocol (HDP)* is modeled after hierarchical protocols that rely on ultra- or super-peers [16] such as Kazaa [17] and Gnutella v0.6 [18]. We heavily relied on the specifications and available RFCs. When the specifications were vague or unavailable, we resorted to our understanding of (open-source) clients currently in use and

[4]Recommendations are obtained through the ping/pong message exchange already described.

[5]We also evaluated the performance of our protocols following the node capacity distribution model suggested by [5], where each node is assigned a maximum connection number based on its capacity class. Nodes' capacities are assigned arbitrarily and have no correlation with the nodes' session times. The results, available upon request, are comparable to those included in the article.

[6]Simulations using the remainder traces yield similar results.

other publicly available documents.

Our experiment run for a total simulation period of 510,000 seconds (about six days), capturing the lifespan of 36,577 peers. Each simulation starts "cold," i.e. without any peer. The number of peers in the system increases during the first day and stabilizes for the remaining time, varying between 700 and 1,000 at any given point. The results reported exclude this warm-up period (∼80,000 sec.). We now quote some of our results; for the complete set, we direct the reader to our position paper [8].

A good indicator of the effect of a protocol on system stability is the ratio of connection breakdowns to the number of effective connections. Table III shows this ratio for every one of the protocols discussed, in terms of a few statistics including average, standard deviation, maximum and minimum. As can be observed, all lifespan-based protocols yield lower ratios of connection breakdowns than random-based protocols (UDP and HDP), a natural result of the UBNE property of peers' lifespan distribution and the former protocols' preference for older peers. The most selective lifespan-based protocol, LSPAN-2, naturally gives the lowest ratio of connection breakdowns over time, with reductions of 42% by comparison with that of UDP and HDP. LSPAN-1 and LSPAN-3 yield comparative savings of 26% to 28% (by contrast with UDP) in the ratio of connection breakdowns in the system and their associated costs.

Protocol	**Avg (%)**	**Std (%)**	**Min (%)**	**Max (%)**
UDP	7.318	1.632	4.142	10.894
HDP	7.533 (-2.9)	1.756 (-7.6)	4.159 (-0.4)	11.645 (-6.9)
LSPAN-1	5.385 (26.4)	1.155 (29.2)	3.101 (25.1)	7.793 (28.5)
LSPAN-2	4.262 (41.8)	0.951 (41.7)	2.443 (41.0)	6.104 (44.0)
LSPAN-3	5.235 (28.5)	1.103 (32.4)	3.017 (27.2)	7.360 (32.4)

TABLE III
RATIO OF CONNECTION BREAKDOWNS TO NUMBER OF EFFECTIVE CONNECTIONS OVER TIME (AGGREGATED OVER 10,000 SEC) AND REDUCTION PERCENTAGE RESPECT TO UDP.

As we previously point out, the peers' preference for long-lived neighbor candidates could result in a high overall number of connection rejections by these long-lived peers. While this is true for our most selective protocol LSPAN-2, LSPAN-1 and LSPAN-3 yield virtually insignificant increases with averages number of connection rejection per peer of 0.859 and 0.158 for every 10,000 seconds, respectively. In other words, under LSPAN-3 a peer will face, on average, one connection rejection every 17.58 hours. In the remainder of this article we employ LSPAN-3 as our lifespan-based unstructured protocol and denote it as *LUDP*.

Lifespan-based ideas can be equally applied to loosely-structured systems, where super-peers are placed at the highest layer of the network and given greater responsibilities toward the community than leaf peers. As with LUDP, super-peers can give preferences to older super-peers when setting up new connections, while leaf peers could, with higher probability, opt for older super-peers when deciding to which node to connect. We use *Lifespan-based HDP*, or *LHDP* to denote this lifespan-based, loosely-structured organizational protocol.

B. Query-Related Strategies

As with organizational protocols, existing query-related strategies can be easily modified to incorporate lifespan-based ideas. We now describe in detail various illustrative lifespan-based strategies for query distribution, caching and replication.

Query dissemination: In the original *k-random-walks* query strategy [14], each visited node randomly picks the next peer where to forward the query walker. While offering good scalability, this "purely blind" approach is oblivious to peers' properties or past history. This basic random walk strategy could be easily extended to give preference to those peers with estimated longer session times when guiding the forwarding of a query walker. Depending on the weight that a peer's estimated session time plays in the forwarding decision, a naïve algorithm could increase the chance of "collision" between different walkers. Collisions will reduce the effectiveness of the approach and can even result in the creation of hotspots at longer-lived peers. For our evaluation we adopt a simple weighted probabilistic approach which has shown to be highly effective while avoiding the aforementioned problems.

Caching: Although directly applicable, the effectiveness of PCX (Path Caching with eXpiration) in less structured P2P systems is unclear, as different searches for the same object may take different paths than previous ones, negating the benefits of caching. Thus, we extend PCX to cover a broader region around the path yielding what we call *Regional Caching with Expiration (RCX)*. Under RCX, a peer routing a query hit message back to the requester will also push the query hit entry (an `<object-ID, hit-peer-ID>` tuple) toward some of its neighbors' caches. Pushing cache indexes with higher probability to older peers can increase the number of queries answered based on these cached entries.

