Residual-Based Measurement of Peer and Link Lifetimes in Gnutella Networks

Xiaoming Wang, Zhongmei Yao, and Dmitri Loguinov*
Department of Computer Science, Texas A&M University
College Station, TX 77843 USA
{xmwang, mayyao, dmitri}@cs.tamu.edu

***Abstract*—Existing methods of measuring lifetimes in P2P systems usually rely on the so-called *Create-Based Method* (CBM) [16], which divides a given observation window into two halves and samples users "created" in the first half every Δ time units until they die or the observation period ends. Despite its frequent use [2], [17], [19], this approach has no rigorous accuracy or overhead analysis in the literature. To shed more light on its performance, we first derive a model for CBM and show that small window size or large Δ may lead to highly inaccurate lifetime distributions. We then show that create-based sampling exhibits an inherent tradeoff between overhead and accuracy, which does not allow any fundamental improvement to the method. Instead, we propose a completely different approach for sampling user dynamics that keeps track of only *residual* lifetimes of peers and uses a simple renewal-process model to recover the actual lifetimes from the observed residuals. Our analysis indicates that for reasonably large systems, the proposed method can reduce bandwidth consumption by several orders of magnitude compared to prior approaches while simultaneously achieving higher accuracy. We finish the paper by implementing a two-tier Gnutella network crawler equipped with the proposed sampling method and obtain the distribution of ultrapeer lifetimes in a network of 6.4 million users and 60 million links. Our experimental results show that ultrapeer lifetimes are Pareto with shape $\alpha \approx 1.1$; however, link lifetimes exhibit much lighter tails with $\alpha \approx 1.9$.**

I. Introduction

Peer-to-peer networks are popular platforms for many applications such as file-sharing, content distribution, and multimedia streaming. Besides modeling and simulating system dynamics of P2P networks under churn (e.g., [3], [6], [8], [11]), validation of proposed techniques in real networks has recently become an important area for understanding P2P performance and design limitations in practice. In this regard, several efforts have been undertaken to characterize peer-to-peer systems by measuring churn-related user behavior (e.g., distribution of lifetime, inter-arrival delays, and availability) [1], [2], [4], [5], [17], [19], topological information (e.g., degree distribution and clustering coefficients) [13], [21], and traffic flow rate [8], [18].

Sampling of large-scale networks usually faces two fundamental problems – 1) obtaining an unbiased distribution of the target quantity and 2) keeping bandwidth overhead reasonable as system size increases. While sampling bias in topology measurement is understood fairly well [20], the same issue in lifetime sampling has not been addressed before. What makes the latter problem different is that sampled users *cannot* be queried for their lifetimes or even arrival instances. Measurement in such cases generally requires taking repeated snapshots of the system every Δ time units, detecting new arrivals by user appearance in a given snapshot, and inferring departures based on user absence in another snapshot. Since Δ cannot be lowered below the delay it takes to crawl the network, the issue of precisely reconstructing the lifetime distribution from measured samples remains open.

In this paper, we aim to formalize the notion of lifetime sampling bias, understand its source in existing methods, and design a robust and bandwidth-efficient sampling mechanism for estimating peer and link lifetime distributions in unstructured P2P networks (e.g., Gnutella [7], KaZaA [9]). Note that peer lifetimes are important for understanding general user behavior, their habits, and application performance offered by the peers to the system. Link lifetimes, on the other hand, have a significant impact on resilience [11], [22] and routing ability [10] of the network since broken links, rather than dead peers, contribute to formation of stale neighbor pointers, network disconnection, and routing failure.[1]

We start by creating a novel analytical framework for understanding and characterizing bias in network sampling. We first explain what constitutes inaccuracy in measuring the target distribution of lifetimes $F(x)$ and define sampling methods to be *biased* if, given an infinite population of sampled users, they cannot reproduce $F(x)$ in all discrete points $j\Delta$ in the interval $[\Delta, T]$. Armed with this definition, we then offer a closed-form model for the measurements obtained by *Create-Based Method* (CBM) [16], which is a widely used heuristic for sampling lifetimes in computer systems. We show that both CBM and its modification in [2], [17], [19] are biased as long as $\Delta > 0$, where the bias is caused by two factors – inconsistent round-offs (i.e., some user lifetimes are rounded up and others down) and missed users (i.e., users arrive and depart within a Δ interval). In fact, we generalize this result to show that *any* sampling technique that attempts to directly measure user lifetimes every Δ time units is biased as long as $\Delta > 0$ and that the bias is not removable regardless of the mathematical manipulation applied to the measured samples.

To overcome the discovered limitations of direct sampling, we next propose a technique called *ResIDual-based Estimator* (RIDE), in which a crawler takes a snapshot of the entire

*Supported by NSF grants CCR-0306246, ANI-0312461, CNS-0434940, and CNS-0519442.

[1]There are many reasons why peer lifetime may be different from link lifetime, which include peers reaching their maximum neighbor capacity and dropping excess links, leaves migrating from one ultrapeer to another to achieve better performance, path outages between certain nodes, and demotion of ultrapeers to leaf status.

network and then tracks the residual (i.e., remaining) lifetimes of the users seen in the first crawl. We show that this approach produces an unbiased version of the residual distribution $H(x)$, which allows us to develop a simple mechanism based on renewal churn models of [11], [22] that accurately reconstructs the lifetime distribution $F(x)$ from the sampled residuals with a negligible amount of error.

The next issue we address is bandwidth consumption. With small Δ and large T, CBM requires significant overhead since it must track *all* users that appear in the system in the observation interval, i.e., old peers discovered early in the crawl and new ones constantly arriving into the system. In RIDE, however, initial users die quickly and the amount of bandwidth needed to sustain the crawl decays to zero proportionally to the tail of the residual lifetime distribution $H(x)$. Additional bandwidth savings are possible if the initial set S_0 of users found in the system is uniformly subsampled and only ϵ-fraction of the users is monitored during the interval $[0, T]$. For example, given Pareto lifetimes with $\alpha = 1.1$ observed in our experiments, window $T = 24$ hours, and sampling interval $\Delta = 3$ minutes, the proposed technique reduces the download overhead compared to that in CBM by a factor of 16 for $\epsilon = 0.1$ and a factor of 125 for $\epsilon = 0.01$.

We finish the paper by implementing a Gnutella crawler that is about 18 times faster than the fastest prior crawler [19], which allows it to cover the entire network of 6.4 million users (1.2 million contacted ultrapeers) in under 3 minutes. Our results using RIDE indicate that ultrapeer lifetimes are Pareto distributed with shape $\alpha \approx 1.1$, which is very close to the results of [2]. At the same time, Gnutella links are much more volatile and can be described by a Pareto distribution with shape $\alpha \approx 1.9$. These results, fed into the latest resilience models for unstructured systems [11], [12], [22], suggest that node isolation among joining ultrapeers in Gnutella and thus partitioning of the network must indeed be extremely rare events.

