Elders Know Best - Handling Churn in Less Structured P2P Systems

Yi Qiao and Fabián E. Bustamante
Department of Computer Science
Northwestern University, Evanston, IL 60201, USA
Email: {yqiao,fabianb}@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.

1. Introduction

Due in part to the autonomous nature of peers, their architectural mutual dependency, and their excessively large populations, the transiency of peer populations (a.k.a. churn) and its implications on P2P systems have recently attracted the attention of the research community [3, 19, 7, 27, 16]. Measurement studies of deployed P2P systems have reported *median session times* [1] varying from one hour to one minute [29, 6, 27]. The implications of such a high degree of transiency on the overall system's performance would clearly depend on the level of nodes' investment in their neighboring peers. At the very least, the amount of maintenance-related messages processed by any node is proportional to the degree of stability of the node's neighboring set. Further, in the context of data-sharing P2P systems, the level of replication, the effectiveness of caches, and the spread and satisfaction rate of queries will all be affected by how dynamic the peers' population is.

We address the problem of highly transient populations in unstructured and loosely-structured peer-to-peer (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 our context, this means that the expected remaining session time of a peer is directly proportional to the session's current length, i.e. the peer's current *age*. This observation forms the basis for a set of new protocols for peer organization and query-related strategies 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 considering lifespan in organizational protocols - i.e. how peers organize themselves in an overlay network - in terms of end-application performance and in the context of dynamic and heterogeneous Internet environments. The lifespan-based approach for organizational protocols was first proposed in our position paper [6], where we show its effectiveness in terms of increased system stability (e.g. an over 42% reduction on the ratio of connection breakdowns and their associated costs) through a simulation study.

[1] Where a node's *session time* is the time from the node's joining to its subsequent leaving from the system. 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 [27].

In this paper we go beyond those preliminary results to evaluate the advantages of the proposed approach in terms of application performance in a dynamic Internet testbed of ∼150 world-wide distributed PlanetLab nodes [25]. 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 go further by applying similar ideas to query-related strategies. Through trace-driven simulation and wide-area experimentation in PlanetLab 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 our proposed lifespan-based organizational protocols. Our results show that lifespan-based strategies can generate over *2-5x* 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 2 provides some background and reviews related work. Section 3 presents results from our study on peers' lifespans and discusses lifespan-based organizational protocols and query-related strategies. Section 4 describes our evaluation methodology and presents results from both trace-based simulations and wide-area experiments. We conclude in Section 5.

2. Background

A peer-to-peer system is structured based on its own organizational protocol. In unstructured and loosely-structured P2P systems [2], nodes join the network by first contacting a set of peers already in the system [14, 15]. Peers define the P2P network overlay through connections with other, randomly chosen, peers. While organizational protocols for unstructured systems, such as Gnutella v0.4 [9], consider all peers as equals, protocols for loosely-structured systems, such as Gnutella v0.6 and Kazaa [31, 14, 15], commonly define a two-level hierarchy distinguishing between common "leaf" peers and resource-richer superpeers [31, 14, 15]. Hereafter we also refer to unstructured and loosely-structured organizational protocols as *Unstructured Decentralized Protocols (UDP)* and *Hybrid Decentralized Protocols (HDP)*, respectively.

Connected peers interact with each other 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 are forwarded on a specific connection on the reverse of the path taken by an associated message. 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 some others that peer knows about. Information on neighboring peers can be provided either by creating pongs on their behalf or by forwarding them pings and back-propagating their replies. *Byes* are optional messages used to report the closing of connections.

2.1. 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 [10]. We review some of them for less structured systems next, including currently used and other proposed techniques for query distribution, caching and replication.

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. [20] 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. Adamic et al. [1] 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. [7] suggest taking node capacity into consideration through biased random walks, directed

[2] We use the classification proposed by Lv et al. [20]

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 [28]. The *value* in the pair is normally a pointer to the node holding a replica of the object associated with the corresponding *key* [26, 33]. Upon receiving a query message, the node will not only check its own shared objects, but will also scan the cache for entries with matching keys. Upon a successful match, the node will reply with the associated pair(s). PCX can be used in both DHT-based and less structured systems to improve query results. With NCU [1, 15, 7], each node maintains caches of metadata for all of its neighbors. As with PCX, a node will send a query hit either on its own behalf or on behalf of its neighbors, effectively increasing the reach of a query [23, 34].

