ROBUST AND SCALABLE SAMPLING ALGORITHMS FOR NETWORK MEASUREMENT

A Dissertation

by

XIAOMING WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2009

Major Subject: Computer Science

ROBUST AND SCALABLE SAMPLING ALGORITHMS FOR NETWORK MEASUREMENT

A Dissertation

by

XIAOMING WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Dmitri Loguinov
Committee Members,	Riccardo Bettati
	Jennifer L. Welch
	Narasimha Annapareddy
Head of Department,	Valerie E. Taylor

August 2009

Major Subject: Computer Science

ABSTRACT

Robust and Scalable Sampling Algorithms for Network Measurement. (August 2009)

Xiaoming Wang, B.S., Beijing University of Posts and Telecommunications;

M.S., Beijing University of Posts and Telecommunications

Chair of Advisory Committee: Dmitri Loguinov

Recent growth of the Internet in both scale and complexity has imposed a number of difficult challenges on existing measurement techniques and approaches, which are essential for both network management and many ongoing research projects. For any measurement algorithm, achieving both accuracy and scalability is very challenging given hard resource constraints (e.g., bandwidth, delay, physical memory, and CPU speed). My thesis research tackles this problem by first proposing a novel mechanism called *residual sampling*, which intentionally introduces a predetermined amount of *bias* into the measurement process. We show that such biased sampling can be extremely scalable; moreover, we develop residual estimation algorithms that can *unbiasedly* recover the original information from the sampled data. Utilizing these results, we further develop two versions of the residual sampling mechanism: a continuous version for characterizing the user lifetime distribution in large-scale peer-to-peer networks and a discrete version for monitoring flow statistics (including per-flow counts and the flow size distribution) in high-speed Internet routers. For the former application in P2P networks, this work presents two methods: *ResIDual-based Estimator* (RIDE), which takes single-point snapshots of the system and assumes systems with stationary arrivals, and *Uniform RIDE* (U-RIDE), which takes multiple snapshots and adapts to systems with arbitrary (including non-stationary) arrival processes. For the latter application in traffic monitoring, we introduce *Discrete*

RIDE (D-RIDE), which allows one to sample each flow with a geometric random variable. Our numerous simulations and experiments with P2P networks and real Internet traces confirm that these algorithms are able to make accurate estimation about the monitored metrics and simultaneously meet the requirements of hard resource constraints. These results show that residual sampling indeed provides an ideal solution to balancing between accuracy and scalability.

To my family

ACKNOWLEDGMENTS

I would like to devote my greatest appreciation to my advisor Professor Dmitri Loguinov. Without his insightful advice, constant encouragement, and generous support, I could not have reached this far. During the course of my doctoral studies, it was him who introduced me to the networking area, taught me how to produce solid results, and helped me improve my research, writing, and presentation skills. I am always surprised by his superb intelligence and creativity and constantly inspired by his thoughtful comments and discussions. His persistent commitment to research, great passion for solving problems, and strong demand for excellence will have a great impact on me for the rest of my research career.

I acknowledge Dr. Riccardo Bettati, Dr. Jennifer Welch, and Dr. Narasimha Reddy for kindly serving on my committee and patiently commenting on my research. I am also indebted to Dr. Daren Cline for his valuable discussions on the problem of non-stationary systems. His insightful comments made the U-RIDE paper possible and generally deepened my understanding of the stochastic process theory in many ways.

Furthermore, I sincerely appreciate my friends and fellow students at Texas A&M University for making my life in College Station an enjoyable experience. My appreciation goes to current and former IRL colleagues for making my stay at the Internet Research Lab a wonderful memory.

Last, but not least, I would like to thank my parents and my family members for their continuous support and encouragement. I am specially grateful to my wife Meng for her endless patience and trust that accompanied me through my hard times during completing this dissertation. Without her support and love, this work would have not been possible.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

1 Overview

During recent years, the Internet has underwent a fast growing period both in scale and complexity, which imposes an urgent need to better manage and engineer both the underlying infrastructure and application-level networks. The ability to monitor and characterize large-scale, complex systems has thus become an important component for supporting and sustaining the current growing speed in the Internet. For network management practices, measuring the Internet helps in traffic engineering, accounting, anomaly detection, etc.; for research purposes, understanding the Internet provides crucial input to analytic models of many performance metrics such as throughput, resilience, response time, etc.

However, for any measurement algorithm, achieving both objectives of *accuracy* and *scalability* is very challenging given resource constraints (e.g., bandwidth, delay, physical memory, and CPU speed) that are commonly found in many large systems such as the Internet. In particular, accuracy stands for the extent of how close measured quantities are to the true values and determines usefulness of measurement results. Scalability refers to the ability to stay within the limit of resources and decides feasibility of a measurement algorithm. These two properties are especially important in measuring the Internet and have received considerable attention in a wide range of contexts from high-speed Internet routers [9], [10], [11], [12], [13], [15], [16], [24], [25],

---

The journal model is *IEEE/ACM Transactions on Networking*.

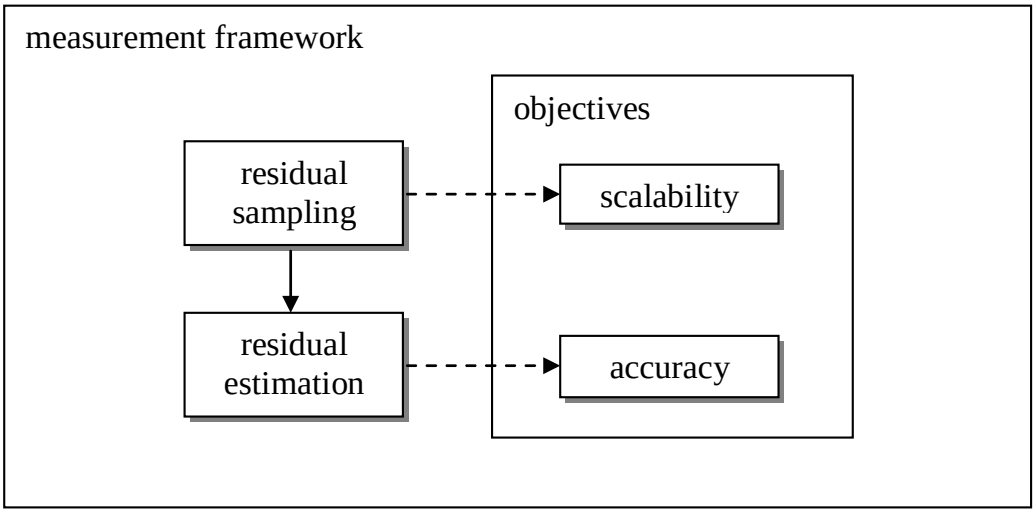


Fig. 1. Illustration of measurement framework.

[28], [30], [31], [32], [33], [38], [43], [64] to peer-to-peer networks [3], [52], [58], [56] , [62]. A tradeoff constantly made between these two objectives by previous efforts is to sacrifice accuracy for scalability and it remains an open problem to design such a measurement algorithm that can produce accurate result while satisfying all imposed constraints.

In this work, we study two cases of such measurement efforts. The first case performs *active measurement* in peer-to-peer (P2P) networks by sending probing queries into the network so as to estimate the *lifetime distribution* of participating users, where each lifetime instance represents the duration of a user's appearance in the system. The second case conducts *passive measurement* in high-speed Internet routers by monitoring the traffic as it passes by and its goal is to measure the statistics of traffic flows, where a *flow* is a sequence of packets that share a common pattern (e.g., the 5-tuple in each packet header). While these two problems are completely different, they can be addressed by the same framework that we propose in this work. We propose novel algorithms in the family of *residual sampling*, which intentionally

introduces a predetermined amount of *bias* into the measurement process, and show that such biased sampling can be extremely scalable; moreover, we develop residual estimation algorithms that can *unbiasedly* recover the original information from the sampled data. Utilizing these results, we implement an accurate and scalable measurement framework as illustrated by Figure 1, where residual sampling provides scalability and residual estimation guarantees accuracy. We then develop two versions of this framework: a continuous version for P2P measurement and a discrete version for traffic monitoring. Next, I briefly discuss these two problems and our solutions in the following two sections, respectively.

2 P2P Measurement

Consider a distributed P2P system with n potential users, all of which alternate between two states: ON (alive) and OFF (dead). Let L be the lifetime (the length of an ON session) of a random user of the system and $F_L(x)$ be the corresponding distribution function, i.e., $F_L(x) = P(L \leq x)$. The goal of this problem is to measure the lifetime distribution $F_L(x)$. The user lifetime distribution is one of the essential metrics in capturing the dynamics of P2P systems and provide useful information to throughput models [19], [46], resilience analysis [34], [65], [66], and system design [21], [35], [52]. Prior efforts [3], [52], [56] use *Create-Based Method* (CBM), which takes periodic snapshots of participating users by crawling the whole system every Δ time units and then directly identifies lifetime samples by comparing adjacent snapshots. As we show in Chapter II, CBM could 1) miss lifetime samples (a user joins and departs within the same sampling interval) and 2) produce inconsistent round-offs (some user lifetimes are rounded up and others down). We provide an analytic model for CBM and show that it generally produces biased estimation of $F_L(x)$ and the bias

cannot be removed without letting $\Delta \to 0$. We also show that the overhead of CBM in measuring a system with n users is proportional to n/Δ. Therefore, to scale to large P2P systems like Gnutella [20] and KaZaA [27], we need to use large Δ, which unfortunately could lead to inaccurate results.

We address this problem using the result from renewal theory [49]. Instead of capturing the lifetimes of new users, we collect the residuals of existing users. In the proposed approach called *residual sampling*, we only take one snapshot of the entire network and then tracks residual (i.e., remaining) lifetimes R of the users seen in the first crawl. From renewal theory [63], residual sampling is biased since it is more likely to capture long-lived users. However, there are two facts that make residual sampling desirable in P2P measurement. First, it provides an accurate estimator of $F_L(x)$. Let $H(x) = P(R \leq x)$ be the residual distribution and $E[L]$ be the expected lifetime. Utilizing the asymptotic results from [34], [65] for systems with a stationary arrival process:

$$H(x) = \frac{1}{E[L]} \int_0^x (1 - F_L(y))dy, \tag{1}$$

we develop a simple mechanism called *ResIDual-based Estimator* (RIDE), which can accurately recover $F_L(x)$ from the measured residuals. Second, residual sampling is able to aggressively reduce traffic overhead since initial users captured by RIDE die quickly and the amount of probing traffic decays to zero accordingly. We also present a subsampling strategy, where each user found in the first crawl is uniformly selected with probability ϵ and only selected users are then probed for residuals. Our analysis and experiments show that this subsampling technique does not change the correctness of RIDE and more importantly, it reduces traffic overhead compared to that in CBM by a factor of up to $1/\epsilon$.

While RIDE can achieve a decent tradeoff between accuracy and bandwidth

consumption, its application is limited to systems with constant arrival rates since (1) assumes a stationary arrival process, which we call *Stationary Renewal Churn Model* (SR-CM). However, many systems have been discovered to exhibit diurnal behavior [22], [52], [55], [60], which motivates us to develop a more general sampling algorithm. As the first step to achieve this goal, we design a novel generic arrival model called *Non-Stationary Periodic Churn Model* (NS-PCM) that can replicate first-order dynamics (i.e., mean arrival rate) of almost any non-stationary arrival process. In NS-PCM, each user again alternates between ON (alive) and OFF (dead) states, but OFF states are now split into two sub-states: REST and WAIT. The former sub-state represents the delay between the user's departure and midnight of the day when it joins the system again, while the latter is the delay from midnight until the user's arrival into the system within a given day. Let A be the length of a random WAIT period and $f_A(x)$ its distribution density. We prove that NS-PCM can mimic any system with a continuous non-stationary periodic arrival rate by adjusting the arrival density $f_A(x)$.

Utilizing this result, we then show that RIDE's estimator under non-stationary arrivals does not converge and sometimes produces completely invalid results (including CDF functions that are non-monotonic). Define $R(t)$ to be the remaining session duration of a random online user at time t and $H(x,t) = P(R(t) \leq x)$ to be the CDF of residuals of currently alive users. Our analysis shows that unlike in prior models where $\lim_{t\to\infty} H(x,t) = H(x)$ existed, NS-PCM does not admit a limiting distribution of $R(t)$, which explains why RIDE's manipulation of the residual distribution $H(x,t)$ produces unpredictable results. To address the problem in RIDE, we suggest a mechanism called *Uniform RIDE* (U-RIDE), which takes multiple snapshots and measures the system in uniformly random points in the observation window. To decide random time instances for residual sampling, we study a scheduling algorithm called *Bernoulli*

Scheduling ($\mathcal{BS}$), which leverages the BASTA principle (Bernoulli Arrival See Time Average) [40] and allows accurate measurement even when the network is small or the arrival process is unknown. We show that U-RIDE can be efficiently implemented in large systems and that it admits a subsampling technique similar to that in RIDE. Our simulation results and experiments with Gnutella networks demonstrate that U-RIDE is able to accurately estimate the actual distribution $F_L(x)$ in various non-stationary systems and at the same time significantly reduce traffic overhead compared to CBM.

3 Traffic Monitoring

The goal of flow measurement is to monitor flow statistics of the packet stream that passes through an Internet router. The flow statistics of interests typically include *single-flow usage* — the number of packets that has been observed from a single flow and *flow size distribution* — the probability of a flow with size i. Accurately monitoring these flow statistics is critical for network management and operation and has received considerable attentions [9], [10], [11], [12], [13], [15], [16], [24], [25], [28], [33], [30], [31], [32], [38], [43], [64]. This can be accomplished by keeping a counter in a table for each flow that passes the router. However, for high-speed Internet links, this approach could lead to a huge flow table since a large number of distinct flows could traverse the router per second. Such a big table not only consumes a significant amount of memory, which makes algorithms using fast SRAM (Static Random Access Memory) prohibitively expensive, but also drastically slows down per-packet processing, which obviates any possibility of using slow DRAM (Dynamic Random Access Memory). Previous efforts in flow measurement suggest to use traffic sampling to reduce memory consumption and per-packet processing time. While these sampling methods could make traffic monitoring affordable for high-speed links, either

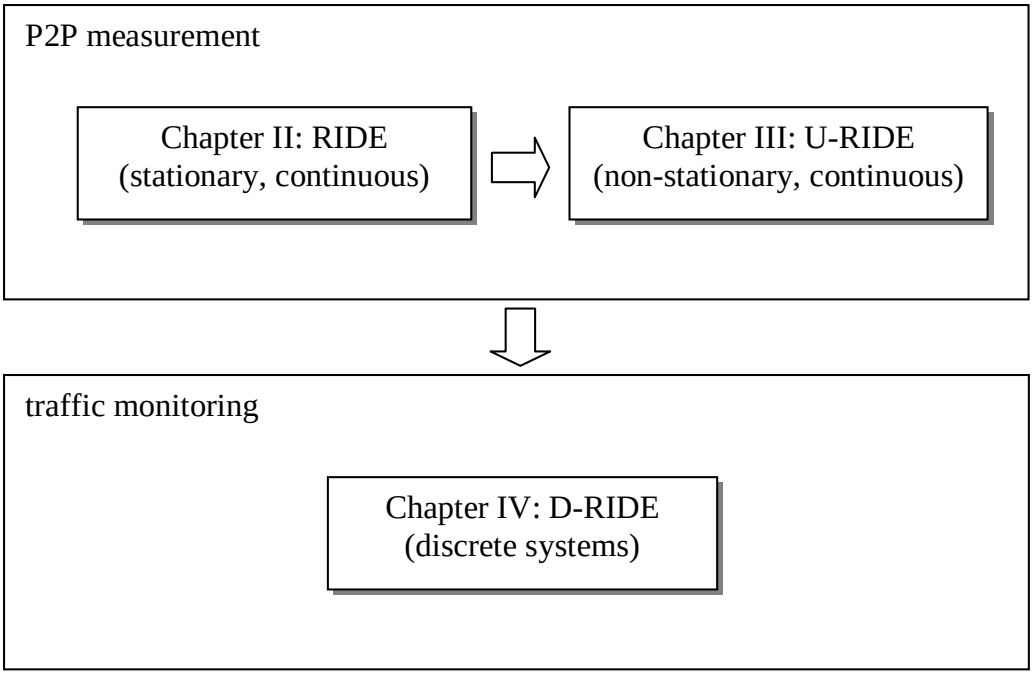


Fig. 2. Dissertation organization.

they make estimation inaccurate and too complicated as shown in [11], [24]; or they do not capture all *elephant flows* and thus cannot support usage-based applications as suggested by [10], [16], [17], [18], [44].

In this work, we analyze a discrete version of *residual-geometric sampling,* which has two appealing properties for traffic monitoring: 1) it can keep all large flows with high probability and provide all necessary information for generating accurate estimation about flow statistics; 2) it has the potential to scale to high-speed links. In the proposed method, each incoming packet is processed as follows: if the packet is from a new flow z, a counter is then created for z with probability p; otherwise, the corresponding counter is then incremented. Denote by R_L the number of remaining packets of a flow with size L. We first show that previous estimator $\hat{L}(R_L) = R_L - 1 + 1/p$ of original size L suggested by [16], [28] can be arbitrarily biased in estimating both the original size L and the flow size distribution $f_i = P(L = i)$. We then develop

two estimation algorithms, which we call *Discrete ResIDual-based Estimators* (D-RIDE), $\tilde{L}(R_L) = R_L - 1 + 1/p - (1-p)^{R_L}/p$ for single-flow usage and $\tilde{f}_i = \frac{M_i - (1-p)M_{i+1}}{Mp + (1-p)M_1}$ for flow size distribution, where M is the number of sampled flows and M_i the number of those with residual size i. Our analysis and experiments with Internet traces indicate that D-RIDE produces accurate estimation about single-flow usage and flow size distribution. Moreover, we propose a technique that periodically removes inactive flows from the flow table and show that residual sampling with periodic removal can be very scalable in terms of memory consumption and per-packet processing time.

The organization of this dissertation is illustrated in Figure 2. We start with P2P measurement and introduce RIDE in Chapter II and U-RIDE in Chapter III. We then discuss the problems in traffic monitoring and present D-RIDE in Chapter IV. Finally, we summarize this dissertation and discuss future work in Chapter V.

CHAPTER II

RESIDUAL-BASED ESTIMATOR (RIDE)

1 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., [5], [19], [23], [34]), 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], [3], [6], [14], [52], [56], topological information (e.g., degree distribution and clustering coefficients) [36], [58], and traffic flow rate [23], [53].

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 [57], 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 chapter, 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 [20], KaZaA [27]). 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 [34], [65] and routing ability [29] 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 lifetime sampling. We first explain what constitutes inaccuracy in measuring the target distribution of lifetimes $F_L(x)$ and define sampling methods to be *biased* if, given an infinite population of sampled users, they cannot reproduce $F_L(x)$ 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) [51], which is a widely used heuristic for sampling lifetimes in computer systems. We show that both CBM and its modification in [3], [52], [56] are generally 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

[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.

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 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 [34], [65] that accurately reconstructs the lifetime distribution $F_L(x)$ from the sampled residuals with a negligible amount of error.

The next issue we address is bandwidth consumption of lifetime sampling. 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.[2] 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

[2]Note that besides lifetimes CBM can measure additional metrics (e.g., arrival/departure process of users) and its overhead might be justified when these metrics are important.

factor of 16 for $\epsilon = 0.1$ and a factor of 125 for $\epsilon = 0.01$.

We finish this chapter by implementing a Gnutella crawler that is about 18 times faster than the fastest prior crawler [56], 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 [3]. At the same time, Gnutella links are much more volatile and can be described by a Pareto distribution with shape $\alpha \approx 1.8$. These results, fed into the latest resilience models for unstructured systems [34], [65], suggest that node isolation among joining ultrapeers in Gnutella and thus partitioning of the network must indeed be extremely rare events.

The remainder of this chapter is organized as follows. In Section 2, we formalize sampling and bias. In Section 3, 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 4. We analyze the subsampling technique in Section 5, examine the bandwidth overhead of the various methods in Section 6 and present our measurement study of Gnutella in Section 7. Section 8 reviews prior work and Section 9 concludes this chapter.

2 Formalizing Lifetime Sampling

2.1 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 joining users is drawn from some distribution $F_L(x)$. This is similar to the heterogeneous churn model proposed in [65]. Then, the goal of the sampling process is to estimate with as much accuracy as possible function $F_L(x)$, which we assume is continuous *almost*

everywhere[3] in the interval $(0, \infty)$. As shown in [65], distribution $F_L(x)$ 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 $F_L(x)$ 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 $F_L(x)$ 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 $F_L(x)$. 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

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

if:

$$\sum_{j=1}^{\infty} P(X^{\Delta} = j\Delta) = 1, \tag{2}$$

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 $P(X^{\Delta} \leq 0) = F_L(0) = 0$ 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 *in all discrete points* x_j *in the interval* $[\Delta, T]$ *for any* $\Delta > 0$*:*

$$E(x_j) = P(X \leq x_j) \tag{3}$$

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. Properly selected indirect sampling may be more accurate and/or may require lower overhead than direct sampling. We demonstrate one such example later in this chapter.

3 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.

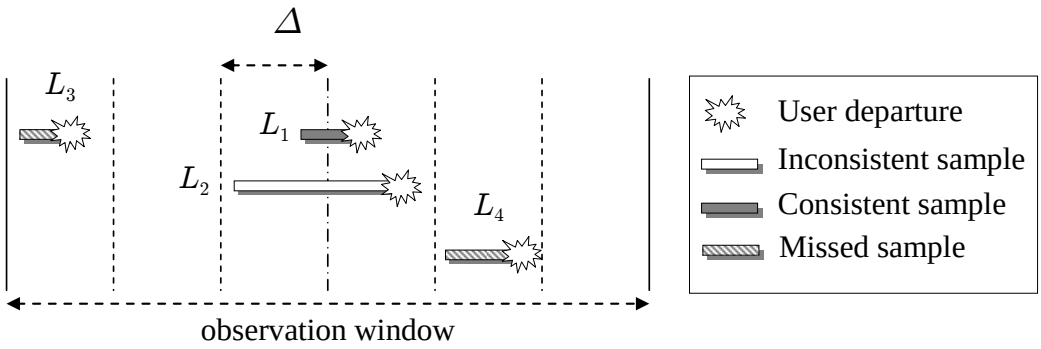


Fig. 3. Round-off inconsistencies in direct sampling.

3.1 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:

$$P(M^{\Delta} \leq x) = P(L \leq x | L \leq T, \text{not missed}), \tag{4}$$

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 Figure 3, 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.

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:*

$$\rho_j = \frac{1}{\Delta} \int_{x_j}^{x_{j+1}} F_L(x)dx - F_L(x_j), \tag{5}$$

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

Proof. Without loss of generality, shift the time axis such that a given user arrives at time $0 \leq t < \Delta$ into the system and its lifetime is $x_j < L \leq x_{j+1}$. Then, the user is sampled inconsistently with probability:

$$g(t) = P(x_j < L \leq x_{j+1} - t) = F_L(x_{j+1} - t) - F_L(x_j). \tag{6}$$

Since arrival point t can be any uniformly random point in $[0, \Delta]$, we have:

$$\rho_j = \int_0^{\Delta} g(t) \frac{1}{\Delta} dt, \tag{7}$$

which leads to the desired result. □

Equipped with (5), 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:*

$$E(x_j) = P(M^{\Delta} \leq x_j) P(L \leq T | Q_0 = 0)(1 - \rho_0) + \rho_0 - \rho_j, \tag{8}$$

where ρ_j is given in (5).

Proof. Note that for a measured lifetime sample $M^{\Delta} \leq x_j$, either the actual lifetime $L \leq x_j$ holds, or $x_j < L < x_{j+1}$ but is inconsistently rounded down to x_j, i.e., $Q_j = 1$. Therefore, (4) becomes:

$$P(M^{\Delta} \leq x_j) = P(L \leq x_j \cup Q_j = 1 | L \leq T, Q_0 = 0). \tag{9}$$

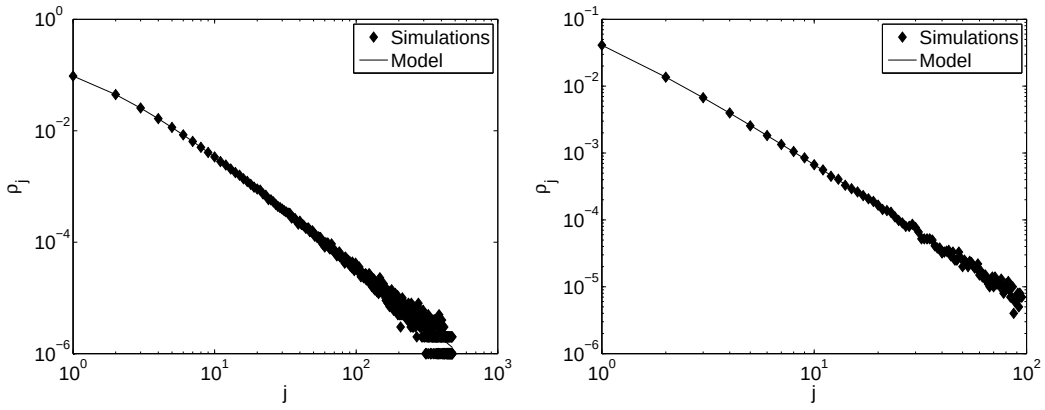
Denoting by B the event $(L \leq x_j \cup Q_j = 1)$ and expanding the conditional probability on the right side of (9), we get:

$$P(M^{\Delta} \leq x_j) = \frac{P(B, L \leq T, Q_0 = 0)}{P(L \leq T, Q_0 = 0)} = \frac{P(B, Q_0 = 0)}{P(L \leq T, Q_0 = 0)}, \tag{10}$$

where the second equality is from the fact that B implies $L \leq T$. Next, we derive the numerator and denominator of the right-hand side of (10) separately.

Rewrite the numerator of (10) as follows:

$$P(B, Q_0 = 0) = P(B) - P(B, Q_0 = 1), \tag{11}$$



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

Fig. 4. Comparison of ρ_j computed from (5) to simulations.

where $P(B)$ can be expanded by:

$$P(B) = P(L \leq x_j) + P(Q_j = 1) = P(L \leq x_j) + \rho_j. \tag{12}$$

Since $Q_0 = 1$ implies $L \leq x_j$, we have:

$$P(B, Q_0 = 1) = P(Q_0 = 1) = \rho_0. \tag{13}$$

Substituting (12)-(13) into (11), we obtain:

$$P(B, Q_0 = 0) = P(L \leq x_j) + \rho_j - \rho_0. \tag{14}$$

Expanding the denominator of (10) establishes that:

$$\begin{aligned} P(L \leq T, Q_0 = 0) &= P(L \leq T | Q_0 = 0) P(Q_0 = 0) \\ &= P(L \leq T | Q_0 = 0)(1 - \rho_0). \end{aligned} \tag{15}$$

Substituting (14)-(15) into (10) gives:

$$P(M^{\Delta} \leq x_j) = \frac{P(L \leq x_j) + \rho_j - \rho_0}{P(L \leq T|Q_0 = 0)(1 - \rho_0)}, \tag{16}$$

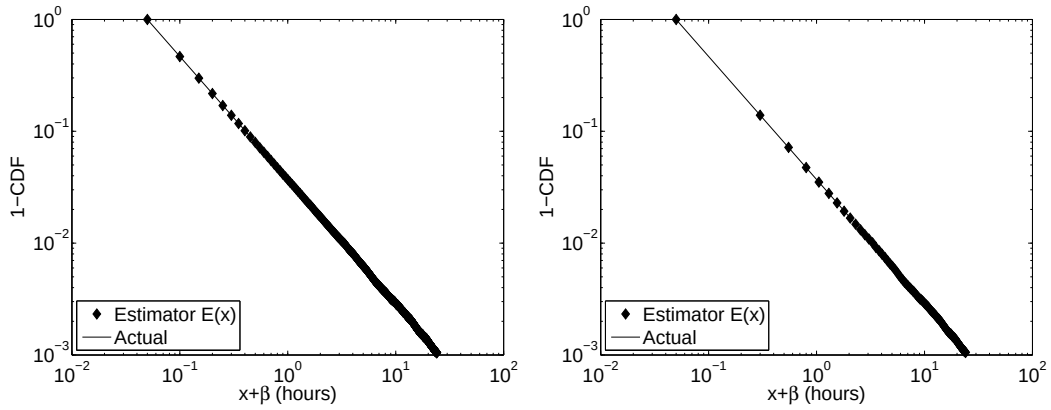
Considering that an unbiased estimator $E(x_j)$ should equal $P(L \leq x_j)$, (8) follows from (16). □

We next verify (5) and (8) in simulations using the example of a Pareto distribution commonly used to model user lifetimes [34], [65]:

$$F_L(x) = 1 - (1 + x/\beta)^{-\alpha}, \alpha > 1, x \geq 0, \tag{17}$$

with $E[L] = \beta/(\alpha - 1)$. We use $\alpha = 1.1$, $\beta = 0.05$, $T = 24$ hours, and $E[L] = 0.5$ hours. We count the number of inconsistent round-offs for each $j = 0, 1, ..., T/\Delta$ and plot the corresponding empirical probability ρ_j in Figure 4, which shows that (5) predicts reality very well. Furthermore, we compute the empirical values of $P(M^{\Delta} \leq x_j)$ and supplement the measured data with the knowledge of ρ_j to obtain $E(x_j)$ according to (8). Figure 5 plots the values of $E(x_j)$ obtained both from model (8) and the actual distribution (17), which indicates a perfect match.

From the result of Theorem 2, it becomes clear that unbiased measurement requires access to the distribution of observed 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}.



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

Fig. 5. Verifying estimator (8) in simulations.

3.2 Create-Based Method (CBM)

We next study how inconsistent round-offs exhibit themselves in a widely used [3], [52], [56] direct sampling algorithm called *Create-Based Method* (CBM), first introduced by [51] in the context of operating systems. Recall from [51] 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, [51] 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. Figure 6 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 or in its second half, a peer that survives beyond time $2T$, and a user whose lifetime is larger than T.

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

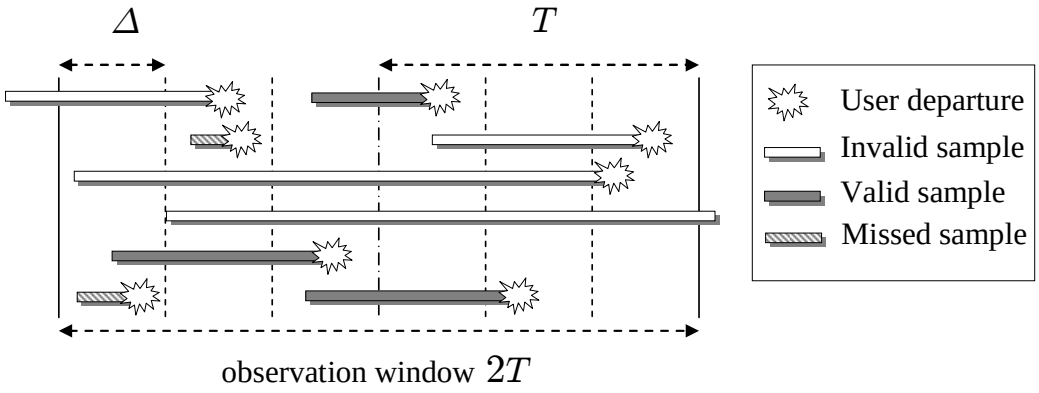


Fig. 6. Illustration of sampling in CBM.

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 $F_L(x)$ 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 [51]:

$$E_A(x_j) = \lim_{N\to\infty} \frac{N(x_j)}{N(T)} = P(M^{\Delta} \leq x_j). \tag{18}$$

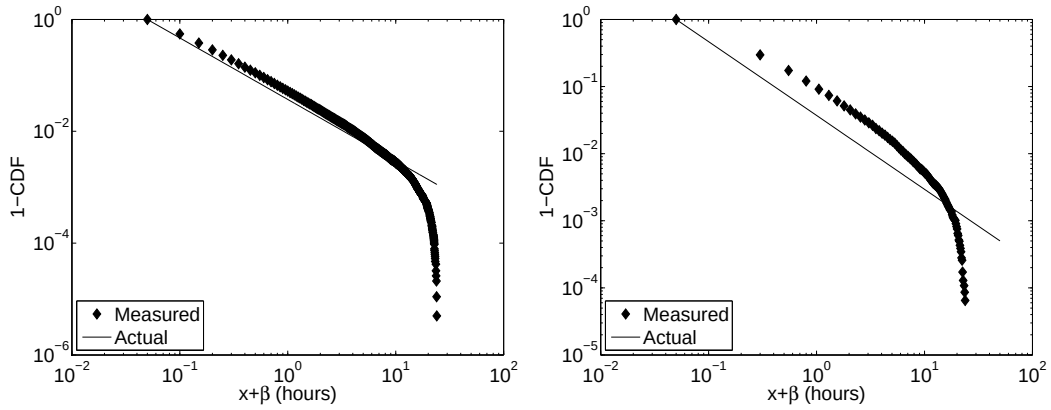
Recent work in [3], [52], [56] normalizes E_A by the percentage of samples no larger than T (i.e., $N(T)/N$) and defines the following modified estimator:

$$E_B(x_j) = \lim_{N\to\infty} \frac{N(x_j)}{N}. \tag{19}$$

With the result in (8), we can express both CBM estimators as functions of the actual distribution $F_L(x_j) = P(L \leq x_j)$.

Theorem 3. *Both CBM estimators* (18)-(19) *are generally biased when any* $\rho_j > 0$ *and produce the following distributions:*

$$E_A(x_j) = \frac{F_L(x_j) - \rho_0 + \rho_j}{F_L(T) - \rho_0}, E_B(x_j) = \frac{F_L(x_j) - \rho_0 + \rho_j}{1 - \rho_0}. \tag{20}$$



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

Fig. 7. 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).

Proof. Expanding $E_A(x_j)$ using (8), we have:

$$E_A(x_j) = \frac{F_L(x_j) - \rho_0 + \rho_j}{P(L \leq T|Q = 0)(1 - \rho_0)}. \tag{21}$$

We now use the following reasoning. Observe that $P(L \leq T|Q = 0)P(Q = 0) = P(L \leq T, Q = 0)$, which can be rewritten as follows:

$$\begin{aligned} P(L \leq T, Q = 0) &= P(L \leq T) - P(L \leq T, Q = 1) \\ &= P(L \leq T) - \rho_0. \end{aligned} \tag{22}$$

The second equality in (22) is from the fact that $Q = 1$ implies $L \leq T$. The result of $E_B(x_j)$ is a direct consequence of (8). □

The result in (20) shows that E_B is closer to $F_L(x)$ 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 (20) and simulations.