The remainder of the paper is organized as follows. In Section II, we formalize sampling and bias. In Section III, we derive the sampling bias of CBM and examine it under different simulation settings. We propose the residual-based method and discuss its simulation results in Section IV. We examine the bandwidth overhead of the various methods in Section V and present our measurement study of Gnutella in Section VI. Section VII reviews prior work and Section VIII concludes the paper.

II. Formalizing Lifetime Sampling

A. Target Distribution

We start by defining the objective of our measurement process. Assume that each user spends a random amount of time in the system, where the lifetime L of the next joining user is drawn from some distribution $F(x)$. This is similar to the heterogeneous churn model proposed in [22]. Then, the goal of the sampling process is to estimate with as much accuracy as possible function $F(x)$, which we assume is continuous *almost everywhere*[2] in the interval $(0, \infty)$. As shown in [22], distribution $F(x)$ represents the lifetimes of *arriving* rather than *existing* peers in the system. The latter metric is known in renewal process theory as the *spread* of user lifetimes and can be obtained from $F(x)$ using simple integration.

The measurement process is assumed to have periodic access to the information about which users are currently present in the system. This process allows the sampler to test whether a given user i is still alive as well as discover the entire population of the system at any time t. However, due to bandwidth and connection-delay constraints on obtaining this information, the sampling process cannot query the system for longer than T or more frequently than once per Δ time units, where Δ usually varies from several minutes to several hours depending on the speed of the crawler and network size.

Given the above requirements, notice that reconstructing the entire $F(x)$ from discrete samples is simply impossible. There are three biases arising from discrete sampling: 1) the measuring process cannot observe any lifetimes larger than T; 2) all samples are rounded to a multiple of Δ; 3) an empirical distribution based on a finite sample size will not necessarily match the theoretical one. We are not concerned with the last issue since *all* methods require an infinitely large sample size to converge to the desired distribution $F(x)$. Instead, we are interested in the bias arising from finite T and non-zero Δ.

We start with the following definition that formalizes samples obtained during periodic measurements.

Definition 1: A non-negative random variable X^Δ for some $\Delta > 0$ is called *lattice* if:

$$\sum_{j=1}^{\infty} P(X^\Delta = j\Delta) = 1, \tag{1}$$

where Δ is called the *periodicity* of X^Δ and points $x_j = j\Delta$ are called the *support* of X^Δ.

For all lattice distributions, we assume that $P(X^\Delta \leq 0) = F(0) = 0$ and that the probability mass of X^Δ starts from the point $x_1 = \Delta$.

We are now ready to define a sampling process.

Definition 2: A (Δ, T)-*sampling process* is a lattice random variable M^Δ with periodicity Δ and $P(\Delta \leq M^\Delta \leq T) = 1$.

Note that the above defines a sampling process using the *limiting* distribution of the values it measures (i.e., assuming an infinite population size). The reason for doing so is to understand whether a method can provide accurate results given a sufficiently large sampling size. As we show below, some methods *always* exhibit bias, no matter how long they measure the system.

Definition 3: For a random variable X, function $E(x)$ is called an *estimator of* X in some interval $[a, b]$ if it is the CDF of some random variable Y that approximates X in $[a, b]$.

Note that Y can be arbitrarily dissimilar to X, in which case the estimator will be biased. We next explain what makes an estimator unbiased.

Definition 4: A (Δ, T)-sampling process with estimator $E(x)$ is *unbiased* with respect to a target continuous random variable X if it can correctly reproduce the distribution of X

[2]The set of points in which $F(x)$ is discontinuous must have measure 0.

in all discrete points x_j in the interval $[\Delta, T]$ for any $\Delta > 0$:

$$E(x_j) = P(X \leq x_j) \tag{2}$$

for $x_j = j\Delta$ and $j = 1, 2, \ldots, T/\Delta$.

Since one may measure different aspects of the system, we finally classify sampling methods based on whether they measures the target random variable or some other related distribution.

Definition 5: A (Δ, T)-sampling process of a random variable X is called *direct*, if it measures quantities whose distribution is the same as that of X. It is called *indirect* otherwise.

For example, direct lifetime sampling must measure session lengths of all arriving users, while indirect sampling may record the lifetimes of peers alive in the system at some time t. Given an established relationship between the two metrics, an estimator can then be used to reconstruct lifetimes L from indirect samples. In another example, direct sampling of network size must count the number of users present in the system at different times t, while indirect sampling may measure the arrival process of peers into the system. With proper selection of the indirect sampling method, estimation may be more accurate and/or may require lower overhead than direct sampling. We demonstrate one such example later in the paper.

III. Direct Sampling

In this section, we first examine the source of bias in direct sampling and study the problem of constructing an unbiased estimator for measuring lifetimes. We then derive a model for the distribution obtained by Create-Based Method (CBM) and demonstrate examples of its bias.

A. General Results

In direct sampling, the measured random variable M^Δ is the lifetime of individual users conditioned on them being smaller than T and being present in the crawl:

$$P(M^\Delta \leq x) = P(L \leq x | L \leq T, \text{not missed}), \tag{3}$$

where missed samples arise when a user joins and departs between consequent crawls. Note, however, that not all users with lifetimes smaller than Δ are missed and that some of them are actually taken into account in the distribution of M^Δ. Another issue that we discover in this work is that some lifetime samples are rounded up and others rounded down during the measurement, which together with missed users gives rise to the bias we derive below. We next formalize round-off errors and explain how they affect direct sampling.

Definition 6: For a continuous random variable X, a (Δ, T) sampling process is *consistent* if measured samples are all rounded up to the nearest multiple of Δ.

Since a crawler in direct sampling never knows the exact arrival time of users it observes, there is an ambiguity in how to round-off the lifetimes of measured peers. Consider the example in Fig. 1, where sample $L_1 = 0.5\Delta$ is indistinguishable from sample $L_2 = 1.8\Delta$ from the perspective of the crawler. This causes both of these lifetimes to be rounded off to Δ, which using our terminology makes L_1 consistent and L_2 inconsistent. Also observe in the figure that samples $L_3 = 0.4\Delta$ and $L_4 = 0.6\Delta$ are completely missed by the crawler, even though sample L_1 is captured. This case can also be treated as inconsistent round-off as we define below.

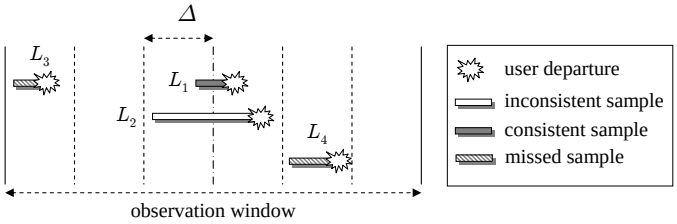


Fig. 1. Round-off inconsistencies in direct sampling.