Replication is a common approach to improve performance when distributed systems need to scale in numbers of users and objects in the system and in 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 aimed at improving query hits while reducing response time have been proposed. Cohen and Shenker [10] modeled various explicit replication strategies; they found square-root replication, which can be efficiently achieved using path replication, to be optimal.

2.2. Transient Populations and P2P Systems

There have been a number of studies of peers' participation and transiency in P2P systems [29, 6, 8, 23, 30, 4, 13]. Through active probing of 17,125 peers during 60 hours, Saroiu et al. [29] found a median peer session time ~60 minutes. Chu et al. [8] 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 works 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.

We performed an independent study [6] of peers' lifespans in the widely deployed Gnutella network (v0.6 [15], i.e. with super-peers), collecting over 1 million peer session times for over half-million peers. To avoid potential measurement errors, each of our probes tries to establish an application-level connection checking for specific Gnutella packet headers. This work was the first one to show that the peer session time distribution can be modeled by a Pareto distribution of the form λT^k $(k < 0)$ (an R^2 value higher than 0.99 verifies the very high *goodness of fit* of the model). In our context, this means that the expected remaining session time of a peer is directly proportional to the session's current length, i.e., its current *age*. Based on this, in our position paper [6] we proposed a number of illustrative lifespan-based organizational protocols and show its performance benefits in terms of increased systems stability through a trace-driven simulation study. In this paper, we go beyond those preliminary results, reporting the advantages of the proposed approach in terms of application performance and in a dynamic Internet testbed. Further, we apply lifespan-based ideas for query-related strategies (query, caching and replication) and demonstrate significant performance benefits of these strategies when layered over currently employed organizational protocols as well as when used in combination with our proposed lifespan-based protocols.

Related research efforts have looked at the performance and maintenance cost of DHT-based systems in the face of churn [18, 27, 22, 17]. Although originally aimed to 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.

3. Lifespan-based Protocols and Strategies

This section presents a number of lifespan-based, illustrative organizational protocols and query-related strategies. For continuity and containment, we first provide an overview of one of the previously proposed lifespan-based organizational protocols [6] and describe its extension for loosely-structured P2P systems. A number of lifespan-based, query-related strategies are also proposed and discussed in detail. We close the section by outlining a lightweight distributed protocol for peers' age discovery.

3.1. Organizational Protocols

The basic idea behind the proposed organizational 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 expected longer sessions times. Given the UBNE (Used-Better-than-New-in-Expectation) nature of lifespan distribution [6], a peer's current age is a fair estimate of its lifespan.

This idea can be equally applied to both unstructured and loosely structured protocols. In our position paper [6] we described its use in an illustrative unstructured protocol, denoted here as *Lifespan-based UDP*, or *LUDP*. [3] Under LUDP, a peer looking for a new neighbor selects the oldest node from among those it knows, a group formed from the random set of recommendations forwarded by other peers in the network. Since the actual number of incoming connections that a peer can accept is commonly bounded, LUDP employs a weighted credit selection scheme that also considers the peer's current age and the (estimated) number of available incoming connection slots.

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

3.2. 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 [20], 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 preferences 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 resulting in 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 lifespan-based 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 remove 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 than previous ones. 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, probabilis-

[3] LUDP corresponds to the LSPAN-3 protocol in [6].

tic 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.

Clearly, these illustrative lifespan-based strategies could be directly employed in original unstructured and loosely structured UDP and HDP systems, as well as in the proposed lifespan-based LUDP and LHDP protocols described in the previous subsection.

3.3. Determining Peers' Age

The effectiveness of the proposed lifespan-based approach depend 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 [6]. To improve the latter, we have designed a *lightweight distributed protocol for peers' age determination* based on previous work on reputation [11, 12, 5].

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:* From the provided list and in order to reduce the chances of collusion, P first trims off proposed witnesses with suspiciously large interaction windows (outliers) and then samples 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 improve 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 are less interested in the real age of a peer than in its relative seniority among other candidate peers. We are currently evaluating the potential impact of estimation error of peers' seniority on the performance of lifespan-based ideas.

4. 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 end applications.

4.1. Query-Related Strategies and Organizational Protocols

For this evaluation, we implemented two random-based organizational protocols, *Unstructured Decentralized Protocol (UDP)* and *Hybrid Decentralized Protocol (HDP)*, as well as our lifespan-based organizational protocols, LUDP and LHDP.