Given the transiency of peer populations, cached entries must be expired after relatively short times to reduce the number of stale ones. For this we can employ a cache expiration technique based on similar ideas – the eviction policy can consider the estimated session-length of the peer referred to in the cache entry in determining the maximum age of a given entry. We have found this strategy to be significantly more effective than the straightforward approach of simply setting a constant maximum age for all cache entries and periodically removing those exceeding it.

Replication: Replication also plays an important role in improving the performance of queries. By replicating files at some intermediate peers along the query path, subsequent queries can be resolved in a more efficient manner. The most straightforward form of proactive replication "leaves" copies of the requested object along the paths taken by query or query hit messages. As with PCX caching, the effectiveness of this approach in less structured P2P systems is unclear given that different searches for the same object may take different paths. Consequently, we modify the path replication approach slightly by leaving copies of the requested objects on some neighbors of each peer along the query/query-hit paths – we refer to this strategy as *regional replication* ($RRep$). Regional replication can easily incorporate lifespan-based ideas by opting for nodes with longer estimated session times as target recipients of object copies. These replicas would be more likely to remain online longer, potentially serving a large number of requests. As with our previously described organizational protocols and query strategies, we use an age-weighted, probabilistic approach to select the target peers for replication. Clearly, a node could always constrain the number of replicas it is willing to host on behalf of others.

C. *Determining Peers' Ages*

The effectiveness of the proposed lifespan-based approach depends in part on the fitness of our session length estimators and the accuracy of peers' age information. The high goodness of fit of our model ensures the former [8]. To improve the latter, we have designed a *lightweight distributed protocol for peers' age determination* based on previous work on reputation [31]–[33].

Assume a system composed of mostly selfish peers. Before a given node can decide who it should attempt connecting to, it must first determine the age of a set of candidate peers. To this end, each peer in the system keeps track of other peers with whom it has interacted (through a connection request, a ping/pong or a query/reply exchange) and the time of their earliest and latest interactions. When a given peer, P, wants to determine the age of a candidate peer C, the following three-phase protocol can be used:

- *Phase 1: Witness Collection:* P first requests from C a list of the peers that C has known the longest and with whom C has interacted most recently. Peers in this list potentially serve as witnesses of C's age.
- *Phase 2: Witness Sampling and Trimming:* To reduce the chances of collusion, P first trims proposed witnesses with suspiciously large interaction windows (outliers), before sampling a subset of the remaining peers to construct the final witness list.
- *Phase 3: Collecting Testimonies and Determining Age:* In the final phase, P verifies the interaction times C reported by directly contacting all peers in the final witness list and determines C's age as a function (e.g. minimum or median) of the collected testimonies, i.e. the verified interaction windows.

The protocol has a number of properties that improves its resilience to cheating. The age of a peer is never directly requested from the peer itself, but determined through the collected testimonies of randomly chosen witnesses. In addition, the trimming of outliers helps in reducing the probability of small cabals. Although the value determined by our protocol may not exactly match the "real" age of the peer in question, it is sufficient for our purposes as our protocols rely less on the real age of a peer than in its relative seniority among other candidate peers.

V. Evaluation

We evaluate the advantage of the proposed lifespan-based approach to both query-related strategies and organizational protocols through simulations and wide-area experiments in PlanetLab. We compare this approach against currently deployed and proposed strategies and organizational protocols. The goal of this evaluation is to determine the efficacy of the proposed approach at improving system stability and, ultimately, enhancing the performance of applications.

A. *Query-Related Strategies and Organizational Protocols*

For our evaluation, we implemented two random-based (*Unstructured Decentralized Protocol (UDP)* and *Hierarchical Decentralized Protocol (HDP)*) and two lifespan-based organizational protocols (*LUDP* and *LHDP*) as well as a number of strategies for query routing, caching and replication. For query routing, we use the original *k-random-walk* query strategy [21] and a lifespan-based variation (denote as *RQuery* and *LQuery*, respectively). For caching we implemented *Path Caching with Expiration (PCX)*, *Neighbor Caching with incremental Update (NCU)* and *Regional Caching with eXpiration (RCX)*. For PCX and RCX, we set the maximum number of object identifiers to 300 and the maximum number of node identifiers per object to 10. We denote our basic random regional caching as *RRCX* and its lifespan-based counterpart as *LRCX*. We use three different replication strategies including its most basic form (denoted as *SRep* for "Simple Replication") and the random and lifespan variations of *Regional Replication* (*RRRep* and *LRRep*, respectively). For both LRRep and RRRep, we set an upper-bound on the number of replicas a peer can hold to be 10. Table IV provides a quick summary of of the different protocols and strategies used throughout the rest of the paper and their associated acronyms.

B. *Metrics*

The effectiveness of the proposed approach is evaluated in terms of the performance improvements to query-related tasks, as captured by three metrics: Query Resolution Time, Query Hits and Z Query Satisfaction. *Query Resolution Time* is the time between query submission and the arrival of the first reply. *Query Hits* is simply the number of query hits associated with a given query. We also analyze the aggregated query hit number for all queries issued during each experiment. *Z Query Satisfaction* [26] is the percentage of queries achieving Z satisfaction, i.e. obtaining at least Z query hits.