3.3 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 (20) that:

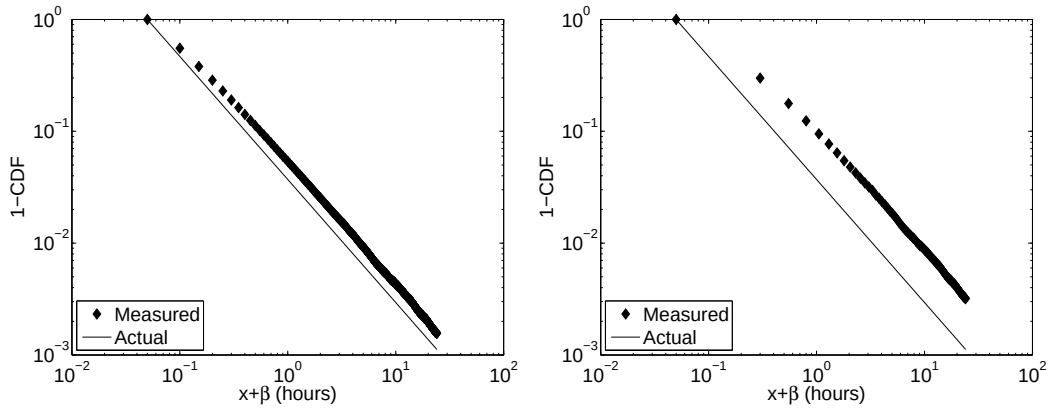
$$\bar{E}_A(x_j) = \frac{\bar{F}_L(x_j) - \bar{F}_L(T) - \rho_j}{F_L(T) - \rho_0}, \tag{23}$$

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 (23) could be illustrated through simulations. 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 [65]. Even though $F_L(T) = 99.8\%$ of users have lifetimes smaller than T, Figure 7 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 Figure 7 does not even resemble that of $\bar{F}_L(x)$, which may lead to erroneous conclusions about the family of distributions $F_L(x)$ belongs to.

We now study how ρ_j affects the shape of E_B. It follows from (20) that for $j = 0, 1, 2, ..., T/\Delta$:

$$\bar{E}_B(x_j) = \frac{\bar{F}_L(x_j) - \rho_j}{1 - \rho_0}, \tag{24}$$

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 Figure 8, 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 $F_L(x)$ does not admit shape invariance during scaling (e.g., Gaussian, uniform), estimator E_B may produce significantly misleading results.



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

Fig. 8. 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).

3.4 Limitations of CBM

From now on, we refer to E_B when mentioning CBM since E_B exhibits less bias than E_A as shown by model (20) and Figure 7-8. We next investigate whether there exists a lifetime distribution such that $\bar{E}_B(x_j) = \bar{F}_L(x_j)$. Notice from (24) that there are two ways for CBM to be unbiased, either $\rho_0 = \rho_j = 0$ or $\rho_j = \rho_0 \bar{F}_L(x_j)$. The next two theorems shows that only special lifetime distributions satisfy these conditions.

Theorem 4. *The only lifetime distribution that allows CBM to avoid all round-off errors (i.e., $\rho_j = 0$, $j = 0, 1, 2, ...T/\Delta - 1$) is a step function with support $m\Delta$ for some integer $m \geq 1$.*

Proof. It follows from (5) that for $\rho_j = 0$ ($j \geq 1$) to hold, we must have:

$$F_L(x) = \begin{cases} 0 & 0 \leq x < \Delta \\ F_L(x_j) & x_j \leq x < x_j + \Delta \end{cases}, \tag{25}$$

which specifies a step function with support $m\Delta$, where $m = 1, 2, \ldots$ □

Note that in real-life systems, the assumption that the lifetime distribution follows a step function with periodicity equal to crawl delay Δ is too restrictive. Therefore, we next seek non-step functions that satisfy the second condition of CBM being unbiased.

Theorem 5. *The only lifetime distribution that allows CBM to be unbiased simultaneously for all $\Delta > 0$ is exponential.*

Proof. We prove the theorem by first verifying that the exponential function satisfies $\bar{E}_B(x_j) = \bar{F}_L(x_j)$ and show its uniqueness. Substituting $F_L(x) = 1 - e^{-x/\mu}$ into (5), we obtain ρ_j for exponential distributions:

$$\rho_j = e^{-x_j/\mu}\Big(1 - \frac{\mu}{\Delta}(1 - e^{\Delta/\mu})\Big) = \bar{F}_L(x_j)\rho_0. \tag{26}$$

Substituting (26) into (24) establishes $\bar{E}_B(x_j) = \bar{F}_L(x_j)$.

We next show that if $\bar{E}_B(x_j) = \bar{F}_L(x_j)$, then $F_L(x)$ must be an exponential function. Assuming $\bar{E}_B(x_j) = \bar{F}_L(x_j)$, we can reduce (24) to:

$$\rho_j = \bar{F}_L(x_j)\rho_0. \tag{27}$$

Expanding ρ_j and ρ_0 in (27) using (5), we get:

$$\int_0^\Delta (F_L(x + x_j) - F_L(x_j))dx = \int_0^\Delta \bar{F}_L(x_j)F_L(x)dx. \tag{28}$$

For (28) to hold for all $\Delta > 0$, we must have for all $x > 0$:

$$F_L(x + x_j) - F_L(x_j) = \bar{F}_L(x_j)F_L(x), \tag{29}$$

which can be simplified to:

$$\bar{F}_L(x + x_j) = \bar{F}_L(x)\bar{F}_L(x_j). \tag{30}$$

Note that the only solution to $\bar{F}_L(x+y) = \bar{F}_L(x)\bar{F}_L(y)$ is given by the exponential function $\bar{F}_L(x) = e^{-x/\mu}$, which establishes the desired result. □

Given that the observed lifetimes in peer-to-peer systems [3], [56] are heavy-tailed, we next explore a different method that works without any assumptions on $F_L(x)$.

4 Indirect Sampling

In this section, we seek a solution to the problem of achieving both high accuracy and low overhead using indirect sampling. It has been suggested [1], [34], [65] 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 [49] to reconstruct $F_L(x)$ from sampled residual lifetimes.

4.1 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 $i = 1, 2, ..., n$:

$$Z_i(t) = \begin{cases} 1 & \text{user } i \text{ is alive at time } t \\ 0 & \text{otherwise} \end{cases}. \tag{31}$$

This framework is illustrated in Figure 20, where X_i and Y_i are i.i.d. 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 variables X_i are drawn from a user-specific distribution $F_i(x) = P(X_i < x)$ and denote by $l_i = E[X_i]$ the expected lifetime and

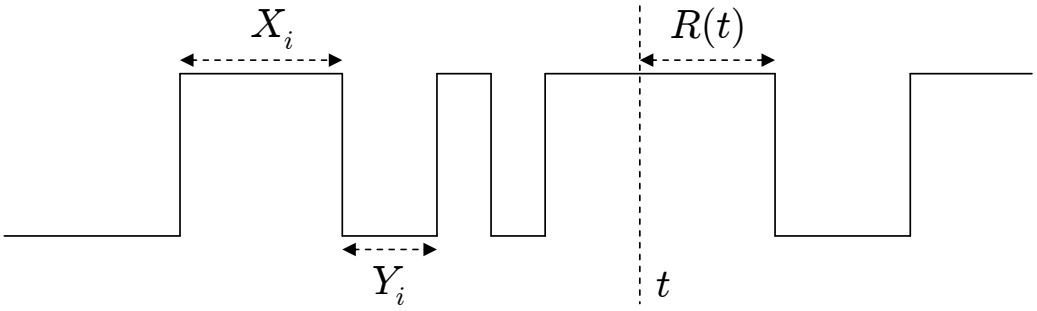


Fig. 9. Process $Z_i(t)$ depicting user i's ON/OFF behavior.

by $d_i = E[Y_i]$ the expected offtime of user i. Further define b_i to be the arrival rate of user i normalized by the total arrival rate into the system [65]:

$$b_i = \frac{1/(l_i + d_i)}{\sum_{k=1}^{n} 1/(l_k + d_k)}. \tag{32}$$

Then, it has been proven in [65] that the aggregate lifetime of all users seen by the system is drawn from a weighted distribution:

$$F_L(x) = \sum_{i=1}^{n} b_i F_i(x). \tag{33}$$

As before, L is the random lifetime of peers visiting the system and 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 $F_L(x) = P(L < x)$. In order to accomplish this task, define $R(t)$ be the residual lifetime of a random alive user at time t and denote by $H(x)$ its limiting distribution:

$$H(x) = \lim_{t \to \infty} P(R(t) \leq x). \tag{34}$$

Then, $F_L(x)$ can be inferred from $H(x)$ using the following relationship established in [65]:

$$H(x) = \frac{1}{\mu} \int_0^x (1 - F_L(u)) du, \tag{35}$$

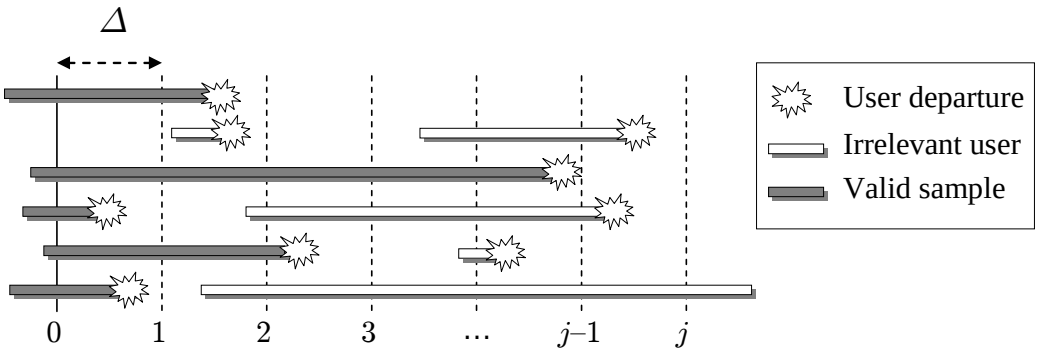


Fig. 10. Sampling residuals in RIDE.

where $\mu = E[L]$ is the expected lifetime of a joining peer and $F_L(x)$ is given by (33).

4.2 RIDE

We first define the sampling algorithm in RIDE and then discuss its estimator $E_R(x)$. At time t_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 since only users who are alive at time $t = t_0$ are valid measurements; 2) no samples can be inconsistently rounded off since all valid residual lifetimes start from the time of the first crawl. Figure 10 illustrates an example of five valid samples captured in the first crawl and five irrelevant lifetimes that are safely ignored by the algorithm.

Define $E_H(x_j)$ to be an estimator of the residual distribution $H(x)$ using users

in set S_0 and assuming the system is in equilibrium:

$$E_H(x_j) = \lim_{|S_0|\to\infty} \frac{N(x_j)}{|S_0|}, \tag{36}$$

where $N(x)$ denotes the number of users in S_0 whose lifetimes are shorter than or equal to x. Since RIDE does not miss or incorrectly round off samples, its estimation of residual lifetimes is unbiased, which leads to $E_H(x_j) = H(x_j)$. Combining with (35), we establish the next theorem.

Theorem 6. *For residual lifetime sampling, the following is an unbiased estimator of L:*

$$E_R(x_j) = 1 - \frac{h(x_j)}{h(0)}, \tag{37}$$

where $x_j = j\Delta$ and $h(x) = H'(x)$ is the PDF of $R(t)$. Furthermore, the expected user lifetime is $E[L] = 1/h(0)$.

Proof. Differentiating both sides of (35), we get $h(x) = (1 - F_L(x))/\mu$. Setting $x = 0$, it follows that $h(0) = 1/\mu$. Solving for $F_L(x)$, we get (37). □

Since $H(x)$ is computed without bias, it is now possible to numerically compute its derivative $h(x)$ using Taylor expansion with error bounded by $O(H^{(k)}(x)\Delta^k/k!)$, where $k = T/\Delta$ is the number of samples in the curve. For any analytic function $H(x)$, the convergence of the error to zero is guaranteed as k becomes large. For $\Delta = 3$ minutes and $T = 24$ hours commonly used in our experiments, the resulting error for Pareto lifetimes with $\alpha = 1.1$ and $\beta = 0.1$ is upper bounded by $H^{(480)}(0)\Delta^{480}/480! \approx 10^{-624}$, 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).

5 Subsampling

In this section, we examine under what conditions CBM and RIDE allow reduction of measurement overhead through some type of user subsampling. We then discuss the algorithm used in RIDE and show its performance.

5.1 Preliminaries

Assume that the target P2P system is fully distributed and supports the operation of building a list S of currently alive peers. We assume that it takes Δ time units to create S and that this process requires bandwidth overhead proportional to the number of alive users, i.e., $C|S|$ bytes where C is some fixed bandwidth needed to find an alive user and download its parameters (e.g., IP address and port number). Note that we do not require that the P2P system provide any other mechanisms that aid our measurement process (e.g., notifications about user arrival, departure, or dynamic link changes).

Once S is built, we assume that each alive user can be monitored using a mechanism independent of the P2P network to detect its departure (e.g., using TCP connection requests). This monitoring also incurs overhead C units per user and can be done no more often than once per Δ time units per peer in order to keep the process non-intrusive and scalable to large n. Decoupling lifetime probing from the initial collection of alive users, we allow for a range of subsampling techniques where only a subset of alive users is monitored at any given time. With such algorithms, the goal is to simultaneously reduce the total overhead in the observation window $[0, T]$ and preserve estimation accuracy.

Our earlier description of CBM required repeated crawls of the system to refresh set S, detect new arrivals, and measure their lifetimes. With certain additional re-

strictions (see below), there are two possible mechanisms for subsampling in CBM. The first method, which we call *direct*, obtains the initial snapshot S_0 of the system at time t_0 and then selects each user with an independent probability ϵ into a smaller subset S_0'. The measurement process then monitors the ON/OFF behavior of each user $v \in S_0'$ for T time units and estimates $F_L(x)$ based on the collected lifetimes (ignoring the first ON cycle of each user since it contains a residual rather than a lifetime). The second method, which we call *indirect*, monitors the *neighbors* of users in S_0' to detect new arrivals and uses their lifetimes for estimating the distribution of L.

RIDE subsampling is similar to the first CBM technique in the sense that only residuals of users in S_0' are monitored and used in estimation.

5.2 Direct CBM Sampling

We start our analysis with direct subsampling. There are several restrictions that this method imposes: 1) users between sessions must appear with the same (IP, port) combination; 2) OFF durations are small in comparison to T; and 3) the distribution of lifetimes for users in S_0' is indeed $F_L(x)$. In many current P2P networks, the user is likely to use a new IP address assigned by its DHCP/PPP server [56] or choose a new port for each join into the system [20], which makes detection of its return into the system and monitoring of its ON/OFF cycles impossible without periodic re-crawls of the system. The second assumption depends on user behavior and may significantly impact CBM subsampling if users do not return into the system within T time units, which has been observed in BitTorrent networks [56]. Even if both conditions 1) and 2) are satisfied, our next result shows that even with $\Delta = 0$ direct CBM subsampling is biased in general networks and does not converge to $F_L(x)$.

Assume that S_0 is a random variable representing the set of currently alive users,

$N(x,T)$ is the average number of lifetimes less than x collected by CBM from users in S_0 during $[0,T]$, and $N(T)$ is the average number of samples observed by CBM in the same interval. Then, define a new CBM estimator that measures the lifetime of users in S_0:

$$E_{BS}(x) = \lim_{T\to\infty} \frac{N(x,T)}{N(T)}. \tag{38}$$

Theorem 7. *For $\Delta = 0$, the distribution of user lifetimes in S_0 is:*

$$E_{BS}(x) = \sum_{i=1}^{n} \frac{l_i/(l_i+d_i)^2}{\sum_{k=1}^{n} l_k/(l_k+d_k)^2} F_i(x). \tag{39}$$

Proof. Denote by $\Omega(T)$ the set of all lifetime samples obtained from monitoring users in S_0 in the interval $[0,T]$. We use average-case analysis where each user is included into S_0 with probability $l_i/(l_i+d_i)$, i.e., with probability that it is alive at time t_0. It then follows that the expected fraction of samples from user i in $\Omega(T)$ as $T\to\infty$ is [65]:

$$p_i = \frac{l_i/(l_i+d_i)^2}{\sum_{k=1}^{n} l_k/(l_k+d_k)^2}. \tag{40}$$

Therefore, the probability that a randomly selected lifetime in an infinite set $\lim_{T\to\infty}\Omega(T)$ is from user i and less than x is $p_i P(L_i < x)$, which leads to:

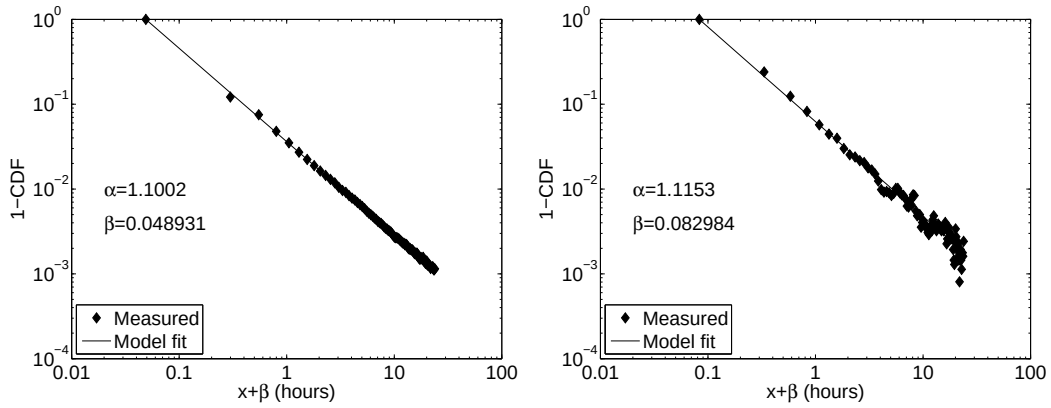
$$E_{BS}(x) = \sum_{i=1}^{n} p_i P(L_i < x), \tag{41}$$

and thus establishes (39). □

Since uniformly random subsampling of S_0 produces users with the same bias as in the original set, we immediately obtain the next result.

Corollary 1. *For sufficiently large $\epsilon|S_0|$, direct CBM subsampling estimates distribution* (39)*.*

Due to limited space, we omit simulations showing the accuracy of (39) and

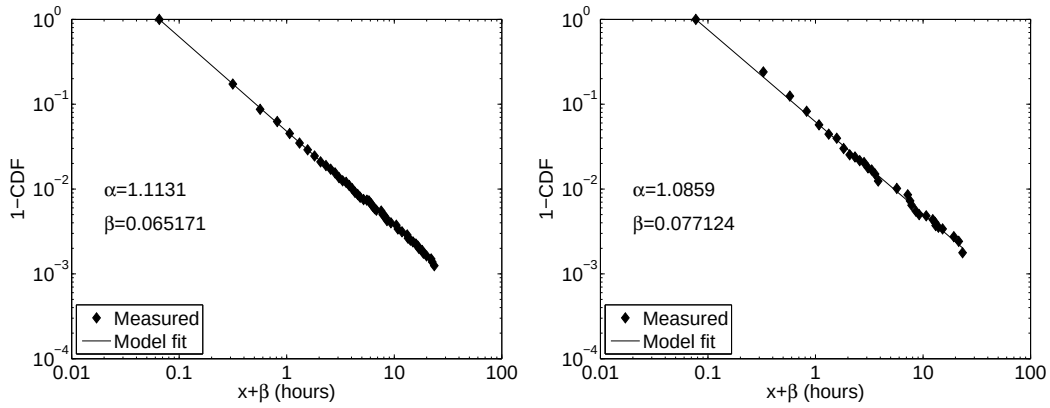


(a) $\epsilon = 1$ (b) $\epsilon = 0.01$

Fig. 11. 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.

the extent of its deviation from $F_L(x)$ in (33). Theorem 7 shows that, compared to peers arriving into the system, set S_0 is more heavily skewed towards users with significant online presence. The key difference between CBM and RIDE is that the former requires observation of *new* arrivals into the system, while the latter needs to monitor users *currently* in the system. Therefore, RIDE naturally expects bias among users in S_0, which it overcomes using a reconstruction technique in (37). CBM has no such mechanism.

Our final observation regarding direct CBM subsampling is that it can be used only in systems consisting of homogeneous peers (i.e., all users exhibit identical characteristics with $F_i(x) = F_L(x)$). However, since measurement studies [56] show that P2P users are often heterogeneous, this assumption is of limited practical use.



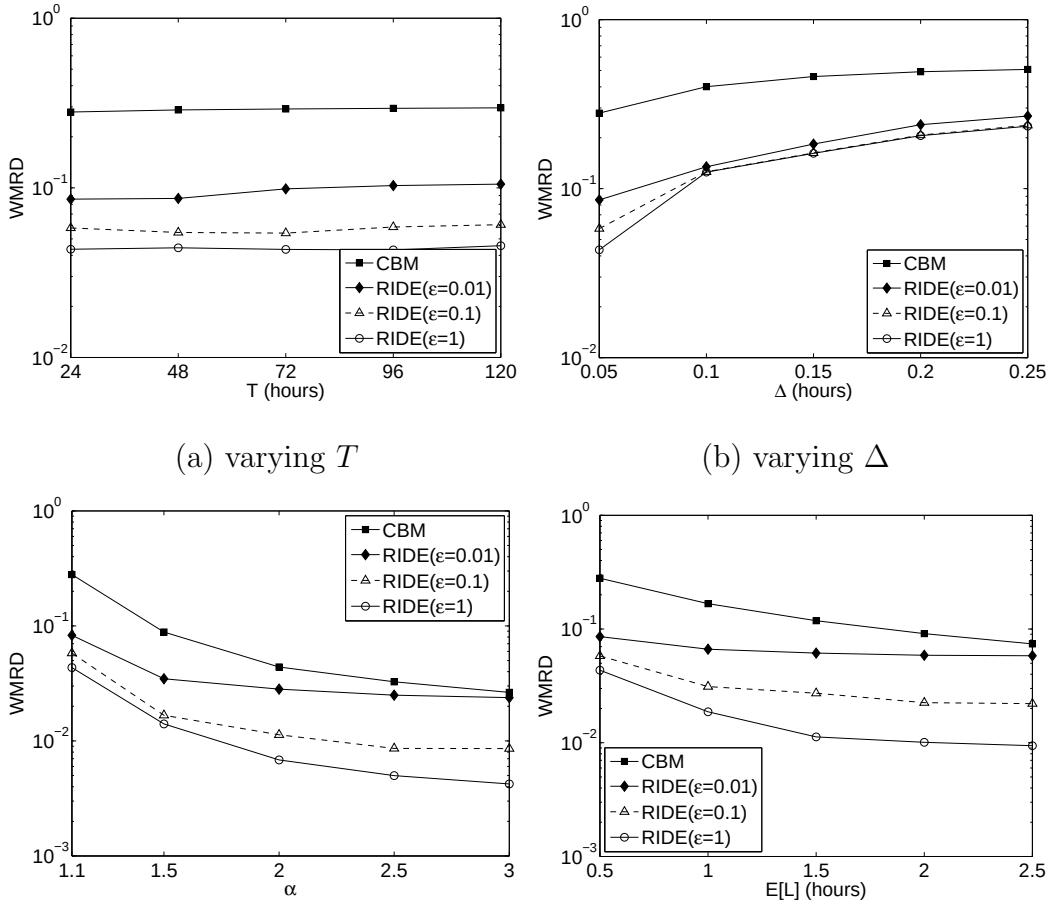
(a) $\epsilon = 0.1$ (b) $\epsilon = 0.01$

Fig. 12. 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.

5.3 Indirect CBM Sampling

Indirect sampling imposes the following restrictions: 1) neighbors of alive users can be monitored using some P2P protocol; 2) arrivals into a certain neighbor set only include new users (i.e., peers do not dynamically switch links); 3) neighbors of peers in set S_0' have unbiased lifetimes; 4) set S_0' is popular enough to attract a sufficient number of new neighbors in $[0, T]$; and 5) users in S_0' depart from the system slower than new users become their neighbors.

The first restriction is relatively easy to satisfy in certain systems (e.g., Gnutella), but may be more problematic in cases when the current population S is known to some server-like entity (e.g., BitTorrent tracker) in the form of a list, but individual peer-to-peer links are unknown. The second restriction does not hold even in Gnutella since users frequently migrate from one ultrapeer to another. These migrating users then arrive into existing neighbor sets with non-zero age, which creates bias in all



(a) varying T (b) varying Δ

(c) varying α (d) varying $E[L]$

Fig. 13. Comparing CBM and RIDE for Pareto lifetimes ($T = 24$ hrs, $\Delta = 3$ min, $\alpha = 1.1$, and $E[L] = 30$ min, 3-point derivatives).

cases except exponential L.

The third constraint depends on the neighbor selection algorithm and may introduce bias in lifetimes of observed arrivals if users make neighbor selection based on the characteristics of their future neighbors. Since set S_0 is already biased from the perspective of CBM, arriving neighbors may also exhibit bias if links are formed based on some similarity between $F_i(x)$ of new users and $F_j(x)$ of existing peers. Without a specific graph-construction algorithm, further analysis of this bias is impossible. The last two restrictions also depend on how P2P graphs are constructed and may become a problem when new users for whatever reason do not choose peers in S_0' as neighbors or when $|S_0'|$ shrinks to zero before enough new arrivals have been detected.

5.4 RIDE Subsampling

The accuracy of RIDE's subsampling algorithm can be inferred from the fact that users in S_0' have the same distribution of residuals as those in S_0.

Corollary 2. *For sufficiently large $\epsilon|S_0|$, RIDE subsampling produces an unbiased estimate of $H(x)$.*

Compared to CBM, RIDE allows subsampling under the most general conditions and requires only two main assumptions introduced in the beginning of this section (i.e., ability to construct S_0 and monitor alive users until they die). RIDE subsampling does not impose any restrictions on link structure, user migration, lifetime homogeneity, user appearance in subsequent sessions, OFF durations, or neighbor selection. As a result, for the most general case assumed throughout this chapter, CBM must perform full crawls to build its estimates of $F_L(x)$, while RIDE can monitor residuals in S_0'.

It is worthwhile mentioning that residual sampling acquires all valid samples

during the very first crawl. Therefore, given that $|S_0|$ is sufficiently large, Corollary 2 shows that 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. We leave the overhead analysis of subsampling to the next section and now show in Figure 11 one example of using this technique, where a system of 1 million users in the same setup as in Figure 8(b) is subsampled by a factor of 100. First notice in Figure 11(a) that RIDE recovers $F_L(x)$ with much higher accuracy than E_B and obtains $\alpha = 1.1002$ and $\beta = 0.049$. Second, observe in Figure 11(b) that RIDE achieves reasonable estimation accuracy ($\alpha = 1.115, \beta = 0.083$) 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. We next deal with this issue.

5.5 Inverse Averaging

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 $\hat{x}(y)$:

$$\hat{x}(y) = \frac{\sum_i x_i \mathbf{1}_{E_R(x_i)=y}}{\sum_k \mathbf{1}_{E_R(x_i)=y}}, \tag{42}$$

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 (42), 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.

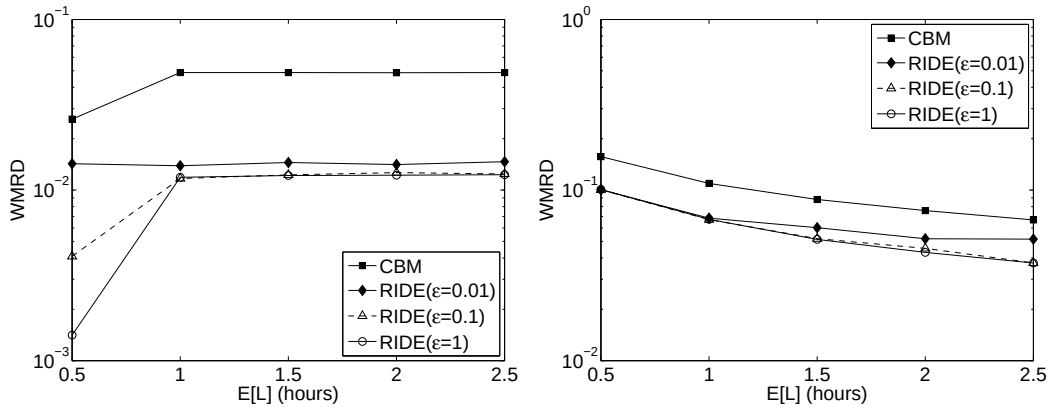
Next, we examine two cases of inverse averaging using the example in Figure 11(a) subsampled with $\epsilon = 0.1$ and $\epsilon = 0.01$. The resulting distributions are shown in Figure 12, which demonstrates much better preservation of the Pareto shape in the tail and less oscillations than without the use of inverse averaging. For $\epsilon|S_0| = 10^5$ in Figure 12(a), curve fitting produces $\alpha = 1.11, \beta = 0.065$, and for $\epsilon|S_0| = 10^4$ in Figure 12(b), we obtain $\alpha = 1.09, \beta = 0.077$. 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.

5.6 Evaluation

In this section, we provide comparison results between subsamplied RIDE and original CBM that cover various parameter settings and lifetime distributions. We start with defining a statistical metric that characterizes the extent of difference between two distribution functions. The metric that we use in this section is the Weighted Mean Relative Difference (WMRD), which is often used for comparing heavy-tailed distributions [11]. Denote by $F_L(x)$ the actual CDF function and by $E(x)$ its estimator. Define δ to be the WMRD distance between $E(x)$ and $F_L(x)$, which is computed as:

$$\delta = \frac{\sum_j |E(x_j) - F_L(x_j)|}{\sum_j (E(x_j) + F_L(x_j))/2}, \tag{43}$$

where $x_j = j\Delta$. Note that the asymptotic value of δ in CBM (i.e., for $N \to \infty$) can be computed from (20) given any lifetime distribution and sampling parameters. It can also be inferred from (20) that CBM's limiting δ is generally non-zero for non-lattice, non-exponential lifetimes. On the other hand, with $\epsilon|S_0| \to \infty$, RIDE's δ converges to zero.



(a) Uniform varying μ (b) Weibull - varying μ

Fig. 14. Comparing CBM and RIDE for uniform and Weibull lifetimes (3-point derivatives).

Setting asymptotics aside, we next examine CBM and RIDE using δ in simulations with finite systems. Figure 13 plots the values of δ obtained from simulations with Pareto lifetimes. In the figure, we vary each of the four parameters while fixing the others according to the basic setting of $T = 24$ hrs, $\Delta = 3$ min, $\alpha = 1.1$, and $E[L] = 30$ min. For each scenario, we simulate a measurement process that captures 10^7 samples, estimates the lifetime distribution using CBM and RIDE, and computes the resulting difference δ between the estimated and actual distributions. The figure shows that RIDE even using 1% sub-sampling ($\epsilon = 0.01$) exhibits smaller errors than the original CBM in all parameter configurations. Figure 14 also compares CBM and RIDE with lifetimes drawn from the uniform distribution in $(\mu - 0.5, \mu + 0.5)$ and the Weibull distribution with shape $\alpha = 0.6$ and scale $\beta = 0.66\mu$, where μ is the expected lifetime. The figure indicates that RIDE achieves similar and even better performance than CBM in all studied cases.

6 Overhead

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

6.1 Models

For general P2P networks where CBM subsampling is impossible, we assume that the algorithm performs full crawls of the system every Δ time units in the interval $[t_0, t_0 + T]$ and then keeps probing peers that were present in the system at time $t = t_0 + T$ until they die or their observed lifetime exceeds T time units. Based on these rules, we formulate in the next theorem the overhead for CBM.

Theorem 8. *Total bandwidth overhead of (Δ, T)-sampling using CBM is given by:*

$$b_{CBM} = \frac{C|S_0|}{\Delta}\left(T + \int_0^T [H(T) - H(x)]dx\right), \tag{44}$$

where $|S_0|$ is the number of alive users in the system, C is the cost of probing or crawling a user, and $H(x)$ is the CDF of residual lifetimes.

Proof. Denote by b_1, b_2 the overhead of CBM in intervals $[t_0, t_0+T]$ and $[t_0+T, t_0+2T]$, respectively. For all T/Δ points in the first interval, each snapshot will capture $|S_0|$ users and thus the overhead is simply $b_1 = C|S_0|T/\Delta$.

Now, we examine the overhead in the second interval. Consider the snapshot that captures $|S_0|$ users at time $t_0 + T$, i.e., the last snapshot in the first interval. Denote by Z the remaining sampling duration of a random user alive at t_0+T. Then, the expected overhead b_2 incurred by probing in $[t_0 + T, t_0 + 2T]$ is given by:

$$b_2 = \frac{C|S_0|E[Z]}{\Delta}. \tag{45}$$

Next, we focus on deriving $E[Z]$. Denote by A the age of a user at time $t_0 + T$.