We implemented the original [21] (RQuery) and a lifespan-based (LQuery) *k-random-walk* query strategies. We also implemented PCX, NCU and RCX for caching. 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. The lifespan-based RCX is denoted as *LRCX*, otherwise is denoted as *RRCX*. The basic form of replication is denoted as SRep (for "Simple Replication"). For Regional Replication (RRep), we use *LRRep* and *RRRep* to denote its lifespan- and random-based variants, respectively. For both LRRep and RRRep, we set an upperbound on the number of replicas a peer can hold to be 10. Figure 1 lists various acronyms for different protocols and strategies used throughout the rest of the paper.

4.2. Metrics

The effectiveness of the proposed approach is evaluated in terms of the performance improvements to query-related tasks, as captured by three metrics:

Organizational Protocol	Random	Lifespan
Unstructured	UDP	LUDP
Loosely Structured	HDP	LHDP

UDP: Unstructured Decentralized Protocol
HDP: Hierarchical Decentralized Protocol

Query−Related Strategies	Random	Lifespan
Query Strategy	RQuery	LQuery
Caching Strategy	RRCX, PCX, NCU	LRCX
Replication Strategy	RRRep, SRep	LRRep

RQuery, LQuery: Random− and Lifespan−based k−random walker
RRCX, LRCX: Random− and Lifespan−based Regional Caching with eXpiration
PCX: Path Caching with eXpiration
NCU: Neighbor Caching with incremental Update
RRRep, LRRep: Random− and Lifespan−based Regional Replication
SRep: Simple Replication

Figure 1. List of organizational protocols and query-related strategies and their acronyms.

Query Resolution Time, Query Hit Number and Z Query Satisfaction. *Query Resolution Time* is the time between query submission and the arrival of the first reply. *Query Hit Number* stands for 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* [34] is the percentage of queries achieving Z satisfaction, i.e. obtaining at least Z query hits.

4.3. 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 a variety of query distribution, caching and replication mechanisms. We ran simulations using 4 of the 20 traces collected,[4] 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 [24], thus inheriting all the expected functionality of a typical P2P file-sharing system. As mentioned 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 [2, 32] 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 [5]. 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 [13] 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.

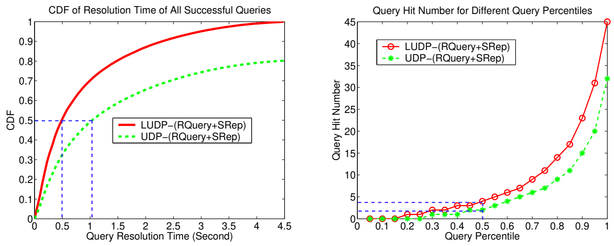
4.4. Simulation Results

4.4.1. Organizational Protocols

We first examine the advantages of a lifespan-based approach in organizational protocols. 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.

[4] 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.

[5] 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.

Figure 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. 2.a) and query hit number at different query percentiles (Fig. 2.b) using simple replication (SRep) and no caching. The advantage of a more stable overlay in LUDP is clear from both graphs: Figure 2.a shows a reduction of between 50% and 70% in query resolution time with the lifespan-based LUDP in contrast to UDP, while Figure 2.b demonstrates 50-75% increase on the query hit number at various query percentiles. As an example, the dashed lines in Figure 2.a show that 50% of 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, Figure 2.b shows a median query hit number (corresponding to 50 percentile of queries) of only 2 for UDP, but 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 (Figure 3). Note that all numbers in Figure 3 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 Figure 3.

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.

Figure 3 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.

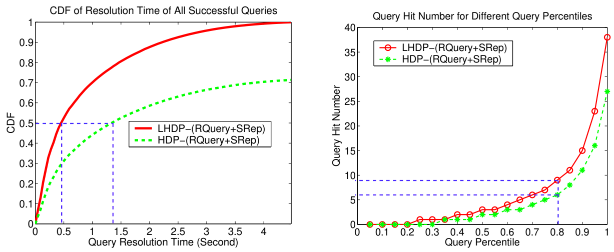
Recall that our lifespan-based protocols differ from UDP (HDP) mainly at the time of connection establishment, where a peer gives older peers (superpeers) higher priority when selecting neighbors. Thus, although the paths that query walkers take in both LUDP and UDP are purely random, a walker in LUDP 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 on 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, 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%.