Let

$$Q_j = \begin{cases} 1 & \text{inconsistently rounded down to } x_j \\ 0 & \text{otherwise} \end{cases}$$

to be an indicator variable of the event that a user's lifetime $x_j \leq L < x_{j+1}$ is inconsistently rounded down to x_j by the sampling process, where rounding down to $x_0 = 0$ represents missing the entire sample. For simplicity of notation, we define $\rho_j = P(Q_j = 1)$ and obtain the probability of inconsistent round-off in the interval $[x_j, x_{j+1})$ in the next theorem.

Theorem 1: In direct sampling, the probability that lifetime samples are inconsistently rounded down to $x_j = j\Delta$ ($j = 0, 1, ..., T/\Delta$) is:

$$\rho_j = \frac{1}{\Delta} \int_{x_j}^{x_{j+1}} F(x)dx - F(x_j), \tag{4}$$

where $F(x)$ is the CDF of the lifetime distribution of samples.

Equipped with result in (4), we next derive an unbiased estimator for the continuous random variable L.

Theorem 2: For direct lifetime sampling, the following is an unbiased estimator of L:

$$E(x_j) = P(M^\Delta \leq x_j)P(L \leq T | Q_0 = 0)(1 - \rho_0) + \rho_0 - \rho_j, \tag{5}$$

where ρ_j is given in (4).

Note that both (4)-(5) are exact, but due to limited space we do not show simulations verifying their accuracy. From the result of Theorem 2, it becomes clear that unbiased measurement requires access to the distribution of samples (i.e., variable M^Δ), the fraction of observed lifetimes that are no larger than T (i.e., $P(L \leq T | Q_0 = 0)$), and all individual ρ_j. While the first two metrics are easily measurable in practice, recovery from inconsistent round-offs requires the exact join time of each sampled user and the number of missed users. Unfortunately, within the constraints of our problem (i.e., crawling of alive users with a period no less than Δ), the effect of round-off errors is impossible to overcome no matter what manipulation is applied to M^Δ.

B. Create-Based Method (CBM)

We next study how inconsistent round-offs exhibit themselves in a widely used [2], [17], [19] direct sampling algorithm called *Create-Based Method* (CBM), first introduced by

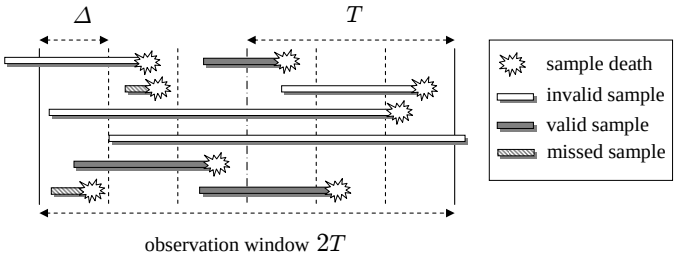
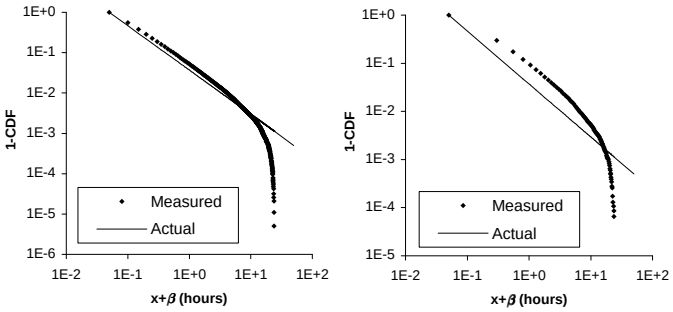


Fig. 2. Illustration of sampling in CBM.



(a) $\Delta = 3$ min (b) $\Delta = 15$ min

Fig. 3. Estimator E_A with Pareto lifetimes ($n = 10^6$ users, $T = 24$ hours, $\alpha = 1.1$, $\beta = 0.05$, and $E[L] = 0.5$ hours).

[16] in the context of operating systems. Recall from [16] that CBM uses an observation window of size $2T$, which is split into small intervals of size Δ. Within the observation window $[0, 2T]$, the algorithm takes a snapshot of the system at the beginning of each interval. To avoid sampling bias, [16] suggests dividing the window into two halves and only including samples that appear during the first half of the window, disappear somewhere within the window, and stay in the system no longer than T time units. Fig. 2 shows an example of create-based sampling with three valid, four invalid, and two missed lifetime samples. The invalid cases include users who join the system before the observation window begins or in its second half, a peer that survives beyond time $2T$, and a user whose lifetime is larger than T.

Based on the collected snapshots, the algorithm obtains individual lifetimes M^Δ and calculates the corresponding distribution $P(M^\Delta \leq x_j)$ for $j = 1, 2, ..., T/\Delta$. Assume that N is the number of users that arrive into the system in the first half of the window $[0, T]$ and $N(x)$ is the number of such users with lifetimes less than or equal to x. Observe that $N(T)$ is the number of valid samples collected by CBM and $\lim_{N\to\infty} N(T)/N$ is the simply metric $P(L \leq T|Q_0 = 0)$ defined earlier. One possible way to estimate $F(x)$ is to take the ratio of $N(x_j)$ to $N(T)$ as the estimator of the probability $P(L \leq x_j)$, which leads to our first CBM estimator [16]:

$$E_A(x_j) = \lim_{N\to\infty} \frac{N(x_j)}{N(T)} = P(M^\Delta \leq x_j). \tag{6}$$

Recent work in [2], [17], [19] normalizes E_A by the percentage of samples no larger than T (i.e., $N(T)/N$) and defines the following modified estimator:

$$E_B(x_j) = \lim_{N\to\infty} \frac{N(x_j)}{N}. \tag{7}$$

With the result in (5), we can express both CBM estimators as functions of the actual distribution $F(x_j) = P(L \leq x_j)$.

Theorem 3: Both CBM estimators (6)-(7) are biased when any $\rho_j > 0$ and produce the following distributions:

$$E_A(x_j) = \frac{F(x_j) - \rho_0 + \rho_j}{F(T) - \rho_0}, E_B(x_j) = \frac{F(x_j) - \rho_0 + \rho_j}{1 - \rho_0}. \tag{8}$$

The result in (8) shows that E_B is closer to $F(x)$ than E_A since its accuracy is not affected by the value of T. Next, we explore in more detail the effect of (Δ, T) on the fidelity of these estimators using model (8) and simulations.