C. *Simulation and Wide-Area Experimental Setup*

For the simulation study we employ an in-house, event-based simulator for P2P systems with support for all membership management related functionalities as well as the evaluated query distribution, caching and replication mechanisms. We ran simulations using 4 of the 20 traces collected,[7] with a total simulation time of 511,000 seconds, capturing the lifespan of 150,033 peers. At any time during a simulation run, there are around 3,000 to 4,000 active peers in the system.

For our wide-area evaluation, we implemented lifespan-based protocols and strategies as extensions to an open source Gnutella client [34], thus inheriting all the expected functionality of a typical P2P file-sharing system. As mentioned

[7]Simulations using the remainder traces yield similar results. On the other hand, due to memory and CPU limitations of the physical machines, for some of our simulation scenarios, it was prohibitively expensive to apply more than 4 traces simultaneously.

Organizational Protocols		
	Random	**Lifespan**
Unstructured	Unstructured Decentralized (UDP)	Lifespan UDP (LUDP)
Loosely structured	Hierarchical Decentralized (HDP)	Lifespan HDP (LHDP)
Query-related Strategies		
	Random	**Lifespan**
Query	k-random walk (RQuery)	Lifespan k-random walk (LQuery)
Caching	Regional Caching with eXpiration (RRCX)	Lifespan RCX (LRCX)
	Path Caching with eXpiration (PCX)	
	Neighbor Caching with incremental Update (NCU)	
Replication	Regional Replication (RRRep)	Lifespan RRep (LRRep)
	Simple Replication (SRep)	

TABLE IV
LIST OF ORGANIZATIONAL PROTOCOLS AND QUERY-RELATED STRATEGIES AND THEIR ACRONYMS.

earlier, we use random- and lifespan-based *k-random-walk* instead of the original flooding in Gnutella as query strategies. We deployed and ran our system using 150 stable PlanetLab nodes, distributed throughout the world. At any time during an experiment, the number of active peers in the whole system ranges between 200 and 300, evenly mapped to the set of PlanetLab hosts. Peers' lifespans are sampled from our collected traces. Each experiment lasts for 511,000 seconds, or about six days. To ensure that peers of the different protocols and/or strategies were exposed to the same network conditions and host load as their counterpart for fair comparison, *all experiments run the compared configurations concurrently.*

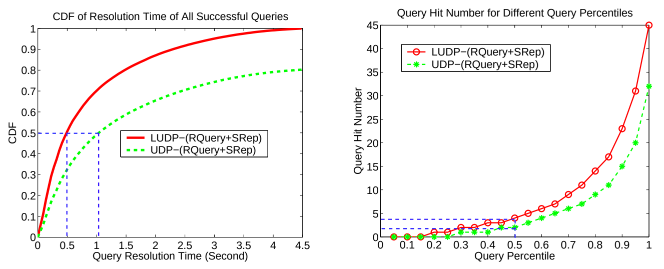
Several studies [35], [36] have shown that object popularity in P2P systems and the Web follows a Zipf-like distribution, where the probability of the i-th most popular object being queried is directly proportional to $1/i^a$. For our evaluation, the popularity of an object is reflected in both the number of queries issued for it and its degree of replication. We set a to 0.6 for the query distribution, and use an object population of 3,000 for all simulations and 500 for our wide-area evaluation. [8] Every new peer brings with it a number of copies of objects (currently two), selected according to the same Zipf-like distribution of object popularity. In our evaluations we obtained a similar "fetch-at-most-once" behavior and query distribution to that reported in [30] by making peers issue queries only for files they do not yet have.

In simulation, unless explicitly stated, we use 4 query walkers per query, with a TTL value of 20 for each walker. In the wide-area, each query consists of three walkers with a TTL value of seven for each. Each active peer issues a query every 600 seconds on average, both in simulation and in wide-area experiments.

D. Simulation Results

1) Organizational Protocols: We first examine the advantages of the lifespan-based approach to organization. To better understand the effects of lifespan-based organizational protocols, the first three sets of simulations isolate the contributions of caching and replication to query performance. We first evaluate the benefits of this approach under simple replication (SRep) without caching. We then demonstrate its advantages employing two different caching strategies, PCX and NCU, respectively, but without replication. The fourth set of experiments show results with both caching and simple replication $(SRep + NCU)$ enabled. These experiments are mainly discussed in unstructured systems, followed by more results in loosely-structured ones.

[8]Object population defines the number of *distinct* objects in the system, not the *total* number of objects which counts each replica of the same objects.



(a) CDF for query resolution time.

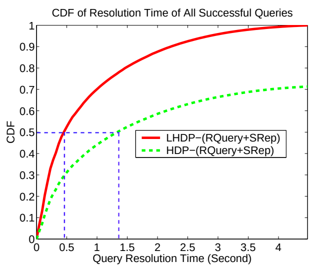
(b) Query hit number for different query percentiles.