From time $t_0 + T$, we only need to track peers whose age A is less than T, since $E[Z|A \geq T] = 0$. Thus, $E[Z]$ can be rewritten as follows:

$$E[Z] = \int_0^\infty E[Z|A = x] f_A(x) dx = \int_0^T E[Z|A = x] f_A(x) dx, \tag{46}$$

where $E[Z|A = x]$ is the expected sampling duration of a user given that its age is $x < T$. Denote by $R(x)$ the residual lifetime of a randomly selected user given that its age $A = x$. Note that from time $t_0 + T$, we keep probing a user with age $x < T$ until either it dies, i.e., for $R(x)$ time units, or the remaining observation duration $T - x$ expires, whichever happens first. Thus, it follows that:

$$\begin{aligned} E[Z|A = x] &= E[\min(R(x), T - x)] \\ &= \int_0^\infty P(\min(R(x), T - x) > y) dy \\ &= \int_0^{T-x} P(R(x) > y) dy. \end{aligned} \tag{47}$$

Note that $P(R(x) > y)$ can be expressed in terms of $F_L(x)$:

$$P(R(x) > y) = \frac{P(L > x + y)}{P(L > x)} = \frac{1 - F_L(x + y)}{1 - F_L(x)},$$

which together with (46), (47) gives:

$$\begin{aligned} E[Z] &= \int_0^T \int_0^{T-x} \frac{1 - F_L(x + y)}{1 - F_L(x)} dy f_A(x) dx \\ &= E[L] \int_0^T \frac{H(T) - H(x)}{1 - F_L(x)} f_A(x) dx. \end{aligned} \tag{48}$$

The second equality in (48) comes from the fact that $H(x) = 1/E[L] \int_0^x (1 - F_L(u)) du$. Substituting $f_A(x) = (1 - F_L(x))/E[L]$ into (48) establishes:

$$E[Z] = \int_0^T [H(T) - H(x)] dx,$$

from which we obtain b_2. Summing up b_1, b_2 gives (44). □

Next, we examine the overhead of RIDE sampling. 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 9. *Total bandwidth overhead of (Δ, T)-sampling using RIDE is given by:*

$$b_{RIDE} = \frac{C|S_0|}{\Delta}\left(\Delta + \epsilon \int_0^T [1 - H(x)]dx\right), \tag{49}$$

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

Proof. In RIDE, we have one full crawl and then each user is sampled until it dies or T expires. The average duration a user survives in the system is $E[\min(R, T)]$ given by:

$$E[\min(R, T)] = \int_0^T P(\min(R, T) > x)dx \tag{50}$$

where for $x \leq T$:

$$P(\min(R, T) > x) = P(R > x) = 1 - H(x).$$

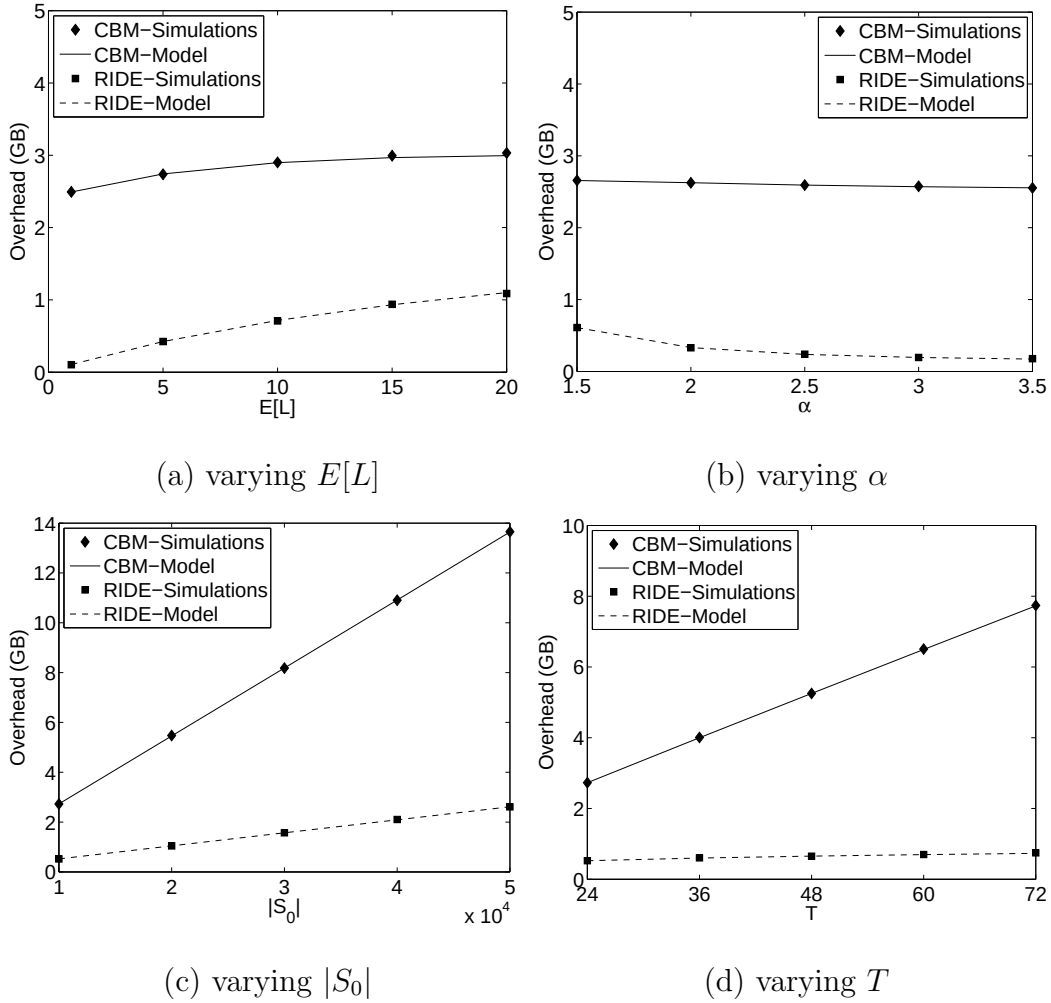
With subsampling, we have one full crawl of cost $C|S_0|$ and then repeated crawls over systems of size $\epsilon|S_0|$, which gives us

$$b_{RIDE} = C|S_0|\left[1 + \frac{\epsilon E[\min(R, T)]}{\Delta}\right], \tag{51}$$

which together with (50) leads to (49). □

6.2 Simulations and Discussion

We first verify models (44) and (49) in simulations and compare RIDE to CBM in terms of overhead. We set $C = 1$ KB and $\Delta = 6$ minutes in (44), (49). When simulating RIDE, we do not use subsampling and set $\epsilon = 1$. Users have lifetimes



(a) varying $E[L]$

(b) varying α

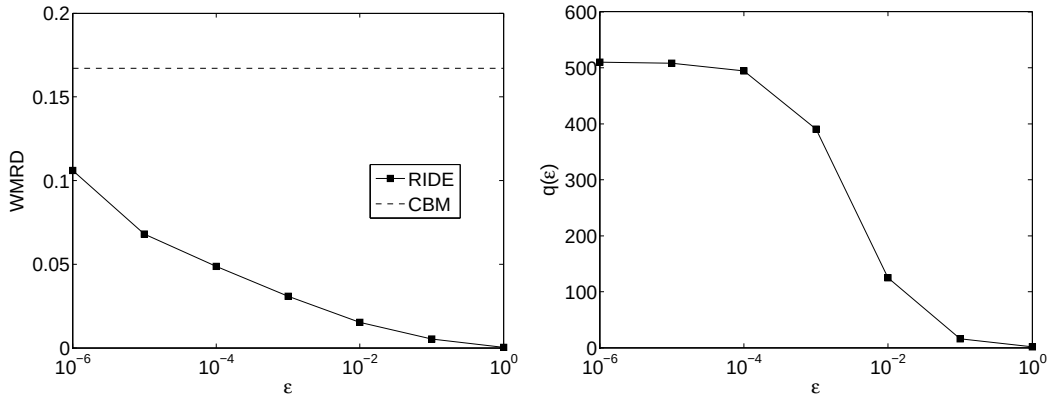
(c) varying $|S_0|$

(d) varying T

Fig. 15. Verification of models (44), (49) against simulations.

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



(a) WMRD (b) overhead ratio

Fig. 16. Effect of ϵ on the accuracy and overhead of RIDE subsampling in simulations (40-point derivatives).

drawn from a power-law distribution and each departure event triggers a new arrival. Additional simulations show that this simplified treatment of departure times does not affect the result.

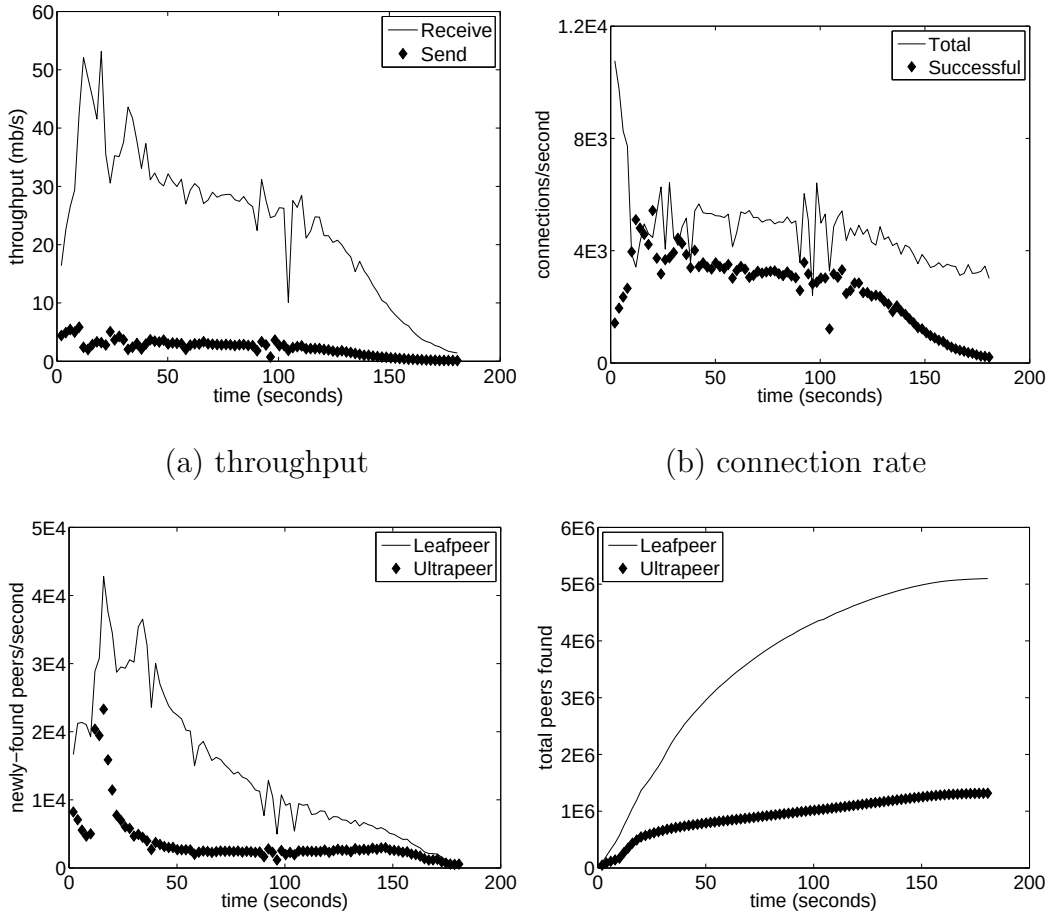
We plot in Figure 15 the overhead of CBM and RIDE obtained from both simulations and models (44), (49) by fixing three parameters from the set $\alpha = 3$, $E[L] = 2$ hours, $|S_0| = 10^4$, $T = 24$ hours and varying the fourth. The figure shows that both models track simulation results pretty accurately for all studied cases and that RIDE (even without subsampling) exhibits significantly less overhead than CBM. The

curves in Figure 15(a)-(b) are almost horizontal, predicting that neither method is very sensitive to the changes in $E[L]$ or α. Figure 15(c)-(d) display linear increasing curves for both methods, but CBM's slope is significantly more aggressive.

Next, define $q(\epsilon)$ to be the ratio b_{CBM}/b_{RIDE} for the same $|S_0|$. Assuming Pareto lifetimes with shape α, Table I shows the exact savings gained by using 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 $F_L(x)$, sampling duration T, and subsampling factor ϵ.

As long as $\epsilon|S_0|$ is sufficiently large, RIDE has the same accuracy as its original (non-subsampled) version, but at significantly smaller overhead. 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, which can be computed using standard methods of statistical inference for any given accuracy requirement specified in terms of confidence intervals [4]. 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 $|S_0|$.

We finish this section with illustrating how $\epsilon|S_0|$ affects the accuracy and overhead of RIDE. We use different values of ϵ in simulations with a fixed set of parameters: $C = 1$ KB, $\Delta = 3$ minutes, $\alpha = 1.1$, $E[L] = 1$ hour, $|S_0| = 10^{10}$, and $T = 24$ hours. Figure 16 plots the resulting metrics δ and $q(\epsilon)$. The figure shows that as $\epsilon|S_0|$ increases, RIDE's δ tends to zero. By tuning ϵ, the measurement application can decide how the solve the tradeoff between accuracy and overhead. For the specific example in Figure 16, RIDE with $\epsilon = 0.01$ reduces overhead by two orders of magnitude compared to CBM while keeping the corresponding WMRD at 10% of CBM's. These observations suggest that RIDE is indeed more suitable for large systems and long measurements than CBM.



(a) throughput

(b) connection rate

(c) discover rate

(d) success percentage

Fig. 17. Statistics of a 3-minute crawl on July 22, 2006 (single-core, dual-CPU Xeon computer @ 3GHz).

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.* [56]	Gnutella	7 min	48 hrs	–	1.3 million	7	2004	$22,500$
Liang *et al.* [36]	FastTrack	5 min	65 hrs	965	–	N/A	$2004 - 2005$	N/A
Bustamante *et al.* [3]	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.* [6]	Gnutella	10 min	9 wks	$5,000$	–	1	2002	N/A
Ripeanu *et al.* [50]	Gnutella	hours	–	–	$48,195$	N/A	2001	< 1000
Saroiu *et al.* [52]	Gnutella	7 min	60 hrs	$17,125$	–	N/A	2001	N/A

7 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.

7.1 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.

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 ultrapeers and then contact them to obtain their neighbor lists, which are then used in a BFS search to discover all currently alive ultrapeers in the system. Neighbor lists in Gnutella include other ultrapeers with whom a connection is currently active, suggested ultrapeers 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 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 17(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* ultrapeer 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 ultrapeers 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 17(c) varies between $40,000$/second and $10,000$/second and that of new ultrapeers 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 17(d), 90% of ultrapeers (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 [56], but it also discovers almost 5 times more concurrent users than any other study.

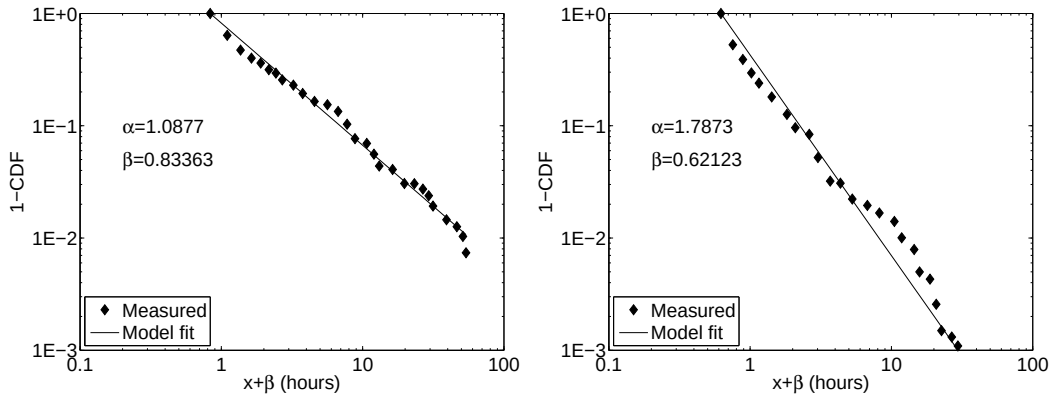
7.2 Peer Lifetimes

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.[4] 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 [29], [34], [65]. 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 sub-sampled 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

[4]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. 18. Inverse-averaged estimator $E_R(x)$ for responsive peers and links in Gnutella. Both cases use 3-point derivatives.

read the GnuSpider logs and applied RIDE to reconstruct $F_L(x)$. Figure 18(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 $\beta = 0.83$, where the shape parameter is very close to the 1.06 observed in [3]. Denote by r the expected residual lifetime 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.

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

7.3 Link Lifetimes

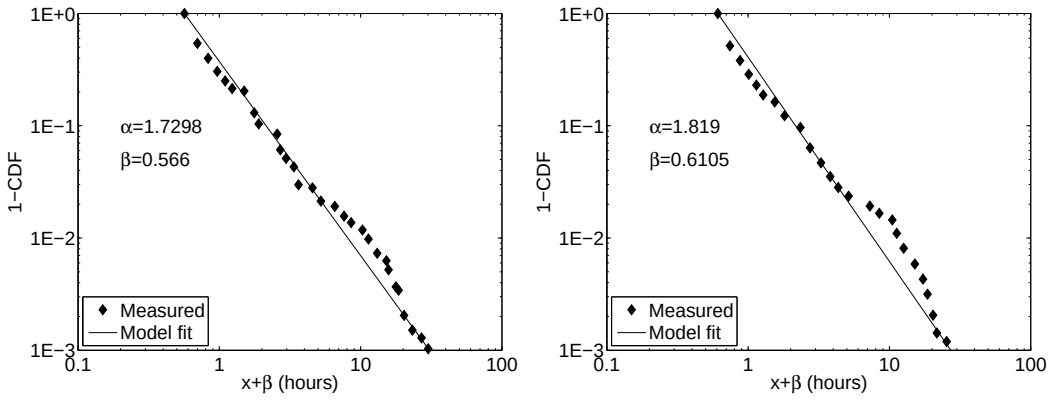
It is straightforward to apply the residual-based algorithm to measure the link lifetime distribution in Gnutella networks. In the experiment of section 7.2, GnuCrawler kept track of the links of responsive ultrapeers found in S_0 and updated their status (i.e., connected or broken) in subsequent crawls. Using this information, we applied the

same processing program to extract link residuals from GnuSpider logs and perform the proposed recovery technique to obtain $E_R(x)$. Figure 18(b) shows that the resulting distribution of all link lifetimes is also power-law, but this time with $\alpha = 1.79$, 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 Figure 19. 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 $\alpha = 1.73$ and the latter has $\alpha = 1.82$; 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.

7.4 Discussion

With the experimental results of this section, we are now able to study resilience properties of Gnutella networks by applying models from [34], 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 study in future work.



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

Fig. 19. Inverse-averaged estimator $E_R(x)$ for different types of links in Gnutella. Both cases use 3-point derivatives.

8 Related Work

Some of the first P2P sampling studies date to 2001 [50], [52] and the first use of CBM can be traced to Saroiu *et al.* [52] 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 [6], Chu *et al.* used a similar method, but probed 5,000 peers every 7 minutes for 10 weeks. Bhagwan *et al.* [1] improved over [6], [52] 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.* [36] 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 [3] and [56]. Bustamante *et al.* [3] 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.* [56] 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.* [57], which focused on capturing unbiased snapshots of joint properties of users currently alive in P2P systems using random walks.

9 Discussion

In this chapter, 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 RIDE, which measured lifetime distributions with high accuracy. Our simulations and Internet experiments showed that RIDE required several orders of magnitude less bandwidth than the prior approaches. However, RIDE assumes systems with stationary arrivals and therefore could be not accurate in those that violate this assumption. We next overcome this limitation by developing a novel algorithm called U-RIDE.

CHAPTER III

UNIFORM-RIDE (U-RIDE)

1 Introduction

The problem of measuring temporal and topological characteristics of large-scale peer-to-peer networks such as Gnutella [20] and KaZaA [27] has recently received considerable attention [1], [3], [6], [36], [52], [56], [62]. One of the central elements in capturing the dynamics of P2P systems is the *lifetime distribution* of participants, which can provide valuable input to throughput models [19], [46], resilience analysis [34], [65], [66], and system design [21], [35], [52].

Previous efforts in sampling lifetimes can be categorized into two classes: *direct sampling* [3], [52], [56], which performs periodic crawls of the system to detect new peer arrival and measures their lifetimes, and *indirect sampling* [62], which scans the entire system only once and monitors all discovered peers until they depart to obtain their residual session lengths. While the latter estimator is unbiased after proper conversion of residuals to lifetimes and requires several orders of magnitude less bandwidth than the former [62], it relies on one crucial assumption – *stationarity* of the arrival process. It thus remains to be seen whether the same benefits can be achieved in systems that exhibit diurnal arrival/departure patterns or any other non-stationary dynamics. However, in order to study this question rigorously, one requires a non-stationary model of user behavior and the corresponding analysis of lifetime sampling. We focus on these issues next.

1.1 Non-Stationary User Churn

Recall that traditional analytical P2P work either directly assumes stationary Poisson arrivals [29], [37], [45], [59] or models users with equilibrium ON/OFF renewal processes [34], [62], [66], whose scaled superposition tends to a stationary Poisson process for sufficiently large system size [65]. In our comparison with related work, we only consider the approach of [65], which we call *Stationary Renewal Churn Model* (SR-CM), since it includes all other models as special cases.

We start this chapter by designing a novel generic arrival model for Internet users that can replicate first-order dynamics (i.e., mean arrival rate) of almost any non-stationary churn process. In the proposed approach, which we call *Non-Stationary Periodic Churn Model* (NS-PCM), each user alternates between ON (alive) and OFF (dead) states. As before, the duration L of ON cycles is drawn from the distribution of user lifetime $F_L(x)$, but OFF states are now split into two sub-states: REST and WAIT. The former sub-state can be visualized as the delay between the user's departure and midnight of the day when he/she joins the system again. The latter sub-state is the delay from midnight until the user's arrival into the system within a given day, which follows its own distribution $F_A(x)$. Unlike prior models, NS-PCM allows OFF periods to be dependent on the time of day and the duration of the previous ON cycle (i.e., user lifetime).

We derive that the average arrival rate $\lambda(t)$ during the day is given by $n\tau f_A(t)/\delta$, where n is the system size, τ is the period of the arrival process, δ is the average inter-arrival delay of a user, and $f_A(t) = F'_A(t)$ is the PDF of WAIT time. Thus, NS-PCM can achieve any continuous non-stationary periodic arrival rate by adjusting density $f_A(x)$ and includes SR-CM as a special case with $f_A(x) = 1/\tau$. We show examples of using NS-PCM to model Gnutella and then analyze its impact on the existing

sampling methods in distributed P2P systems.

1.2 Analysis of Existing Methods

Equipped with the new model, we examine two major existing paradigms for measuring the lifetime distribution: *Create-Based Method* (CBM) [51] and *ResIDual-based Estimator* (RIDE) [62]. The former takes snapshots of the system every Δ time units within some window W and builds a distribution of observed lifetimes as an estimate of $F_L(x)$. The latter takes only *one* full snapshot of the system and probes discovered users every Δ units until they die or the observation window ends. The measured residuals are used to infer the target distribution $F_L(x)$ using equilibrium renewal-theory assumptions. While [62] analyzes both approaches for accuracy, it does so assuming stationary arrivals into the system under SR-CM. We perform the same task using the new model NS-PCM and obtain several interesting results.

First, we show that the bias in CBM is now affected not only by Δ and the lifetime distribution $F_L(x)$, but also by the arrival CDF $F_A(x)$. This makes removal of the bias much harder as it requires knowing the arrival pattern of users. Second, we derive the exact distribution produced by CBM and establish that it is unbiased only when $\Delta = 0$ or $F_L(x)$ is exponential. Third, we find that RIDE's estimator under non-stationary churn does not converge and sometimes produces completely invalid results (including CDF functions that are non-monotonic). To understand the cause of sampling bias in RIDE, we investigate the distribution of residual lifetimes in systems driven by NS-PCM. Define $R(t)$ to be the remaining session duration of a random online user at time t and $H(x,t) = P(R(t) \leq x)$ to be the CDF of residuals of currently alive users. Our analysis shows that unlike in prior models where $\lim_{t\to\infty} H(x,t) = H(x)$ existed, NS-PCM does not admit a limiting distribution of $R(t)$, which explains why RIDE's manipulation of non-existing metrics produces

unpredictable results. Finally, we show that RIDE's bias under NS-PCM cannot be eliminated even with $\Delta = 0$ and that accurate estimation is possible only when $\lambda(t) = \lambda$ is a constant (i.e., stationary churn) or lifetimes $F_L(x)$ are exponential, neither of which is a realistic assumption in practice [22], [52], [55], [60].

It therefore remains an open problem to design a low-overhead and robust lifetime estimator for distributed systems commonly found in real life. We perform this task next.

1.3 U-RIDE

To preserve the advantage of residual sampling in terms of overhead, we design a novel sampling algorithm called *Uniform ResIDual-based Estimator* (U-RIDE), which measures the system in uniformly random points in the observation window. The naive approach would be to compute the expected residual distribution $E[H(x, U)]$, where U is a uniformly random sampling time within the period τ of the arrival process; however, we show that this expectation does not allow reconstruction of user lifetimes and is generally not related to $F_L(x)$ in closed-form. Instead, we derive a different estimator using renewal-reward theory and show that it allows unbiased estimation of $F_L(x)$ under the most general conditions of NS-PCM.

The first component of U-RIDE is a *sample-scheduling algorithm*, which decides random time instances for residual sampling. We study one such algorithm that we call *Bernoulli Scheduling* ($\mathcal{BS}$), which leverages the BASTA principle (Bernoulli Arrival See Time Average) [40] and allows accurate measurement even when the network is small or the period τ of $\lambda(t)$ is unknown. The second component of U-RIDE is a *residual processing algorithm*, which aggregates residual samples obtained by the first component and outputs a statistical quantity that can be used to estimate $F_L(x)$. We show that our aggregation algorithm can be efficiently implemented in

large systems and that it admits a subsampling technique similar to the one in [62]. Simulation results demonstrate that U-RIDE is able to accurately estimate the actual lifetime distribution $F_L(x)$ in a variety of non-stationary systems driven by NS-PCM.

1.4 Experiments

We finish this chapter with deploying U-RIDE in the Gnutella network [20], a large P2P file sharing system of roughly 6 million concurrent users. We evaluate U-RIDE using over 260M peer lifetime samples and show that RIDE [62] indeed exhibits non-trivial error compared to CBM whose bias we neglect given the small $\Delta \approx 0$ used in the experiments. On the other hand, the proposed algorithm U-RIDE produces very accurate estimation and tracks CBM distributions precisely, but at the same time reduces overhead by two orders of magnitude. Since U-RIDE is a generic sampling method that does not assume anything specific to Gnutella, it is suitable for many large, non-stationary distributed systems found in today's Internet.

The remainder of this chapter is organized as follows. We introduce a new user churn model in Section 2 and examine the existing sampling algorithms in Section 3. We propose our new method in Section 4 and evaluate it in Gnutella experiments in Section 5. Section 6 reviews prior work and Section 7 concludes this chapter.

2 Non-Stationary User Churn

In this section, we cover basic definitions, briefly discuss prior arrival models, present our simple approach for generating non-stationary churn, and examine its ability to replicate arrival rates in Gnutella [20].

2.1 Basics

Two important metrics of interest in any churn model are the arrival *process* and its *rate*. Let $M_i(t)$ be the number of arrivals from user i into the system in $[0, t]$ and assume $\lambda_i(t)$ is the corresponding arrival rate (whose existence we prove below under certain assumptions):

$$\lambda_i(t) = \lim_{h \to 0} \frac{E[M_i(t+h) - M_i(t)]}{h}. \tag{52}$$

The aggregate arrival process of the system is then $M(t) = \sum_{i=1}^{n} M_i(t)$ and its rate is $\lambda(t) = \sum_{i=1}^{n} \lambda_i(t)$, where n is the total number of participating users. Our interest in stationarity of a process is solely related to its rate as defined next.

Definition 7. *Arrival process $M(t)$ is called* rate-stationary *if $\lambda(t) = \lambda$ is simply a constant and* non-stationary *otherwise.*

To understand the properties of non-stationary processes, define:

$$\tau = \inf\{\tau : \lambda(t+\tau) = \lambda(t), \forall t \geq 0\}$$

to be the period of arrival rate $\lambda(t)$. Note that $\tau = 0$ implies a stationary process and $\tau > 0$ non-stationary. The latter type can be further classified as follows.

Definition 8. *Non-stationary process $M(t)$ is called* rate-periodic *if $0 < \tau < \infty$ and* rate-aperiodic *if $\tau = \infty$.*

Note that most real-life churn falls under the category of rate-periodic. We are now ready to examine prior churn models and overcome their limitations.

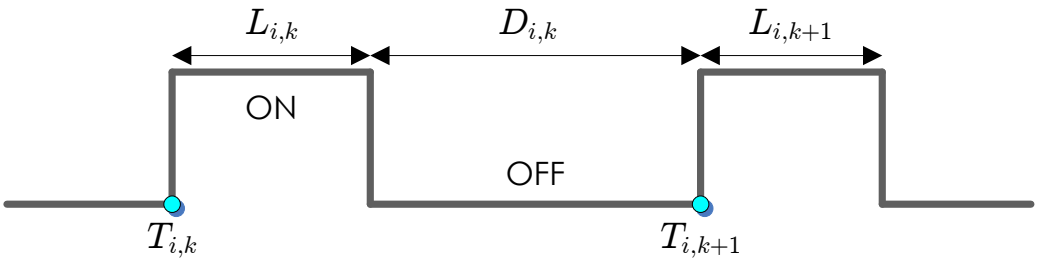


Fig. 20. User process $Z_i(t)$ under SR-CM.

2.2 Stationary Renewal Churn Model (SR-CM)

Recall that [65] models each user in P2P systems using an alternating ON/OFF renewal process:

$$Z_i(t) = \begin{cases} 1 & \text{user } i \text{ is alive at } t \text{ (ON)} \\ 0 & \text{otherwise (OFF)} \end{cases}, \tag{53}$$

which is illustrated in Figure 20, where $\{L_{i,k}\}_{k=1}^{\infty}$ are random ON durations, $\{D_{i,k}\}_{k=1}^{\infty}$ are random OFF durations, and $\{T_{i,k}\}_{k=1}^{\infty}$ are arrival times of user i.

Note that the renewal nature of this process implies that all ON/OFF durations are *independent* of each other, which makes each $Z_i(t)$ stationary as $t \to \infty$. As a result, superposition of n such arrival processes converges to a stationary point process with constant rate $\lambda(t) = \lambda$. Since this stationarity does not match churn characteristics observed in Gnutella and other P2P systems [22], [52], [55], [60], one requires a much more general approach, which we offer next.

2.3 Non-Stationary Periodic Churn Model (NS-PCM)

As before, assume that each user i is modeled by an alternating ON/OFF point process $Z_i(t)$ in (53); however, it is no longer renewal as we allow OFF cycles $\{D_{i,k}\}$ to depend on both lifetimes $\{L_{i,k}\}$ and the time when the current OFF cycle starts. Specifically, assume $0 \leq \tau < \infty$ is the period of the system that we aim to model

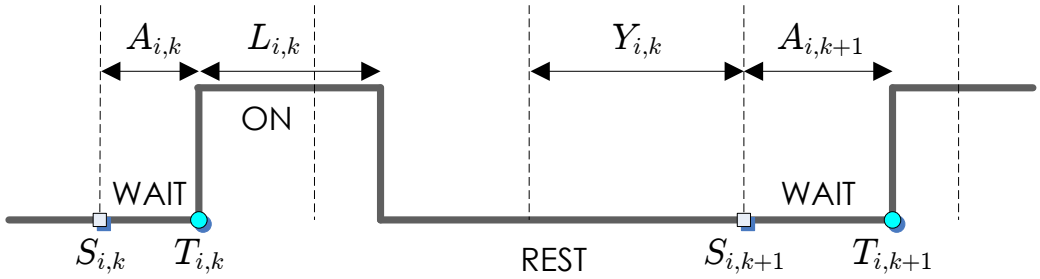


Fig. 21. User process $Z_i(t)$ under NS-PCM, where dashed vertical lines represent bin boundaries.

(e.g., for common human activity, $\tau = 24$ hours) and partition time t into bins of τ units each. For any point $t \in [0, \infty)$, define $b(t) = \tau\lfloor t/\tau \rfloor$ to be the start of the corresponding bin, $e(t) = \tau\lceil t/\tau \rceil$ to be its end, and $t^\star = t - b(t)$ to be the offset of t within its bin. Further denote by $S_{i,k} = b(T_{i,k})$ the beginning of the bin where user i arrives for the k-th time and assume each arrival occurs only once per bin (a user arriving m times in a given bin can be represented by m different users with arrivals scattered throughout the day).

As shown in Figure 21, the OFF period in the current bin $[S_{i,k}, S_{i,k}+\tau]$ starts with a WAIT duration $A_{i,k}$, which models the habits of users and their arrival preferences during the day. After process $Z_i(t)$ transitions to the ON state, the user stays logged in for a random lifetime $L_{i,k}$ and then departs from the system. Afterwards, the user stays in the REST state until time $S_{i,k+1}$ (i.e., the beginning of the bin when i decides to return into the system), from which point the process repeats. Note that the combination of REST and WAIT sub-states comprises the OFF state of (53) and that each REST duration may include a random number of full bins $Y_{i,k}$, which represent long-term absence cycles of the user from the Internet. Furthermore, observe in the figure that OFF durations are clearly dependent not only on user lifetimes in the same cycle, but also on the time of departure.

We next make two assumptions that allow this system to be tractable in closed-form.

Assumption 1. *Sequence* $\{L_{i,k}\}_{k=1}^{\infty}$ *consists of i.i.d. variables with CDF* $F_L(x)$, $\{A_{i,k}\}_{k=1}^{\infty}$ *is i.i.d. with differentiable CDF* $F_A(x)$, *and* $\{Y_{i,k}\}_{k=1}^{\infty}$ *is i.i.d. with CDF* $F_Y(x)$. *Furthermore, these sequences are pair-wise independent.*

Given this assumption, we can replace each user i's lifetimes with a random variable $L \sim F_L(x)$ such that $0 < E[L] < \infty$, its WAIT durations with $A \sim F_A(x)$ where $F_A(\tau) = 1$, and its absence times with $Y \sim F_Y(x)$. Pair-wise independence means that the lattice process defined on points $\{S_{i,k}\}_{k=1}^{\infty}$ for each user i is renewal (formally established below), even though $Z_i(t)$ is not. Additionally, notice that inter-arrival delays $\{T_{i,k+1} - T_{i,k}\}_{k=1}^{\infty}$ are i.i.d. and do not depend on user i.

Our second assumption prevents synchronization between different users and ensures sufficient variety of samples collected from crawling the system.

Assumption 2. *Processes* $\{Z_i(t)\}_{i=1}^{n}$ *are mutually independent.*

We call the system defined by the above rules and assumptions *Non-Stationary Periodic Churn Model* (NS-PCM). The following lemma reveals an important property of the point process formed by $\{S_{i,k}\}$, i.e., bin boundaries before each arrival (see Figure 21).

Lemma 1. *Point process* $\{S_{i,k}\}_{k=1}^{\infty}$ *is lattice and renewal.*

Proof. Notice that $S_{i,k+1} - S_{i,k}$ can be expressed as:

$$S_{i,k+1} - S_{i,k} = e(A_{i,k} + L_{i,k}) + Y_{i,k},$$

where $e(t) = \tau\lceil t/\tau \rceil$ is the end of the bin that contains t. From Assumption 1, it follows that interval $\{S_{i,k+1} - S_{i,k}\}_{k=1}^{\infty}$ is i.i.d. and lattice, which establishes the desired result. □

Next, we use Lemma 1 to show that arrival rate $\lambda(t)$ under NS-PCM is a simple periodic function determined by $F_A(x)$.