C. Effect of Bias on CBM

We first explain how T and Δ skew the shape of estimator E_A. To simplify the discussion below, define $\bar{E}(x) = 1 - E(x)$ to be the tail distribution of any CDF function $E(x)$. It then follows from (8) that:

$$\bar{E}_A(x) = \frac{\bar{F}(x) - \bar{F}(T) - \rho_j}{F(T) - \rho_0}, \tag{9}$$

which shows that the measured tail distribution is a shifted and scaled version of the true tail. The influence of the shift/scale factors on the right side of (9) could be illustrated using the example of a Pareto distribution commonly used to model user lifetimes [11], [22]:

$$F(x) = 1 - (1 + x/\beta)^{-\alpha}, \alpha > 1, x \geq 0, \tag{10}$$

with $E[L] = \beta/(\alpha - 1)$. We use CBM with $T = 24$ hours in a hypothetical network with $n = 1$ million users that join and depart using the churn model of [22]. Even though $F(T) = 99.8\%$ of users have lifetimes smaller than T, Fig. 3 shows that E_A suffers from significant bias that increases as Δ becomes larger. Not only does the measured distribution E_A produce incorrect estimates $\alpha \approx 2.4, \beta \approx 0.5$ of Pareto parameters when fitted with the corresponding curve, but the shape of the tail in Fig. 3 does not even resemble that of $\bar{F}(x)$, which may lead to erroneous conclusions about the family of distributions $F(x)$ belongs to.

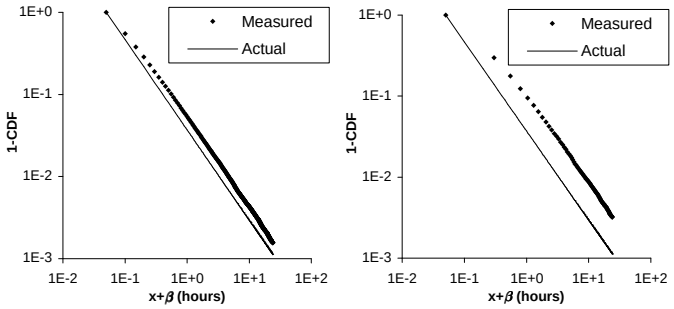
We now study how ρ_j affects the shape of E_B. It follows from (8) that for $j = 0, 1, 2, ..., T/\Delta$:

$$\bar{E}_B(x_j) = \frac{\bar{F}(x_j) - \rho_j}{1 - \rho_0}, \tag{11}$$

which is the true tail shifted by ρ_j and then scaled by $1 - \rho_0$. For small $\rho_j \approx 0$, this transformation on log scale preserves the Pareto shape parameter α as seen in Fig. 4, but makes scale parameter β inaccurate (i.e., $\alpha \approx 1.14, \beta \approx 0.15$ for $\Delta = 15$ minutes). For cases of non-negligible ρ_j that arise when Δ is very large or when distribution $F(x)$ does not admit shape invariance during scaling (e.g., Gaussian, uniform), estimator E_B may produce significantly misleading results.

IV. Indirect Sampling

In this section, we seek a solution to the problem of achieving both high accuracy and low overhead using indirect



(a) $\Delta = 3$ min (b) $\Delta = 15$ min

Fig. 4. Estimator E_B with Pareto lifetimes ($n = 10^6$ users, $T = 24$ hours, $\alpha = 1.1$, $\beta = 0.05$, and $E[L] = 0.5$ hours).

sampling. It has been suggested [1], [11], [22] that users in peer-to-peer systems can be modeled as alternating between available (ON) and unavailable (OFF) states. Inspired by these efforts, we now propose our measurement algorithm, called *ResIDual-based Estimator* (RIDE), that exploits renewal theory [14] to reconstruct $F(x)$ from sampled residual lifetimes.

A. Churn Model

Consider a P2P system with n participating users, where each user i is either alive (i.e., present in the system) at time t or dead (i.e., logged off). This behavior can be modeled by an ON/OFF process $\{Z_i(t)\}$ for each user $i = 1, 2, ..., n$:

$$Z_i(t) = \begin{cases} 1 & \text{user } i \text{ is alive at time } t \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

This framework is illustrated in Fig. 5, where X_i and Y_i are the durations of user i's ON (life) and OFF (death) periods, respectively, and $R(t)$ is the remaining lifetime of user i at time t. Assume that $\{X_i\}$ are drawn from distribution $F_i(x)$ and $\{Y_i\}$ from $G_i(x)$. It has been proven in [22] that the lifetime of the next joining user into the system is drawn from a weighted distribution:

$$F(x) = \omega \sum_{i=1}^{n} \frac{F_i(x)}{E[X_i] + E[Y_i]}, \tag{13}$$

where $\omega = 1/\sum_{l=1}^{n} (E[X_l] + E[Y_l])^{-1}$. As before, define L to be the lifetime of the next joining peer whose distribution is specified by $F(x)$. Then, the goal of our and other measurement studies is not to sample each of $F_i(x)$, but rather to measure the users' aggregate behavior $F(x) = P(L < x)$.

B. RIDE

We first define the sampling algorithm in RIDE and then discuss its estimator $E_R(x)$. At time 0, RIDE takes a snapshot of the whole system and records in set S_0 all users found to be alive. For all subsequent intervals j ($j = 1, 2, ..., T/\Delta$) of Δ time units, the algorithm keeps probing peers in set S_0 either until they die or T expires. After the observation window is over, the algorithm obtains the distribution of residual lifetime M^{Δ} of the users in set S_0.

Two important properties about residual sampling can be drawn from its definition: 1) no valid samples can be missed

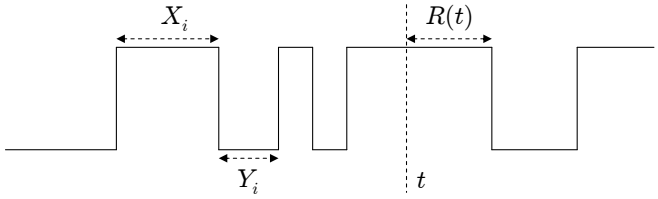


Fig. 5. Process $Z_i(t)$ depicting user i's ON/OFF behavior.

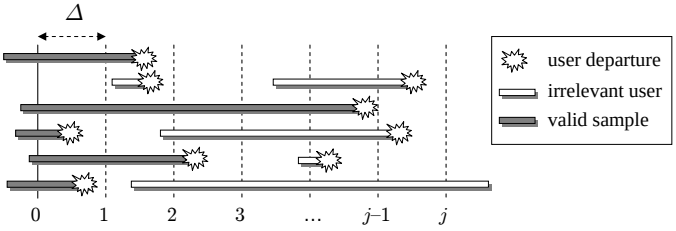


Fig. 6. Sampling residuals in RIDE.

since only users who are alive at time $t = 0$ are valid measurements; 2) no samples can be inconsistently rounded off since all valid residual lifetimes start from the time of the first crawl. Fig. 6 illustrates an example of five valid samples captured in the first crawl and five irrelevant lifetimes that are safely ignored by the algorithm.