Fig. 2. Query performance for UDP and LUDP using simple replication (SRep) and no caching.

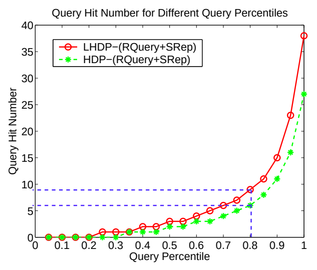
Figure 2 shows the Cumulative Distribution Function (CDF) of query resolution time (Fig. 2a) and query hit number at different query percentiles (Fig. 2b) using simple replication (SRep) and no caching. The advantage of a more stable overlay in LUDP is clear from both graphs: Figure 2a shows a reduction of between 50% and 70% in query resolution time with the lifespan-based LUDP in contrast to UDP, while Fig. 2b demonstrates 50-75% increase in the query hit number at various query percentiles. As an example, the dashed lines in Fig. 2a show that 50% of the queries can be resolved in 0.5 seconds for LUDP while it takes UDP over 1 second to resolve the same percentage of queries. Similarly, Fig. 2b shows a median query hit number (corresponding to 50 percentile of queries) of only 2 for UDP, and as many as 4 for the LUDP case. The lifespan-based protocol results in considerably larger aggregated query hit number than UDP (57% more) and higher query satisfaction at different satisfaction levels (Table V). Note that all numbers in Table V are relative improvements of LUDP over UDP. The first simulation scenario - using simple replication (SRep) and no caching - corresponds to the first data row in Table V.

For the second and third simulation scenarios, using PCX or NCU caching strategies without replication, LUDP also shows clear advantages over UDP, reducing query resolution time by 40% to 70% for different percentages of queries. We omit the graphs for PCX or NCU for brevity.

Table V offers the complete picture, showing the relative aggregated query hit number with different combinations of replication and caching strategies running over LUDP. UDP is used as the comparison baseline. Note that when using PCX, LUDP does not result in higher query hits but in faster query resolution. NCU with LUDP, on the other hand, yields faster query responses and 22% more query hits than with UDP.



(a) CDF for query resolution time.

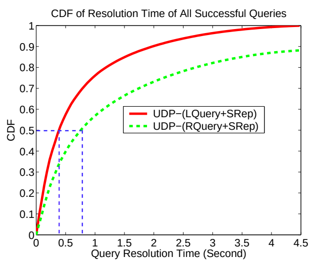


(b) Query hit number for different query percentiles.

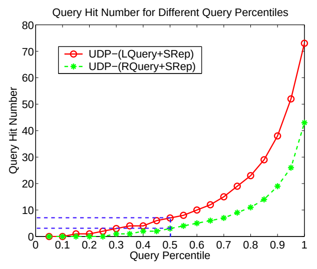
Fig. 3. Query performance for LHDP and HDP using simple replication (SRep) and no caching.

Replication	Caching	Aggregated Query Hits	Query Satisfaction (Z)		
			5	10	20
SRep	None	1.57	1.21	1.50	1.65
None	NCU	1.22	1.13	1.18	1.21
None	PCX	1.00	1.00	1.00	1.00
SRep	NCU	1.67	1.15	1.22	1.37

TABLE V
RELATIVE PERFORMANCE COMPARISON FOR UNSTRUCTURED DECENTRALIZED ORGANIZATIONAL PROTOCOLS; LIFESPAN-BASED (LUDP) OVER RANDOM-BASED (UDP).



(a) CDF for query resolution time.



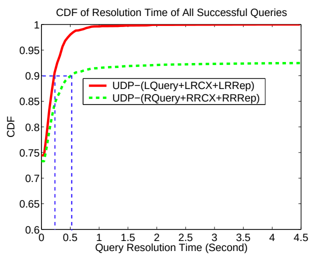
(b) Query hit number for different query percentiles.

Fig. 4. Query performance for lifespan-based (LQuery) and random-based (RQuery) query using simple replication (SRep) and no caching.

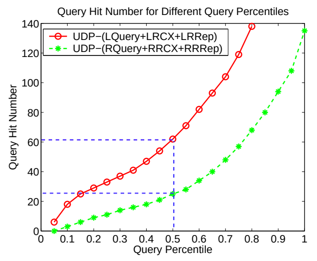
Recall that our lifespan-based protocols differ from UDP (HDP) mainly at the time of connection establishment, where a peer gives older peers (super-peers) higher priority when selecting neighbors. Thus, although the paths that query walkers take in both LUDP and UDP are purely random, a walker in LUDP still has a better chance of meeting long-lived peers along its path. In our experiments with simple replication, long-lived peers are more likely to store more shared objects than short-lived peers. For the second set of experiments, using PCX and no replication, long-lived peers contain more valid cache indexes than short-lived ones and can thus respond to more queries on behalf of other nodes, resulting in faster replies. In the case of NCU, long-lived peers have a better chance of being connected with more neighbors and thus contain more metadata for shared objects, resulting in higher query hit ratios, a larger number of hits per query, and shorter resolution time. Finally, independent of the query, caching and replication strategies, the more stable LUDP overlay also reduces the chance of link breakdowns along the query path, thus guaranteeing safer delivery of a query message and its associated query hits.