Lemma 2. *Suppose that time t is sufficiently large. Then, arrival rate $\lambda(t)$ of an NS-PCM system with n users exists and is a periodic function given by:*

$$\lambda(t) = \frac{n\tau f_A(t^\star)}{\delta}, \tag{54}$$

where $t^\star \in [0, \tau)$ is the offset of t within the bin, $\delta = E[T_{i,k+1} - T_{i,k}]$ is the average inter-arrival delay of a user, and $f_A(x) = F'_A(x)$ is the PDF of arrival time A.

Proof. First of all, denote by $I_i(t, t+h)$ a random variable indicating whether user i has an arrival within interval $[t, t+h)$, where $h > 0$ is small enough so that t and $t+h$ are in the same bin. Then, we rewrite arrival rate $\lambda(t)$ defined in (52) as follows:

$$\begin{aligned}
\lambda(t) &= \lim_{h\to 0} \frac{E[M(t+h) - M(t)]}{h} \\
&= \lim_{h\to 0} \frac{E[\sum_{i=1}^{n} I_i(t, t+h)]}{h} \\
&= \lim_{h\to 0} \frac{\sum_{i=1}^{n} P(I_i(t, t+h) = 1)}{h}.
\end{aligned} \tag{55}$$

Next, we derive $P(I_i(t, t+h) = 1)$, which is the probability for a user i to have an arrival in interval $[t, t+h)$. Notice that $I_i(t, t+h)$ is equivalent to the event that there exists an integer k such that arrival time $T_{i,k} \in [t, t+h)$:

$$P(I_i(t, t+h) = 1) = P(T_{i,k} \in [t, t+h)). \tag{56}$$

Further notice that $T_{i,k} = S_{i,k} + A_{i,k}$ by definition, where $S_{i,k}$ and $A_{i,k}$ are the start and offset of the bin containing $T_{i,k}$. Thus, $T_{i,k} \in [t, t+h)$ is equivalent to the event that $T_{i,k}$ is contained by the same bin as t and the offset of $T_{i,k}$ is included by interval $[t^\star, t^\star + h]$:

$$T_{i,k} \in [t, t+h) \Leftrightarrow (S_{i,k} = b(t)) \cap (A_{i,k} \in [t^\star, t^\star + h]). \tag{57}$$

Since $S_{i,k}$ and $A_{i,k}$ are independent by Assumption 1, it thus follows from (56)-(57) that:

$$P(I_i(t,t+h)=1) = P(S_{i,k}=b(t)) \times P(A_{i,k}\in[t^\star,t^\star+h)). \tag{58}$$

For $P(A_{i,k}\in[t^\star,t^\star+h))$, we simply have that:

$$P(A_{i,k}\in[t^\star,t^\star+h)) = F_A(t^\star+h)-F_A(t^\star). \tag{59}$$

It remains to derive the probability $P(S_{i,k}=b(t))$. Notice that $S_{i,k}=b(t)$ for any k is equivalent to the event that t hits the first bin of the k-th cycle. Note that point process $\{S_{i,k}\}$ is proved in Lemma 1 to be a lattice renewal process. Using renewal theory, we obtain that time t hits the first bin with probability $1/\eta$, where η is the expected number of bins in cycle $[S_{i,k},S_{i,k+1})$. Further note that $\eta=\delta/\tau$, where δ is the expected length of cycle $[S_{i,k},S_{i,k+1})$:

$$\delta = E[S_{i,k+1}-S_{i,k}] = E[T_{i,k+1}-T_{i,k}], \tag{60}$$

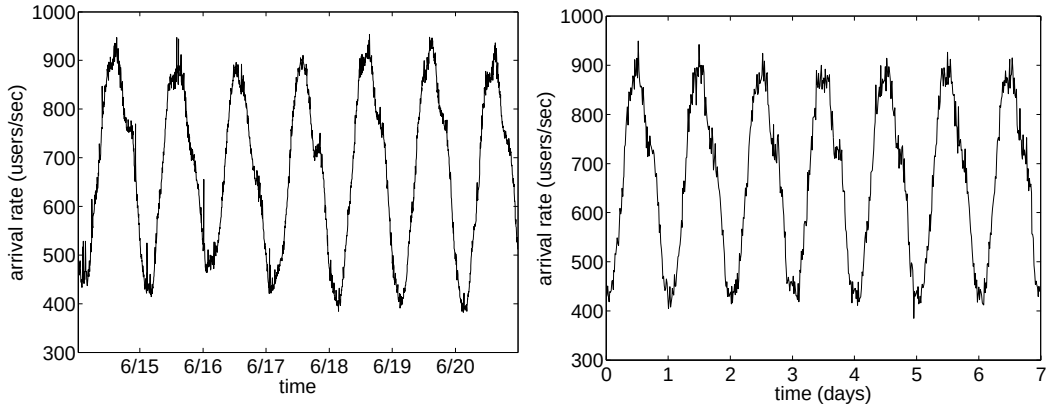
where the second equality comes from the fact that $E[A_{i,k}]=E[A_{i,k+1}]$ by Assumption 1. It thus follows that:

$$P(P(S_{i,k}=b(t))) = 1/\eta = \tau/\delta. \tag{61}$$

Substituting (59) and (61) into (58) establishes that:

$$P(I_i(t,t+h)=1) = \frac{\tau(F_A(t^\star+h)-F_A(t^\star))}{\delta}. \tag{62}$$

Further utilizing (62) and considering that n users are homogeneous, we reduce (55)



(a) Gnutella (b) NS-PCM simulations

Fig. 22. User arrival rate: a) observed in Gnutella during June 14-20, 2007; b) obtained from NS-PCM simulations.

into:

$$\lambda(t) = \lim_{h \to 0} \frac{n\tau(F_A(t^\star + h) - F_A(t^\star))}{\delta h} = \frac{n\tau f_A(t^\star)}{\delta}, \tag{63}$$

where the second equality is guaranteed by the assumption that $F_A(x)$ is differentiable every in $[0, \tau)$. The desired result follows immediately. □

Using (54), one can approximate first-order dynamics of a wide class of systems with both stationary and non-stationary arrivals. For example, setting $f_A(x) = 1/\tau$, we obtain $\lambda(t) = n/\delta = \lambda$, which is identical to SR-CM. To illustrate a more interesting example, we first collect arrival rates from a 7-day measurements of the Gnutella network (see Section 5 for details) and plot them in Figure 22(a), which indicates a clear pattern of diurnal churn. Then, we average the empirical arrival rate $\lambda(t)$ over the observed 7 days to obtain the parameters of NS-PCM. Specifically, integrating

(54), we get for $x \in [0, \tau)$:

$$f_A(x) = \frac{\lambda(x)}{\int_0^\tau \lambda(t)dt}, \quad \delta = \frac{n\tau}{\int_0^\tau \lambda(t)dt}. \tag{64}$$

Finally, we generate a system of $n = 100,000$ users with A drawn from $f_A(x)$ and plot the resulting instantaneous arrival rates in Figure 22(b), which shows a random arrival pattern very similar to that of Gnutella.

3 Analysis of Existing Methods

In this section, we characterize the accuracy of existing measurement methods under NS-PCM. Discussion of the associated overhead is presented in Section 4.

3.1 Basics

Suppose that the target P2P system is fully decentralized and the sampling process has recurring access to the information about which users are currently present in the system. This process allows us to test whether a given user i is still alive as well as discover the entire population of the network at any time t (e.g., using crawls). The goal of the sampling process is to estimate with as much accuracy as possible function $F_L(x)$, which we assume is continuous everywhere in the interval $(0, \infty)$. However, due to bandwidth and connection-delay constraints on obtaining this information, the sampling process cannot query the system for longer than W time units or more frequently than once per Δ interval, where Δ usually varies from several minutes to several hours depending on the speed of the measurement facility and network size. These constraints lead to the following two properties: 1) all lifetime samples are discrete and rounded to a multiple of Δ; and 2) all samples are no larger than W.

Denote by $\mathcal{A}$ the sampling algorithm of interests and by $\mathcal{V}_\mathcal{A}$ its sample set after an

infinite measurement. Further define $E_{\mathcal{A}}(x)$ to be the *estimator* function computed from set $\mathcal{V}_{\mathcal{A}}$ for approximating the value of $F_L(x) = P(L \leq x)$. Note that $E_{\mathcal{A}}(x)$ is discrete and can be arbitrarily different from target distribution $F_L(x)$.

Definition 9 ([62])**.** *Estimator $E_{\mathcal{A}}(x)$ of algorithm $\mathcal{A}$ is unbiased with respect to a target continuous random variable L if it matches the distribution of L at all discrete points $x_j = j\Delta, j = 1, 2, \ldots, W/\Delta$ in the interval $[\Delta, W]$ for any $\Delta > 0$:*

$$E_{\mathcal{A}}(x_j) = P(L \leq x_j). \tag{65}$$

Notice that empirical distributions based on a *finite* set $\mathcal{V}_{\mathcal{A}}$ will not generally match the target distribution $F_L(x)$, which is not a source of bias but rather a limitation of the finite measurement process. Definition 9 instead refers to errors that cannot be eliminated by sampling the system indefinitely.

3.2 Create-Based Method (CBM)

CBM was first introduced by [51] in the context of operating systems and later applied to peer-to-peer networks by [3], [52], [56]. Recall from [51] that CBM uses an observation window of size $2W$, which is split into small intervals of size Δ. Within the observation window $[0, 2W]$, the algorithm takes snapshots of the system at points $x_j = j\Delta$, i.e., at the beginning of each interval. To avoid sampling bias, [51] suggests dividing the window into two halves and only including in sample set $\mathcal{V}_C$ lifetimes that appear during the first half of the window. Based on $\mathcal{V}_C$, define $E_C(x_j)$ to be the CBM estimator of the lifetime distribution $F_L(x)$:

$$E_C(x_j) = \lim_{N_C \to \infty} \frac{N_C(x_j)}{N_C}, \tag{66}$$

where $N_C = |\mathcal{V}_C|$ is the size of the sample set and $N_C(x)$ is the number of seen users with lifetimes no larger than x.

As formalized by [62], there are two possible causes of bias in CBM sampling: 1) missed peers that join and depart between consequent crawls; 2) random direction of round-offs (i.e., some samples rounded up and others down). We say a user's lifetime L such that $x_j \leq L < x_{j+1}$ is *inconsistently* sampled if it is rounded down to x_j and *consistently* sampled otherwise (i.e., rounded up to x_{j+1}). Define ρ_j to be the probability of inconsistent round-offs for lifetimes in the interval $[x_j, x_{j+1})$, where ρ_0 refers to the probability of missing a user. The next theorem indicates that the bias in CBM under NS-PCM is determined not only by Δ and lifetime distribution $F_L(x)$, but also by the arrival distribution $F_A(x)$.

Theorem 10. *Under NS-PCM, CBM estimator* (66) *produces the following distribution:*

$$E_C(x_j) = \frac{F_L(x_j) - \rho_0 + \rho_j}{1 - \rho_0}, \tag{67}$$

where ρ_j is given by:

$$\rho_j = \sum_{v=0}^{W/\Delta - 1} \int_{x_v}^{x_{v+1}} \frac{(F_L(x_{v+j+1} - y) - F_L(x_j)) f_A(y^\star) dy}{\int_0^W f_A(u^\star) du}, \tag{68}$$

$F_L(x)$ is the CDF of the lifetime distribution, and $f_A(x)$ is the PDF of arrivals.

Proof. Note that (67) can be proved using exactly the same reasoning as in [62, Theorem 3]. In what follows, we thus focus on deriving the result of ρ_j in (68), the probability of inconsistent round-offs for lifetimes in the interval $[x_j, x_{j+1})$.

Without loss of generality, we shift the time origin to the start time t_0, i.e., $t_0 = 0$ so that the first half of the window is given by $[0, W]$. Suppose that a user arrives at time $X \in [0, W]$, where X is relative to t_0, and its lifetime is $x_j < L \leq x_{j+1}$, where $x_j = j\Delta$. Further assume that arrives time $X \in [x_v, x_{v+1}]$. Denote by $d = X - x_v$

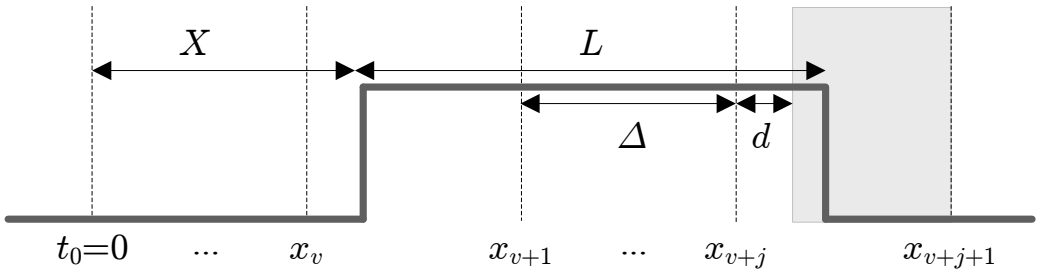


Fig. 23. Illustration of inconsistent round-off in CBM, where arrival time $X \in [x_v, x_{v+1}]$ and lifetime $L \in (x_j, x_{j+1}]$. Vertical dotted lines stand for CBM sampling points with interval Δ. Gray area represents the region of inconsistent sampling, that is, if the user departs within the gray area, its lifetime could be inconsistently rounded to x_j. The gap between x_{v+j} and the beginning of the gray area is given by d.

the gap between X and x_v. Then, we establish that the user lifetime is inconsistently rounded off to x_j if and only if departure time $X + L$ is within $[x_{v+j} + d, x_{v+j+1})$, which is illustrated in Figure 23 as gray area.

Denote by $g_v(X)$ the probability of inconsistent sampling given that $X \in [x_v, x_{v+1}]$. Then, we express $g_v(X)$ as:

$$\begin{aligned} g_v(X) &= P(x_{v+j} + d < X + L \leq x_{v+j+1} | X \in [x_v, x_{v+1}]) \\ &= P(x_j < L \leq x_{v+j+1} - X | X \in [x_v, x_{v+1}]) \\ &= F_L(x_{v+j+1} - X) - F_L(x_j). \end{aligned} \tag{69}$$

Next, we relax the condition on X to obtain roundoff-error probability ρ_j. Denote by $f_X(x)$ the PDF of arrival time X. Thus, we have:

$$\rho_j = \sum_{v=0}^{W/\Delta - 1} \int_{x_v}^{x_{v+1}} g_v(y) f_X(y) dy. \tag{70}$$

It remains to derive the distribution $f_X(x)$ of arrival time X. Notice that X could be anywhere within $[0, W]$. We first consider the probability of a user arriving before

time x in the first half of the window, i.e., $F_X(x) = P(X \leq x)$. In fact, $P(X \leq x)$ is given by the ratio of the number of users arriving before x over the total number of users that appear in the first half of the window. Utilizing the result of $\lambda(t)$ in (54), we obtain that:

$$F_X(x) = \frac{\int_0^x \lambda(t)dt}{\int_0^W \lambda(u)du} = \frac{\int_0^x f_A(t^\star)dt}{\int_0^W f_A(u^\star)du}. \tag{71}$$

The last step is to differentiate $F_X(x)$, which gives:

$$f_X(x) = \frac{f_A(x^\star)}{\int_0^W f_A(u^\star)du}. \tag{72}$$

The desired result immediately follows from substituting (69) and (72) into (70). □

Note that Theorem 10 generalizes the result developed in [62] to non-stationary systems. It is easy to verify that for stationary arrivals, i.e., $f_A(x) = 1/\tau$ for $x \in [0, \tau)$, the result in (68) becomes:

$$\rho_j = \frac{1}{\Delta} \int_{x_j}^{x_{j+1}} F_L(x)dx - F_L(x_j), \tag{73}$$

which together with (67) gives the same expression for the CBM estimator as in [62]. We next investigate whether there exist cases that make CBM unbiased under the new churn model.

Corollary 3. *Under NS-PCM, the only lifetime distribution that allows CBM to be unbiased simultaneously for all $\Delta > 0$ is exponential. Furthermore, as $\Delta \to 0$, probability $\rho_j \to 0$ and $E_C(x_j) \to F_L(x_j)$, i.e., CBM becomes unbiased for any $F_L(x)$ and $F_A(x)$.*

Proof. We can prove the first statement by a) deriving the necessary and sufficient condition for CBM to be unbiased simultaneously for all $\Delta > 0$ and b) showing that the exponential distribution is the only one that satisfies the condition.

For a), we substitute $E_C(x_j) = F_L(x_j)$ into (67) to establish that the necessary and sufficient condition is given by:

$$\rho_j = \bar{F}_L(x_j)\rho_0, \tag{74}$$

where $\bar{F}_L(x_j) = 1 - F_L(x_j)$ is the complementary CDF of lifetimes.

For b), it is easy to verify that any exponential distribution satisfies (74). With $F_L(x) = 1 - e^{-x/\mu}$, inconsistency probability ρ_j becomes that:

$$\begin{aligned}
\rho_j &= \sum_{v=0}^{W/\Delta-1} \int_{x_v}^{x_{v+1}} \left(e^{-x_j/\mu} - e^{-x_{v+j+1}+y}\right) f_A(y^\star)dy \\
&= e^{-x_j/\mu} \sum_{v=0}^{W/\Delta-1} \int_{x_v}^{x_{v+1}} \left(1 - e^{-x_{v+1}+y}\right) f_A(y^\star)dy \\
&= \bar{F}_L(x_j)\rho_0.
\end{aligned} \tag{75}$$

Now, we prove that the only lifetime distribution satisfying (74) is exponential. Substituting (68) into (74) establishes:

$$\begin{aligned}
&\sum_{v=0}^{W/\Delta-1} \int_{x_v}^{x_{v+1}} \left(F_L(x_j + x_{v+1} - y) - F_L(x_j)\right) f_A(y^\star)dy \\
&= \bar{F}_L(x_j) \sum_{v=0}^{W/\Delta-1} \int_{x_v}^{x_{v+1}} F_L(x_{v+1} - y) f_A(y^\star)dy \\
&= \sum_{v=0}^{W/\Delta-1} \int_{x_v}^{x_{v+1}} \bar{F}_L(x_j) F_L(x_{v+1} - y) f_A(y^\star)dy.
\end{aligned} \tag{76}$$

For (76) to hold for all $\Delta > 0$, we need to have:

$$F_L(u + v) - F_L(u) = \bar{F}_L(u)F_L(v), \tag{77}$$

for any $u > 0$ and $v > 0$. Note that (77) simplifies to $\bar{F}_L(u + v) = \bar{F}_L(u)\bar{F}_L(v)$, to which the only solution is $\bar{F}_L(x) = e^{-x/\mu}$. The first statement of this corollary thus follows immediately.

The second statement of this corollary can be easily proved by showing that ρ_j tends to zero as $\Delta \to 0$. From Taylor expansion, we have that:

$$\begin{aligned} F_L(x_{v+j+1} - y) - F_L(x_j) &= f_L(x_j)(x_{v+1} - y) + \Theta(\Delta^2) \\ &\leq f_L(x_j)\Delta + \Theta(\Delta^2), \end{aligned} \tag{78}$$

and

$$F_A(x^\star_{v+1}) - F_A(x^\star_v) = f_A(x^\star_j)\Delta + \Theta(\Delta^2). \tag{79}$$

Thus, ρ_j can be upper bounded as follows:

$$\begin{aligned} \rho_j &\leq \sum_{v=0}^{W/\Delta-1} \frac{(f_L(x_j)\Delta + \Theta(\Delta^2))(f_A(x^\star_j)\Delta + \Theta(\Delta^2))}{\int_0^W f_A(u^\star)du} \\ &= \frac{\Delta W f_L(x_j) f_A(x^\star_j) + \Theta(\Delta^2)}{\int_0^W f_A(u^\star)du}, \end{aligned} \tag{80}$$

which indicates that $\rho_j \to 0$ as $\Delta \to 0$. □

Interestingly, CBM's conditions for removing bias did not change from those under stationary churn (and are still impossible to satisfy in practice), despite the fact that its bias in all other cases became a much more complex function of both $F_L(x)$ and $F_A(x)$. We next examine how RIDE is impacted by NS-PCM.

3.3 ResIDual-based Estimator (RIDE)

Wang *et al.* [62] proposed RIDE to address potential problems of overhead and bias in CBM. At time t_0, RIDE takes a snapshot of the whole system and records in set $\mathcal{V}_R$ all users found to be alive. For all subsequent intervals j ($j = 1, 2, ..., W/\Delta$) of Δ time units, the algorithm keeps probing peers in set $\mathcal{V}_R$ either until they die or W expires. After the observation window W is over, the algorithm collects the residual

lifetimes of users in $\mathcal{V}_R$. Define $E_H(x_j)$ to be the empirical residual distribution based on sample set $\mathcal{V}_R$:

$$E_H(x_j, t_0) = \lim_{N_R \to \infty} \frac{N_R(x_j)}{N_R}, \tag{81}$$

where $N_R = |\mathcal{V}_R|$ is the number of acquired samples and $N_R(x)$ is the number of them no larger than x. Denote by $E_R(x_j, t_0)$ the RIDE estimator of $F_L(x)$ obtained using a single crawl at time t_0:

$$E_R(x_j, t_0) = 1 - \frac{h(x_j, t_0)}{h(0, t_0)}, \tag{82}$$

where $h(x, t_0)$ is the numerical derivative of $E_H(x_j, t_0)$.

To quantify the accuracy of (82), we must first determine how its companion $E_H(x_j, t_0)$ relates to $F_L(x)$. Notice that $E_H(x_j, t_0)$ measures the *residual lifetime distribution* of users alive at t_0. Specifically, denote by $R(t)$ the actual remaining lifetime of a random user alive at time t and by $H(x, t) = P(R(t) \leq x)$ its CDF. Then, we immediately have the following result.

Lemma 3. *Under NS-PCM, $E_H(x_j, t_0)$ is an unbiased estimator of $H(x, t_0)$, i.e., $E_H(x_j, t_0) = H(x_j, t_0)$ for $j = 1, \ldots, W/\Delta$.*

Proof. Notice that if a user is found to be alive at time t_0, its residual lifetime according to the definition is the time from t_0 till it dies. This observation is consistent with the sampling rule of RIDE as described above. Therefore, sample set $\mathcal{V}_R$ contains instances of residuals $R(t_0)$ and estimator $E_H(x_j, t_0)$ computed from set $\mathcal{V}_R$ gives the empirical distribution of $R(t_0)$. It follows that when the sample size $|\mathcal{V}_R| \to \infty$, empirical distribution $E_H(x_j, t_0)$ converges to the actual distribution $P(R(t_0) \leq x_j)$. □

Then, the problem of analyzing RIDE's accuracy reduces to deriving the residual distribution $H(x, t_0)$, which can be obtained by applying the lattice version of the

Renewal-Reward Theorem [63, page 60] to point process $\{S_{i,k}\}$.

Theorem 11. *Under NS-PCM, residual lifetime distribution $H(x, t_0)$ is a periodic function of time t_0 for sufficiently large t_0:*

$$H(x, t_0) = 1 - \frac{\int_x^\infty \omega(z - x, t_0^\star) dF_L(z)}{\int_0^\infty \omega(z, t_0^\star) dF_L(z)}, \tag{83}$$

where $\omega(x, u)$ for $u \in [0, \tau)$ is given by:

$$\begin{aligned} \omega(x, u) = {} & F_A(u) - F_A(\max(u - x^\star, 0)) + 1 \\ & - F_A(1 + \min(u - x^\star, 0)) + b(x)/\tau. \end{aligned} \tag{84}$$

Proof. See Appendix A. □

Now, we are ready to derive what values RIDE's estimator $E_R(x_j)$ produces. Differentiating (83) and substituting the result into (82), we immediately establish the next corollary.

Corollary 4. *Under NS-PCM, RIDE estimator $E_R(x, t_0)$ is a periodic function of time t_0 for sufficiently large t_0:*

$$E_R(x_j, t_0) = 1 - \frac{\int_{x_j}^\infty \omega(z - x_j, t_0^\star) df_L(z)}{\int_0^\infty \omega(z, t_0^\star) df_L(z)}, \tag{85}$$

where $\omega(.)$ is given in (84) and $f_L(x) = F_L'(x)$ is the PDF of user lifetimes.

Proof. Differentiating both sides of (83), we obtain:

$$h(x, t_0) = H'(x, t_0) = \frac{\int_x^\infty \omega(z - x, t_0^\star) df_L(z)}{\int_0^\infty \omega(z, t_0^\star) dF_L(z)},$$

which combining with (82) establishes the desired result. □

Note from (85) that the RIDE estimator $E_R(x, t_0)$ is a complex function of $F_L(x)$, arrival pattern $F_A(x)$, and initial sample time t_0. To make estimation possible out of

this result, one requires either exponential lifetimes or stationary arrivals as shown next.

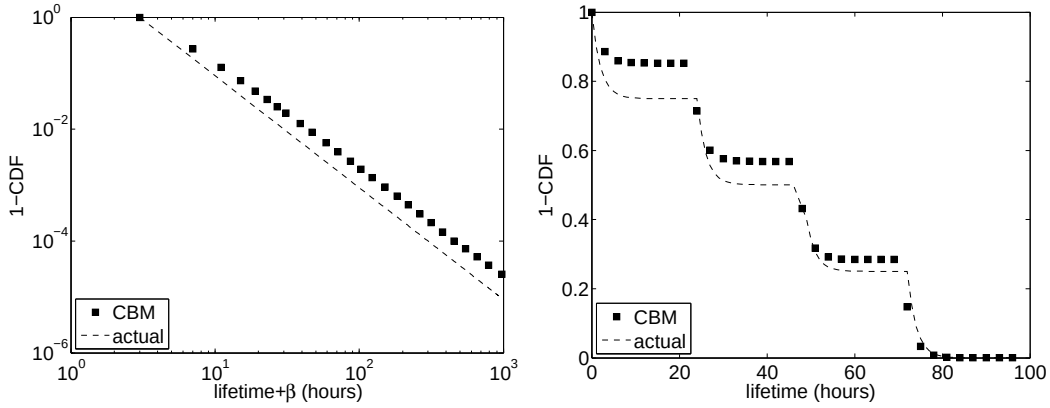
Corollary 5. *Under NS-PCM, RIDE is unbiased for all lifetime distributions iff the arrival pattern is uniform, i.e.,* $f_A(x) = 1/\tau$ *for* $x \in [0, \tau)$. *Similarly, RIDE is unbiased for all arrival patterns iff* $F_L(x)$ *is exponential.*

Interestingly, sampling interval Δ has no impact on the bias in (85), which means that no matter how fast RIDE samples the system, the bias cannot be eliminated (unlike in CBM, where it is actually possible).

3.4 Simulations

We now examine CBM and RIDE in simulations to show examples of their bias. In all simulations, we use $\tau = 24$ hours and the arrival pattern $F_A(x)$ observed in the Gnutella network. We consider two lifetime distributions: 1) Pareto with $F_L(x) = 1 - (1 + x/\beta)^{-\alpha}$, where shape $\alpha = 2$ and scale β such that $E[L] = 3$ hours; 2) periodic $L = J_1 + J_2$, where J_1 is uniformly discrete among $\{0, \tau, 2\tau, 3\tau\}$ and $J_2 \in [0, \tau)$ is a truncated exponential random variable with mean 2 hours. The former case models users with heavy-tailed lifetimes, which is fairly standard in evaluating churn models [34], [65]. The latter case covers peers that leave their computers logged in for J_1 full days and then spend a random amount of time J_2 browsing the system on the last day before finally departing.

Using sampling interval $\Delta = 3$ hours and $n = 10^6$ users, we apply CBM and RIDE to obtain the corresponding estimates of target distribution $F_L(x)$. We observe from simulations that both (67) and (85) are very accurate in predicting the errors of these methods. Due to limited space, we omit this discussion and instead focus on the actual bias. Figure 24 shows that CBM's estimates clearly deviate from both



(a) Pareto ($E[L] = 3$ hrs) (b) periodic ($E[L] = 38$ hrs)

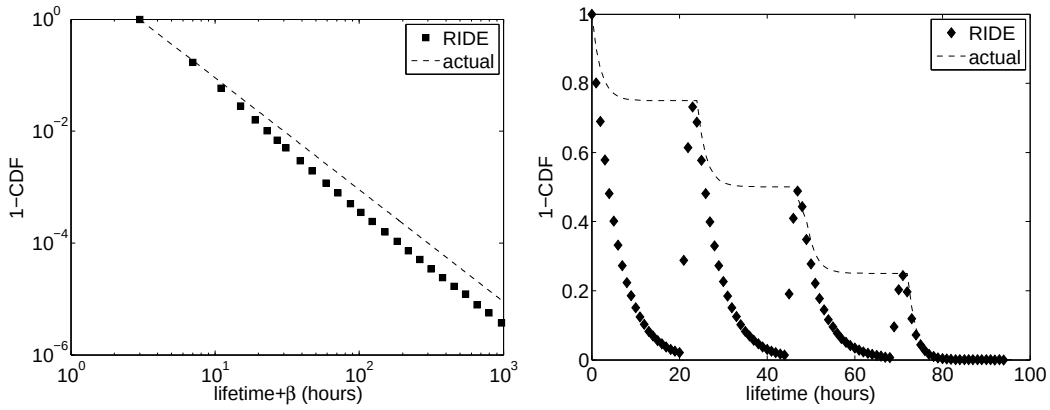
Fig. 24. CBM estimator (66) under NS-PCM.

target distributions. Even though smaller intervals (i.e., $\Delta \ll E[L]$) can oftentimes reduce the bias in CBM to negligible levels, this improvement comes at the expense of a sharp increase in overhead.

RIDE results for the Pareto case are shown in Figure 25(a), whose deviation distance from $F_L(x)$ resembles that of CBM in Figure 24(a). However, the periodic case in Figure 25(b) produces completely different results. Not only is the shape of the estimated distribution completely different from that of $F_L(x)$, but the estimated values do not even represent a valid CDF function (i.e., $E_R(x_j, t_0)$ is non-monotonic in variable x_j). Increasing overhead (i.e., lowering Δ) in this case has no impact and RIDE remains biased regardless of manipulations to the sampling process.

3.5 Discussion

In summary, all existing methods suffer from bias under NS-PCM and, to be complete accurate, require either high overhead (i.e., $\Delta \approx 0$) or unrealistic assumptions (i.e., exponential lifetimes, stationary arrivals), which cannot be satisfied in practice. In



(a) Pareto ($E[L] = 3$ hrs) (b) periodic ($E[L] = 38$ hrs)

Fig. 25. RIDE estimator (82) under NS-PCM.

what follows, we seek a better solution by adapting residual sampling to remain robust under general non-stationary arrivals while preserving its advantage over CBM in terms of overhead.

4 U-RIDE

This section generalizes RIDE by varying its sample point t_0 uniformly within the period of the arrival process $\lambda(t)$. The main issue is to decide the location of sampling points without knowing period τ and build a provably unbiased estimator from collected samples. In what follows, we first develop a general framework that can produce an unbiased estimator for $F_L(x)$ and then present an algorithm to implement it. Toward the end of this section, we validate the proposed algorithm in simulations and compare its traffic overhead to that of prior methods.

4.1 General Framework

Instead of just one snapshot at time t_0, assume that we can crawl the entire system at multiple time points $t_1, t_2, \ldots, t_M$, where M is the number of snapshots permitted by the overhead-accuracy tradeoff. For each snapshot m, we identify all live users and independently track their residuals using recurring probing every Δ time units. We call set $\mathcal{T}_M = \{t_1, t_2, \ldots, t_M\}$ a *sampling schedule* and set $\mathcal{O}_M = \{t_1^\star, t_2^\star, \ldots, t_M^\star\}$ an *offset schedule*. We further assume that all t_m are within some *snapshot window* $W_S \leq W$, i.e., $t_m \in [t_1, t_1 + W_S]$ for all m.

Definition 10. *Schedule $\mathcal{T}_M$ is called* uniform *if its offset schedule $\mathcal{O}_M$ forms a realization of a uniform random variable in $[0, \tau)$ as $M \to \infty$.*

Given a uniform schedule $\mathcal{T}_M$, we present a sampling algorithm that can construct an unbiased estimator of target distribution $F_L(x)$.

Algorithm 1. *Assuming schedule $\mathcal{T}_M$ is uniform, obtain a snapshot of the entire system at each time $t_m \in \mathcal{T}_M$. For snapshot m, record the number of alive users $N_R(t_m)$ and the number of them $N_R(x, t_m)$ with residual lifetimes no larger than x. Then, output the following ratio for each x_j:*

$$r(M, x_j) = \frac{\sum_{m=1}^{M} N_R(x_j, t_m)}{\sum_{m=1}^{M} N_R(t_m)}. \tag{86}$$

We make two comments on Algorithm 1. First, notice that at each time t_m, the sampling process does not know the exact number of discovered users that have residual lifetime $R(t_m)$ no greater than x. Therefore, values $N_R(x, t_m)$ remain unknown until the end of the measurement, at which time they are updated simultaneously for all $m \in [1, M]$. Second, it can be shown that if a user is alive during two snapshots at times t_m and t_j, it must be sampled at both instances as if these were two independent users. Doing otherwise leads to incorrect estimation and bias in the result. For

brevity, we omit additional discussion of this issue and the corresponding simulations.

The next theorem indicates that Algorithm 1 can be used to infer target distribution $F_L(x)$.

Theorem 12. *The output of Algorithm 1 under NS-PCM converges as following:*

$$E_H^*(x_j) := \lim_{M\to\infty} r(M, x_j) = \frac{1}{E[L]} \int_0^{x_j} (1 - F_L(t))dt. \tag{87}$$

Proof. As before, we first reduce $E_H^*(x)$ to reward functions $W_{i,k}(\theta)$ and $R_{i,k}(x, \theta)$ and then derive their formulas.