Given the above observations, it immediately follows that the measured distribution can be used to obtain an unbiased estimator of the actual residual distribution:

$$P(R(t) \le x_j) = \lim_{|S_0| \to \infty} \frac{N(x_j)}{|S_0|}, \tag{14}$$

where as before $N(x)$ denotes the number of samples in S_0 smaller than or equal to x. At any time $t \gg 0$, the distribution of the remaining lifetime $R(t)$ of users present in the system is given by [22]:

$$H(x) = P(R(t) \le x) = \frac{1}{\mu} \int_0^x (1 - F(u)) du, \tag{15}$$

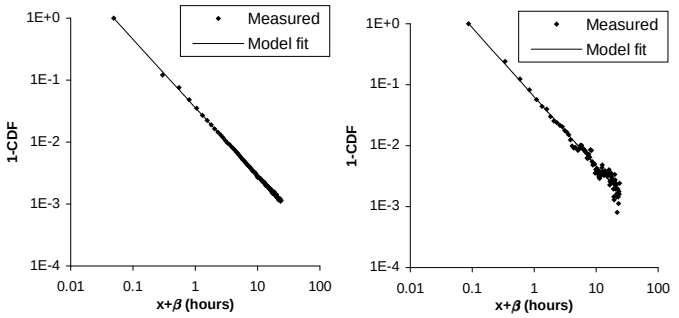
where $\mu = E[L]$ is the expected lifetime of a joining peer and system size n is sufficiently large for all $Z_i(t)$ to be in equilibrium. This leads to the following result.

Theorem 4: For residual lifetime sampling, the following is an unbiased estimator of L:

$$E_R(x_j) = 1 - \frac{h(x_j)}{h(0)}, \tag{16}$$

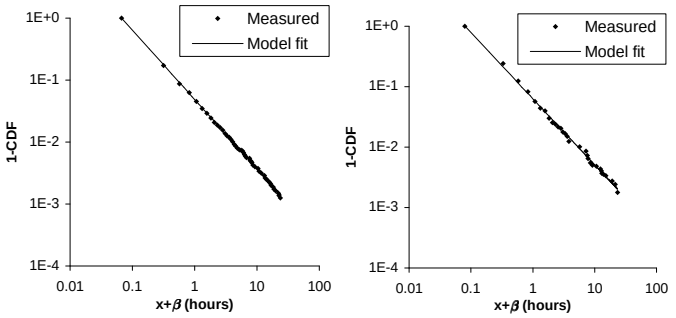
where $x_j = j\Delta$ and $h(x_j)$ is the PDF of $R(t)$. Furthermore, the expected user lifetime is given by $E[L] = 1/h(0)$.

Since $H(x)$ is obtained without bias, it is now possible to numerically compute its derivative $h(x)$ using Taylor expansion with error bounded by $O(\Delta^k/k!)$, where $k = T/\Delta$ is the number of samples in the curve. For $\Delta = 3$ minutes and $T = 24$ hours commonly used in our experiments, the resulting error is $\Delta^{480}/480! \approx 10^{-1960}$, which for all practical purposes can be considered zero. In simulations, however, we find that using only 3 points is often sufficient for achieving good estimation accuracy (see below).



(a) $\epsilon = 1$ (b) $\epsilon = 0.01$

Fig. 7. Original and subsampled estimator E_R with Pareto lifetimes ($|S_0| = 10^6$ users, $T = 24$ hours, $\Delta = 15$ minutes, $\alpha = 1.1$, $\beta = 0.05$, and $E[L] = 0.5$ hours). Both examples use 3-point derivatives.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.01$

Fig. 8. Inverse averaging applied to E_R for Pareto lifetimes ($|S_0| = 10^6$ users, $T = 24$ hours, $\Delta = 15$ minutes, $\alpha = 1.1$, $\beta = 0.05$, and $E[L] = 0.5$ hours). Both examples use 3-point derivatives.

C. Subsampling

It is worthwhile mentioning that residual sampling acquires all valid samples during the very first crawl. Therefore, given that $|S_0|$ is sufficiently large, it is possible to randomly subsample the initial set of users and track the residuals of only ϵ percent of the entire user population. This significantly reduces traffic requirements and allows RIDE to achieve orders of magnitude lower bandwidth overhead in practice compared to CBM. One example of applying subsampling is shown in Fig. 7, where a system of 1 million users in the same setup as in Fig. 4(b) is subsampled by a factor of 100. First notice in Fig. 7(a) that RIDE recovers $F(x)$ with much higher accuracy than E_B and obtains $\alpha = 1.099$ and $\beta = 0.049$. Second, observe in Fig. 7(b) that RIDE achieves reasonable estimation accuracy ($\alpha = 1.13, \beta = 0.087$) even with just $10,000$ users; however, the tail of the subsampled distribution is highly variable, which potentially makes it difficult to understand the distribution's qualitative behavior.

To overcome the tail noise arising when $|S_0|$ is heavily subsampled, we next present an algorithm for reducing the variance in the measured distribution $E_R(x)$. Notice that $E_R(x)$ is a mapping between two discrete sets, i.e., from set $\mathcal{X} = \{j\Delta\}$ to set $\mathcal{Y} = \{j/|S_0|\}$ for $j = 1, 2, \ldots, T/\Delta$. For each $y \in \mathcal{Y}$, we find all $x_i \in \mathcal{X}$ such that $E_R(x_i) = y$ and calculate the corresponding average $\hat{x}(y)$:

$$\hat{x}(y) = \frac{\sum_i x_i \mathbf{1}_{E_R(x_i)=y}}{\sum_k \mathbf{1}_{E_R(x_i)=y}}, \tag{17}$$

where $\mathbf{1}_A$ is the indicator function of event A. Denote by $\hat{\mathcal{X}}$ the set of all possible $\hat{x}(y)$ from (17), i.e., $\hat{\mathcal{X}} = \{\hat{x}(y) | y \in \mathcal{Y}\}$ and define *inverse averaging* to be a relation $(\hat{x}(y), y)$ for all $y \in \mathcal{Y}$. By smoothing out the tail, inverse averaging improves the shape of the distribution and allows better accuracy in estimation.

Next, we examine two cases of inverse averaging using the example in Fig. 7(a) subsampled with $\epsilon = 0.1$ and $\epsilon = 0.01$. The resulting distributions are shown in Fig. 8, which demonstrates much better preservation of the Pareto shape in the tail and less oscillations than without the use of inverse averaging. For $\epsilon|S_0| = 10^5$ in Fig. 8(a), curve fitting produces $\alpha = 1.12, \beta = 0.067$, and for $\epsilon|S_0| = 10^4$ in Fig. 8(b), we obtain $\alpha = 1.09, \beta = 0.079$. This shows that even when Δ is comparable to the average lifetime $E[L]$ and with very few samples, RIDE is capable of reasonably accurate estimation. Due to limited space, we omit additional simulations that show insensitivity of RIDE to selection of Δ and T, significant improvement in accuracy for less heavy-tailed cases, and convergence of $E(x)$ to $F(x)$ as $|S_0| \to \infty$.