All the results presented so far have intentionally factored out some components of the query mechanism to better understand the effects of our approach. We also conducted experiments with both object simple replication and caching (SRep+NCU) enabled. Our results show the advantages of the lifespan-based approach in terms of system scalability. For example, query performance of *k random walk* with 2 walkers for LUDP are almost identical with that of 5 walkers for UDP. In other words, LUDP achieves similar performance as UDP while reducing the number of required walkers per query and its associated costs by almost 60%.

So far we have been looking at lifespan-based organizational protocols for unstructured P2P systems. Illustrating the benefits of lifespan-based protocols for loosely-structured systems, Fig. 3 shows query performance of the lifespan-based, loosely-structured LHDP system versus the alternative HDP system. As the figure shows, 50% of the queries can be answered in about 0.45 seconds with LHDP, while they take over 1.3 seconds with HDP. LHDP has a query hit number of 9 for 80 percentile queries while HDP can only guarantee a value of 6. Clearly, lifespan-based organizational protocols can equally benefit loosely-structured P2P systems, yielding faster query resolution times and higher query hit numbers.



(a) CDF for query resolution time.



(b) Query hit number for different query percentiles.

Fig. 5. Query performance for lifespan-based query (LQuery), regional caching (LRCX) and replication (LRRep) and its random-based counterpart (RQuery, RRCX and RRRep).

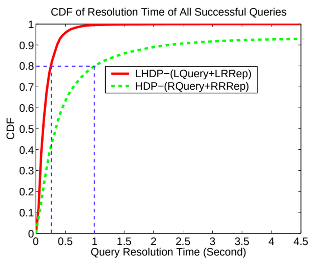
2) Query-related Strategies: We now evaluate the performance of lifespan-based query-related strategies. We show, for unstructured P2P systems, how lifespan-based strategies can boost search performance. Applying these strategies to loosely-structured systems yields even more significant performance gains, which we omit here due to space constraints.

We first demonstrate the benefits of employing lifespan-

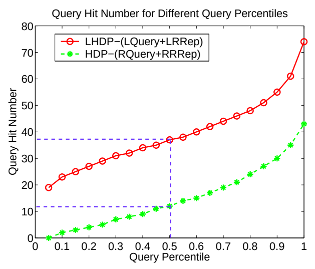
Replication	Caching	Aggregated Query Hits	Query Satisfaction (Z)		
			5	10	20
SRep	None	2.02	1.46	1.77	2.13
RRep	None	1.68	1.14	1.30	1.66
RRep	RCX	2.53	1.08	1.20	1.45

TABLE VI
RELATIVE PERFORMANCE COMPARISON FOR QUERY-RELATED STRATEGIES; LIFESPAN-BASED QUERY-RELATED STRATEGIES OVER ALTERNATIVE ONES.

based ideas only for the query strategy (LQuery). This corresponds to the scenario in which only implicit, simple replication (SRep) is used upon successful queries, while explicit replication and caching strategies are disabled. Note that this is the common case for currently deployed P2P systems. Figure 4 shows the CDF of query resolution time (Fig. 4a) and query hit number at various percentiles (Fig. 4b) for k-random walk query strategy (RQuery) and our lifespan-based LQuery, respectively. About half the queries can be resolved in 0.4 seconds when using LQuery, while it takes 0.8 seconds with RQuery. Similarly, there is a 100% increase in median query hit number (from 4 to 8) when switching from RQuery to LQuery. Since no caching or explicit replication is presented in this scenario, the difference between the two can only be attributed to query strategies themselves. LQuery walkers, i.e. random walkers biased toward old peers, are more likely to run into peers with more shared objects, making possible to answer queries more effectively.



(a) CDF for query resolution time.



(b) Query hit number for different query percentiles.

Fig. 6. Query performance for lifespan-based query (LQuery) and replication (LRRep) on a lifespan-based hybrid organizational protocol (LHDP) and its random-based counterpart (HDP, RQuery and RRRep).

Next, we determine how much could be gained by combining lifespan-based query, caching, and explicit replication. Figure 5 shows the query performance of two cases, one with RQuery, random regional caching with expiration (RRCX), and random regional replication (RRRep), another with lifespan-based LQuery, LRCX, and LRRep. As Fig. 5a shows, the original random approach needs 0.55 seconds to answer 90% of all queries while the lifespan-based strategies reduce this to 0.2 seconds. Query hit numbers of the lifespan-based system (Fig. 5b), are typically two to four times larger than that of the alternate system at different query percentiles. These big advantages of lifespan strategies can be easily explained by earlier description in Subsection IV-B.

Table VI offers a summary of the performance of several simulation scenarios where different query, caching and replication strategies were applied (simple replication (SRep), regional replication (RRep), and regional caching with expiration (RCX)). As with Table V, in each scenario we show the relative performance of the lifespan-based strategies over its random-based counterpart (in terms of aggregated query hit numbers and degree of query satisfaction). Systems relying on lifespan-based strategies consistently result in significantly better performance, both in terms of aggregate query hit number and query satisfaction at different levels, than random-based ones.