Notice from Algorithm 1 that $N_R(t_m)$ records the number of alive users at time t_m and can be expressed using $Z_i(t_m)$:

$$N_R(t_m) = \sum_{i=1}^{n} Z_i(t_m), \tag{88}$$

which leads to:

$$\sum_{m=1}^{M} N_R(t_m) = \sum_{m=1}^{M} \sum_{i=1}^{n} Z_i(t_m). \tag{89}$$

Dividing both sides of (89) by product nM, it thus follows that:

$$\begin{aligned} \lim_{M\to\infty} \sum_{m=1}^{M} \frac{N_R(t_m)}{nM} &= \lim_{M\to\infty} \sum_{i=1}^{n} \sum_{m=1}^{M} \frac{Z_i(t_m)}{nM} \\ &= \sum_{i=1}^{n} \lim_{M\to\infty} \sum_{m=1}^{M} \frac{Z_i(t_m)}{nM} \\ &= \sum_{i=1}^{n} \frac{E[Z_i(t_1 + \Theta)]}{n}. \end{aligned} \tag{90}$$

The third equality of (90) comes from the fact that the offset schedule of $\{t_m\}_{m=1}^{M}$ forms a realization of a uniform random variable in $[0, \tau)$. From Assumptions 1-2, n

users in the system are homogeneous, which leads to:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}=E[Z_i(t_1+\Theta)]. \tag{91}$$

Now, we let $t_1\to\infty$ and rewrite $E[Z_i(t_1+\Theta)]$ using conditional expectation:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}=\lim_{t_1\to\infty}E[E[Z_i(t_1+\Theta)]|\Theta]. \tag{92}$$

Applying the Dominated Convergence Theorem to (92) establishes that:

$$\begin{aligned}\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}&=E[\lim_{t_1\to\infty}E[Z_i(t_1+\Theta)]|\Theta]\\&=E[\lim_{t_1\to\infty}P(Z_i(t_1+\Theta)=1)|\Theta].\end{aligned} \tag{93}$$

We apply (182) to $P(Z_i(t_1+\Theta)=1)$ and replace it with $E[W_{i,k}(\Theta)]/E[\eta_{i,k}]$ in (93). Since $E[\eta_{i,k}]$ does not depend on Θ, it thus follows that (93) can be reduced to:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}=\frac{E[E[W_{i,k}(\Theta)]|\Theta]}{E[\eta_{i,k}]}. \tag{94}$$

Notice from conditional expectation that for given Θ:

$$E[W_{i,k}(\Theta)]=\int_0^\infty\omega(z,\Theta)dF_L(z), \tag{95}$$

where function $\omega(.)$ is given in (84). It follows that:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}=\frac{1}{E[\eta_{i,k}]}\int_0^\infty E[\omega(z,\Theta)]dF_L(z). \tag{96}$$

Note that Θ is uniformly distributed in $[0,\tau)$, i.e., $P(\Theta\le\theta)=x/\tau$ for $\theta\in[0,\tau)$. It thus follows from (84) that:

$$E[\omega(z,\Theta)]=z, \tag{97}$$

which leads to:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(t_m)}{nM}=\frac{E[L]}{E[\eta_{i,k}]}. \tag{98}$$

Similarly, we have:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(x,t_m)}{nM}=\frac{1}{E[\eta_{i,k}]}\int_0^{\infty}E[\varphi(x,z,\Theta)]dF_L(z), \tag{99}$$

where function $\varphi(.)$ is given in (192). It follows from (192) that for uniform Θ:

$$E[\varphi(x,z,\Theta)]=\min(x,z), \tag{100}$$

which establishes:

$$\lim_{M\to\infty}\sum_{m=1}^{M}\frac{N_R(x,t_m)}{nM}=\frac{1}{E[\eta_{i,k}]}\int_0^{x}(1-F_L(z))dz, \tag{101}$$

Combining (98) and (101), we obtain:

$$\lim_{M\to\infty}r(M,x)=\frac{1}{E[L]}\int_0^{x}(1-F_L(z))dz, \tag{102}$$

which is the desired result. □

Taking the derivative of $E_H^*(x)$ in (87), we immediately obtain the desired result.

Corollary 6. *For all $\Delta\geq 0$, the following is an unbiased estimator of $F_L(x)$:*

$$E_R^*(x_j)=1-\frac{h^*(x_j)}{h^*(0)}, \tag{103}$$

where $h^*(x)$ is the numerical derivative of $E_H^*(x)$.

We call Algorithm 1 in combination with (103) *Uniform ResIDual-based Estimator* (U-RIDE) and examine how to implement it below. In the meantime, it is worth mentioning that performing RIDE sampling at uniformly randomized time points $U\in[0,\tau)$ and then taking the expectation of the resulting CDF, i.e., $E[H(x,U)]$, does

not produce the same result as $E^*_H(x)$ in (87). According to our analysis, $E[H(x,U)]$ is heavily dependent on the arrival pattern $F_A(x)$ and thus cannot be used to reconstruct $F_L(x)$. This observation distinguishes the new method from simply applying RIDE a number of times and averaging the result.

4.2 Scheduling

The last piece of our algorithm is to find a uniform schedule $\mathcal{T}_M$ for Algorithm 1. We use a simple approach that we call *Bernoulli Scheduling* ($\mathcal{BS}$). Suppose that the algorithm starts at time t_1 and the smallest sampling interval is Δ as before. Then $\mathcal{BS}$ generates sequence $\mathcal{T}_M$ using:

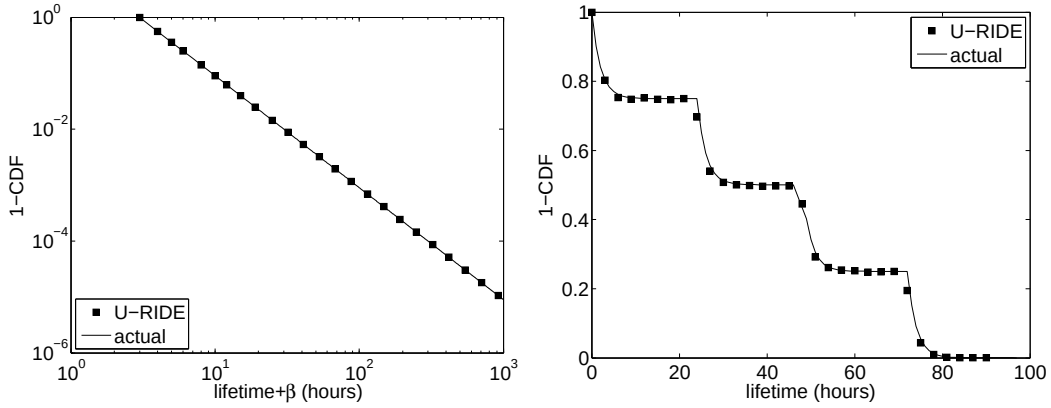
$$t_{m+1} = t_m + v_m\Delta + u_m, \quad m \geq 1,$$

where v_m is drawn from a geometric distribution with success probability p and u_m is drawn from a uniform distribution within $[0, \Delta)$. From the property of BASTA (Bernoulli Arrival See Time Average) [40], it is straightforward to show that the $\mathcal{BS}$ algorithm produces uniform schedules.

Corollary 7. *Sampling schedule $\mathcal{T}_M$ generated by $\mathcal{BS}$ is uniform for any period τ.*

Notice that the expected duration of a $\mathcal{BS}$ schedule is given by $M\Delta/p$. Therefore, p can achieve both dense (i.e., large p) and sparse (i.e., small p) sampling. The former allows the sampling process to complete in a short time, while the latter spreads traffic overhead over time and thus avoids overloading network resources. In addition, while our analysis earlier in the section implicitly assumed that period τ was known, $\mathcal{BS}$ does not require this knowledge and thus can be used in a wide variety of periodic systems without any additional input.

Next, we examine U-RIDE under NS-PCM using the same parameters as in



(a) Pareto ($E[L] = 3$ hrs) (b) periodic ($E[L] = 38$ hrs)

Fig. 26. U-RIDE estimator (103) with $\mathcal{BS}$ under NS-PCM.

Figure 25. We set $p = 0.05$ and $M = 24$ in $\mathcal{BS}$ scheduling. Figure 26 plots the lifetime distributions estimated from the output of Algorithm 1 along with the actual $F_L(x)$, indicating a very accurate match between the two. Other simulations with Weibull, discrete, uniform, and exponential lifetimes, as well as various arrival patterns $F_A(x)$, indicate that U-RIDE is extremely accurate. We omit them for brevity.

4.3 Overhead

We next study the question of how U-RIDE in its current shape compares to the other two methods in terms of overhead. To address this issue, we first derive a formula to show how U-RIDE compares to RIDE. We assume unit cost for contacting a Gnutella peer, which makes traffic overhead directly equal to the number of users contacted during the sampling process. Denote by c_j the number contacts made at the j-th step of the sampling process for $j = 1, 2, \ldots, W/\Delta$. Then, define $B_{\mathcal{A}}$ to be the sampling overhead of an algorithm $\mathcal{A}$:

$$B_{\mathcal{A}} = \sum_{j=1}^{W/\Delta} c_j. \tag{104}$$

We use B_C to represent the overhead of CBM, B_U that of U-RIDE, and B_R that of RIDE. Define $q_{xy} = B_x/B_y$ to be one of the overhead ratios of interest, where $x, y \in \{C, U, R\}$.

Theorem 13. *Assume $\mathcal{BS}$ scheduling with p and M and that U-RIDE starts at midnight. Then, overhead ratio q_{UR} is given by:*

$$q_{UR} = 1 + \frac{\tau}{E[L]} \sum_{m=2}^{M} \int_{y_{m-1}}^{y_m} f_A(t^\star)(1 - F_L(y_m - t))dt, \tag{105}$$

where $y_m = m\Delta(1-p)/p$, τ is the period of the arrival process, $E[L]$ is the expected user lifetime, $f_A(x)$ is the PDF of arrivals, and $F_L(x)$ is the CDF of lifetimes.

Proof. Note that both RIDE and U-RIDE use residual sampling, which keeps probing each discovered alive user until it dies or window T expires. Denote by $\mathcal{V}_x$ the probing set obtained by residual sampling algorithm x, where $x \in \{\text{RIDE, U-RIDE}\}$. Then, the overhead B_x of residual sampling x is proportional to the product of the probing set size and the expected residual lifetime, $B_x = |\mathcal{V}_x|E[R]$. Since expected residual lifetime $E[R]$ is the same for RIDE and U-RIDE, we only need to compare the probing set size $|\mathcal{V}_x|$ of these two methods.

Notice that probing set size $|\mathcal{V}_R|$ of RIDE equals to the size of the first snapshot made by U-RIDE at time t_1:

$$|\mathcal{V}_R| = \frac{nE[L]}{\delta}, \tag{106}$$

where δ is the expected inter-arrival delay. Note that (106) is simply the average number of alive users in steady-state. Therefore, we only need to count the number of *new* residual samples discovered by U-RIDE at time points $t_2, \ldots, t_M$.

Denote by N_m^* the number of new residual samples found at time t_m for $m = 2, \ldots, M$. Further note that we do not make multiple probes of the same user when it has multiple residual samples according to our algorithm. It thus follows that N_m^*

is the number of users that arrive during interval $[t_{m-1}, t_m)$ and live through time t_m:

$$\begin{aligned} N_m^* &= \int_{t_{m-1}}^{t_m} \lambda(t^\star)(1 - F_L(t_m - t))dt \\ &= \frac{n\tau}{\delta} \int_{t_{m-1}}^{t_m} f_A(t^\star)(1 - F_L(t_m - t))dt. \end{aligned} \tag{107}$$

Then, we obtain the probing set size $|\mathcal{V}_U|$ of U-RIDE:

$$\begin{aligned} |\mathcal{V}_U| &= \frac{nE[L]}{\delta} + \sum_{m=2}^{M} N_m^* \\ &= \frac{nE[L]}{\delta} + \frac{n\tau}{\delta} \sum_{m=2}^{M} \int_{t_{m-1}}^{t_m} f_A(t^\star)(1 - F_L(t_m - t))dt. \end{aligned} \tag{108}$$

Notice from the definition of $\mathcal{BS}$ that t_m can be modeled by $Y\Delta$, where Y is a negative binomial random variable $NegBin(m, p)$. We thus approximate t_m with its expectation $y_m \equiv E[t_m] = m\Delta(1-p)/p$ in (108). It follows from (106) and (108) that overhead ratio q_{UR} is given by:

$$\begin{aligned} q_{UR} &= \frac{B_U}{B_R} = \frac{|\mathcal{V}_U|}{|\mathcal{V}_R|} \\ &= 1 + \frac{\tau}{E[L]} \sum_{m=2}^{M} \int_{y_{m-1}}^{y_m} f_A(t^\star)(1 - F_L(y_m - t))dt, \end{aligned} \tag{109}$$

which is exactly (105). □

The result in (105) shows that q_{UR} is a function of M, Δ, and p. Under uniform arrivals, (105) becomes:

$$q_{UR} = 1 + (M-1)H(\Delta/p), \tag{110}$$

where $H(x) = \frac{1}{E[L]} \int_0^x (1 - F_L(u))du$ is the CDF of residual lifetimes in stationary systems. Notice that overhead ratio q_{UR} is an increasing function of M for constant $\Delta > 0$ and p and tends to M as $p \to 0$ or $\Delta \to \infty$. This observation motivates us to seek a more efficient way to execute U-RIDE.

Table III. Overhead ratio q_{CU} using uniform arrivals, Pareto lifetimes with shape α, $E[L] = 1$ hour, $\Delta = 3$ minutes, and U-RIDE with $M = 8$ and $p = 1/60$

α	W	q_{CU}		
		$\epsilon_U = 0.1$	$\epsilon_U = 0.01$	$\epsilon_U = 0.001$
1.1	48 hrs	5.7	50	213
	72 hrs	6	54	274
	96 hrs	6.2	57	322
2	48 hrs	19	92	151
	72 hrs	25	130	222
	96 hrs	31	166	292

4.4 Subsampling

Next, we propose a subsampling technique aimed at reducing the overhead of U-RIDE. In Algorithm 1, we apply ϵ-subsampling as follows: *for each discovered user, toss an unfair coin with success probability ϵ to decide whether the sample should be kept (i.e., added to both $N_R(t_m)$ and $N_R(x, t_m)$) or discarded.* This approach reduces measurement traffic by approximately a factor of $1/\epsilon$. Using simple renewal-process arguments, it can be shown that subsampling does not affect the properties of users collected by U-RIDE and has no effect on its ability to avoid bias.

In order to select ϵ, notice that U-RIDE (as described above) obtains many more residual samples than RIDE, most of which are not necessary for accurate estimation. As long as the total number of samples $\sum_{i=1}^{M} N_R(t_m)$ is above some threshold, U-RIDE will converge by the law of large numbers. Therefore, keeping the same number of snapshots M, but reducing the *size* of each snapshot, U-RIDE can match the overhead of RIDE without sacrificing accuracy. Denote by $\mathcal{V}_R$ and $\mathcal{V}_U$ the original sample sets of

Table IV. Number of lifetime samples in the subsets of country and ISP

Country	Samples	Percentage	Unique IPs	ISP	Samples	Percentage	Unique IPs
United States	120.8M	48.3%	21M	FDC Servers	21.5M	8.6%	3.6M
Brazil	35.7M	14.3%	6.4M	Level 3	18.2M	7.3%	3M
Canada	16M	6.4%	2.6M	Tele. Santa Catarina	11.3M	4.5%	2.1M
United Kindom	13.3M	5.3%	2M	Tele. Bahia	8.7M	3.5%	1.5M
Germany	6M	2.4%	1M	SBC	8.2M	3.3%	1.3M
Australia	5M	2%	0.93M	Verizon	6.2M	2.5%	1M
Japan	4.6M	1.9%	0.91M	Tele. Sao Paulo	5.5M	2.2%	0.96M
Netherlands	4.5M	1.8%	0.87M	Shaw	4.8M	1.9%	9.3M
Poland	4.4M	1.7%	0.82M	Cablevision	4.1M	1.6%	0.76M
Austria	4.3M	1.7%	0.7M	Cox	4.0M	1.6%	0.72M

RIDE and U-RIDE, respectively. Further, define ϵ_R and ϵ_U to be the corresponding subsampling factors. The following theorem ensures that U-RIDE with can be as efficient as RIDE.

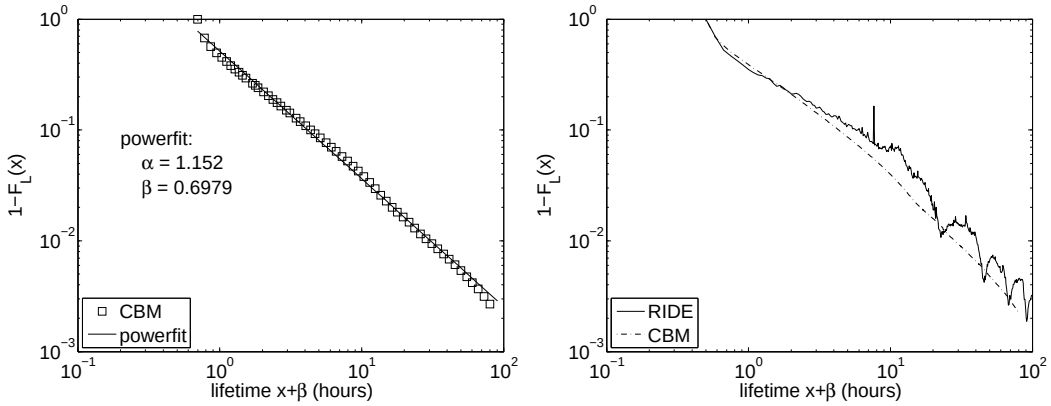
Theorem 14. *Assuming $\epsilon_R|\mathcal{V}_R| = \epsilon_U|\mathcal{V}_U|$, the overhead of U-RIDE is upper bounded by that of RIDE for all Δ, i.e., $q_{UR} \leq 1$.*

Proof. It is easy to prove the statement of this theorem by the fact that both RIDE and U-RIDE use residual sampling. Note that after discovering an alive user, residual sampling keep probing the user until it dies or window T expires. Therefore, the overhead of U-RIDE is the same as that of RIDE as long as the number of residual samples discovered by them are the same, which is guaranteed by $\epsilon_R|\mathcal{V}_R| = \epsilon_U|\mathcal{V}_U|$. Note that U-RIDE might have several samples from the same user. It thus follows that U-RIDE might have less overhead than RIDE given the condition $\epsilon_R|\mathcal{V}_R| = \epsilon_U|\mathcal{V}_U|$. □

As network size $n \to \infty$, one can always choose $\epsilon_U(n) \sim 1/n$ such that $\epsilon_U(n)|\mathcal{V}_U|$ remains constant at some predetermined threshold needed to invoke the law of large numbers. With this modification, U-RIDE retains the overhead advantages of RIDE compared to CBM and better scales to larger systems as shown in Table III for small ϵ_U.

5 Experiments

In this section, we compare U-RIDE with RIDE based on Gnutella measurements. In what follows, we first introduce our data collection process, then discuss comparison methodology, and finally present our results.



(a) CBM with a power fit (b) RIDE

Fig. 27. Estimated lifetime distribution of all observed peers using CBM and RIDE.

5.1 Dataset

Gnutella [20] is a popular peer-to-peer file sharing network that organizes users into a two-tier overlay structure. Each peer is identified by its (IP address, port) pair and can serve in one of two roles: ultrapeer or leaf. The former type of users connect to other ultrapeers to form the Gnutella overlay and route search messages between each other to find content. The latter type of users attach to a handful of ultrapeers and do not provide any routing services to other members of the system. Note that Gnutella has no central administration and its global structure at any given time is hidden from the user.

Leveraging the crawl option supported in Gnutella/0.6, our crawler requests neighbors of each visited ultrapeer and runs a BFS-like algorithm to capture snapshots of the entire system at different times t_m. In a continuous experiment that lasted $W = 7$ days during June 14-20, 2007, we performed repeated crawls of Gnutella every $\Delta = 3$ minutes, which approximated the behavior of CBM and provided enough data to emulate both U-RIDE and RIDE using offline processing. The dataset recorded

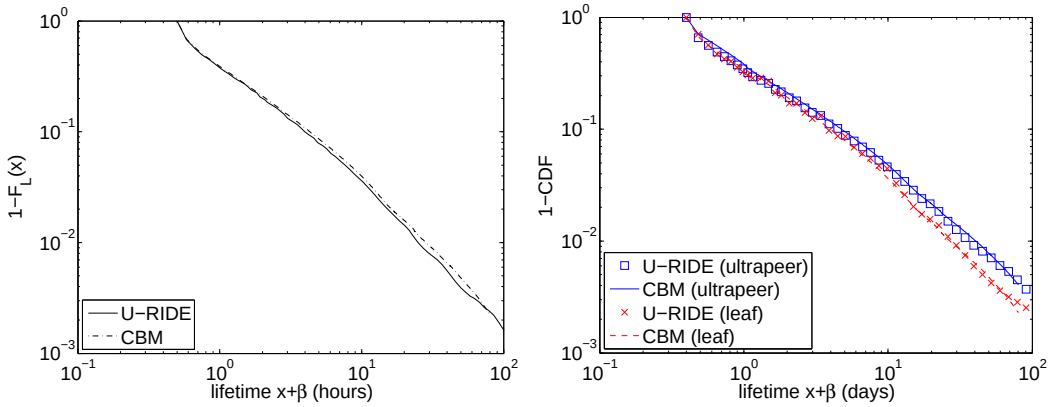
over 250M user instances (36.9M ultrapeers and 219.1M leaves) from 50.5M unique IPs. Due to the dynamic nature of ports and IPs, we were unable to determine the total number of *unique* peers that participated in the system; however, the average number of concurrent users during this period has stayed close to 6.5M.

We also split the dataset based on two criteria: geographic location and service provider. Table IV lists the numbers of samples and their percentages along with unique IPs of the top-10 subsets in both categories. We observe from the table that while the collected samples concentrate in a few countries with almost 50% from US, the distribution of users among service providers is much more even with all ISPs receiving less than 10% of the samples.

5.2 Comparison Methodology

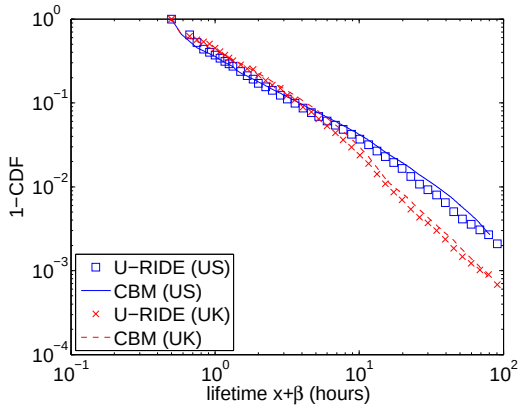
To compare U-RIDE with RIDE, we first need to obtain $F_L(x)$ as ground-truth. While this task is impossible with absolute accuracy, our earlier results (see Corollary 3) have shown that CBM has a diminishing bias under NS-PCM as $\Delta \to 0$. In particular, this condition can often be assumed to hold when $\Delta \ll E[L]$ (simulations omitted for brevity), which is satisfied in our crawls given $E[L] \approx 2$ hours.

We processed the dataset with all observed peers using CBM after discarding 30.4M invalid samples, but RIDE uses the original dataset. Figure 27(a) plots the resulting distribution on a log-log scale along with a power-law fit, which indicates that lifetimes of Gnutella users follow a power-law distribution with shape $\alpha = 1.15$ and $\beta = 0.69$, which is consistent with the result in [3] and other prior papers. With the data collected from CBM sampling, we are now ready to compare the other two methods.

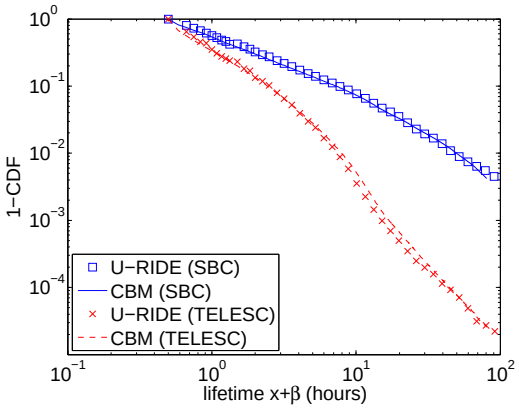


(a) all peers

(b) ultrapeer vs. leaf



(c) geographic location



(d) service provider

Fig. 28. Comparison of U-RIDE with CBM with $M = 24, \epsilon_U = 1$ in different datasets.

5.3 U-RIDE vs. RIDE

We apply the two residual sampling algorithms to the collected dataset. For RIDE, we use residual samples from a single snapshot taken at time t_0 (i.e., 5 AM on June 14th, 2007) and estimate the corresponding lifetime distributions. Figure 27(b) plots the 1-CDF of RIDE's estimator along with that of CBM. The figure shows that RIDE exhibits a non-trivial deviation from CBM and again violates the monotonicity requirement of a valid distribution function. While in general the two curves have a similar trend, significant variance near the tail compromises estimation accuracy. We also discover from experiments that the gap between RIDE and CBM is consistently non-trivial for different values of t_0. It should be noted that under different arrival conditions $F_A(x)$ and/or distributions $F_L(x)$, the bias in RIDE can be much more drastic as shown in Figure 25(b).

For U-RIDE, we use $p = 1/20$ and collect 24 full snapshots (approximately one for each hour) during the first sampling day (i.e., $W_S = \tau$ and $W = 7$ days). We then apply the corresponding estimator to the original dataset of all peers and plot in Figure 28(a) the curve computed by U-RIDE along with that of CBM. Observe in the figure that U-RIDE exhibits an almost identical match to CBM. Figure 28(b) shows a similar match of U-RIDE in the datasets containing only ultrapeers and leaves.

We also examine U-RIDE with four subsets of samples selected from Table IV. For the geographic location, we use US and UK peers to show the difference in their $F_L(x)$; and for the service provider, we select a US ISP SBC Internet Services (SBC) and a Brazilian company Telecomunicacoes de Santa Catarina SA (TELESC). Figure 28(c)-(d) indicate that U-RIDE is accurate in measuring the lifetime distribution for all studied subsets. Our additional experiments (omitted) with other subsets based on criteria such as time zone, protocol version, and software vendor of Gnutella peers

also confirm the accuracy of U-RIDE.

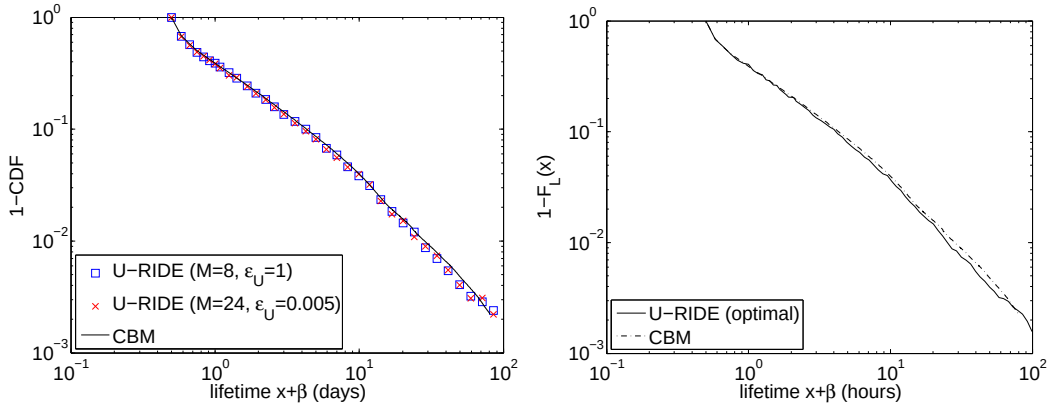
5.4 Balancing Accuracy and Overhead

Note that Gnutella experiments above took $M = 24$ system snapshots in one day and used U-RIDE without subsampling (i.e., $\epsilon_U = 1$). Figure 29(a) shows that U-RIDE with other choices of M and ϵ_U can also produce accurate estimation of the lifetime distribution. In what follows, we explore the parameter space of M and ϵ_U to strike a balance between accuracy and overhead (W_S is kept constant at one day). To assess accuracy, we employ *Weighted Mean Relative Difference* (WMRD), which is often used for comparing distribution functions [11], [30]. Given estimator function $E(x)$ and target function $F_L(x)$, the distance is defined as:

$$WMRD = \frac{\sum_{j=1}^{W/\Delta} |E(x_j) - F_L(x_j)|}{\sum_{j=1}^{W/\Delta} (E(x_j) + F_L(x_j))/2}, \tag{111}$$

where $x_j = j\Delta$. Small WMRD distances imply that estimator $E(x)$ is close to the target distribution. For comparison, RIDE exhibits $WMRD = 0.2$ and overhead ratio $q_{CR} = 9.8$ in Figure 27(b), while U-RIDE achieves $WMRD = 0.048$ and $q_{CU} = 4.6$ in Figure 29(a), where both methods use their most inefficient versions with $\epsilon_U = \epsilon_R = 1$.

Next, we illustrate a more interesting example that solves the tradeoff between accuracy and overhead. We run U-RIDE with a set of 72 combinations of parameters M (from 1 to 288) and ϵ_U (from 0.0001 to 1). To find the optimal choice for M and ϵ_U, we admit only such pairs that keep $WMRD < 0.1$ and simultaneously $q_{CU} > 100$. Among the 5 candidates that pass this criteria, we select the pair with the smallest WMRD. The resulting choice is $M = 8$ and $\epsilon_U = 0.005$, which reduces the overhead of U-RIDE by a factor of 126 compared to CBM, while achieving a very decent $WMRD = 0.055$. Figure 29(b) plots the estimated results using the optimized



(a) $M = 8, \epsilon_U = 1$ and $M = 24, \epsilon_U = 0.005$

(b) $M = 8, \epsilon_U = 0.005$

Fig. 29. Comparison of U-RIDE with CBM.

parameters, indicating a very good match despite the heavy subsampling. Since CBM does not admit similar reduction in overhead through subsampling (see [61, theorem 7]), U-RIDE emerges as the most viable solution for estimating lifetime distributions in large, non-stationary distributed systems.

6 Related Work

The Create-Based Method (CBM) for lifetime sampling was first proposed by Roselli *et al.* [51] to characterize lifetime distributions of data blocks in file systems and later introduced by Saroiu *et al.* [52] to peer-to-peer networks in order to measure session length distributions. More studies following [52] were presented by Bustamante *et al.* [3], Chu *et al.* [6], and Stutzbach *et al.* [56]. Wang *et al.* [62] proposed residual sampling as a way of overcoming potential inaccuracy and high overhead of CBM.

7 Discussion

This chapter studied the tradeoff between accuracy and overhead in sampling user lifetimes in distributed systems with non-stationary arrivals. We first proposed a novel non-stationary churn model NS-PCM, which was then used to show that existing methods could not simultaneously achieve high accuracy and low overhead given non-stationary user arrivals. To overcome this problem, we introduced a simple sampling algorithm U-RIDE that achieves unbiased estimation of the lifetime distribution and offers considerable reduction in bandwidth compared to the traditional approaches.

CHAPTER IV

DISCRETE RIDE (D-RIDE)

1 Introduction

Recent growth of the Internet in both scale and complexity has imposed a number of challenges on network management, operation, and traffic monitoring. The main problem in this line of work is to scale measurement algorithms to achieve certain objectives (e.g., accuracy) while satisfying real-time resource constraints (e.g., fixed memory consumption and per-packet processing delay) of high-speed Internet routers. This is commonly accomplished (e.g., [9], [10], [11], [12], [13], [15], [16], [24], [25], [28], [33], [30], [31], [32], [38], [43], [64]) by reducing the number of information a router has to store in its internal tables, which comes at the expense deploying special estimation techniques that can recover metrics of interest from the collected samples.

In this chapter, we focus on two problems in the general area of *measuring flow sizes* – determining the number of packets transmitted by "elephant" flows [16], [25], [28], [33], [32], [38] and building the distribution of flow sizes seen by the router in some time window [11], [30], [64] – coupled in a single measurement technique. The former problem often arises in usage-based accounting and traffic engineering [10], [16], [17], [18], [44], while the latter has a number of security applications such as anomaly and intrusion detection [2], [39], [26].

Our interest falls within the family of *residual sampling*, which selects a random point A within each flow and then samples the remainder R of that flow until it ends. Denoting by L the size (in packets) of a random flow, sampled residuals R are simply $L - A$. Stochastically larger A results in fewer flows being sampled and leads to lower

overhead in terms of both CPU and RAM consumption. Besides reduced overhead arising from omission of many small-size flows from counter tables, residual sampling guarantees to capture large flows with probability $1 - o(1)$ as their size $L \to \infty$. This allows ISPs to determine "heavy-hitters" and charge the corresponding customers for generated traffic.

While in P2P networks residual sampling distributes the initial point A uniformly within user lifetimes [62], flow-based estimation [16], [28] usually employs geometric A since it can be easily implemented with a sequence of independent Bernoulli variables. We call the resulting approach *residual-geometric sampling* and note that it has received some limited analytical attention in [16], [28]; however, unbiased estimation of individual flow sizes, analysis of the resulting error as a function of L, asymptotically accurate recovery of flow-size distribution $P(L = i)$ and the number of original flows n from sampled residuals R, and analysis of space-CPU requirements (i.e., memory and lookup overhead in steady-state) have not been explored. We overcome these issues below.

1.1 Single-Flow Usage

We start with the problem of obtaining sizes of individual flows for accounting purposes. Since residual sampling requires an estimator to convert residuals into the metrics of interest, our first task is to define proper notation and desired properties for the estimation algorithm. Assume that for a flow of size L the sampling algorithm produces residual R_L, where both L and R_L are random variables. We call an estimator $e(R_L)$ *unbiased* if its expectation produces the correct flow size, i.e., $E[e(R_L)|L = l] = E[e(R_l)] = l$. Unbiased estimation allows one to average the estimated size of several flows of a given size l and accurately estimate their total contribution. We further call an estimator *elephant-accurate* if ratio $e(R_l)/l$ converges to

1 in mean-square as $l \to \infty$. Elephant-accuracy ensures that the variance of $e(R_l)/l$ tends to zero as $l \to \infty$, which means that the amount of relative error between $e(R_l)$ and l decays to zero as $l \to \infty$.

Prior work on residual-geometric sampling [16], [28] has suggested the following estimator:

$$e(R_L) = R_L - 1 + 1/p, \tag{112}$$

where $0 < p \leq 1$ is the parameter of variable A. To understand the performance of (112), we first build a general probabilistic model for residual-geometric sampling and derive the relationship between flow size L and its residual R_L. Using this result, we prove that:

$$E[e(R_l)] = \frac{l}{1 - (1-p)^l}, \tag{113}$$

which indicates that (113) is generally biased and on average tends to overestimate the original flow size by a factor of up to $1/p$. To address this problem, we propose a different estimator:

$$\hat{e}(R_L) = R_L - 1 + 1/p - (1-p)^{R_L}/p \tag{114}$$

and prove that it is both unbiased and elephant-accurate. We also derive in closed-form the mean-square error $\delta_l = E[(\hat{e}(R_l)/l - 1)^2]$ for finite l, which can be used to determine when (114) approximates the true flow size with accuracy sufficient for billing purposes.

