Friendships that last: Peer lifespan and its role in P2P protocols

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

Abstract

We consider the problem of choosing who to "befriend" among a collection of known peers in distributed P2P systems. In particular, our work explores a number of P2P protocols that, by considering peers' lifespan distribution a key attribute, can yield systems with performance characteristics more resilient to the natural instability of their environments.

This article presents results from our initial efforts, focusing on currently deployed decentralized P2P systems. We measure the observed lifespan of more than 500,000 peers in a popular P2P system for over a week and propose a functional form that fits the distribution well. We consider a number of P2P protocols based on this distribution, and use a trace-driven simulator to compare them against alternative protocols for decentralized and unstructured or loosely-structured P2P systems. We find that simple lifespan-based protocols can reduce the ratio of connection breakdowns and their associated costs by over 42%.

Keywords: Peer-to-Peer, Lifespan, Pareto Distribution, Protocols, Empirical Study.

1 Introduction

Peer-to-peer computing has been defined as the sharing of computer resources and services by direct exchange between the participating nodes. Since Napster's [17] introduction in 1999, the area has received increasing attention from the research community and the general public, as the model's many advantages have being recognized.

In its purest form, the P2P model has no concept of a server, but rather considers all participants as equals, regardless of resource capacity, connectivity or "commitment" to the common good. While this assumption of equality enables very simple protocols, it could also translate into the loss of some of the model's most appealing attributes [1].

Part of the problem stems from the clash between this equality assumption and the degree of heterogeneity and transiency found in recent studies of current P2P systems. Far from being equal, peers' populations have been shown to exhibit significant variations in attributes such as storage, bandwidth, latency and their degree of sharing. Peers' commitment to the system, in particular, has been found to differ by more than four orders of magnitude [24].

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. The implication of the degree of peer transiency on the overall system's performance is 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 directly related to 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 ultimately is.

We consider the problem of selecting who to "befriend," among a collection of known peers in distributed P2P systems. Our work explores new protocols that, by considering peers' lifespan distribution a key attribute, can yield systems with performance characteristics more resilient to the natural instability of their environments. This article presents results from our initial efforts, focusing on currently-deployed unstructured and loosely structured P2P systems.

We have measured the observed lifespan of peers in a widely-deployed P2P system and identified a functional form that fits the distribution well. We have designed a number of P2P protocols based on this distribution and used a trace-driven simulator to compare them against alternative protocols for decentralized and unstructured or loosely-structured P2P systems. We found that P2P protocols based on simple heuristics that prioritize long lived peers when selecting a peer's new "friends" can significantly reduce (up to 42%) the ratio of connection breakdowns and their associated costs.

The remainder of this article is structured as follows. In Section 2 we present some background and review related work. We discuss the results of our study on peers' lifespans in Section 3. Sections 4 and 5 present a number of examples of how lifespan could be taken into consideration in P2P protocols and illustrate its benefits using a trace-driven simulation of decentralized unstructured and loosely-structured P2P systems. We summarize our results and discuss future work in Section 6.

2 Background

P2P computing has experienced an explosive growth in the last few years and a number of widely-deployed and research-oriented protocols have become available. Although the goal of most P2P systems is to provide general distributed resource sharing among the participating peers [26, 9, 10, 12], one of the most popular applications is content distribution.

In general, a set of participant nodes in a P2P system carries the system traffic consisting of functionality as well as control-related messages. P2P systems, or more precisely the protocols they implement, can be classified based on the participating nodes' reliance on centralized servers and the set's degree of structure [15].

In protocols adopting a centralized architecture, e.g. Napster, a central server is approached first to obtain meta-information, such as the identity of the peer in which some information is stored, and all subsequent communication is done directly between the peers themselves. In order to improve query performance and/or reduce control traffic, a number of protocols with more structured but decentralized architectures have been proposed. In loosely structured protocols the location of objects could be more or less controlled [7], or some degree of hierarchy may be imposed among peers [13, 14]. Within highly-structured protocols, both the network topology and the placement of resources are precisely determined [19, 22, 29, 30]. Decentralized and unstructured protocols such as early versions of Gnutella [8] neither rely on centralized directories nor enforce any precise control over the network topology or object placement, resulting in systems that are highly resilient to the transient nature of P2P populations.