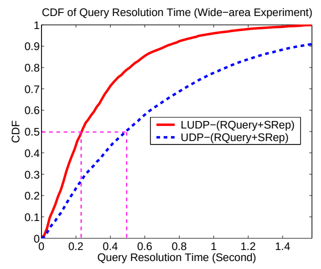
3) Combined Lifespan-based Protocols & Strategies: Clearly, the biggest advantage of lifespan-based approaches would come from the combination of lifespan-based organizational protocols and query-related strategies. To demonstrate this, we compare the performance of a purely random-based with a purely lifespan-based system. The random-based system uses UDP as the organizational protocol, basic k-random-walks (RQuery) for query dissemination, plus random regional caching (RRCX) and replication (RRRep). The lifespan-based system employs LUDP as its organizational protocol, and lifespan-based query (LQuery), caching (LRCX) and replication strategies (LRRep). Combining the advantages of lifespan-based approaches at both levels results in between three- and five-times increase in query hits at different query percentiles, and over four-time speed-up for query resolution.

The advantage of combining lifespan-based organizational protocols and query-related strategies also holds for loosely-structured systems. Figure 6 compares the performance of two loosely-structured P2P systems: a random-based one using HDP as its organizational protocol, along with RQuery and RRRep; and a lifespan-based one relying on LHDP for its organizational protocol, and LQuery and LRRep for query dissemination and replication. No caching strategy is used here, since super-peers already provide object index caching to leaf-peers. As Fig. 6 illustrates, the combined benefits of lifespan-based ideas results in a significant improvement in query resolution time (resolving 80% queries in about 0.2 seconds instead of 1.0 seconds) and median query hit number (from 12 to 37 hits).

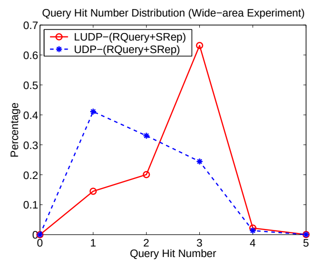
E. Wide-Area Results

We first illustrate the effectiveness of lifespan-based organizational protocols in wide-area. All queries take two purely random walkers, no caching or proactive replication strategies are used. Figure 7a compares CDF of query resolution time of the LUDP and UDP systems in the wide area. The results show that query resolution time is typically two times faster for LUDP than for UDP. Figure 7b gives query hit number distribution, i.e., the percentage of queries that have a certain number of hits. For this experiment, most queries (over 60%) on the lifespan-based LUDP system have a query hit number of 3, while most queries (44%) on the UDP deliver only one hit. In general, LUDP protocol delivers over 40% more query hits than UDP.

We also evaluated the wide-area performance of our lifespan-based query-related strategies. Two systems were deployed, one with LQuery for query and LRRep for replication



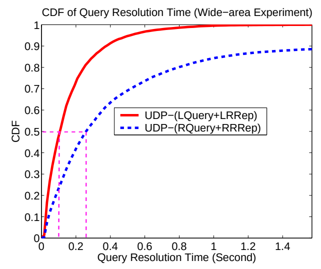
(a) CDF for query resolution time (wide-area).



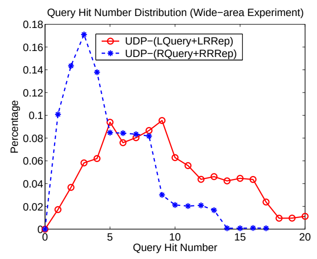
(b) Query hit number distribution (wide-area).

Fig. 7. Query performance for lifespan and random UDP in PlanetLab.

and the second one employing RQuery and RRRep. Both systems use UDP as the organizational protocol. As Fig. 8a shows, lifespan-based strategies deliver between 2-3X faster query resolution. Figure 8b indicates a much higher chance for a lifespan-based query to return 5 or more query hits than its random equivalent. Overall, lifespan strategies provide 2.11 times more query hits than their alternatives. These results, consistent with those found through simulation, clearly demonstrate the advantages of using lifespan-based ideas for query-related strategies.



(a) CDF for query resolution time (wide-area).



(b) Query hit number distribution (wide-area).

Fig. 8. Query performance for the combined lifespan query and replication strategies (LQuery and LRRep) and their random alternative (RQuery and RRRep) in PlanetLab; we employ UDP as the organization protocol in both experiment.

VI. Related Work

Our work builds on many related previous efforts. Due to space constraints, this section only briefly describes some of the work not already addressed in previous sections.

Some related studies adopt a model of uniformly random, concurrent node failure when evaluating the impact of churn on single peers or the overall network [24], [37]–[40]. More realistic failure models [8], [9], [13], [29] better take into account the intrinsic behavior of P2P user populations, with most users spending few minutes per day searching for content to download and a few others remain logged in for weeks exhibiting a more server-like behavior (as summarized by Rhea et al. [6]).