1.2 Flow-Size Distribution

Our second problem is estimation of the original flow-size PMF $f_i = P(L = i), i = 1, 2, \ldots$ We call PMF estimator q_i *asymptotically unbiased* if it converges in probability to f_i for all i as the number of sampled flows $M \to \infty$. One may be at first tempted

to define:

$$q_i = P(e(R_L) = i), \qquad \hat{q}_i = P(\hat{e}(R_L) = i) \tag{115}$$

using the estimates produced by (112) or (114); however, we show that (115) almost always differs from the original distribution f_i and the bias persists as sample size $M \to \infty$. The reason for this discrepancy is that $e(.)$ and $\hat{e}(.)$ both estimate the sizes of flows *that have been sampled* by the algorithm, which are not representative of the entire population passing through the router. Since longer flows are more likely to be selected by residual sampling, (115) severely overestimates their fraction and thus skews the PMF towards the tail.

Denote by M_i the number of sampled flows with $R_L = i$ and define a new estimator:

$$\tilde{q}_i = \frac{M_i - (1-p)M_{i+1}}{Mp + (1-p)M_1}. \tag{116}$$

Using the general model of residual-geometric sampling derived earlier in the chapter, we prove that $\tilde{q}_i$ tends to f_i in probability as $M = \sum_i M_i \to \infty$ and derive the amount of error $|\tilde{q}_i - f_i|$ for finite M. We also provide asymptotically unbiased estimators for the total number of flows n:

$$\tilde{n} = M + (1-p)M_1/p \tag{117}$$

and the number of flows n_i with exactly i packets:

$$\tilde{n}_i = (M_i - (1-p)M_{i+1})/p, \tag{118}$$

where we prove that $\tilde{n}/n \to 1$ and $\tilde{n}_i/n_i \to 1$, both in probability, as $M \to \infty$. We call the resulting combination (114), (116)-(118) *Discrete ResIDual-based Estimators* (D-RIDE).

1.3 Implementation and Evaluation

We finish the chapter by discussing an efficient implementation of the above algorithms and evaluating their accuracy/performance using several Internet traces. Prior work has not discussed how residual sampling should be implemented or its overhead in steady-state, which prompts a fairly detailed exposition below.

We assume D-RIDE uses a chain-linked hash table of size K, which keeps individual flow counters. Each linked list is sorted according to the flow ID and is traversed linearly until a match is found or an ID larger than the one being sought is encountered. Keeping the list sorted (as opposed to FIFO) reduces the lookup delay by half for flows not already in the table. To reduce RAM overhead, we remove flows from the table if they have completed (i.e., FIN, RST packets detected) or if no packets from these flows arrive within some timeout τ. To keep the overhead manageable, the removal process is run over the entire table on the timescale of seconds or even minutes.

As before, assume that the router sees a total of n flows in window $[0, T]$. Then, denote by $N(t)$ the number of *active* flows at time t and by $M(t)$ the number of them sampled by the router. It then follows that the average memory consumption at time t is $W(t) = K + E[M(t)]$ and the average lookup delay is $D(t) = 1 + E[M(t)]/2K$ operations. Under certain stationarity assumptions, we obtain a simple result on $E[M(t)]$ and show that even as the total number of flows $n \to \infty$, both RAM usage and overhead remain constant for any $p > 0$ as long as *flow density* $\rho = n/T$ is bounded.

We then explore how to satisfy the tradeoff between three design objectives – memory consumption, processing speed, and accuracy – using parameters K and p. Given upper bounds on memory usage W_0 and per-packet processing delay D_0, we

propose a technique for deciding K based on the above analysis such that $W \leq W_0$ and $D \leq D_0$ are satisfied, while maximizing p at the same time (i.e., achieving the best accuracy within the constraints).

We finish this chapter by evaluating D-RIDE with real Internet traces obtained from NLANR [41] and CAIDA [7]. Our experiments reveal that the proposed algorithm produces very accurate estimation of flow metrics and thus allows one to perform more aggressive sampling (i.e., smaller probability p) of the monitored traffic. With $p = 0.01$, we find that $E[M(t)]$ is $40-4000$ times smaller than n and $3-100$ times smaller than M, where most lookups require 1-2 operations with $K = E[M(t)]$. We also discover in the experiments with small traces that D-RIDE does not degrade significantly in terms of accuracy even for moderate sample sizes, which makes it suitable for monitoring individual customer networks and certain protocols.

The remainder of the chapter is organized as follows. We review prior work on traffic monitoring in Section 2. We then develop a probabilistic model for residual-geometric sampling in Section 3, analyze previous methods in Section 4, and propose two new estimators in Section 5. We explore the implementation of the suggested framework in Section 6, evaluate its performance in Section 7, and conclude the chapter in Section 8.

2 Related Work

In this section, we review several sampling algorithms in the area of traffic monitoring. In particular, we classify existing work into two categories: *packet sampling* and *flow sampling*, where the former makes *per-packet* and the latter *per-flow* decisions to sample incoming traffic.

2.1 Packet Sampling

Sampled NetFlow (SNF) [43] is a widely used technique in which incoming packets are sampled with a fixed probability p. The general goal of SNF is to obtain the PMF of flow sizes; however, [24] shows that it is impossible to accurately recover the original flow-size distribution from sampled SNF data. Estan *et al.* [15] propose *Adaptive NetFlow* (ANF), which adjusts the sampling probability p according to the size of the flow table; however, ANF's bias in the sampled data is equivalent to that in SNF and is similarly difficult to overcome in practice.

Instead of using one uniform probability for all flows as in [15], [43], another direction in packet sampling is to compute $p_i(c)$ for each flow i based on its currently observed size c. This approach has been studied by two independent papers, *Sketch-Guided Sampling* (SGS) [32] and *Adaptive Non-Linear Sampling* (ANLS) [25]. A common feature of these two methods is to sample a new flow with probability 1 and then monotonically decrease $p_i(c)$ as c grows. Both methods must maintain a counter for each flow present in the network and are difficult to scale due to the high RAM/CPU usage.

2.2 Flow Sampling

In *flow thinning* [24], each flow is sampled independently with probability p and then all packets in sampled flows are counted. Hohn *et al.* [24] show that flow thinning is able to accurately estimate the flow size distribution; however, this method typically misses $1-p$ percent of elephant flows and thus does not support applications such as usage-based accounting and traffic engineering [10], [16], [17], [18], [44]. For highly skewed distributions with a few extremely large flows and many short ones (which is typical for Internet links), this method may also take a long time to converge.

To address these problem of flow thinning, Estan *et. al.* [16] introduce a size-dependent flow sampling algorithm called *Sample-and-Hold* (S&H), which is proposed to identify elephant flows. For each packet from a new flow, the algorithm creates a flow counter with probability p; once a flow is sampled, all of its subsequent packets are then counted. It is easy to verify that S&H samples a flow with size L with probability $1-(1-p)^L$, which quickly approaches 1 as L grows.

Another direction of size-dependent flow sampling has been explored by Duffield *et al.* in [9], [10], [12], which present another size-dependent flow measurement method called *Smart Sampling*. Their approach selects each flow of size L with probability $p(L) = \min(1, L/z)$, where z is some constant. Since this method requires flow size L before deciding whether to sample it or not, it can only be applied off-line.

Kompella *et. al.* [28] examine a method called *Flow Slicing* (FS), which combines SNF and S&H with a variant of smart sampling. Other non-sampling methods include *exact counting* [42], [47], [54], [67] and *lossy counting* [30], [38], which are orthogonal to our work.

3 Underlying Model

In this section, we build a general probabilistic model of residual-geometric sampling and establish the necessary analytical foundation for this work.

3.1 Definitions

Consider a sequence of packets traversing a router and assume that residual-geometric sampling checks each packet's flow identifier x in some RAM table. If x is found in the table, the corresponding counter is incremented by 1; otherwise, with probability p a new entry for x is created in the table (with counter value 1) and with probability

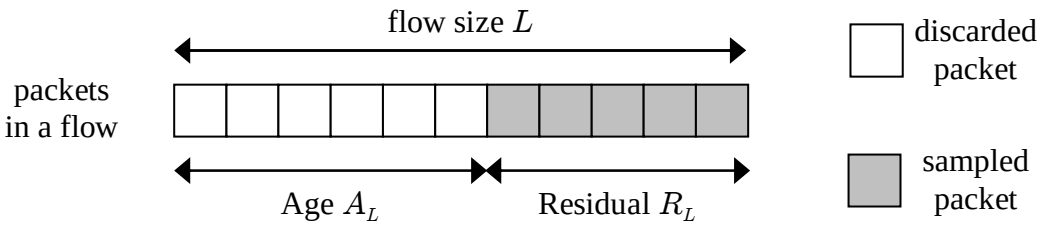


Fig. 30. Residual-geometric sampling of a flow with size L.

$1-p$ the packet is ignored.

To model this process, we first need several definitions. Assume that flow sizes are i.i.d. random variables and define *geometric age* A_L to be the number of packets discarded from the front of a flow with size L before it is sampled (see Figure 30). Let G be a shifted geometric random variable with success probability p, i.e., $P(G = j) = (1-p)^j p$. It thus follows that A_L is simply:

$$A_L = \min(G, L). \tag{119}$$

Now define *geometric residual* R_L to be the final counter value of a flow of size L conditioned on the fact that it has been sampled (i.e., $A_L < L$):

$$R_L = L - A_L, \tag{120}$$

which is also illustrated in Figure 30. From the perspective of traffic monitoring in this chapter, geometric residual R_L is the only quantity collected during measurement and available to an estimation algorithm. We study its distribution next.

3.2 Geometric Residual

Assume that L has a PMF $f_i = P(L = i)$, where $i = 1, 2, \ldots$, and denote by $p_s = P(A_L < L)$ the probability that a random flow is sampled. Then, we have the following result.

Lemma 4. *The probability p_s that a flow is selected by residual-geometric sampling is given by:*

$$p_s = E[1-(1-p)^L] = 1 - \sum_{i=1}^{\infty} f_i(1-p)^i. \tag{121}$$

Proof. Observe that for a fixed flow size $L = l$, we have $P(A_l < l) = 1-(1-p)^l$. Unconditioning L, we immediately get (121). □

Next, let $h_i = P(R_L = i)$ be the PMF of geometric residual R_L. The following lemma expresses h_i in terms of f_i.

Lemma 5. *The PMF of geometric residual R_L is:*

$$h_i = \frac{p\sum_{j=i}^{\infty} f_j(1-p)^{j-i}}{p_s}, \tag{122}$$

where p_s is given by (121).

Proof. Using (120), we have:

$$\begin{aligned} h_i = P(R_L = i) &= P(L - A_L = i | A_L < L) \\ &= \frac{P(L - A_L = i \cap A_L < L)}{p_s}, \end{aligned} \tag{123}$$

where $p_s = P(A_L < L)$. Substituting (119) into (123), we get:

$$h_i = \frac{P(L - G = i \cap G < L)}{p_s}. \tag{124}$$

Since $L - G = i \geq 1$, (124) becomes:

$$h_i = \frac{P(L-G=i)}{p_s} = \frac{\sum_{j=0}^{\infty} P(G = j-i)f_j}{p_s}, \tag{125}$$

which gives the desired result in (122) by substituting the PMF of G into (125). □

The result of Lemma 5 is fundamental as most of the results of this chapter are conveniently derived from (122).

3.3 Fixed Flow Size

We next analyze a special case of residual sampling where the original flow size is fixed at $L = l$. Note that residuals are now R_l instead of R_L since the original flow size is no longer a random variable. Recall that the goal of single-flow size estimation is to obtain l from R_l for each sampled flow. The next corollary follows from (122) and gives the distribution and expectation of geometric residual R_l.

Corollary 8. *Given flow size $L = l$, the PMF of R_l is given by:*

$$P(R_l = i) = \frac{(1-p)^{l-i}p}{1-(1-p)^l}, \tag{126}$$

and its expectation is:

$$E[R_l] = \frac{l}{1-(1-p)^l} + 1 - 1/p. \tag{127}$$

Proof. For $L = l$, we have $f_l = 1$ and $f_i = 0$ for all $i \neq l$. Writing $p_s = 1-(1-p)^l$, we get from (122):

$$P(R_l = i) = \frac{\sum_{j=i}^{\infty} f_j (1-p)^{j-i}p}{1-(1-p)^l} = \frac{(1-p)^{l-i}p}{1-(1-p)^l}, \tag{128}$$

which is exactly (126).

We next derive expectation $E[R_l]$, which can be expanded into:

$$E[R_l] = E[l - A_l | A_l < l] = l - E[G|G < l]. \tag{129}$$

Recall that for any non-negative discrete random variable Y taking values over the integer set $\{0, 1, \ldots\}$, its expectation is given by $E[Y] = \sum_{y=0}^{\infty} P(Y > y)$. It thus

follows that (129) reduces to:

$$E[R_l] = l - \sum_{j=0}^{l-1} P(G > j | G < l)$$
$$= \sum_{j=0}^{l-1} P(G \leq j | G < l) = \frac{\sum_{j=0}^{l-1} P(G \leq j)}{P(G < l)}. \quad (130)$$

Substituting $P(G \leq j) = 1 - (1-p)^{j+1}$ into (130), we have:

$$E[R_l] = \frac{\sum_{j=0}^{l-1} [1 - (1-p)^{j+1}]}{1 - (1-p)^l}$$
$$= \frac{l - (1-p)(1-(1-p)^l)/p}{1-(1-p)^l}, \quad (131)$$

which can be simplified to (127). □

Next, we apply the results derived in this section to analyze existing estimation methods that have been proposed for residual-geometric sampling.

4 Analysis of Existing Methods

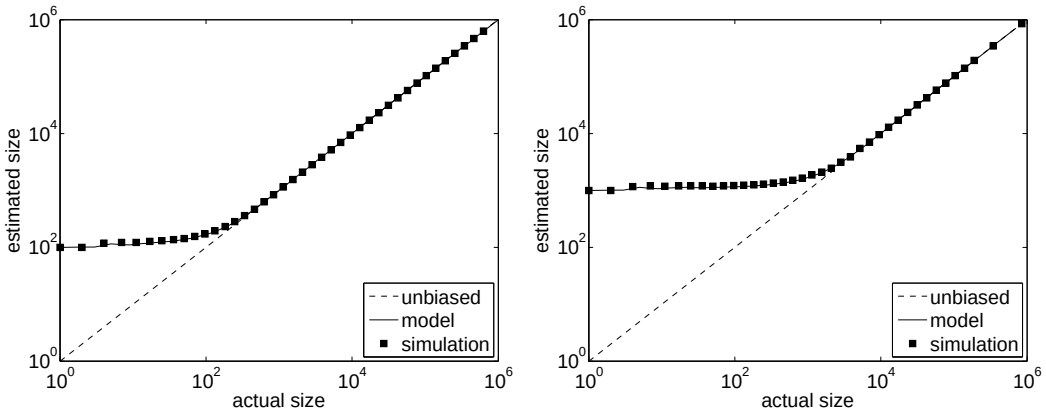
In this section, we examine prior approaches [16], [28] to estimating single-flow usage and whether their results can be generalized to recover the PMF of L.

4.1 Single-Flow Usage

To evaluate single-flow estimators, we use the following definition that is commonly used in statistics [4].

Definition 11. *Estimator $e(R_l)$ is called* unbiased *if $E[e(R_l)] = l$ for all $l \geq 1$.*

Unbiased estimation is a key property of an estimator as it allows accurate estimation of the total contribution from a sufficiently large pool of flows (e.g., one customer network). However, since large flows are typically rare, one commonly faces



(a) $p = 0.01$ (b) $p = 0.001$

Fig. 31. Expectation of estimator (132) in simulations and its model (133).

an additional requirement to estimate their size *with just a single sample* $e(R_l)$, which is formalized in the next definition.

Definition 12. *Estimator* $e(R_l)$ *is called* elephant-accurate *if* $e(R_l)/l \to 1$ *in mean-square as* $l \to \infty$.

Elephant-accuracy guarantees that the amount of relative error between $e(R_l)$ and l decays to zero as $l \to \infty$. As before, suppose that a flow of size L produces a counter with value R_L. Recall that [16], [28] suggest the following estimator:

$$e(R_L) = R_L + 1/p - 1, \tag{132}$$

where p is the probability of residual-geometric sampling. The next result directly follows from (127).

Theorem 15. *Expectation* $E[e(R_l)]$ *is given by:*

$$E[e(R_l)] = \frac{l}{1-(1-p)^l}. \tag{133}$$

Note that (133) indicates that (132) is generally biased, especially when lp is

small. Indeed, for $lp \approx 0$, we have $1-(1-p)^l \approx lp$ and $E[e(R_l)] \approx 1/p$ regardless of l, which shows that in such cases $E[e(R_l)]$ *carries no information about the original flow size*. However, as $l \to \infty$, it is straightforward to verify that the bias in $e(R_l)$ vanishes exponentially, which is consistent with the analysis in [28], which has only considered the case of $l \to \infty$.

To see the extent of bias in (132) and verify (133), we apply residual-geometric sampling to flows of size l ranging from 1 to 10^6, feed the measured sizes to (132), and average the result after 1000 iterations for each l. Figure 31 plots the obtained $E[e(R_l)]$ along with model (133). The figure indicates that (133) indeed captures the bias and that (132) tends to over-estimate the size of short flows *even in expectation*, where smaller sampling probability p leads to more error.

To quantify the error of individual values $e(R_l)$ in estimating flow size l and to understand elephant-accuracy, denote by $Y_l = e(R_l)/l$ and define the *Relative Root Mean Square Error* (RRMSE) to be:

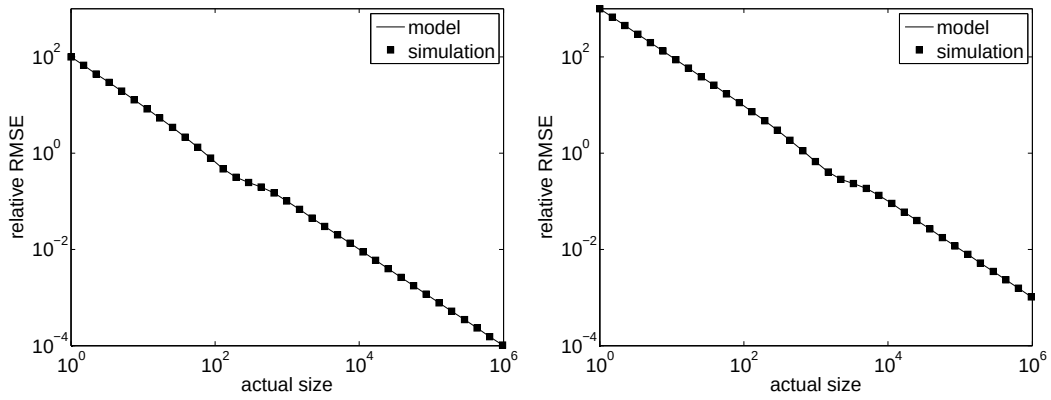
$$\delta_l = \sqrt{E[(Y_l - 1)^2]}. \tag{134}$$

Note that $\delta_l \to 0$ indicates that $Y_l \to 1$ in mean-square and thus implies elephant-accurate estimation. The next result derives δ_l in closed form. We omit the derivations due to limited space.

Theorem 16. *The RRMSE of* (132) *is given by:*

$$\delta_l = \sqrt{\frac{1-p-l(l-1)p^2(1-p)^l-(1-p)^{l+1}}{l^2p^2(1-(1-p)^l)}}. \tag{135}$$

Observe from (135) that for flows with size $l = 1$, the relative error is $\sqrt{1-p}/p$, but as $l \to \infty$, $\delta_l \to 0$ and the estimator is elephant-accurate. Figure 32 plots (135) against simulations, indicating a close match. The figure also shows that the RRMSE



(a) $p = 0.01$ (b) $p = 0.001$

Fig. 32. RRMSE of (132) in simulations and its model (135).

starts from $1/p$ and decreases towards zero as $\Theta(1/l)$ as $l \to \infty$.

4.2 Flow-Size Distribution

We now investigate whether $e(R_L)$ defined in (132) can be used to estimate the actual flow-size distribution $\{f_i\}_{i=1}^{\infty}$. Denote by $q_i = P(e(R_L) = i)$ the PMF of estimated sizes among the sampled flows. To understand our objectives with approximating the PMF of L, the following definition is in order.

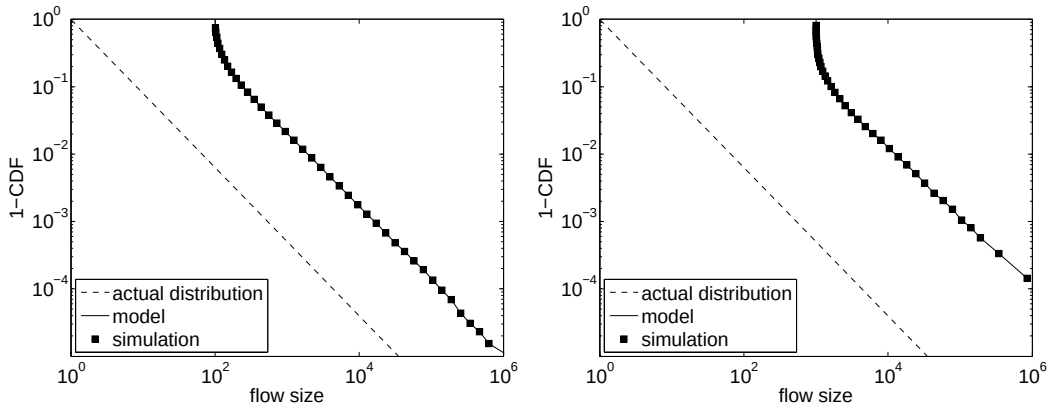
Definition 13. *An estimator* $\{q_i\}_{i=1}^{\infty}$ *of PMF* $\{f_i\}_{i=1}^{\infty}$ *is called* asymptotically unbiased *if* q_i *converges in probability to* f_i *for all* i *as the number of sampled flows* $M \to \infty$.

The next theorem follows directly from (122).

Theorem 17. *The PMF of flow sizes estimated from* (132) *is given by:*

$$q_i = \frac{\sum_{j=y(i)}^{\infty} f_j (1-p)^{j-y(i)} p}{p_s}, \tag{136}$$

where $y(i) = \lceil i + 1 - 1/p \rceil$ *and* p_s *is in* (121).



(a) $p = 0.01$ (b) $p = 0.001$

Fig. 33. Distribution $\{q_i\}$ in simulations and its model (136).

The result in (136) indicates that each q_i is different from f_i regardless of the sampling duration and thus cannot be used to approximate the flow-size distribution. We verify (136) with a simulated packet stream with 5M flows, where flow sizes follow a power-law distribution $P(L \leq i) = 1 - i^{-\alpha}$ for $i = 1, 2, \ldots$ and $\alpha = 1.1$. Figure 33 plots the CCDF of random variable $e(R_L)$ obtained from simulations as well as model (136), both in comparison to the tail of the actual distribution. The figure shows that (136) accurately predicts the values obtained from simulations and that PMF $\{q_i\}$ is indeed quite different from $\{f_i\}$.

So far, our study of existing methods in residual-geometric sampling has shown that they are not only generally biased, but also unable to recover the flow-size distribution from residuals R_L. This motivates us to seek better estimation approaches, which we perform next.

5 D-RIDE

This section proposes a family of algorithms called *Discrete ResIDual-based Estimators* (D-RIDE), proves their accuracy, and verifies them in simulations.

5.1 Single-Flow Usage

For estimating individual flow sizes, we first consider an estimator directly implied by the result in (127). Notice that solving (127) for l and expressing flow size l in terms of $E[R_l]$, we get:

$$l = u - \frac{1}{\log(1-p)} W\Big(u(1-p)^u \log(1-p)\Big), \tag{137}$$

where $u = E[R_l] + 1/p - 1$ and $W(z)$ is Lambert's function (i.e., a multi-valued solution to $We^W = z$) [8]. Thus, a possible estimator can be computed from (137) with $E[R_l]$ replaced by the measured value of geometric residual R_l. However, there are two reasons that (137) is a bad estimator of flow sizes. First, Lambert's function $W(z)$ has no closed form solution and has to be numerically solved using tools such as Matlab. Second, it can be verified (not shown here for brevity) that (137) is not an unbiased estimator. Instead, we define a new estimator:

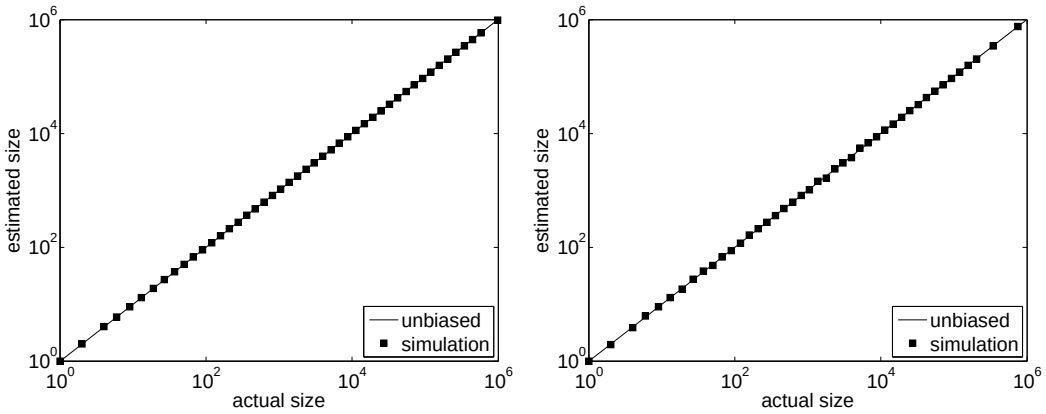
$$\hat{e}(R_l) = R_l + 1/p - 1 - (1-p)^{R_l}/p. \tag{138}$$

and next show that it is unbiased.

Lemma 6. *Estimator* $\hat{e}(R_l)$ *in* (138) *is unbiased, i.e.,*

$$E[\hat{e}(R_l)] = l. \tag{139}$$

Proof. We prove (139) by deriving such function $\hat{e}(.)$ that satisfies $E[\hat{e}(R_l)] = l$. First,



(a) $p = 0.01$ (b) $p = 0.001$

Fig. 34. Expectation of estimator (138) in simulations.

it follows from (126) that expectation $E[\hat{e}(R_l)]$ can be rewritten as:

$$E[\hat{e}(R_l)] = \frac{\sum_{j=1}^{l} \hat{e}(j)(1-p)^{l-j}p}{1-(1-p)^l}. \tag{140}$$

For $E[\hat{e}(R_l)] = l$ to hold, we must have:

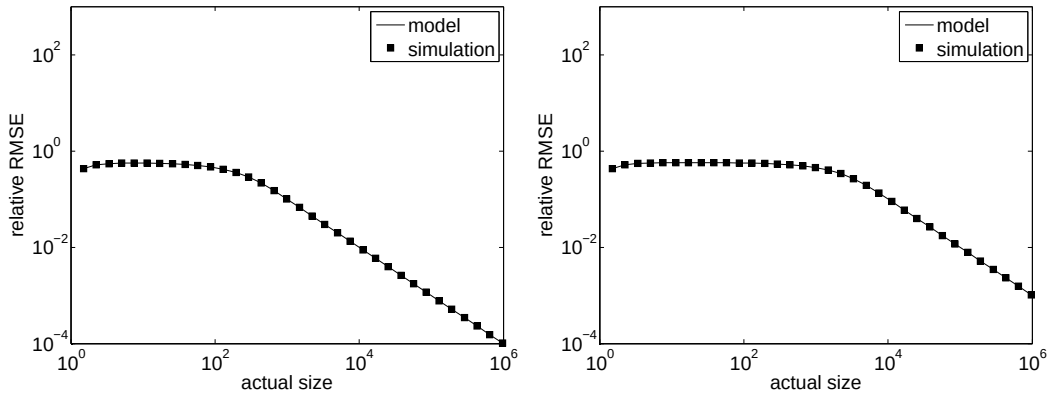
$$\sum_{j=1}^{l} \hat{e}(j)(1-p)^{-j} = \frac{l\left(1-(1-p)^l\right)}{p(1-p)^l}. \tag{141}$$

Writing (141) twice for l and $l-1$ and subtracting the two equations from each other, we get:

$$\hat{e}(l)(1-p)^{-l} = \frac{1+p(l-1)-(1-p)^l}{p(1-p)^l}. \tag{142}$$

Simplifying (142) and replacing l with R_l immediately gives (138). □

We plot in Figure 34 simulation results obtained from (138). The figure indicates that $\hat{e}(R_l)$ accurately estimates actual sizes for all flows in both cases of p. Next, we derive the RRMSE of D-RIDE.



(a) $p = 0.01$ (b) $p = 0.001$

Fig. 35. RRMSE of (138) in simulations and model (143).

Theorem 18. *The RRMSE of* (138) *is given by:*

$$\hat{\delta}_l = \sqrt{\frac{1-p+lp(p-2)(1-p)^l-(1-p)^{2l+1}}{l^2p^2(1-(1-p)^l)}}. \tag{143}$$

It is easy to verify from (143) that D-RIDE has zero RRMSE for $l = 1$ or $l \to \infty$, confirming its elephant-accuracy. We plot $\hat{\delta}_l$ obtained from simulations along with the model in Figure 35, which shows that (143) accurately tracks the actual relative error. From Figures 34-35, it is clear that $\hat{e}(R_l)$ significantly improves the accuracy of estimating small flow sizes compared to $e(R_l)$. In practice, (143) can be used to determine threshold l_0, which leads to desired bounds on error for all $l \geq l_0$ and allows ISPs to use $e(R_l)$ instead of l.

5.2 Flow-Size Distribution

It is worth mentioning that while (138) produces unbiased estimation of flow sizes, $\hat{e}(R_L)$ is not suitable for producing the flow-size distribution as we show below. Denote by $\hat{q}_i = P(\hat{e}(R_L) = i)$ the PMF of $\hat{e}(R_L)$. Then, we have the following result.

Lemma 7. *PMF of $\hat{e}(R_L)$ is given by:*

$$\hat{q}_i = \frac{1}{p_s} \sum_{j=y(i)}^{\infty} (1-p)^{j-y(i)} f_j p, \tag{144}$$

where p_s is in (121)*, function $y(i)$ is:*

$$y(i) = \lceil i + 1 - 1/p - \omega \rceil, \tag{145}$$

and $\omega = W\left(-(1-p)^{i+1-1/p} \log(1-p)\right)$.

Proof. We first solve

$$R_L + 1/p - 1 - (1-p)^{R_L}/p = i, \tag{146}$$

for R_L and express it in terms of i, i.e., $R_L = y(i)$, where $y(i)$ is given by (145), ignoring approximate round-offs to the nearest integer. Combining with (122), we have:

$$\hat{q}_i = P(R_L = y(i)) = h_{y(i)}, \tag{147}$$

where h_i is in (122). This directly leads to (144). □

Notice from (144)-(145) that distribution $\hat{q}_i$ does not even remotely approximate the original PMF f_i. This problem is fundamental since residual sampling exhibits bias towards larger flows and even if we could record L from R_L exactly, the distribution of sampled flow sizes would not accurately approximate that of all flows passing through the router.

We thus explore another technique for estimating the flow-size distribution. Before doing that, we need the next lemma.

Lemma 8. *The flow size distribution f_i can be expressed using the PMF of geometric*

residuals $\{h_i\}$ *given by* (122) *as following:*

$$f_i = \frac{h_i - (1-p)h_{i+1}}{p + (1-p)h_1}. \tag{148}$$

Proof. We first rewrite (122) as:

$$h_i = \frac{p}{p_s(1-p)^i} \sum_{j=i}^{\infty} f_j (1-p)^j, \tag{149}$$

and then subtract $(1-p)h_{i+1}$ from both sides of (149):

$$h_i - (1-p)h_{i+1} = \frac{p}{p_s} f_i. \tag{150}$$

It immediately follows that f_i is given by:

$$f_i = \frac{p_s(h_i - (1-p)h_{i+1})}{p}, \tag{151}$$

Notice that p_s in (121) is a function of f_i's, which are unknown from the measurement perspective. The last step of the proof is to express p_s in terms of known quantities $\{h_i\}$, which can be accomplished by applying the normalization condition $\sum_{i=1}^{\infty} f_i = 1$. It is easy to verify that:

$$\sum_{i=1}^{\infty} h_i = 1, \qquad \sum_{i=1}^{\infty} h_{i+1} = 1 - h_1, \tag{152}$$

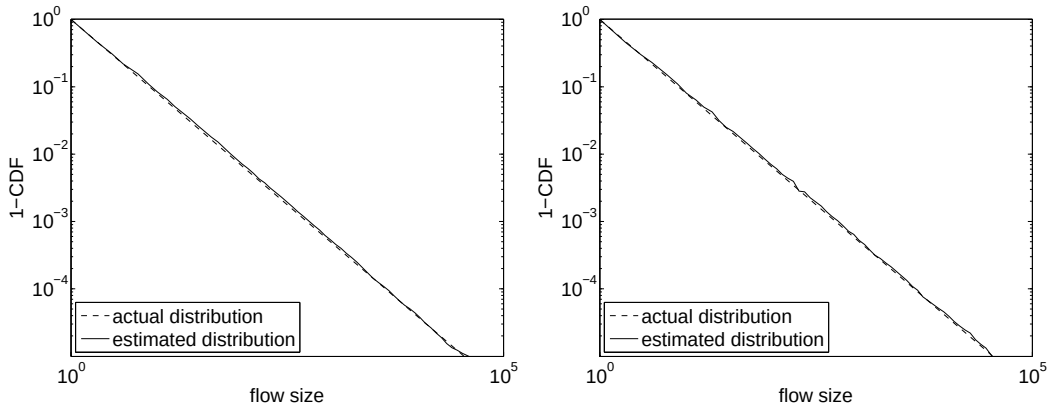
Then, summing up both sides of (151) for i from 1 to infinity gives us:

$$p_s = \frac{p}{p + (1-p)h_1}. \tag{153}$$

Substituting (153) into (151) establishes the desired result in (148). □

The result in (148) leads a new estimator for the flow-size distribution:

$$\tilde{q}_i = \frac{M_i - (1-p)M_{i+1}}{Mp + (1-p)M_1}, \tag{154}$$



(a) $p = 0.01,\ M = 194,208$ (b) $p = 0.001,\ M = 26,233$

Fig. 36. Estimator (154) in simulations.