V. Overhead

This section formalizes the overhead of the various studied sampling methods and compares bandwidth requirement of RIDE to that of CBM.

In CBM, we are interested in the expected overhead of crawling all users in the graph every Δ time units in the interval $[0, T]$ and then probing peers that were present in the system at time $t = T$ under they die or their lifetime exceeds T time units. After some technical manipulation, we obtain the following result.

Theorem 5: Total bandwidth overhead of (Δ, T)-sampling using CBM is given by:

$$b_{CBM} = \frac{Cn}{\Delta}\left(T + \int_0^T [H(T) - H(x)]dx\right), \tag{18}$$

where n is the number of users in the system, C is the cost of probing or crawling a user, and $H(x)$ is the CDF of residual lifetimes.

Note that RIDE only probes users that are captured in the first crawl until they die or T expires. Taking into account ϵ-subsampling, we have the following theorem.

Theorem 6: Total bandwidth overhead of (Δ, T)-sampling using RIDE is given by:

$$b_{RIDE} = \frac{C|S_0|}{\Delta}\left(\Delta + \epsilon\int_0^T [1 - H(x)]dx\right), \tag{19}$$

where ϵ is the fraction of peers retained in the initial set S_0.

As long as $\epsilon|S_0|$ is sufficiently large, RIDE has the same accuracy as $E_R(x)$, but at significantly smaller overhead. Define $q(\epsilon)$ to be the ratio b_{CBM}/b_{RIDE} for $|S_0| = n$ and notice that for very small ϵ/Δ, metric $q(\epsilon)$ reduces to a simple formula T/Δ, which for example is 480 for $T = 24$ hours and $\Delta = 3$ minutes. Assuming Pareto lifetimes with shape α, Table I shows the exact savings gained by using

TABLE I
COMPARISON OF OVERHEAD USING $E[L] = 1$ HOUR, $\Delta = 3$ MINUTES

α	T	$q(0.1)$	$q(0.01)$	α	T	$q(0.1)$	$q(0.01)$
1.1	24 hrs	16	125	2	24 hrs	71	319
	48 hrs	17	151		48 hrs	116	573
	72 hrs	18	164		72 hrs	157	811

residual subsampling. The table shows that RIDE can reduce traffic overhead by a factor of $16 - 800$ compared to CBM depending on the tail weight of $F(x)$, sampling duration T, and subsampling factor ϵ.

In practice, one can choose ϵ based on the size of the initial set S_0 such that $\epsilon|S_0|$ is fixed at some pre-determined value. Simulations show that $\epsilon|S_0|$ between 10^4 and 10^5 users allow accurate reconstruction of $F(x)$ even when lifetimes are very heavy-tailed and Δ is large. Given this dynamic selection of ϵ, it becomes clear that RIDE can scale to arbitrarily large systems since it requires monitoring only a fixed number of users that does not depend on system size $|S_0|$. Note that similar subsampling is not possible in CBM since the latter requires *full* system crawls to discover new users.

VI. EXPERIMENTS

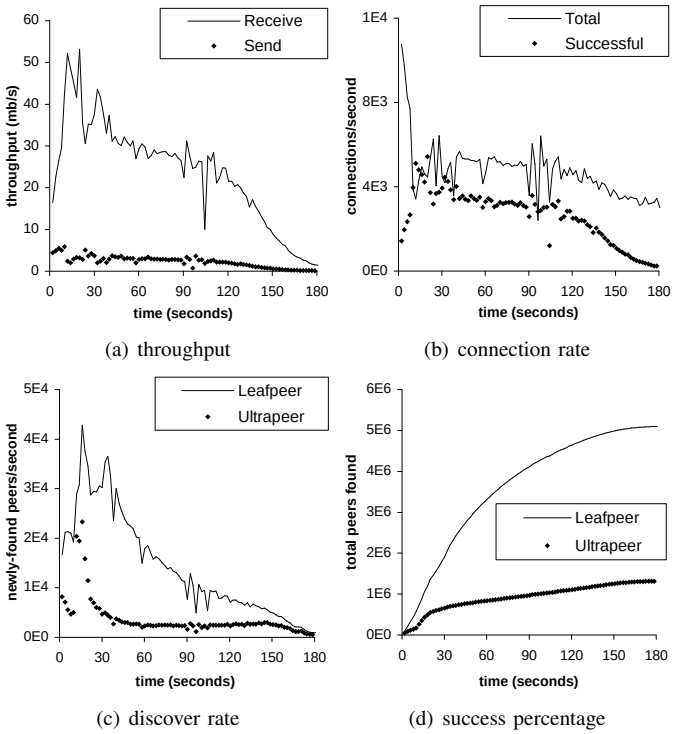
In what follows in this section, we apply the residual-based algorithm to crawl the Gnutella network and estimate the distributions of peer/link lifetimes.

A. Gnutella Crawler

Recent Gnutella networks are implemented in a two-tier structure that contains ultrapeers and leaves. Ultrapeers are responsible for forwarding search requests between each other, while leaves stay at the "edge" of the network and connect to several ultrapeers that provide them with search capabilities. A recent extension to the Gnutella protocol provides a crawler-friendly mechanism: upon receiving a crawl request (i.e., a handshake message with the "Crawler" field), a Gnutella client replies with a complete list of the identities of its neighboring peers. According to our experiments, an ultrapeer on average connects to 28.5 ultrapeers and 25.7 leaves, while a leaf is usually attached to only one or two ultrapeers.

To sample lifetimes of real Internet users using RIDE, we designed and built a scalable Gnutella crawler called GnuSpider that can operate in networks with millions of hosts and maintain reasonably small values of sampling period Δ. As most other crawlers, GnuSpider starts the crawl using a default seed file of ultra-peers and then contact them to obtain their neighbor lists, which are then used in a BFS search to discover all currently alive ultra-peers in the system. Neighbor lists in Gnutella include other ultra-peers with whom a connection is currently active, suggested ultra-peers who may or may not be online, and leaf peers currently attached as children. The crawler records leaf peers for statistical purposes, but only contacts nodes found in the other two lists.

Our GnuSpider implementation is a single-threaded Windows process that uses asynchronous completion ports (IOCP) to manage up to $60,000$ simultaneous connections to other



(a) throughput (b) connection rate

(c) discover rate (d) success percentage

Fig. 9. Statistics of a 3-minute crawl on July 22, 2006 (single-core, dual-CPU Xeon computer @ 3GHz).

hosts. To reduce the effect of timeouts and allow scalability, GnuSpider limits all TCP connection timeouts to 9 seconds, includes a low-overhead management of the BFS queue, and avoids socket re-binding between connections. Figures 9(a)-(b) show bandwidth consumption in one crawling example and the number of connections per second generated by the crawler. As seen in the figure, the crawler downloads data at sustained rates of 30 mb/s and attempts on average $400,000$ connections per minute. Since a certain percentage of SYN requests encounter dead or firewalled peers, the number of *successful* ultra-peer contacts lingers at $216,000$/min.