Replication	Caching	Aggregate Query Hit Number	Z=5 Satisfaction	Z=10 Satisfaction	Z=20 Satisfaction
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

Figure 3. Relative performance comparison between LUDP and UDP organizational protocols. Different simulation scenarios are indicated by the combination of SRep, PCX and NCU. All numbers in the table are relative performance of LUDP over UDP, including relative aggregated query hit number and relative Z query satisfaction with different levels of Z.

So far we have been looking at lifespan-based or-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.

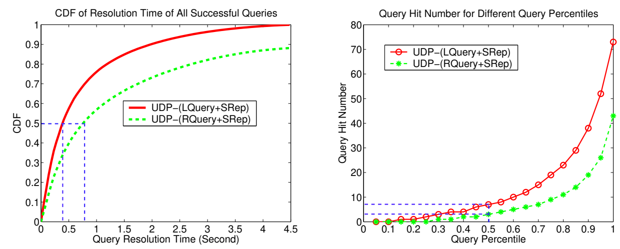
Figure 4. Query performance for LHDP and HDP using simple replication (SRep) and no caching.

ganizational protocols for unstructured P2P systems. Illustrating the benefits of lifespan-based protocols for loosely-structured systems, Figure 4 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 benefit loosely-structured P2P systems in the similar way as they benefit unstructured ones, yielding faster query resolution times and higher query hit numbers.

4.4.2. 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 constraint.

We first demonstrate the benefits of employing lifespan-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 5 shows the CDF of query resolution time (Fig. 5.a) and query hit number at various percentiles (Fig. 5.b) for k random walk query strategy (RQuery) and our lifespan-based LQuery, respectively. 50% of all queries can be resolved in 0.4 seconds when using LQuery, while it takes 0.8 seconds with RQuery. Similarly, there



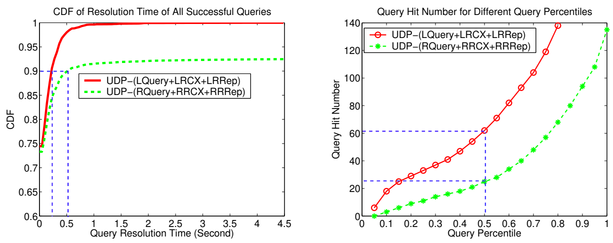
(a) CDF for query resolution time.

(b) Query hit number for different query percentiles.

Figure 5. Query performance for LQuery and RQuery using simple replication (SRep) and no caching.

Next, we determine how much could be gained by combining lifespan-based query, caching, and explicit replication. Figure 6 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 Figure 6.a 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. 6.b), 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 3.2.

Figure 7 offers a summary of the performance of several simulation scenarios where different query, caching and replication strategies were applied. As with Figure 3, in each scenario we show the relative performance of the lifespan-based strategies over its random-based counterpart. 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.



(a) CDF for query resolution time.

(b) Query hit number for different query percentiles.

Figure 6. Query performance for LQuery +LRCX +LRRep and RQuery +RRCX +RRRep.

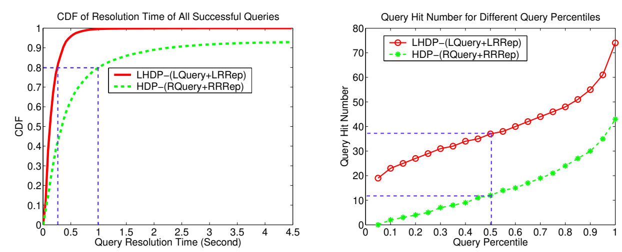
Replication	Caching	Aggregate Query Hit Number	Z=5 Satisfaction	Z=10 Satisfaction	Z=20 Satisfaction
SRep	None	2.02	1.46	1.77	2.33
RRep	None	1.68	1.14	1.30	1.66
RRep	RCX	2.53	1.08	1.20	1.45

Figure 7. Relative performance comparison for query-related strategies. Different simulation scenarios are indicated by the combination of simple replication (SRep), regional replication (RRep), and regional caching with expiration (RCX). All numbers in the table are relative performance of lifespan-based query-related strategies over alternative ones. Relative aggregated query hit number and relative Z query satisfaction with different levels of Z are given.

4.4.3. 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 3-5X increase in query hits at different query percentiles, and over 4X speed-up for query resolution.

The advantage of combining lifespan-based organizational protocols and query-related strategies also holds for loosely-structured systems. Figure 8 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 superpeers already provide object index caching to leaf-peers. As Figure 8 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).



(a) CDF for query resolution time.