In most unstructured and loosely-structured P2P protocols nodes join the network by first contacting a set of peers already in the system (whose contact information may have been obtained through a well-known web site). 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 open connections. Back-propagated messages are forwarded on a specific connection on the reverse of the path taken by an associated broadcasted message. A user wishing to find a given resource issues a query to its own peer. Queries are forwarded among peers for as long as they are alive, determined by a Time-To-Live field associated with the query itself and decremented after each forward. Besides queries and replies, 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. Ping messages are replied with *pongs* containing information on the responding peer and all others this peer knows about. Information on neighbor nodes can be provided either by creating pongs on their behalf or by forwarding the ping to them and back-propagating the replies. Pong messages include the contact point of a peer as well as information on what resources it makes available. *Byes* are optional messages used to report the closing of connections.

2.1 Related Work

There have been a number of studies reporting on experimental data collected from currently deployed P2P systems [28, 20, 24, 6, 16, 25, 4]. Ripenau et al. [20] identify a mismatch between the topologies of the Gnutella application-level network and that of the underlying Internet that leads to ineffective use of the physical networking infrastructure. More relevant to our work, the authors found that, by November 2000, only 36% of the total traffic (in bytes) was user-generated (query), while 55% of the remaining traffic was used to maintain group membership. While these numbers have significantly improved with the last modifications to the protocol, they are still a good indication of some of the potential effects of instability.

A few of these studies have looked at peers' participation in P2P systems. Saroiu et al. [24] examine node uptime and a range of other attributes such as reported bandwidth, latency, and degree of sharing, and found large degrees of heterogeneity among peers in the systems, with variations of three and up to five orders of magnitudes in the characteristics sampled. For their lifetime study, they recorded the uptime of 17,125 peers during 60 hours. Chu et al. [6] present results from a considerably longer experiment (over six weeks) on a smaller number of peers (5,000 IP:port pairs), focusing on node availability and object transfer. Their experiments results show serious fluctuations in the number of available nodes, a highly transient population and significant time-of-day effects. The authors also found a high level of locality in the stored and transferred objects, suggesting that the use of caches may significantly reduce network traffic and improve the user experience. Sen and Wang [25] analyze P2P traffic collected passively at multiple border routers across a large ISP network and report similar high-level system dynamics. The node availability measurements in both Saroiu et al. [24] and Chu et al. [6] were gathered by actively probing previously collected TCP/IP addresses of peers. Due to their methodology, their probes can only determine if a node is or is not accepting TCP connections in the requested port without distinguishing what application is connected to it. In our experiments we collected around 1 million lifespan entries for over a half-million peers; to avoid potential errors in our measurements, we tried to set application-level connections (checking for the specific packet header in Gnutella messages). In their study on availability [4], Bhagwan et al. discuss the potential effects of aliasing on modeling host availability.[1] The authors rightly point out that, in trying to accurately capture the availability characteristics of hosts, IP address aliasing can result in great overestimation of the number of hosts in the systems and the underestimation of their availability. By comparison with the authors' study, our work aims at characterizing the lifespan distribution of individual sessions, during which a peer's IP:port tuple will not change.

Our research is partially motivated by the seminal work of Harchol-Balter and Downey [11] 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.

We consider the problem of selecting who to "befriend" so as to yield systems with performance characteristics more resilient to the dynamic nature of their environment. Bernstein et al. [3] propose the use of machine learning for the selection of peers as sources from which to download. Banerjee et al. [2] introduce a scalable unicast-based technique to locate nearby peers. While efficiently choosing nearest peers is an important problem for many applications, such as overlay multicast and content distribution networks, selecting among similarly near peers using our proposed lifespan-based heuristics could significantly improve system stability, further reducing network load.

3 Peer Lifespan Distribution

Because of the potential implications of high degrees of transiency in P2P populations, we performed an independent study of peers' lifespans in a current and widely deployed P2P network with the intention of developing a model that accurately describes its distribution. In the remainder of this section we describe our methodology and discuss our findings.