Experiments with GnuSpider show that we can cover the entire Gnutella network in 3 minutes and typically discover close to 6.4 million users in the process (1.2 million of which are the ultra-peers that we attempt to contact and 5.2 million are leaf nodes). During the first 120 seconds of the crawl, the discovery rate of new leaves shown in Figure 9(c) varies between $40,000$/second and $10,000$/second and that of new ultra-peers stays on average at $3,000$/second. It can also be seen from the figures that the last 60 seconds of the crawl usually produce a very small number of new peers since most of these connections experience a timeout. As illustrated in Figure 9(d), 90% of ultra-peers (i.e., 1.1 million) and leaf nodes (i.e., 4.5 million) can be discovered in just 100 seconds.

Comparison of GnuSpider to crawlers in prior experimental P2P work is shown in Table II, which provides the sampling period Δ, window duration T, the number of peers periodically probed with SYN packets or discovered during an actual crawl, and the crawling speed in terms of contacted hosts per minute. Observe in the table that GnuSpider is not only 18 times faster than the fastest crawler in prior literature [19],

TABLE II
COMPARISON OF P2P MEASUREMENT STUDIES

Approach	Measured network	Interval Δ	Duration T	Peers seen		Crawling machines	Year crawled	Connections per minute
				Probed	Crawled			
GnuSpider	Gnutella	3 min	$24-72$ hrs	–	6.4 million	1	2006	$400,000$
Stutzbach *et al*. [19]	Gnutella	7 min	48 hrs	–	1.3 million	7	2004	$22,500$
Liang *et al*. [13]	FastTrack	5 min	65 hrs	965	–	N/A	$2004-2005$	N/A
Bustamante *et al*. [2]	Gnutella	21 min	7 days	–	$500,000$	17	2003	<5000
Bhagwan *et al*. [1]	Overnet	20 min	7 days	$2,400$	–	1	2003	N/A
Chu *et al*. [4]	Gnutella	10 min	9 wks	$5,000$	–	1	2002	N/A
Ripeanu *et al*. [15]	Gnutella	hours	–	–	$48,195$	N/A	2001	<1000
Saroiu *et al*. [17]	Gnutella	7 min	60 hrs	$17,125$	–	N/A	2001	N/A

but it also discovers almost 5 times more concurrent users than any other study.

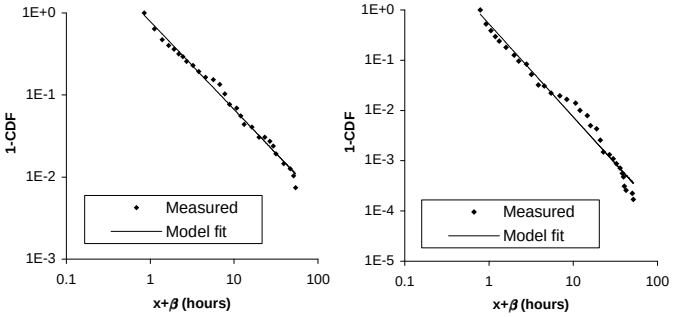
B. Peer Lifetimes

Users arriving into Gnutella immediately attempt to establish several neighboring connections to other peers currently in the system to increase their own resilience and enable themselves to route requests into the network. However, since leaves and users behind firewalls do not generally accept connection requests, selection of neighbors is often limited to non-firewalled, or as we call them *responsive*, ultrapeers.[3] Therefore, measurement of responsive ultrapeers provides the most useful information about the lifetime of future neighbors acquired by arriving users and allows parameter selection for existing P2P models based on lifetime distributions [10], [11], [12], [22]. Thus, our experiments below focus only on lifetimes of ultrapeers that respond to our connection requests and the links associated with them.

To measure peer lifetimes for the plots shown below, we first obtained using GnuSpider the initial set S_0 of about 468 thousand responsive ultrapeers and subsampled it using $\epsilon = 0.213$. Then, GnuSpider probed $\epsilon|S_0| = 100,000$ users for $T = 72$ hours checking if each peer was alive every 3 minutes. It should be noted that we found that in our experiments that a certain amount of peers exhibited erratic behavior, i.e., they would respond to one request, then become silent for several subsequent requests, and eventually become responsive again. This phenomenon appeared when a peer either was too busy to reply or implemented a certain rate-limiting strategy. To filter out the effect of this behavior, we set a threshold u for how many times a peer must appear unresponsive before we declare that user dead. In the crawls below, we use $u = 3$.

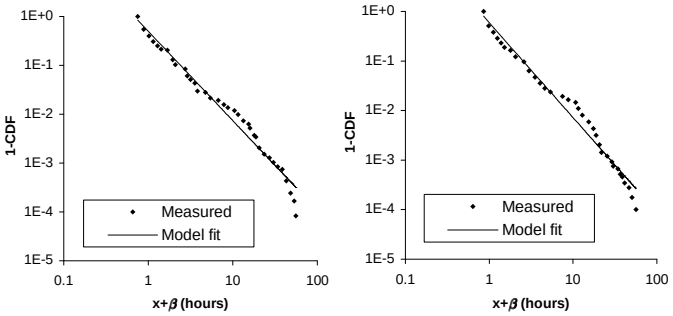
After the observation window in GnuSpider had expired, an off-line program read the GnuSpider logs and applied RIDE to reconstruct $F(x)$. Fig. 10(a) shows the resulting inverse-averaged tail distribution $1 - E_R(x)$ for the set of responsive ultrapeers. The figure matches well with a Pareto distribution with $\alpha = 1.09$ and $\beta = 0.85$, where the shape parameter is very close to that observed in [2]. Denote by r the expectation of residual lifetimes conditioned on the fact that $R(t)$ is within the observation window T, i.e., $r = E[R(t)|R(t) \leq 72]$. Crawl results show that $r = 10.5$ hours, but 5% of the peers in S_0 leave the system in just 8 minutes.

[3]The Gnutella protocol suggest that peers behind firewalls should not become an ultrapeer. But in our measurement, about 5% of users behind firewalls act as an ultrapeer.



(a) responsive ultrapeer (b) all links

Fig. 10. Inverse-averaged estimator $E_R(x)$ for responsive peers and links in Gnutella. Both cases use 3-point derivatives.



(a) ultra-leaf links (b) ultra-ultra links

Fig. 11. Inverse-averaged estimator $E_R(x)$ for different types of links in Gnutella. Both cases use 3-point derivatives.