A number of efforts have studied churn and its implications to P2P systems. A few of these studies have opted for passive monitoring techniques. For example, an early study by Sen and Wang [28] analyze P2P traffic collected passively at multiple border routers across a large ISP network. The authors report high-level system dynamics with *ontime*, the duration an IP address is active, showing a heavy-tailed distribution. More recently, in their study on workload characterization of P2P file sharing, Gummadi et al. [30] report session lengths based on passive monitoring of a router at the University of Washington. It has been pointed out [11] that passive monitoring studies may tend to underestimate peers' lifespans as some peers may not be continuously generating traffic through the instrumented host.

Through active probing of 17,125 peers during 60 hours, Saroiu et al. [13] found a median peer session time $\sim$60 minutes. Chu et al. [27] present results from a considerably longer experiment on a smaller set of peers (5,000 IP:port pairs). Their results show a highly transient population and significant time-of-day effects. Both Saroiu et al. [13] and Chu et al. [27] measured session times by actively probing previously collected TCP/IP addresses of peers following an approach that can only determine if a node is or not accepting TCP connections in the requested port without distinguishing what application is connected to it. In their study on availability in the Overnet DHT [29], Bhagwan et al. rightly point out the potential effects of aliasing on modeling host availability.[9] IP address aliasing can result in great overestimation of the number of hosts in the systems and the underestimation of their availability. A more recent work by Nurmi et al. [12] discuss the suitability of different statistical distribution for describing machine availability in three different data sets. Their results indicate that Weibull model more accurately represent machine availability in some of these settings. In contrast, our work characterizes the lifespan distribution of individual sessions, during which a peer's IP:port tuple will not change, and builds on this characterization to propose new organization protocols and query-related strategies. The different algorithms we have presented throughout this paper rely on the predictability of peer's remaining time based on its current uptime. In a recent study [11] Stutzbach and Rejaie present a thorough analysis of churn in Gnutella, Kad and BitTorrent and, while questioning the use of Pareto distribution to describe the session lengths, they show that current uptime is still a good indicator of remaining uptime in both Gnutella and Kad.

Leonard et al. [9] investigate the resilience of random graphs to lifespan-based node failures and derives the expected delay (and associated probability) before a peer is forcefully isolated from the graph. Godfrey et al. [10] presents a comparative study of different strategies aimed at reducing churn rate by intelligently selecting a subset of the available nodes. The compared strategies differ in the amount of node information they rely on for selection and in whether they replace a failed node with a new one. The authors show, through trace-based evaluation, that replacement strategies can yield significant reduction in churn over fixed strategies, and that random replacement can outperform a preference-list strategy, which occurs because of optimizing for a metric other than churn,

[9]Aliasing effects could be due, for example, to the use of DHCP and NATs, as well as the sharing of a host by multiple users.

in scenarios with fairly skewed distributions of session times. Godfrey et al. also explore how different designs or parameter choices in distributed systems can "accidentally" lead to particular replacement strategies. Our work experimentally derives the distribution of session times and proposes and evaluates a number of organizational protocols and query-related strategies that leverage this to yield systems with performance characteristics more resilient to their environments' expected high churn.

Other related efforts have focused on the performance and maintenance cost of DHT-based systems in the face of churn [6], [41]–[43]. Although originally targeted at non-DHT protocols, our lifespan-based approach could be straightforwardly combined with some of the techniques proposed in the literature to yield better structurally churn-resilient DHT systems. We plan to explore this as part of our future work.

This research is partially motivated by the seminal work of Harchol-Balter and Downey [44] on process lifetime distribution and its implications on load-balancing techniques. The authors measured the distribution of Unix processes and propose a UBNE (used-better-than-new-in-expectation) distribution that fits it well. Based on their finding, Harchol-Balter and Downey present a new policy for preemptive process migration in clusters of workstations.

VII. Conclusions

This paper addresses the problem of highly transient populations in unstructured and loosely structured P2P systems. Using a number of illustrative organizational protocols and query-related strategies, we present trace-driven simulation and wide-area experiment results that illustrate the performance advantages of considering peers' estimated session time as a key system attribute in the design of churn-resilient P2P systems. The benefits of lifespan-based ideas are not constrained to control-related traffic, but extend to applications, resulting in improved query satisfaction and resolution time, as well as significantly higher system scalability. An area of future work is to understand the implications of partial adoption of the presented approach.

Acknowledgments

We would like to thank NUIT and our Computing Support Group for their aid and understanding while obtaining the trace data used for our experiments. We are also grateful to Jeanine Casler, Stefan Birrer, David Choffnes and Kevin Livingston for their helpful comments on early drafts of this article.