[1] 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.

3.1 Collecting Observed Peers' Lifespans

To actively measure the lifespan of peers in Gnutella, we modified an open source Gnutella client[2] to both keep track of every peer found and periodically check its availability. Our monitoring peer maintains a hash table, initially empty, of peers it has seen so far. Each entry in the hash table includes fields for (1) IP:port of peer, (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 new ones as it finds new peers. Since it only knows with certainty the TOB of previously known and reborn peers, first time found (live) peers are included in the table 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[3]) or an unexpected response is received. Table 1 summarized the strategy used for updating peer lifespan information.

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 1. Strategy used for updating the peer table in each iteration (*T:* Current Time). As described 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 Time Of Birth (TOB) is set to the current time. A peer is considered dead when a connection attempt fails or an unexpected response is received.

A single monitoring peer scanning the whole table will clearly be too slow, resulting in too coarse a granularity for our lifespan measurements. To avoid this we evenly distribute the peer table (based on the hash values of peers) over 20 monitoring peers running across 17 hosts. This approach allows us to achieve a granularity of 1,300 seconds (about 21 minutes), when scanning over 30k to 40k entries per client.

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

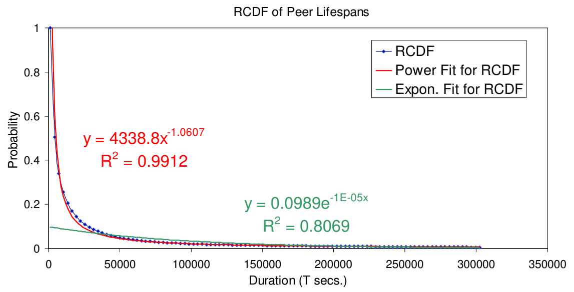
[3]We use the default timeout value of 10 seconds.

3.2 Peer Lifespan Distribution

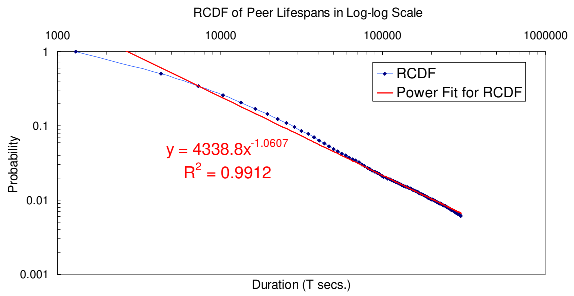
We measured the lifespans of more than 500,000 peers for over 7 consecutive days between March 1st and 8th, 2003. To account for the fact that sessions may be active (or inactive) for times longer than our sampling duration, we resort to the *create-based method* [21, 24]: we divide the captured trace 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. In addition, due to the granularity of our measurement, we could only give lifespan distribution for sessions longer than 1,300 seconds.

Figure 1 presents the Reverse Cumulative Distribution Function (RCDF) of peers' observed lifespans shorter than 3.5 days (and longer than 1,300 seconds). 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.

The Pareto distribution belongs to the UBNE class of distributions. In our context, this means that the expected remaining lifetime of a peer is directly proportional to its current age: the older a peer is, the longer we can expect it to remain in the system. In the remainder of this article we introduce a number of P2P protocols that take advantage of this observation. This set of protocols is not meant to be an exhaustive one, but is only used to illustrate the potential advantages of the proposed approach.



(a) Normal plot.



(b) Log-Log plot.

Figure 1. Distribution of lifetimes in a current P2P system (Gnutella) over a period of 7 days. The two additional lines in subfigure (a) show two attempts to fit a curve to these data: one a Pareto distribution and the other one an exponential curve. Subfigure (b) shows the same distribution on a log-log scale; the straight line in the log-log space 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.

4 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, 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 close friends 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 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.

It is clear that the amount of control messages processed by any node would be in direct relation to the degree of stability of their neighboring set. Beyond this, the implications of peers' transiency on the overall system's performance would be directly related to the degree of investment peers place on their friends. In the context of content distribution networks, the degree of replication, the effectiveness of caches, and the spread and hit ratio of queries would all be affected by how dynamic the peers' population ends up being.