We next proceed to compare the associated link lifetime distribution with that of peers in terms of α and r.

C. Link Lifetimes

It is straightforward to apply the residual-based algorithm to measure the link lifetime distribution in Gnutella networks. In the experiment of section VI-B, GnuCrawler kept track of the links of responsive ultrapeers found in S_0 and updated their status (i.e., connected or broken) in subsequent crawls. Using this information, we applied the same processing program to extract link residuals from GnuSpider logs and perform the proposed recovery technique to obtain $E_R(x)$. Fig. 10(b) shows that the resulting distribution of all link lifetimes is also power-law, but this time with $\alpha = 1.92$, which is much larger

than that in the peer lifetime distribution. This observation establishes that *the lifetime of a link is probabilistically smaller than that of a peer and one may expect more frequent changes in neighboring relationships*. We also find that r is 3.8 hours and 16.4% of links disappear within 8 minutes.

Next, we treat the links between ultrapeers and leaves separately from those among ultrapeers and plot the corresponding distributions in Fig. 11. Interestingly, the figure shows that the ultra-leaf links are slightly more stable (i.e., exhibit a heavier tail) than ultra-ultra links: the former has $\alpha = 1.88$ and the latter has $\alpha = 1.99$; the conditional expected lifetimes r of the two types of links are 3.9 and 3.5 hours, respectively. This can be plausibly explained by the fact that a leaf is usually inactive in collecting information about alternate ultrapeers and is thus less likely to switch its attachment point.

D. Discussion

With the experimental results of this section, we are now able to study resilience properties of Gnutella networks by applying models from [11], which use the average residual link lifetime and average node degree d as input parameters. Given $d = 28.5$ neighbors observed in our experiments and a 1-minute failed-neighbor replacement delay, we obtain that the probability for the network to disconnect at the ultrapeer level is below 10^{-64}. However, leaves may be isolated with a non-negligible probability, because they only have one or two attachment points, i.e., $d \leq 2$, which we plan to explicitly verify in future work.

VII. Related Work

Some of the first P2P sampling studies date to 2001 [15], [17] and the first use of CBM can be traced to Saroiu *et al.* [17] who sampled $17,000$ Gnutella peers every 7 minutes using TCP SYN packets over a period of 60 hours. In a follow-up effort in [4], Chu *et al.* used a similar method, but probed $5,000$ peers every 7 minutes for 10 weeks. Bhagwan *et al.* [1] improved over [4], [17] by implementing the Overnet protocol and probing a randomly chosen subset of peers in the system to measure their availability (i.e., the portion of time they were present online). Their experiment selected $2,400$ out of around $90,000$ peers and kept probing them every 20 minutes for 7 days. Liang *et al.* [13] measured lifetime distributions of links in the KaZaa network, but these experiments were limited to the connections passing through the authors' monitoring hosts.

More related work can be found in [2] and [19]. Bustamante *et al.* [2] implemented a Gnutella sampler using 17 monitoring clients that periodically probed $500,000$ peers in the network every 21 minutes for 7 days. In more recent work, Stutzbach *et al.* [19] developed a much faster crawler that in 2004 was able to cover the entire Gnutella network of $158,000$ ultrapeers within 7 minutes. The closest approach to understanding sampling bias is another recent paper by Stutzbach *et al.* [20], which focused on capturing unbiased snapshots of joint properties of users currently alive in P2P systems using random walks.

VIII. Conclusion

In this paper, we showed that direct lifetime sampling suffered from estimation bias and did not admit any fundamental improvement besides reducing probing interval Δ. To overcome this limitation, we proposed and analyzed a novel residual-based lifetime sampling algorithm, which measured lifetime distributions with high accuracy and required several orders of magnitude less bandwidth than the prior approaches. Using this method, we sampled Gnutella users and discovered that lifetimes of peers and links exhibited power-law distributions, but with different shape parameters, where links are indeed much more volatile than actual peers.

References

[1] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability," in *Proc. IPTPS*, Feb. 2003, pp. 256–267.

[2] F. E. Bustamante and Y. Qiao, "Friendships that Last: Peer Lifespan and its Role in P2P Protocols," in *Proc. Intl. Workshop on Web Content Caching and Distribution*, Sep. 2003.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 407–418.

[4] J. Chu, K. Labonte, and B. N. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," in *Proc. ITCom Conference*, vol. 4868, Jul. 2002, pp. 310–321.

[5] R. J. Dunn, J. Zahorjan, S. D. Gribble, and H. M. Levy, "Presence-Based Availability and P2P Systems," in *Proc. IEEE International Conference on Peer-to-Peer Computing*, Aug. 2005, pp. 209–216.

[6] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2003, pp. 2188–2198.

[7] Gnutella. [Online]. Available: http://www.gnutella.com/.

[8] 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. ACM SIGCOMM*, Aug. 2003, pp. 381–394.

[9] KaZaA. [Online]. Available: http://www.kazaa.com/.

[10] S. Krishnamurthy, S. El-Ansarh, E. Aurell, and S. Haridi, "A Statistical Theory of Chord under Churn," in *Proc. IPTPS*, Feb. 2005, pp. 93–103.

[11] D. Leonard, V. Rai, and D. Loguinov, "On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks," in *Proc. ACM SIGMETRICS*, Jun. 2005, pp. 26–37.

[12] D. Leonard, Z. Yao, X. Wang, and D. Loguinov, "On Static and Dynamic Partitioning Behavior of Large-Scale Networks," in *Proc. IEEE ICNP*, Nov. 2005, pp. 345–357.

[13] J. Liang, R. Kumar, and K. W. Ross, "The FastTrack Overlay: A Measurement Study," *Computer Networks*, vol. 50, no. 6, pp. 842–858, Apr. 2006.

[14] S. Resnick, *Adventures in Stochastic Processes*. Birkhäuser, 2002.

[15] 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 Comput. J.*, vol. 6, no. 1, pp. 50–57, Jan.-Feb. 2002.

[16] D. Roselli, J. R. Lorch, and T. E. Anderson, "A Comparison of File System Workloads," in *Proc. USENIX Annual Technical Conference*, Jun. 2000, pp. 41–54.

[17] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. SPIE/ACM Multimedia Computing and Networking*, vol. 4673, Jan. 2002, pp. 156–170.

[18] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 219–232, Apr. 2004.

[19] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," in *Proc. ACM IMC*, Oct. 2006, pp. 189–202.

[20] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On Unbiased Sampling for Unstructured Peer-to-Peer Networks," in *Proc. ACM IMC*, Apr. 2006, pp. 27–40.

[21] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems," in *Proc. ACM IMC*, Oct. 2005, pp. 49–62.

[22] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks," in *Proc. IEEE ICNP*, Nov. 2006, pp. 32–41.