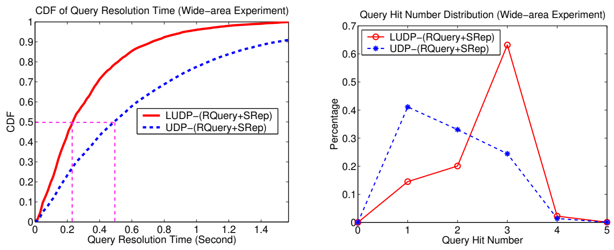
(b) Query hit number for different query percentiles.

Figure 8. Query performance for LHDP-LQuery+LRRep and HDP-RQuery+RRRep.

4.5. 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 9.a 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 9.b 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 and the second one employing RQuery and RRRep. Both systems use UDP as the organizational protocol. As Figure 10.a shows, lifespan-based strategies deliver between 2-3X faster query resolution. Figure 10.b indicates a much higher chance for a lifespan-based query to return 5 or more query hits

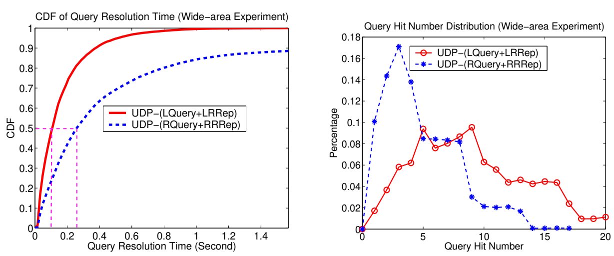


(a) CDF for query resolution time (wide-area).

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

Figure 9. Query performance for LUDP and UDP from wide-area experiment.

than its random equivalent. Overall, lifespan strategies provide 2.31 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).

Figure 10. Query performance for LQuery+LRRep and RQuery+RRRep from wide-area experiments, UDP are used in both experiment.

5. 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.

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 J. Casler, S. Birrer and K. Livingston for their helpful comments on early drafts of this paper.

References

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review*, 64(046135), 2001.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the WWW. In *Proc. of IEEE PDIS*, 1996.

[3] M. Bawa, H. Deshpande, and H. Garcia-Molina. Transience of peers and streaming media. In *Proc. of HotNets-I*, 2002.

[4] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *Proc. of IPTPS*, 2003.

[5] S. Buchegger and J.-Y. L. Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proc. of P2PEcon Workshop*, 2004.

[6] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proc. of 8th WCW Workshop*, 2003.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proc. of ACM SIGCOMM*, 2003.

[8] J. Chu, K. Labonte, and B. N. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proc. of ITCom*, 2002.

[9] Clip2. The Gnutella protocol specification v0.4. RFC, The Gnutella RFC, 2000.

[10] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. of ACM SIGCOMM*, 2002.

[11] 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.

[12] 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.

[13] 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 19th ACM SOSP*, 2003.

[14] KaZaA. http://www.kazaa.com. 2001.

[15] T. Klingberg and R. Manfredi. Gnutella 0.6. RFC, The Gnutella RFC, 2002.

[16] J. Li, J. Stribling, R. Morris, T. Gil, and F. Kaashoek. Routing tradeoffs in peer-to-peer DHT systems with churn. In *Proc. of IPTPS*, 2004.

[17] 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.
[18] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proc. of ACM PODC*, 2002.
[19] 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.
[20] 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.
[21] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make gnutella scalable? In *Proc. of IPTPS*, 2002.
[22] R. Mahajan, M. Castro, and A. Rowstron. Controlling the cost of reliability in peer-to-peer overlays. In *Proc. of IPTPS 03*, 2003.
[23] E. P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *Proc. of CCGrid*, 2002.
[24] Mutella. http://mutella.sourceforge.net. 2003.
[25] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the Internet. In *Proc. of ACM HotNets-1*, October 2002.
[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, 2001.
[27] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. June 2004.
[28] M. Roussopoulos and M. Baker. Cup: Controlled update propagation in peer-to-peer networks. In *Proc. of USENIX Annual Tech.*, 2003.
[29] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN*, 2002.
[30] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. of ACM SIGCOMM-IM Workshop*, 2002.
[31] A. Singla and C. Rohrs. Ultrapeers: Another step towards Gnutella scalability. Working draft, Lime Wire LLC, 2001.
[32] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. In *O'Reilly's OpenP2P*, 2001.
[33] 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 ACM SIGCOMM*, 2001.
[34] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proc. of ICDCS*, 2002.