It is surprisingly, then, to find that peers in most P2P protocols are hardly selective when choosing friends; consistent with the pure P2P model assumption on peers, the most widely-used strategy to select whom to befriend and/or choose which acquaintances to recommend is just random.

The basic idea behind the proposed protocols is to dynamically increase the system's degree of dependency on a node as the node's commitment to the community becomes clear. One way of achieving this is to give preference to peers with longer expected lives. Given the UBNE nature of peers' observed lifespan distribution (Section 3), a fair estimate for a peer's remaining lifetime can be derived from its current age.[4]

The rest of this section describes three lifespan-based protocols. Table 2 highlights their main aspects.

[4]This approach could also be seen as an *incentive* based system, where long standing members of the peer population have a higher degree of connectivity than new-comers [24].

4.1 Lifespan-Based Friend Selection

A very simple protocol using this heuristic, *LSPAN-1*, 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, after the departure of a friend for example, 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.[5]

4.2 Lifespan-Based Friend Selection and Recommendation

A more selective scheme, *LSPAN-2*, uses lifespan in both opportunities: when selecting who to connect to 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 insures 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.

4.3 Taking Available Connection into Consideration

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 of connections and often stay active for several days [24]. Correspondingly, the number of maximum connections allowed by a given peer in our protocols is related to the peer's current lifespan. For our experiments this number ranged between 5 and 50, with an average value of 20.[6]

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

Table 2. Lifespan-based protocols and the different strategies used to select which acquaintances should a peer "befriend" and which ones should it recommend.

The following section presents evaluation results of the three protocols and compares them with two alternative ones that do not rely on lifespan information.

5 Evaluation

To explore the role of peers' lifespan distribution in P2P protocols, we have implemented a trace-driven simulator for P2P systems. Using a subset of the collected trace-data we evaluate the proposed lifespan-based protocols and compare them with two others that

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

[6]For completeness, we evaluated the performance of our protocols following the node capacity distribution model suggested by Chawathe et al. [5], where each node belongs to one of five capacity levels and has a maximum connection number directly proportional to its level. Nodes' capacities are assigned arbitrarily and have no correlation with the nodes' lifespans. The results, available upon request, are comparable to those included in the article.

represent decentralized unstructured and loosely structured P2P systems.

The remainder of this section describes our experimental setup and the different strategies evaluated. We then present results showing that P2P systems more resilient to the transient nature of their environments are possible with simple heuristics that prioritize long lived peers when selecting new "friends."

5.1 Experimental Setup

Our trace-driven, event-based simulator for P2P systems consists of about 3,500 lines of commented C++ code, implementing all membership management related messages. We are currently extending it to include different protocols for object searching and replication.

We ran our simulation (on a cluster of Linux PCs) using one of the 20 traces collected,[7] with a total simulation period of 510,000 seconds (or about six days), capturing the lifespan of 36,577 peers. The 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 in this section exclude this warm-up period (∼80,000 sec.).

5.1.1 Strategies

We evaluate three different protocols based on lifespan distribution and compare them with two decentralized protocols based on currently used systems. The lifespan-based protocols were introduced in Section 4; the remainder of this subsection describes the two alternative 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 [8] and *Hybrid Decentralized Protocol (HDP)* is modeled after hybrid protocols that rely on ultra- or super-peers [27] such as KaZaa [13] and Gnutella v0.6 [14]. We heavily rely on the specifications and available RFCs. When the specifications are vague or unavailable, we resort to our understanding of (open-source) clients currently in use and other publically available documents.

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

Both UDP and HDP utilize separate pools for cached pongs, one pool per connection. Upon receiving a ping message, the protocols *randomly* choose a specified number of pong entries from their caches (currently 10) and respond to the request.