References

[1] Y. Qiao and F. E. Bustamante, "Elders know best - handling churn in less structured P2P systems," in *Proc. of the IEEE International Conference on Peer-to-Peer Computing (P2P)*, Konstanz, Germany, August-September 2005.
[2] Napster, "http://www.napster.com," 2003.
[3] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of peers and streaming media," in *Proc. of HotNets-I*, 2002.
[4] P. Linga, I. Gupta, and K. Birman, "A churn-resistant peer-to-peer web caching system," in *ACM Workshop on Survivable and Self-Renegerative Systems*, 2003.
[5] Y. Chawathe, S. Ratnasamy, L. Breslau, and S. Shenker, "Making Gnutella-like P2P systems scalable," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2003.
[6] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. of the USENIX Annual Technical Conference*, June 2004.
[7] J. Li, J. Stribling, R. Morris, T. Gil, and F. Kaashoek, "Routing tradeoffs in peer-to-peer DHT systems with churn," in *Proc. of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, 2004.
[8] F. E. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in P2P protocols," in *Proc. of the 8th International Workshop on Web Content and Caching Distribution (WCW)*, 2003.
[9] D. Leonard, V. Rai, and D. Loguinov, "On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks," in *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Banff, Alberta, Canada, June 2005.
[10] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*.
[11] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. of ACM Internet Measurement Conference (IMC)*, October 2006.
[12] D. Nurmi, J. Brevik, and R. Wolski, "Modeling machine availability in enterprise and wide-area distributed computing environments," in *Proc. of Euro-Par*, August 2005.
[13] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of MMCN*, 2002.
[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. of ICS*, 2002.
[15] Clip2, "The Gnutella protocol specification v0.4," The Gnutella RFC," RFC, 2000. [Online]. Available: http://rfc-gnutella.sourceforge.net
[16] A. Singla and C. Rohrs, "Ultrapeers: Another step towards Gnutella scalability," Lime Wire LLC," Working Draft, 2001.
[17] KaZaA, "http://www.kazaa.com," 2001.
[18] T. Klingberg and R. Manfredi, "Gnutella 0.6," The Gnutella RFC," RFC, 2002. [Online]. Available: http://rfc-gnutella.sourceforge.net
[19] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2002.
[20] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical Review*, vol. 64, no. 046135, 2001.
[21] 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'02)*, 2002.
[22] M. Roussopoulos and M. Baker, "Cup: Controlled update propagation in peer-to-peer networks," in *Proc. of the USENIX Annual Technical Conference*, 2003.
[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2001.
[24] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2001.
[25] E. P. Markatos, "Tracing a large-scale peer to peer system: an hour in the life of gnutella," in *Proc. of CCGrid*, 2002.
[26] B. Yang and H. Garcia-Molina, "Efficient search in peer-to-peer networks," in *Proc. of the International Conference on Distributed Computing Systems (ICDCS)*, 2002.
[27] J. Chu, K. Labonte, and B. N. Levine, "Availability and locality measurements of peer-to-peer file systems," in *Proc. of ITCom*, 2002.
[28] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *Proc. of ACM SIGCOMM-IM Workshop*, 2002.
[29] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding availability," in *Proc. of the 2st International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.
[30] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. of the 19th ACM Symposium on Operating System Principles (SOSP)*, 2003.
[31] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proc. of 9th ACM CCCS*, 2002.

[32] D. Dutta, A. Goel, R. Govindan, and H. Zhang, "The design of a distributed rating scheme for peer-to-peer systems," in *Proc. of P2PEcon Workshop*, 2003.
[33] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for p2p and mobile ad-hoc networks," in *Proc. of P2PEcon Workshop*, 2004.
[34] Mutella, "http://mutella.sourceforge.net," 2003.
[35] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," in *Proc. of IEEE PDIS*, 1996.
[36] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability," in *O'Reilly's OpenP2P*, 2001.
[37] M. F. Kaashoek and D. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proc. of the 2st International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, February.
[38] J. Aspens, Z. Diamadi, and G. Shah, "Fault tolerant routing in peer-to-peer systems," in *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, Monterrey, CA, July 2002.
[39] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Karlsruhe, Germany, August 2003.
[40] L. Massoulié, A.-M. Kermarrec, and A. J. Ganesh, "Network awareness and failure resilience in self-organizing overlay networks," in *Proc. of the IEEE Symposium on Reliable and Distributed Systems (SRDS)*, Florence, Italy, October 2003.
[41] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer systems," in *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.
[42] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *Proc. of the 2st International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.
[43] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance vs. cost framework for evaluating dht design tradeoffs under churn," in *Proc. of 24th INFOCOM*, 2005.
[44] M. Harchol-Balter and A. B. Downey, "Exploting process lifetime distribution for dynamic load balancing," *ACM Transactions on Computer Systems*, vol. 15, no. 3, 1997.

Fabián E. Bustamante is an assistant professor of computer science in the Department of Electrical Engineering and Computer Science at Northwestern University. His research interests include distributed systems and networks, and operating systems support for massively distributed systems, including Internet and vehicular network services. Bustamante has an M.S. and Ph.D. in computer science from the Georgia Institute of Technology. He is a member of the IEEE Computer Society, the ACM, and USENIX.

Yi Qiao is a Ph.D. student in the Department of Electrical Engineering and Computer Science at Northwestern University. His research interests are in distributed systems and networking with an emphasis on measurement, modeling and design of scalable distributed systems. Qiao has a M.S. and a B.S. in Electrical Engineering from Tsinghua University and a M.S. in Computer Science from Northwestern University. He is a member of the IEEE Computer Society and ACM.