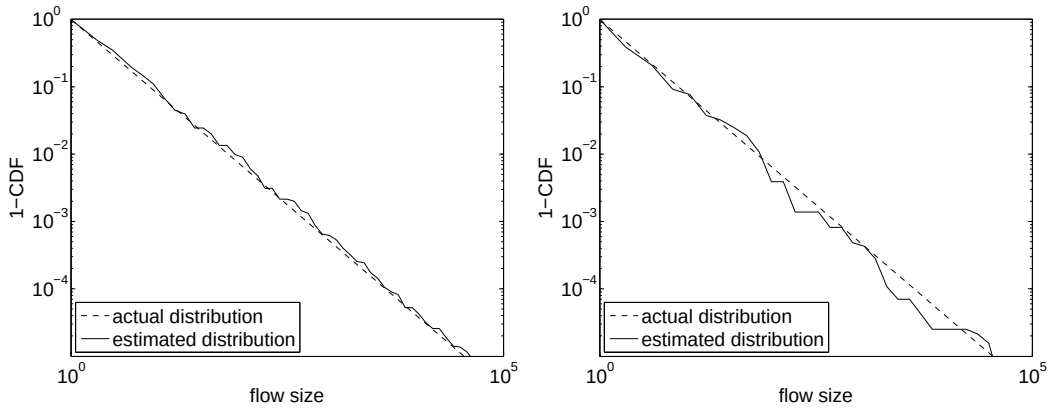
where M is the total number of sampled flows and M_i is the number of them with geometric residual equal to i. Since $M_i/M \to h_i$ in probability as $M \to \infty$ (from the weak law of large numbers), we immediately get the following result.

Corollary 9. *The estimator in* (154) *is asymptotically unbiased.*

We next verify the accuracy of $\tilde{q}_i$ in simulations with 5M flows in the same setting as in the previous section. We plot in Figure 36 the CCDF estimated from (154) along with the actual distribution. The figure shows that $\tilde{q}_i$ accurately follows the actual distribution for both cases of p.

5.3 Convergence Speed

We next examine the effect of sample size M on the convergence of estimator $\tilde{q}_i$. To illustrate the problems arising from small M, we study (154) with $p = 10^{-4}$ and 10^{-5} in simulations with the same 5M flows. The estimator obtained $M = 3,090$ flows for $p = 10^{-4}$ and just $M = 337$ for $p = 10^{-5}$. Figure 37 indicates that while the estimated curves under both choices of p still approximate the trend of the original



(a) $p = 10^{-4}$, $M = 3,090$ (b) $p = 10^{-5}$, $M = 337$

Fig. 37. Estimator (154) in simulations with very small p.

distribution, they exhibit different levels of noise. As the next result indicates, small p leads to a small sample size M and thus more noise in the estimated values.

Corollary 10. *Suppose that M flows are selected by residual-geometric sampling from a total of n flows. Then, the expected value of M is given by:*

$$E[M] = np_s = nE[1 - (1-p)^L]. \tag{155}$$

To shed light on the choice of proper p for residual-geometric sampling, we show how to determine the minimum M that would guarantee a certain level of accuracy in $\tilde{q}_i$. Define $\tilde{h}_i = M_i/M$ to be an estimate of $h_i = P(R_L = i)$. The next lemma follows from Lemma 8 and Corollary 9 and indicates that the accuracy of $\tilde{q}_i$ directly depends on whether $\tilde{h}_i$ approximates h_i accurately.

Lemma 9. *Suppose that $|\tilde{h}_j - h_j| \leq \eta h_j$ holds with probability $1 - \xi$ for $j \in [1, i+1]$ and small constants η and ξ. Then, there exists a constant ζ:*

$$\zeta = \frac{\eta(p + 2\eta(1-p)h_1)}{p + (1-p)(1-\eta)h_1} \tag{156}$$

such that $\zeta \to 0$ *as* $\eta \to 0$ *and* $P(|\tilde{q}_i - f_i| \leq \zeta f_i) = 1 - \xi$.

Proof. We prove the result by deriving ζ that satisfies $|\tilde{q}_i - f_i| \leq \zeta f_i$ given that $|\tilde{h}_j - h_j| \leq \eta h_j$. From (148) and (154), we have:

$$|\tilde{q}_i - f_i| = \frac{|a_1|}{a_2}, \tag{157}$$

where

$$\begin{aligned} a_1 = p(\tilde{h}_i - h_i) + p(1-p)(\tilde{h}_{i+1} - h_{i+1}) + (1-p)\times \\ (h_1\tilde{h}_i - \tilde{h}_1 h_i) + (1-p)^2(h_1\tilde{h}_{i+1} - \tilde{h}_1 h_{i+1}), \end{aligned} \tag{158}$$

and

$$a_2 = (p + (1-p)h_1)(p + (1-p)\tilde{h}_1). \tag{159}$$

From the condition $|\tilde{h}_j - h_j| \leq \eta h_j$, we bound $|a_1|$ and a_2 as follows:

$$\begin{aligned} |a_1| &\leq \eta p h_i + 2\eta(1-p)h_1 h_i + \eta p(1-p)h_{i+1} \\ &\quad + 2\eta(1-p)^2 h_1 h_{i+1} \\ &= \eta(h_i + (1-p)h_{i+1})(p + 2\eta(1-p)h_1), \end{aligned} \tag{160}$$

and

$$a_2 \geq (p + (1-p)h_1)(p + (1-p)(1-\eta)h_1). \tag{161}$$

It thus follows from (148) and (160)-(161) that $|\tilde{q}_i - f_i| \leq \zeta f_i$, where constant ζ is given by:

$$\zeta = \frac{\eta(p + 2\eta(1-p)h_1)}{p + (1-p)(1-\eta)h_1}, \tag{162}$$

and that $\zeta \to 0$ as $\eta \to 0$. □

Next, we obtain a bound on M from the requirement that $\tilde{h}_i$ be bounded in probability within a given range $[h_i(1-\eta), h_i(1+\eta)]$.

Theorem 19. *For small constants η and ξ, $|\tilde{h}_i - h_i| \leq \eta h_i$ holds with probability $1-\xi$ if sample size M is no less than:*

$$M \geq \frac{(1-h_i)}{h_i \eta^2} \left(\Phi^{-1}(1-\xi/2)\right)^2, \tag{163}$$

where $\Phi(x)$ is the CDF of the standard Gaussian distribution $\mathcal{N}(0,1)$.

Proof. Notice that $\tilde{h}_i$ can be approximated by a Gaussian random variable with mean $\mu_i = h_i$ and variance $\sigma_i^2 = h_i(1-h_i)/M$. Define

$$Z = \frac{\tilde{h}_i - \mu_i}{\sigma_i}. \tag{164}$$

Notice that Z is a standard Gaussian random variable. It follows that:

$$P(|Z| \leq z) = 2\Phi(z) - 1, \tag{165}$$

where $\Phi(.)$ is the CDF function of the standard Gaussian distribution $N(0,1)$. Therefore, we establish that:

$$P(|\tilde{h}_i - h_i| \leq z\sigma_i) = 2\Phi(z) - 1. \tag{166}$$

We can guarantee target accuracy by setting $z\sigma_i = \eta\phi_i$ and $2\Phi(z) - 1 = 1 - \xi$, which gives the following equality:

$$\frac{\eta h_i}{\sigma_i} = \Phi^{-1}(1-\xi/2). \tag{167}$$

Substituting $\sigma_i = \sqrt{h_i(1-h_i)/M}$ into the above equation and solving for M, we obtain (163). □

For example, to bound $\tilde{h}_i$ within 10% percent of h_i (i.e., $\eta = 0.1$) with probability

$1-\xi = 95\%$ for all $h_i \geq 10^{-2}$, the following must hold:

$$M \geq \frac{(1-10^{-2}) \times 1.96^2}{10^{-2} \times 0.1^2} \approx 3.8 \times 10^4, \tag{168}$$

which indicates that $M = 38$K flows must be sampled to achieve target accuracy. If we reduce η to 1%, increase $1-\xi$ to 99%, and require the approximation to hold for all $h_i \geq 10^{-3}$, then M must be at least 66M flows. Converting η into ζ using (156), one can establish similar bounds on the deviation of $\tilde{q}_i$ from f_i.

5.4 Estimation of Other Flow Metrics

Besides flow sizes and the flow size distribution, D-RIDE also provides estimators for the total number of flows and the number of them with size i. Before introducing these estimators, we need the next lemma.

Lemma 10. *The expected number of flows with sampled residuals $R_L = i$ is:*

$$E[M_i] = E[M]h_i = nh_i p_s, \tag{169}$$

where h_i is the PMF of geometric residuals and p_s is given by (121).

Based on (169), we next develop two estimators and prove their accuracy. Define $\tilde{n}$ to be an estimator of the total number of flows n observed in the measurement window $[0, T]$:

$$\tilde{n} = M + (1-p)M_1/p. \tag{170}$$

and $\tilde{n}_i$ to be an estimator of the number of flows n_i with size i:

$$\tilde{n}_i = (M_i - (1-p)M_{i+1})/p. \tag{171}$$

Then, the next result shows that both of these estimators are asymptotically unbiased.

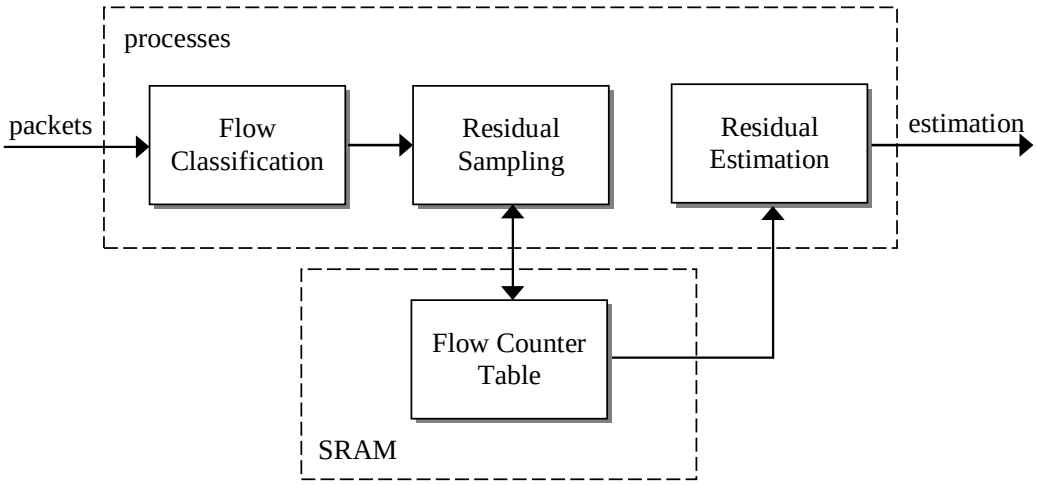


Fig. 38. The D-RIDE framework.

Lemma 11. *Ratios* $\tilde{n}/n$ *and* $\tilde{n}_i/n_i$ *converge to* 1 *in probability as* $M \to \infty$.

Note that [28] provided a similar estimator as (170) and proved $E[\tilde{n}] = n$ using a different approach from ours; however, our results are stronger as they show convergence in probability and additionally address estimation of n_i. Simulations verifying the accuracy of (170)-(171) are omitted for brevity.

6 Implementation

In this section, we implement D-RIDE and examine its memory consumption and processing speed.

6.1 General Structure

Figure 38 illustrates a framework that implements the various D-RIDE algorithms. The framework contains three processes — flow classification, residual-geometric sampling, and estimation — as well as one data structure containing the flow counter table.

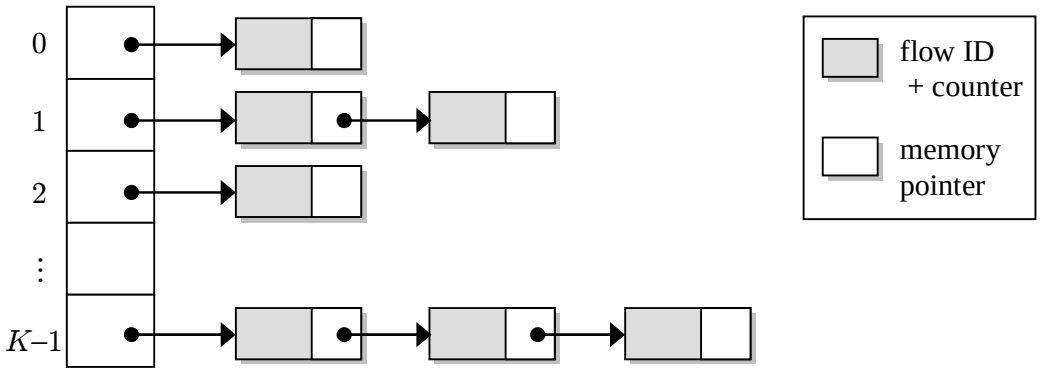


Fig. 39. Illustration of a chained hash table for maintaining flow counters.

Flow classification processes each incoming packet for flow ID and then forwards it to residual-geometric sampling. For each flow ID x arriving from flow classification, *residual-geometric sampling* first checks if the flow table has an existing entry for x and increment the counter by 1; if an entry does not exist, it is created with probability p and its counter is initialized to 1. The geometric estimation process collects counter values from the flow table and then uses D-RIDE to estimate flow statistics.

The flow table keeps a mapping between flow IDs and associated counters. The table supports three operations: 1) $lookup(x)$ to retrieve the record of flow x; 2) $add(x)$ to insert a new entry for flow x in the table with the initial counter value 1; and 3) $increment(x)$ to add 1 to the counter of flow x. We display in Figure 39 an implementation of the counter table, which is based on a *chained hash table*. Assume a hash function $hash(x)$ that produces an integer value in $[0, 1, \ldots, K-1]$. We assume that the generated hash values are uniformly distributed within interval $[0, K-1]$ and the implementation of function $hash(.)$ is fast enough. Efficient hardware hash functions can be found in [48].

We maintain an array A of size K and each entry $A[k]$ points to a liked list that keeps the set of flows whose IDs have the same hash value k. Each node in the

list contains two fields: 1) flow data that keeps the flow ID, the packet counter, and the timestamp of the last packet; and 2) a pointer to the next node. An important element of our algorithm is to ensure that the table only keeps *active* flows, which is accomplished by periodic sweeps through the table and removal of all flows that have completed using FIN/RST packets or have been idle for longer than τ time units. Removal means either saving flow information to disk (single-flow usage) or aggregating statistics of departed flows into a PMF table. Operations $add(x)$ and $increment(x)$ automatically modify the timestamps associated with each flow.

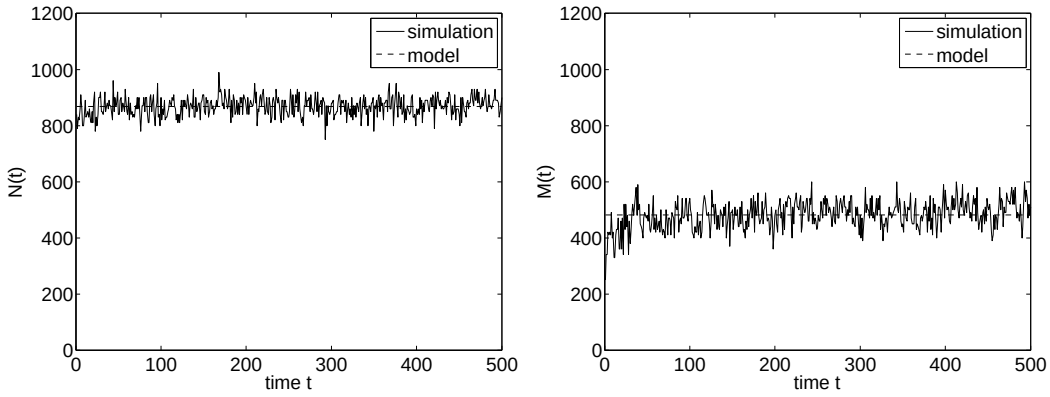
Notice that the flow table is accessed by residual-geometric sampling upon each packet arrival. Therefore, the scalability of the measurement algorithm essentially depends on the access speed to the table. In what follows, we analyze the design of the flow table and quantify its two important properties: memory consumption and processing speed.

6.2 Active Flows

To understand how much benefits removal of dead flows provides to memory consumption, we next derive the expected number of active flows and their fraction sampled by the algorithm. Assume a measurement window $[0, T]$ with some fixed flow density $\rho = n/T$ as $T \to \infty$. For each flow i, let inter-packet delays within the flow be given by a random variable Δ_i, which counts the number of packet arrivals from other flows between adjacent packets of i. Denoting by $\Delta = E[\Delta_i]$, we have the following result.

Lemma 12. *Assuming stationary flow arrivals in* $[0, T]$*, the expected number of active flows* $N(t)$ *at time* t *is given by:*

$$E[N(t)] = \Delta E[L]\rho, \tag{172}$$



(a) alive flows (b) sampled flows

Fig. 40. Verifying models (172) and (173).

where $E[L]$ is the expected flow size.

Our baseline reduction in flow volume comes from geometric sampling in previous sections and reduces the number of flows by a factor $r_1 = n/E[M]$. Now additionally define $r_2 = n/E[N(t)] = T/\Delta E[L]$ and observe that longer observation windows (i.e., larger T), smaller flow sizes (i.e., smaller $E[L]$), and denser arrivals (i.e., smaller Δ) imply more savings of memory. In fact, $T \to \infty$ results in $r_2 \to \infty$ as well assuming the other parameters are fixed. More reduction is possible by geometric sampling. Denote by $M(t)$ the number of sampled flows that are still alive at t and consider the next result.

Lemma 13. *Assuming Δ_i are constant and the flow arrival process is stationary in $[0, T]$, the expected number of sampled active flows at time t is given by:*

$$E[M(t)] = \rho\Delta \left(E[L] - \sum_{i=1}^{\infty} (1-p)^i (1 - F_i) \right), \tag{173}$$

where $F_i = P(L \leq i)$ is the CDF of flow sizes.

Performing a self-check and comparing (172) to (173), observe that $E[M(t)] \leq$

Table V. Comparing models (172) and (173) to simulation results

time t	$E[N(t)]$		$E[M(t)]$	
	simulation	model (172)	simulation	model (173)
100	867.1	866.6	487.0	486.8
200	866.4	866.6	487.0	486.8
300	866.7	866.6	486.5	486.8
400	866.3	866.6	486.4	486.8
500	866.9	866.6	486.8	486.8

$E[N(t)]$ and thus the former always results in more reduction in table size. Denote by $r_3 = n/E[M(t)]$ and notice that it increases not only as T grows, but also when p decreases. We evaluate models (172) and (173) in simulations with $1,000$ iterations through window $[0, T]$ with randomly generated flows from the a distribution with flow-size CDF $F_i = 1 - i^{-\alpha}$, where $\alpha = 1.1$ and $p = 0.01$. Figure 40 plots the evolution of $N(t)$ and $M(t)$ along with the expected values computed from the models. Table V compares the models with $E[N(t)]$ and $E[M(t)]$ computed in simulations, where each value is averaged using $1,000$ iterations of the traffic stream. Figure 40 and Table V indicating a close match even though the simulation does not follow the assumptions on constant Δ_i in Lemma 13.

6.3 Memory Consumption

The memory used by the flow table can be divided into two parts: one for the hash table, which contains an array of pointers, and the other for flow records, which are organized in a set of linked lists. Define w_p to be the number of bytes used by each memory pointer and w_f to be that needed for flow counter, timestamp, and flow

ID. Then, the following theorem gives the memory required for the measurement algorithm.

Theorem 20. *The average number of bytes required by D-RIDE in steady-state is:*

$$E[W(t)] = Kw_p + E[M(t)](w_c + w_f), \tag{174}$$

where $E[M(t)]$ is the average number of sampled active flows at time t given by (173).

From (174), observe that for n original flows with a given distribution of L, memory consumption $E[W(t)]$ can be reduced by lowering either $M(t)$ or K. As discussed in the previous section, $M(t)$ cannot be arbitrarily small as it would lead to lower accuracy. At the same time, small K leads to more conflicts in the hash table, longer linked lists, and thus may slow down the sampling process.

6.4 Processing Time

The time spent in processing each packet depends on how linked lists are built. We examine an approach that sorts flow entries of each linked list based on flow IDs. In this approach, function $lookup(x)$ returns a pointer to the entry of flow x if it exists in the table; otherwise, the function returns a pointer to where the new entry should be inserted.

For each packet with flow ID x, we perform the following steps in sequential order: 1) compute the $k = hash(x)$; 2) retrieve the linked-list head pointer $A[k]$ from the hash table; 3) iterate through the linked list until a flow record is matched or a flow with ID larger than x is reached; 4) if x is not found, a new entry for x is created with probability p and inserted to the location returned by $lookup(x)$.

Denote by t_h the time spent in computing a hash, by t_p that of memory access, and by t_c that of each comparison of flow IDs. Define $D(t)$ to be the processing time

Table VI. Constants used in (174) and (175)

RAM constant	value	CPU constant	value
w_p	4B	t_h	12ns
w_f	17B	t_p	9ns
W_0	1.65MB	t_c	3ns
		T_0	24ns

of incoming packets at t. Then, noticing that the expected list length is $E[M(t)]/K$ entries and on average traversal stops in the middle of a list, we have the next result.

Theorem 21. *The expected per-packet processing time is:*

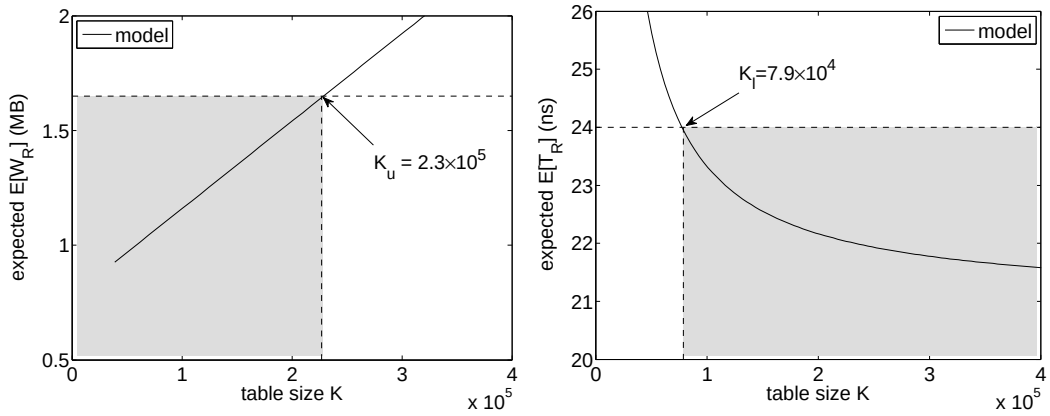
$$E[D(t)] = t_h + t_p + (t_c + t_p)\frac{E[M(t)]}{2K}. \tag{175}$$

The result in (175) indicates that both large hash table size K and small sample size $M(t)$ can contribute to a faster sampling process. We next examine how to properly select K and p to satisfy certain target constraints on $E[W(t)]$ and $E[D(t)]$.

6.5 Tradeoff Analysis

Now, we are ready to explore the design space of constants (K, p) to strike a balance between accuracy and scalability. Suppose that a router requires that $E[W(t)] \leq W_0$ and $E[D(t)] \leq D_0$. Further assume that the number of sampled flows $E[M(t)]$ is known and fixed (i.e., fixed p, window T, and flow-size distribution). Define two constants:

$$K_l = \frac{(t_c + t_p)E[M(t)]}{2(T_0 - (t_h + t_p))}, \tag{176}$$



(a) $E[W(t)]$ (b) $E[D(t)]$

Fig. 41. Tradeoff: (a) memory consumption and (b) processing time with $E[M(t)] = 3.9 \times 10^4$. Gray areas display the acceptable ranges of K.

and

$$K_u = \frac{W_0 - E[M(t)](w_c + w_f)}{w_p}. \tag{177}$$

It thus follows from (174) and (175) that assuming $K_l \leq K_u$, one can choose any value $K \in [K_l, K_u]$ to satisfy the constraints. We show below how by varying p, one can always maximize accuracy while ensuring $K_l \leq K_u$.

To understand this better, consider the following example. Assume that the original traffic contains $n = 10^6$ flows with a power-law distribution $P(L \leq i) = 1 - i^{-1.1}$. With $p = 0.01$, residual-geometric sampling obtains $E[M(t)] = 3.9 \times 10^4$ sampled flows. Table VI gives the constants we use to compute expected memory consumption and processing time from (174) and (175). We also impose the following constraints on memory and delay: $W_0 = 1.65$MB and $D_0 = 24$ns. The memory of 1.65MB can hold an array of about 10^5 flow records, each with a flow ID and a counter. Per-packet processing time must be less than 24ns for OC-768 links with an average packet size of 1 Kbit. Figure 41 illustrates the acceptable ranges of table size

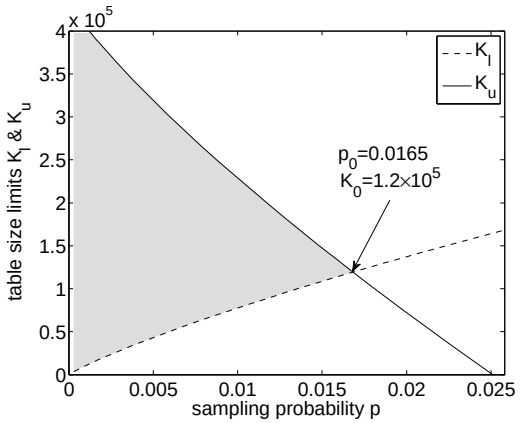


Fig. 42. Lower and upper bounds on table size K with varying probability p. Gray areas display the acceptable range of K and p.

K derived from the models. The figure indicates that table size K can be any value between $K_l = 7.9 \times 10^4$ and $K_u = 2.3 \times 10^5$ (obtained from (176)-(177)) to satisfy the requirements of both memory and processing speed.

Notice that it is possible that for some values of $E[M(t)]$, K_l is larger than K_u and thus the constraints cannot be met. Therefore, we next vary p to see how the choice of K will be affected. Figure 42 plots K_u and K_l as functions of p, where both curves are obtained from the corresponding models. Notice from the figure that as p increases, the interval $[K_l, K_u]$ shrinks to one single point K_0, which is achieved when $p = p_0$. As discussed in the previous section, larger p implies more accurate estimation results since the router sees more flows M in the interval $[0, T]$ and thus estimates h_i more accurately. We therefore call pair (K_0, p_0) *optimal* since it allows memory-delay constraints (W_0, D_0) to be satisfied, while simultaneously maximizing the accuracy of estimation. In this example, we get $p_0 = 0.0165$ and $K_0 = 1.2 \times 10^5$.

Table VII. Reduction in the number of flows using residual sampling with $p = 0.01$ and periodic removal of dead flows

source	trace	#total flows n	#total pkts $nE[L]$	sampling only		removal only		both	
				M	r_1	$E[N(t)]$	r_2	$E[M(t)]$	r_3
NLANR	FRG	1,756,702	131,821,685	117,995	15	21,645	81	2,669	658
	Web	239,174	6,497,894	26,051	9	9,698	24	985	240
	DNS	120,446	292,977	2,073	44	600	152	19	4,797
	NTP	382,489	720,447	4,086	54	3,036	73	77	2,887
CAIDA	LARGE	9,653,609	117,250,415	519,144	19	262,525	37	21,590	447
	MEDIUM	2,317,369	43,837,666	139,316	17	281,137	8	53,903	43
	SMALL	200,910	2,179,574	12,862	16	44,414	5	5,948	34

Table VIII. Performance of D-RIDE implementation with $p = 0.001$ and $K = E[M(t)]$

source	trace	$E[W(t)]$	$E[D(t)]$	#flows			#size-one flows		
				actual	estimated	error	actual	estimated	error
NLANR	FRG	31KB	24.1ns	1,756,702	1,736,261	1.16%	768,742	749,958	2.44%
	Web	10KB	21.4ns	239,174	253,996	6.2%	13,686	13,922	1.72%
	DNS	257B	21ns	120,446	124,176	3.1%	76,607	78,045	1.88%
	NTP	752B	21.1ns	382,489	375,326	1.87%	281,370	279,096	0.8%
CAIDA	LARGE	132KB	28.1ns	9,653,609	9,717,315	0.66%	4,535,449	4,630,037	2.09%
	MEDIUM	341KB	23.7ns	2,317,369	2,278,984	1.66%	1,299,343	1,273,989	1.95%
	SMALL	23KB	21.2ns	200,910	202,604	0.84%	93,575	95,106	1.64%

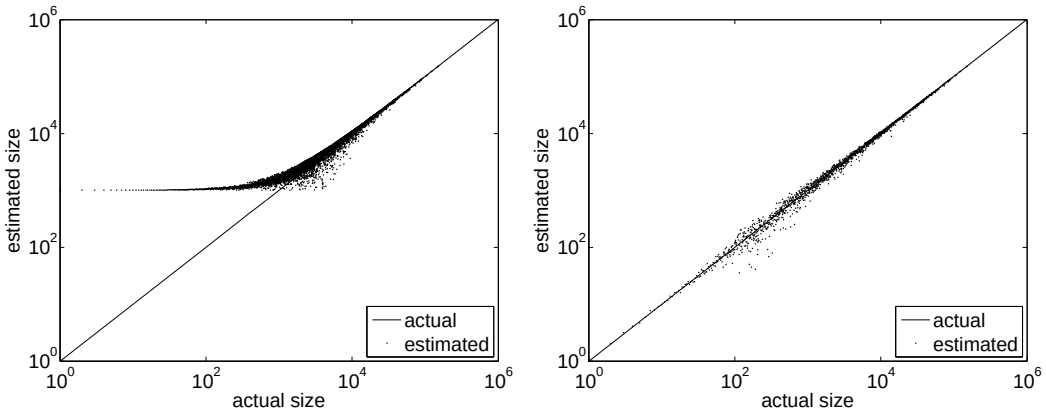
7 Performance Evaluation

In this section, we evaluate our proposed method using several Internet traces in Table VII from NLANR [41] and CAIDA [7]. Trace FRG was collected from a gigabit link between UCSD and Abilene in 2006. We extracted from the NLANR FRG trace additional traces with only Web, DNS, and NTP flows. We also use three traces from CAIDA: LARGE – a one-hour trace from an OC48 link, MEDIUM – a one-minute trace from a OC192 link, and SMALL – a 7-minute trace from a gigabit link.

As the table shows, D-RIDE typically sees a reasonably large number of flows M over the entire interval $[0, T]$; however, the number of active flows $N(t)$ and those constantly kept in memory $M(t)$ is much smaller. For the FRG trace, for example, M is 15 times smaller than n, while $E[N(t)]$ is 81 and $E[M(t)]$ a whopping 658 times smaller. In general, NLANR traces benefit more from the removal of dead flows than CAIDA data. This can be explained by the fact that NLANR traces are collected from two consecutive days and thus have a larger observation window T, which leads to larger ratios r_2 and r_3. The same reasoning also explains the fact that the LARGE trace exhibits much larger benefit from removing dead flows than MEDIUM and SMALL traces.

7.1 Memory and Speed

We use the settings of Table VI to compute the amount of memory consumed by D-RIDE according to (174). As shown in the third column of Table VIII for $p = 0.001$ and $K = E[M(t)]$, the required memory size is small and rarely exceeds 40 KB. Even for the LARGE trace that has the most flows in this comparison, D-RIDE only needs 132 KB of RAM, much smaller than roughly 120 MB required for keeping all flow counters. We also compute per-packet processing time from (175) based on Table VI



(a) $E[e(R_l)]$ (b) $E[\hat{e}(R_l)]$

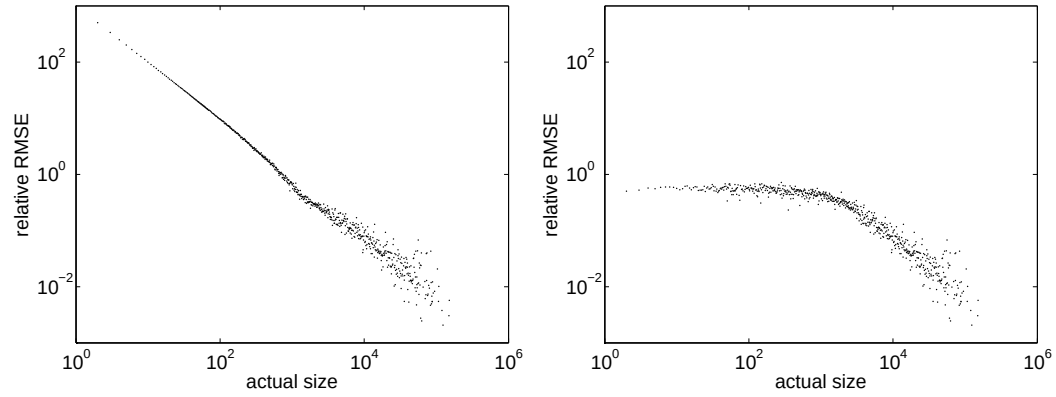
Fig. 43. Estimating single-flow usage in the FRG trace with $p = 0.001$.

and show in the fourth column of Table VIII that $E[D(t)] \leq 25$ ns in the majority of the studied cases, which can be explained by the ratio $E[M(t)]/2K$ being fixed at 0.5 (i.e., 1.5 memory references and 0.5 comparisons per packet on average).

7.2 Estimation Accuracy

First, we examine the problem of estimating the total number of flows in $[0, T]$ and size-one flows in this interval. The fifth and sixth columns of Table VIII list the estimated values of n and n_1 computed from (170) and (171), respectively. The table indicates that these estimates are commonly within 2.5% of the correct value.

We next evaluate the performance of D-RIDE in estimating single-flow usage. Figure 43 plots the estimated flow sizes (averaged over 100 iterations) along with the actual values obtained from the FRG trace using $p = 0.001$. The figure shows that the estimator $e(R_l)$ from previous work tends to overestimate the sizes of small flows, while D-RIDE's estimator $\hat{e}(R_l)$ accurately follows the actual values. We also compare the relative errors of the two studied methods in Figure 44, which indicates



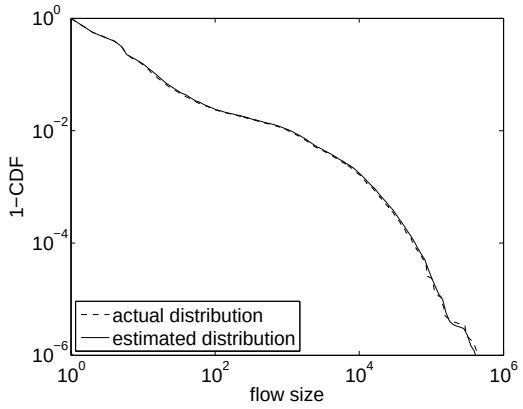
(a) δ_l (b) $\hat{\delta}_l$

Fig. 44. RRMSE of estimating single-flow usage in the FRG trace with $p = 0.001$.