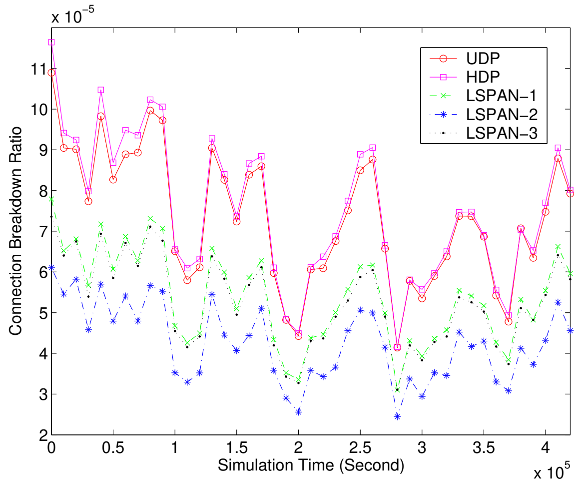
HDP distinguishes between leaf-peers and ultra-peers: ultra-peers can connect to any other peer (ultra or leaf), while leaf peers can only connect to ultra-peers. The scheme basically creates a two-level hierarchy among participating nodes, where more powerful, faster ultra-peers take over much of the load from slower ones. For our experiments we mark each peer as either leaf- or ultra-peer as indicated by our trace information.

5.2 Comparison

A good indicator of the effect of a protocol on system stability is the ratio of connection breakdowns to the number of effective connections. Figures 2 shows this ratio for every one of the protocols discussed, while the associated table presents some basic statistics including the average and standard deviation. As can be observed, all lifespan-based protocols yield much 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 42-43% by comparison with that of UDP and HDP. LSPAN-1 and LSPAN-3 yield comparative savings of 26% to 30% (by contrast with UDP) in the ratio of connection breakdowns in the system and their associated costs. Figure 3 show the closely related connection breakdowns per peer over time.

Both graphs show a clear *sawtooth* shape resulting from time-of-day patterns in our Gnutella-originated traces, something also observed in other studies of peer-to-peer systems [6, 25]. These time-of-day patterns are especially interesting when one considers the expected independence of Gnutella logical topology from geographic location [6].

It would be expected that the lifespan-based protocols' preference for long-lived peers will lead to higher numbers of connection requests to these nodes and,

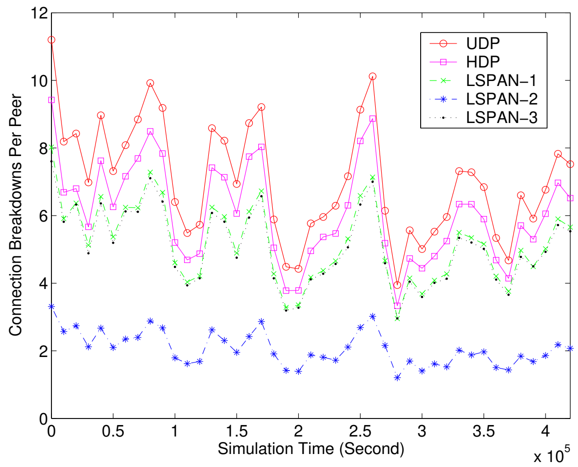


Connection Breakdown Ratio – Main Statistics

Protocol	Avg	Std	Min	Max
UDP	7.318	1.632	4.142	10.894
HDP	7.533	1.756	4.159	11.645
LSPAN–1	5.385	1.155	3.101	7.793
LSPAN–2	4.262	0.951	2.443	6.104
LSPAN–3	5.235	1.103	3.017	7.360

* Values are in 10E–5

Figure 2. Ratio of connection breakdowns to number of effective connections over time (aggregated over 10,000 sec.). The associated table shows some basic statistics including average, standard deviation as well as minimum and maximum observed values.

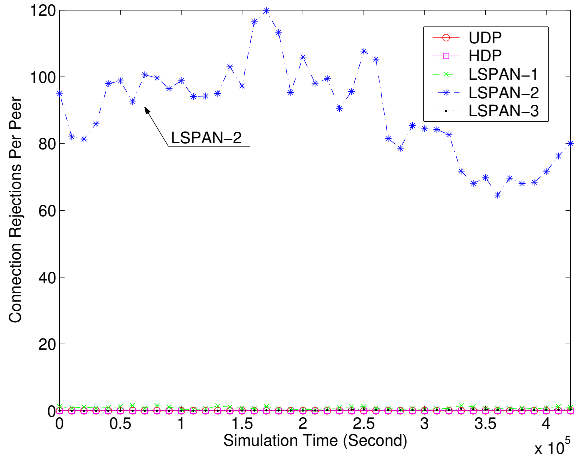


Connection Breakdowns per Peer – Main Statistics

Protocol	Avg	Std	Min	Max
UDP	7.067	1.694	3.395	11.199
HDP	6.100	1.462	3.331	9.420
LSPAN–1	5.199	1.184	2.954	8.023
LSPAN–2	2.072	0.504	1.205	3.309
LSPAN–3	5.064	1.137	2.961	7.602

Figure 3. Connection breakdowns per peer over time (aggregated over 10,000 sec.). The associated table shows some basic statistics including average, standard deviation as well as minimum and maximum observed values.

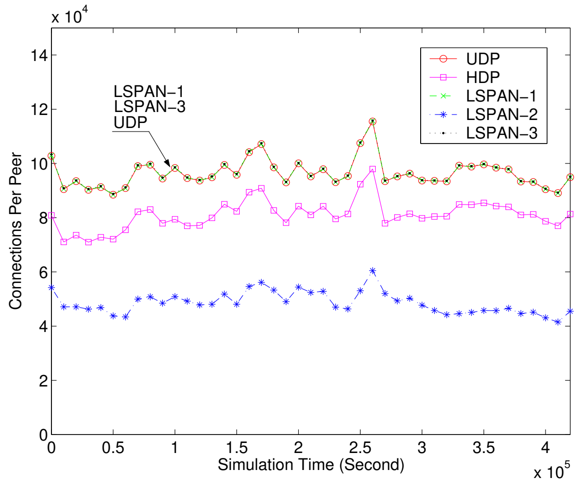
consequently, a higher overall number of connection rejections. This higher rejection number may be seen as a reasonable price to pay for longer-lasting "friendships," and its cost would become comparatively less important as peers increase their degree of investment in their friends. While our most selective protocol



Connection Rejections per Peer – Main Statistics

Protocol	Avg	Std	Min	Max
UDP	3.46E−5	2.27E−4	0	0.002
HDP	0.003	0.002	0	0.007
LSPAN−1	0.859	0.348	0.343	1.561
LSPAN−2	89.89	14.03	64.57	1.2E+2
LSPAN−3	0.158	0.158	0	0.621

Figure 4. Connection rejections per peer over time (aggregated over 10,000 sec.). The associated table shows some basic statistics including average, standard deviation as well as minimum and maximum observed values.



Connections per Peer – Main Statistics

Protocol	Avg	Std	Min	Max
UDP	0.964	0.053	0.886	1.154
HDP	0.811	0.054	0.709	0.980
LSPAN−1	0.965	0.053	0.884	1.157
LSPAN−2	0.486	0.040	0.415	0.605
LSPAN−3	0.966	0.053	0.888	1.158

* Values are in 10E+5

Figure 5. Connections per peer over time (aggregated over 10,000 sec.). The associated table shows some basic statistics including average, standard deviation as well as minimum and maximum observed values.

(LSPAN-2) has, indeed, a high rejection number, the rejection number of LSPAN-1 and LSPAN-3, although higher than that of UDP or HDP, is still low enough that it can be ignored (Figure 4). From the associated table of statistics in Figure 4, for example, we could see that the average connection rejection number per peer for LSPAN-1 is only 0.859 for every 10,000 seconds. This number further drops to 0.158 for LSPAN-3, meaning that, in average, a peer will only face a connection rejection every 17.58 hours!

It is interesting to note that the number of actual connections per peer does not directly follow from the number of rejections experienced by it. While all LSPANs protocols result in higher, if only minor, rejection numbers than UDP and HDP, LSPAN-1 and LSPAN-3 have even higher ratios of connections per peer than both UDP and HDP (Figure 5). Recall that LSPAN-1 and LSPAN-3 select long lasting peers from a random set of candidates and, in the case of LSPAN-3, with some knowledge of the candidate peers' available incoming connections. This can explain their high connection numbers, resulting from both long lasting connections (compared with UDP and HDP) and low rejection numbers (by contrast with LSPAN-2).

6 Conclusions and Future Work

We have presented trace-driven simulation results that illustrate the potential advantages of considering peers' observed lifespan distributions as a key system attribute in the design of P2P protocols. From independent measurements of peer lifespan in a current P2P system we developed a functional form that fits the distribution well. We use this distribution as the basis for a number of illustrative P2P protocols which we compare against two alternative ones for decentralized and unstructured or loosely-structured P2P systems. We find that even simple lifespan-based protocols can achieve up to 42% reductions in the ratio of connection breakdowns and their associated costs.

We are currently exploring the effect of lifespan-based protocols in different query and caching algorithms for P2P systems. While decentralized and unstructured protocols could yield more resilient systems, naive implementations of query mechanisms in these systems have been shown to scale poorly [15]. As part of our future work, we are investigating the benefits of the higher resilience of lifespan-based systems on the hit rates and response times of more scalable search mechanisms [15, 27]. Similarly, while potentially highly beneficial [28, 23], the effectiveness of caching in P2P systems will be directly connected to the lifespan of the caching peer. We have also started to experiment with these and similar ideas in the context of highly structured protocols [29, 30, 22, 18].

Acknowledgments

We would like to thank the Northwestern University Information Technology group and the Computing Support Group of the Department of Computer Science, in particular Roger A. Safian, Scott Hoover and Geoffrey Pagel, for their aid and understanding while obtaining the trace data used for our experiments. We are also grateful to Peter Dinda for stimulating conversations and to Jeanine Casler, Stefan Birrer and the anonymous reviewers for their helpful comments on early drafts of this article.

References

[1] E. Adar and B. A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), September 2000.

[2] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable peer finding on the Internet. In *Proc. of Globecom*, Taipei, Taiwan, R.O.C., November 2002.

[3] D. S. Bernstein, Z. Feng, B. N. Levine, and S. Zilberstein. Adaptive peer selection. In *Proc. of the 2nd IPTPS*, Berkeley, CA, USA, February 2003.

[4] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *Proc. of the 2nd IPTPS*, Berkeley, CA, USA, February 2003.

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

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

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability*, pages 43–58, Berkeley, CA, USA, July 2000.

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

[9] Entropia, Inc. http://www.entropia.com. 2003.

[10] Groobe Networks, Inc. http://www.groove.net. 2003.

[11] M. Harchol-Balter and A. B. Downey. Exploting process lifetime distribution for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, August 1997.

[12] S. Iyer, A. Rowstron, and P. Druschel. SQUIRREL: A decentralized, peer-to-peer web cache. In *Proc. of the ACM PODC*, Monterey, CA, USA, July 2002.

[13] KaZaa. http://www.kazaa.com. 2001.

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

[15] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of ICS*, New York, NY, June 2002.

[16] E. P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *Proc. of CC-Grid*, Berlin, Germany, May 2002.

[17] Napster. http://www.napster.com. 2003.

[18] Overnet. http://www.overnet.com. 2003.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of SIGCOMM*, pages 161–172, San Diego, CA, USA, August 2001.

[20] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1):50–57, January/February 2002.

[21] D. Roselli, J. R. Lorch, and T. E. Anderson. A comparison of file systems workloads. In *Proc. of the USENIX Annual Technical Conference*, San Diego, CA, June 2000.

[22] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the IFIP/ACM Middleware*, Heidelberg, Germany, November 2001.

[23] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. Boston, MA, USA, December 2002.

[24] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of Multimedia Computing and Networking*, San Jose, CA, USA, January 2002.

[25] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[26] SETI@Home. http://setiathome.ssl.berkeley.edu/. 2003.

[27] A. Singla and C. Rohrs. Ultrapeers: Another step towards Gnutella scalability. Working draft, Lime Wire LLC, December 2001.

[28] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. In *O'Reilly's http://www.openp2p.com*, February 2001.

[29] 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 SIGCOMM*, pages 149–160, San Diego, CA, USA, August 2001.

[30] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCV/CSD-01-1141, Computer Science Division, University of California, Berkeley, CA, USA, April 2001.