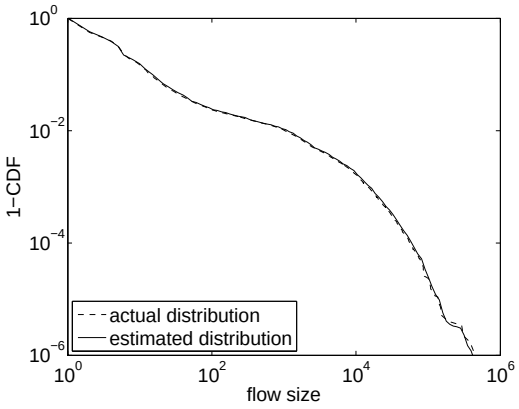
that D-RIDE has bounded relative errors for all flows, while $e(R_l)$ exhibits very large δ_l for small and medium flows, which is an increasing function of $1/p$.

For the flow-size distribution, we first examine three values of p to compare its effect on the accuracy of D-RIDE in the FRG trace. Figure 45 indicates that estimation for all three values of p are very consistent and all of them follow the actual distribution accurately. In our experiments with $p = 0.0001$, D-RIDE recovered the original PMF $\{f_i\}$ using only $M = 7,616$ total flows out of $n = 1.75$M.

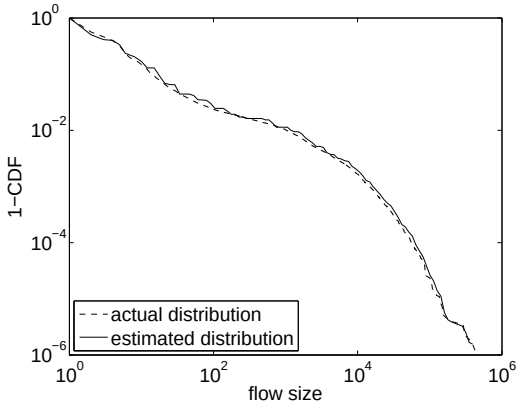
Finally, we apply D-RIDE with $p = 0.001$ to NLANR traces of different traffic types and plot in Figure 46 the estimated distributions along with the actual ones. As the figure shows, the flow statistics of different applications can be accurately estimated by D-RIDE. We observe a similar match in our experiments with three CAIDA traces as shown in Figure 47.



(a) $p = 0.01$

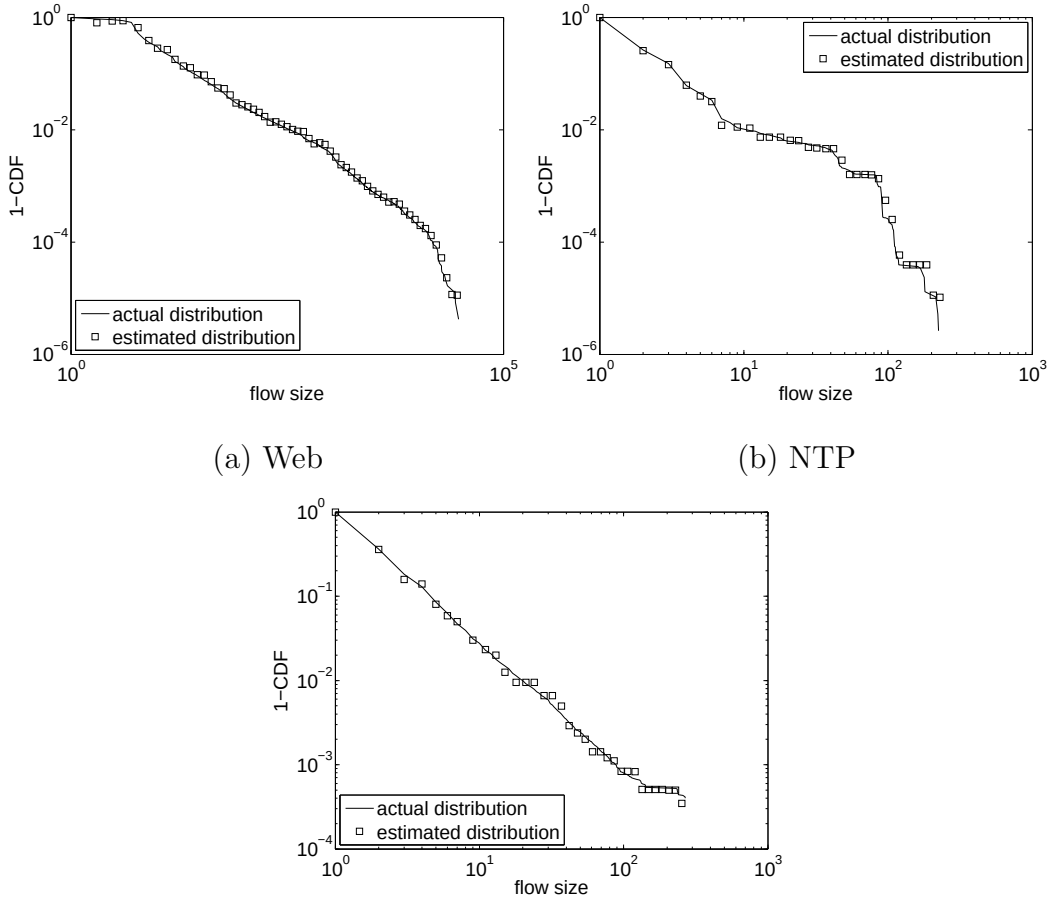


(b) $p = 0.001$



(c) $p = 0.0001$

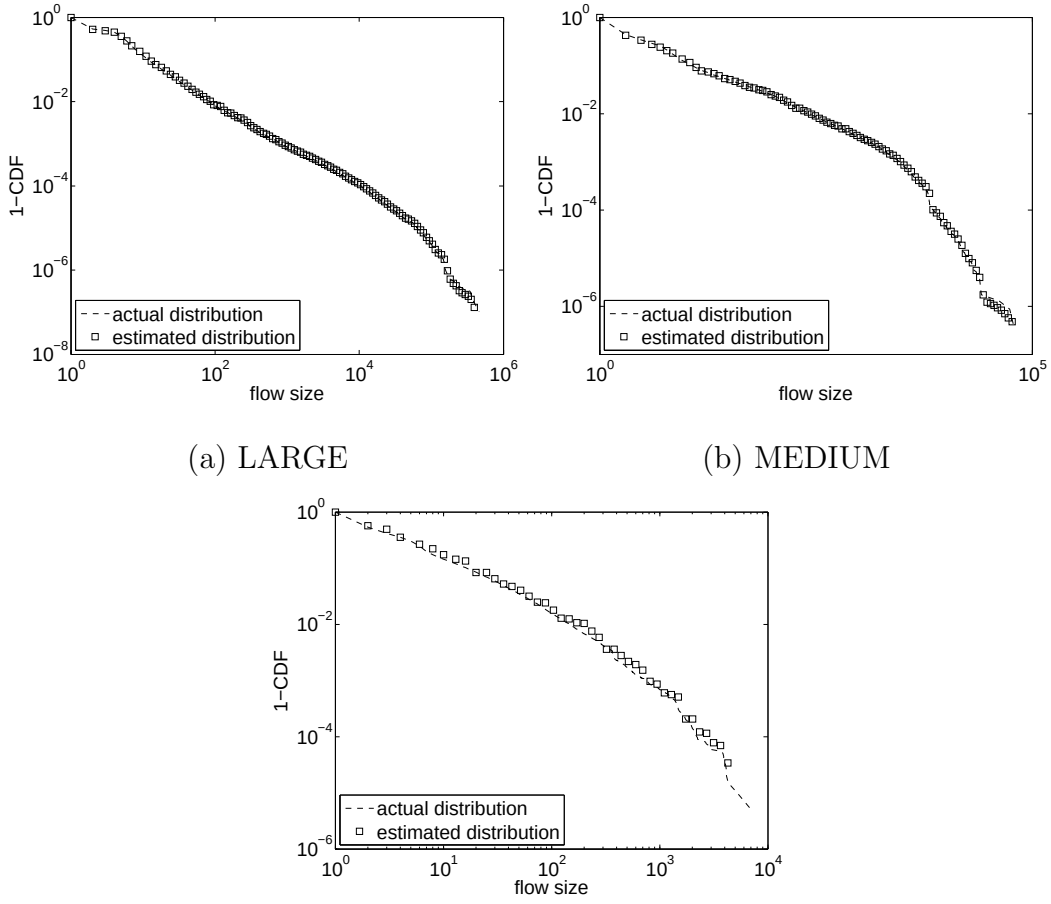
Fig. 45. Estimating the flow size distribution using D-RIDE in the FRG trace.



(a) Web (b) NTP

(c) DNS

Fig. 46. Estimating the flow size distribution using D-RIDE in NLANR traces with $p = 0.001$.



(a) LARGE (b) MEDIUM

(c) SMALL

Fig. 47. Estimating the flow size distribution using D-RIDE in CAIDA traces with $p = 0.001$.

8 Discussion

In this chapter, we proved that previous methods based on residual-geometric sampling had certain bias in estimating single-flow usage and were unable to recover the flow-size distribution from the sampled residuals. To overcome this limitation, we proposed a novel set of estimation algorithms and implemented a scalable framework called D-RIDE for residual-geometric sampling. We applied the proposed methods to Internet traces and showed that these techniques were able to produce accurate estimation of flow statistics, even in the presence of extremely small RAM tables and highly efficient operation (i.e., 1-2 memory pointer references and comparisons per packet).

CHAPTER V

SUMMARY AND FUTURE WORK

1 Summary

This work studied residual sampling in the problem of making tradeoff between accuracy and scalability found in two important measurement efforts P2P measurement and traffic monitoring. We developed three novel algorithms in the family of residual sampling — RIDE, U-RIDE, and D-RIDE. We first designed RIDE to measure the user lifetime distribution in large-scale P2P networks, where the scalability issue was from bandwidth consumption. We showed through analysis, simulations, and experiments that RIDE with the ϵ-subsampling technique was able to achieve more accurate estimation than previous method CBM while reducing traffic overhead by several orders of magnitude. However, due to its assumption of stationary arrivals, RIDE was limited to systems with a constant arrival rate. To make our results more general, we then presented NS-PCM to describe any periodic non-stationary arrival pattern and proved that RIDE could be arbitrarily biased under NS-PCM systems. We thus developed a generic framework U-RIDE and showed that it was able to produce accurate estimation in non-stationary systems. Moreover, we illustrated using simulations and experiments that U-RIDE also supported subsampling and could be as scalable as RIDE. We finally applied our understanding of residual sampling to traffic monitoring, where high-speed links presented the issues of scalability in terms of memory and CPU. We hence devised an accurate and scalable mechanism called D-RIDE and showed that it could precisely estimate flow statistics from the sampled data while reducing the requirement of memory and CPU speed. These results

showed that residual sampling indeed provided a viable way to address the tradeoff between accuracy and scalability in network management.

2 Future Work

As showed by this work, residual sampling has a great potential in addressing the problems in measuring many existing large-scale networks. Future work includes several open directions as follows.

- Many existing efforts in P2P performance analysis assume stationary systems and their correctness could be susceptible to this assumption. Therefore, we need to apply NS-PCM to understand how it affects their results and suggest methods to fix potential problems.
- Many previous performance models require the information of the arrival process. However, the current implementation of U-RIDE does not provide a way to measure the arrival process. We thus need to extend U-RIDE to characterize the properties of user arrivals.
- While this work only examined Gnutella in experiments, U-RIDE is a generic sampling method that does not assume anything specific to Gnutella. We plan to apply U-RIDE to measure other distributed systems.
- Notice that the traffic volume in a router could vary over time, which requires a dynamic mechanism that adjusts the sampling probability used by D-RIDE. To do so, we need to design a special algorithm to recover flow statistics from such adaptive sampling.

REFERENCES

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

[2] D. Brauckhoff, B. Tellenbach, A. Wagner, A. Lakhina, and M. May, "Impact of traffic sampling on anomaly detection metrics," in *Proc. ACM IMC*, Oct. 2006, pp. 159–164.

[3] 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.

[4] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Duxbury/Thomson Learning, 2002.

[5] 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.

[6] 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.

[7] Cooperative Association for Internet Data Analysis (CAIDA). [Online]. Available: http://www.caida.org/.

[8] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the lambert W function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996.

[9] N. Duffield and C. Lund, "Predicting resource usage and estimation accuracy

in an IP flow measurement collection infrastructure," in *Proc. ACM IMC*, 2003, pp. 179–191.

[10] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," in *Proc. ACM IMW*, 2001, pp. 245–256.

[11] N. Duffield, C. Lund, and M. Thorup, "Estimating flow distributions from sampled flow statistics," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 325–336.

[12] N. Duffield, C. Lund, and M. Thorup, "Flow sampling under hard resource constraints," in *Proc. ACM SIGMETRICS*, vol. 32, no. 1, Jun. 2004, pp. 85–96.

[13] N. Duffield, C. Lund, and M. Thorup, "Learn more, sample less: Control of volume and variance in network measurement," *IEEE Transactions in Information Theory*, vol. 51, pp. 1756–1775, 2005.

[14] 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.

[15] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 245–256.

[16] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 323–336.

[17] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proc. GLOBECOM*, Dec. 1999, pp. 1859–1868.

[18] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 265–280, 2001.

[19] 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.

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

[21] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proc. ACM SIGCOMM*, Sep. 2006.

[22] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," in *Proc. IPTPS*, 2006.

[23] 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.

[24] N. Hohn and D. Veitch, "Inverting sampled traffic," in *Proc. ACM IMC*, Oct. 2003, pp. 222–233.

[25] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen, "Accurate and efficient traffic monitoring using adaptive non-linear sampling method," in *Proc. IEEE INFOCOM*, Jun. 2008.

[26] K. Ishibashi, R. Kawahara, T. Mori, T. Kondoh, , and S. Asano, "Effect of sampling rate and monitoring granularity on anomaly detectability," in *Proc. 10th IEEE Global Internet Symposium*, May 2007.

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

[28] R. R. Kompella and C. Estan, "The power of slicing in internet flow measurement," in *Proc. USENIX/ACM IMC*, Oct. 2005.

[29] S. Krishnamurthy, S. El-Ansary, E. Aurell, and S. Haridi, "A statistical theory of chord under churn," in *Proc. IPTPS*, Feb. 2005, pp. 93–103.

[30] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *Proc. ACM SIGMETRICS*, Jun. 2004, pp. 177–188.

[31] A. Kumar, M. Sung, J. Xu, and E. Zegura, "A data streaming algorithm for estimating subpopulation flow size distribution," in *Proc. ACM SIGMETRICS*, Jun. 2005, pp. 61–72.

[32] A. Kumar and J. Xu, "Sketch guided sampling - using on-line estimates of flow size for adaptive data collection," in *Proc. INFOCOM*, Apr. 2006.

[33] A. Kumar, J. Xu, J. Wang, O. Spatschek, and L. Li, "Space-code bloom filter for efficient per-flow traffic measurement," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2004, pp. 1762–1773.

[34] 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.

[35] 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.

[36] 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.

[37] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer networks," in *Proc. ACM PODC*, Jul. 2002, pp. 233–242.

[38] Y. Lu, A. Montanari, B.Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter braids: A novel counter architecture for per-flow measurement," in *Proc. ACM SIGMETRICS*, Jun. 2008.

[39] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proc. ACM IMC*, Oct. 2006, pp. 165–176.

[40] A. Makowski, B. Melamed, and W. Whitt, "On averages seen by arrivals in discrete time," in *Proc. IEEE CDC*, Dec. 1989, pp. 1084–1086.

[41] National Laboratory for Applied Network Research (NLANR). [Online]. Available: http://moat.nlanr.net/.

[42] NetFlow. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt.

[43] Cisco Sampled NetFlow. [Online]. Available: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html.

[44] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fairness through differential dropping," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 33, no. 2, pp. 23–39, 2003.

[45] G. Pandurangan, P. Raghavan, and E. Upfal, "Building low-diameter peer-to-peer networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 995–1002, Aug. 2003.

[46] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 367–378.

[47] S. Ramabhadran and G. Varghese, "Efficient implementation of a statistics counter architecture," in *Proc. ACM SIGMETRICS*, vol. 31, no. 1, Jun. 2003, pp. 261–271.

[48] M. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient hardware hashing functions for high performance computers," *IEEE Trans. Comput.*, vol. 46, no. 12, pp. 1378–1381, 1997.

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] D. Shah, S. Iyer, B. Prabhakar, and N. McKeown, "Analysis of a statistics counter architecture," in *Proc. Hot Interconnects*, Aug. 2001.

[55] M. Steiner, E. W. Biersack, and T. Ennajjary, "Actively monitoring peers in KAD," in *Proc. IPTPS*, Feb. 2007.

[56] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. ACM IMC*, Oct. 2006, pp. 189–202.

[57] 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.

[58] 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.

[59] G. Tan and S. Jarvis, "Stochastic analysis and improvement of the reliability of DHT-based multicast," in *Proc. IEEE INFOCOM*, May 2007, pp. 2198–2206.

[60] J. Tian and Y. Dai, "Understanding the dynamic of peer-to-peer systems," in *Proc. IPTPS*, Feb. 2007.

[61] X. Wang, Z. Yao, and D. Loguinov, "Residual-based measurement of peer and link lifetimes in Gnutella networks," *To Appear in IEEE/ACM Trans. Networking*, 2009.

[62] X. Wang, Z. Yao, and D. Loguinov, "Residual-based measurement of peer and link lifetimes in Gnutella networks," in *Proc. IEEE INFOCOM*, May 2007, pp. 391–399.

[63] R. W. Wolff, *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[64] L. Yang and M. Michailidis, "Sampled based estimation of network traffic flow characteristics," in *Proc. IEEE INFOCOM*, May 2007, pp. 1775–1783.

[65] 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.

[66] Z. Yao, X. Wang, D. Leonard, and D. Loguinov, "On node isolation under churn in unstructured P2P networks with heavy-tailed lifetimes," in *Proc. IEEE INFOCOM*, May 2007, pp. 2126–2134.

[67] Q. Zhao, J. Xu, and Z. Liu, "Design of a novel statistics counter architecture with optimal space and time efficiency," in *Proc. ACM SIGMETRICS*, vol. 34, no. 1, Jun. 2006, pp. 323–334.

APPENDIX A

PROOF OF THEOREM 11

To understand our next derivation for Theorem 11, several definitions and lemmas are in order. For lattice process $\{S_{i,k}\}_{k=1}^{\infty}$, define $\eta_{i,k}$ to be the number bins in interval $[S_{i,k}, S_{i,k+1})$:

$$\eta_{i,k} = (S_{i,k+1} - S_{i,k})/\tau. \tag{178}$$

We are interested of among these bins how many of them have certain special properties, which are defined as follows. For any $\theta \in [0, \tau)$, we define reward $W_{i,k}(\theta)$ of the k-th interval $[S_{i,k}, S_{i,k+1})$ to be the number of bins in which process $Z_i(t)$ is ON at offset θ:

$$W_{i,k}(\theta) = \sum_{j=0}^{\eta_{i,k}-1} Z_i(S_{i,k} + j\tau + \theta). \tag{179}$$

Denote by $I_i(x,t)$ a process associated with $Z_i(t)$:

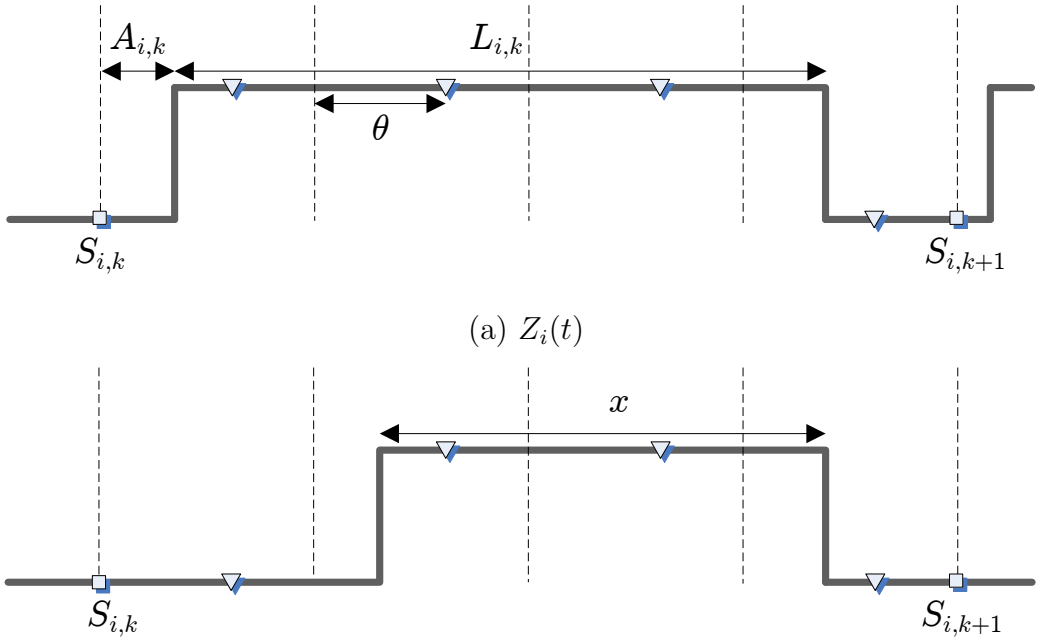
$$I_i(x,t) = \begin{cases} 1 & R(t) \leq x, Z_i(t) = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{180}$$

Similarly, we define reward $R_{i,k}(x,\theta)$ of the k-th interval $[S_{i,k}, S_{i,k+1})$ to be the number of bins in which process $I_i(x,t)$ equals 1 at offset θ:

$$R_{i,k}(x,\theta) = \sum_{j=0}^{\eta_{i,k}-1} I_i(x, S_{i,k} + j\tau + \theta). \tag{181}$$

Figure 48 illustrates an example of process $Z_i(t)$ in a cycle of 4 bins and the corresponding process $I_i(x,t)$ with a given x. It is easy to verify that $W_{i,k}(\theta) = 3$ and $R_{i,k}(x,\theta) = 2$ in the example of Figure 48.

To prove Theorem 11, we first expand residual distribution $H(x,t)$ using rewards $W_{i,k}(t^\star)$ and $R_{i,k}(x,t^\star)$ and then derive these rewards functions. We first need the



(a) $Z_i(t)$

(b) $I_i(x,t)$

Fig. 48. An example of processes $Z_i(t)$ and $I_i(x,t)$ in NS-PCM systems, where $A_{i,k} = 0.3$, $L_{i,k} = 3.1$, $\theta = 0.6$ and $x = 2.1$. Triangles represent points with offset θ and squares are renewal points $\{S_k\}$. Vertical dashed lines stand for bin boundaries with bin size τ.

next lemma.

Lemma 14. *For sufficiently large t_0, probabilities $P(Z_i(t_0))$ and $P(I_i(x,t_0)=1)$ are periodic functions of time t_0:*

$$P(Z_i(t_0)) = \frac{E[W_{i,k}(t_0^\star)]}{E[\eta_{i,k}]},$$
$$P(I_i(x,t_0)) = \frac{E[R_{i,k}(x,t_0^\star)]}{E[\eta_{i,k}]}, \tag{182}$$

where $\eta_{i,k}$ is the number bins in interval $[S_{i,k}, S_{i,k+1})$, $W_{i,k}(.)$ and $R_{i,k}(.)$ are reward functions.

Proof. We first convert the continuous-time ON/OFF process $Z_i(t_0)$ into its discrete-time equivalent $Z_i^\theta(j)$, where $Z_i^\theta(j) = Z_i(j\tau+\theta)$. We make a similar conversion for process $I_i(x,t_0)$ into $I_i^\theta(x,j)$, where $I_i^\theta(x,j) = I_i(x,j\tau+\theta)$. As shown in Figure 48, $Z_i^\theta(j)$ samples marked with triangles are $(1,1,1,0)$ and $I_i^\theta(x,j)$ switches through $(0,1,1,0)$.

We next examine reward functions $W_{i,k}(\theta)$ and $R_{i,k}(x,\theta)$, which also have their corresponding discrete-time versions $W_{i,k}^\theta$ and $R_{i,k}^\theta(x)$, respectively:

$$W_{i,k}^\theta = \sum_{m=0}^{\eta_{i,k}-1} Z_i^\theta(S_{i,k}^\tau + m),$$
$$R_{i,k}^\theta(x) = \sum_{m=0}^{\eta_{i,k}-1} I_i^\theta(x, S_{i,k}^\tau + m). \tag{183}$$

We further convert process $\{S_{i,k}\}_{k=1}^\infty$ into an equivalent discrete point process $\{S_{i,k}^\tau\}_{k=1}^\infty$, where $S_{i,k}^\tau = S_{i,k}/\tau$. Since τ is a constant and the conversion from $S_{i,k}$to $S_{i,k}^\tau$ is linear, it follows from Lemma 1 that process $\{S_{i,k}^\tau\}_{k=1}^\infty$ is a renewal process on discrete time. For any discrete time $j \geq 0$, denote by discrete processes $A(j)$ and $B(j)$

the age and residual of the process $\{S^{\tau}_{i,k}\}_{k=1}^{\infty}$ when it is sampled at time $j = 0, 1, \ldots$:

$$A(j) = j - S^{\tau}_{i,k} + 1, \quad B(j) = S^{\tau}_{i,k+1} - j, \tag{184}$$

given that $S^{\tau}_{i,k} \leq j < S^{\tau}_{i,k+1}$. It follows that $(A(j), B(j))$ is asymptotically stationary as $j \to \infty$.

From stationarity of process $(A(j), B(j))$, we establish that the probability for the k-th interval $[S^{\tau}_{i,k}, S^{\tau}_{i,k+1})$ to be hit by time j is proportional to the corresponding interval length

$$S^{\tau}_{i,k+1} - S^{\tau}_{i,k} = (S_{i,k+1} - S_{i,k})/\tau = \eta_{i,k}.$$

Moreover, given that the k-th interval $[S^{\tau}_{i,k}, S^{\tau}_{i,k+1})$ is hit by time j, i.e., $S^{\tau}_{i,k} \leq j < S^{\tau}_{i,k+1}$, then j is uniformly distributed within the interval, hitting each discrete point with probability $1/(S^{\tau}_{i,k+1} - S^{\tau}_{i,k})$.

Utilizing these results, we thus apply the Renewal-Reward Theorem [63] to process $\{S^{\tau}_{i,k}\}_{k=1}^{\infty}$ and establish The next limiting probabilities:

$$\begin{aligned}
\lim_{j\to\infty} P(Z^{\theta}_i(j) = 1) &= \lim_{j\to\infty} \sum_{m=0}^{j} \frac{Z^{\theta}_i(m)}{j} = \frac{E[W^{\theta}_{i,k}]}{E[\eta_{i,k}]}, \\
\lim_{j\to\infty} P(I^{\theta}_i(x, j) = 1) &= \lim_{j\to\infty} \sum_{m=0}^{j} \frac{I^{\theta}_i(x, m)}{j} = \frac{E[R^{\theta}_{i,k}(x)]}{E[\eta_{i,k}]}.
\end{aligned} \tag{185}$$

The desired result follows from combining $W_{i,k}(\theta) = W^{\theta}_{i,k}$, $R_{i,k}(x, \theta) = R^{\theta}_{i,k}(x)$, along with $\theta = t^{\star}_0$ and $j = \lfloor t_0/\tau \rfloor$. □

Lemma 14 allows us to prove an important asymptotic result on residual distribution $H(x, t_0)$ in the next lemma.

Lemma 15. *For sufficiently large t_0, residual distribution $H(x, t_0)$ can be expressed*

using a periodic function of time t_0:

$$H(x,t_0)=\frac{E[R_{i,k}(x,t_0^\star)]}{E[W_{i,k}(t_0^\star)]}. \tag{186}$$

Proof. Notice from conditional probability that $H(x,t_0)$ can be rewritten as:

$$H(x,t_0)=\frac{P(I_i(x,t_0)=1)}{P(Z_i(t_0)=1)}. \tag{187}$$

The result in (186) immediately follows from substituting (182) into (187). □

Our analysis below indicates that reward functions $W_{i,k}(\theta)$ and $R_{i,k}(x,\theta)$ are solely determined by the arrival time of user i and its lifetime in the k-th interval $[S_{i,k},S_{i,k+1})$. In the next lemma, we derive $W_{i,k}(\theta)$ and $R_{i,k}(x,\theta)$ by conditioning on arrival offset $A_{i,k}=A$ and lifetime $L_{i,k}=L$.

Lemma 16. *Assume that in the k-th interval $[S_{i,k},S_{i,k+1})$, user i arrives at offset A and its lifetime is L. Then, rewards $W_{i,k}(\theta)$ and $R_{i,k}(x,\theta)$ are given by:*

$$\begin{aligned} W_{i,k}(\theta)&=\kappa(A,A+L,\theta)\\ R_{i,k}(x,\theta)&=\kappa(A+\max(L-x,0),A+L,\theta). \end{aligned} \tag{188}$$

where $\kappa(u,v,\theta)$ is defined for $u<v$:

$$\kappa(u,v,\theta)=\begin{cases} \mathbf{1}(u^\star\le\theta,v^\star>\theta) & b(u)=b(v)\\ \mathbf{1}(u^\star\le\theta)+\mathbf{1}(v^\star>\theta) & \\ \qquad +(b(v)-e(u))/\tau & b(u)<b(v) \end{cases}, \tag{189}$$

and $\mathbf{1}(.)$ is an indicator function.

Proof. We first develop a general formula for counting rewards and then apply it to reward functions $W_{i,k}(\theta)$ and $R_{i,k}(x,\theta)$. For any given time interval $\mathcal{I}\equiv[S_{i,k}+u,S_{i,k}+v)\subseteq[S_{i,k},S_{i,k+1})$, we denote by $\kappa(u,v,\theta)$ the number of time points in interval

$\mathcal{I}$ whose offsets are θ.

For $b(u) = b(v)$, i.e., u and v are in the same bin, the only possible point with offset θ is included in interval $\mathcal{I}$ if $u^\star \leq \theta \leq v^\star$, which gives the first line in (189). For $b(u) < b(v)$, i.e., u and v are in different bins and interval $\mathcal{I}$ covers $(b(v) - e(u))/\tau + 2$ bins. In the first bin, we need to have $u^\star \leq \theta$ for the point with offset θ to be included by $\mathcal{I}$; in the last bin, we need to have $v^\star \geq \theta$ for the point with offset θ to be in $\mathcal{I}$; the rest of the bins yield $(b(v) - e(u))/\tau$ points with offset θ in interval $\mathcal{I}$. The second line in (189) follows from summing up these three parts.

According to definition, $W_{i,k}(\theta)$ is given by counting the points included by the ON period $(S_{i,k}+A, S_{i,k}+A+L)$, which gives the first line in (188). Similarly, $R_{i,k}(x, \theta)$ can be computed by counting the ON points in interval $(S_{i,k} + A, S_{i,k} + A + L)$ for $L \leq x$ or $(S_{i,k} + A + L - x, S_{i,k} + A + L)$ for $L > x$, since only those sampling points make the residual lifetime less than x. Then, the second line in (188) follows. □

Next, we simplify (188) by deriving conditional expectations of $E[W_{i,k}(\theta)]$ and $E[R_{i,k}(x, \theta)]$. Denote by $\omega(z, \theta)$ and $\varphi(x, z, \theta)$ the conditional expectation of $W_{i,k}(\theta)$ and $R_{i,k}(x, \theta)$, respectively, given $L = z$:

$$\begin{aligned} \omega(z, \theta) &\equiv E[W_{i,k}(\theta)|L = z], \\ \varphi(x, z, \theta) &\equiv E[R_{i,k}(x, \theta)|L = z]. \end{aligned} \tag{190}$$

The next lemma follows from taking the conditional expectation of both sides of (188) and expressing $\omega(z, \theta)$ and $\varphi(x, z, \theta)$ in terms of arrival time distribution F_A.

Lemma 17. *Assume that the k-th ON duration is $L = z$. Then, the conditional*

expectations of rewards $W_{i,k}(\theta)$ *and* $R_{i,k}(x,\theta)$ *are given by:*

$$\begin{aligned}\omega(z,\theta) &= F_A(\theta) - F_A(\max(\theta - z^\star, 0)) + 1 \\ &\quad - F_A(1 + \min(\theta - z^\star, 0)) + b(z)/\tau,\end{aligned} \tag{191}$$

and

$$\varphi(x,z,\theta) = \begin{cases} \omega(z,\theta) & z \le x \\ \omega(z,\theta) - \omega(z-x,\theta) & z > z \end{cases}. \tag{192}$$

Proof. Notice that conditioning on whether $A \le 1-z$ or not, we can split $E[W_{i,k}(\theta)|L = z]$ into the following two parts:

$$E[W_{i,k}(\theta)|L = z] = w_l + w_g,$$

where

$$w_l = E[W_{i,k}(\theta)|L = z, A \le 1 - z]P(A \le 1 - z), \tag{193}$$

and

$$w_g = E[W_{i,k}(\theta)|L = z, A > 1 - z]P(A > 1 - z). \tag{194}$$

For $A \le 1 - z$, it follows from (188) that:

$$\begin{aligned} w_l &= E[\mathbf{1}(\theta - z < A \le \theta)|A \le 1 - z] \\ &\qquad \times P(A \le 1 - z) \\ &= P(\theta - z < A \le \theta, A \le 1 - z). \end{aligned} \tag{195}$$

Splitting (195) into two cases depending on whether z is larger than 1 or not, we establish that:

$$w_l = \begin{cases} F_A(\min(\theta, 1-z)) & \\ \qquad - F_A(\max(\theta - z, 0)) & z \le 1 \\ 0 & z > 1 \end{cases}. \tag{196}$$

For $A > 1 - z$, we split w_g into three parts, $w_g = w_{g1} + w_{g2} + w_{g3}$, where:

$$w_{g1} = E[\mathbf{1}(A \leq \theta)|A > 1 - z]P(A > 1 - z),$$
$$w_{g2} = E[\mathbf{1}(r(z + A) > \theta)|A > 1 - z]P(A > 1 - z),$$
$$w_{g3} = E[\lfloor z + A \rfloor - 1|A > 1 - z]P(A > 1 - z).$$

It is easy to verify that:

$$w_{g1} = \begin{cases} F_A(\theta) - F_A(\max(1 - z, \theta)) & z \leq 1 \\ F_A(\theta) & z > 1 \end{cases}, \tag{197}$$

$$w_{g2} = \begin{cases} 1 - F_A(\min(1 + \theta - z, 1)) & z \leq 1 \\ 1 - F_A(\min(1 + \theta - r(z), 1)) & \\ \quad + F_A(1 - r(z)) & z > 1 \\ \quad - F_A(\max(\theta - r(z), 0)) & \end{cases}, \tag{198}$$

and

$$w_{g3} = \begin{cases} 0 & z \leq 1 \\ b(z)/\tau - F_A(1 - r(z)) & z > 1 \end{cases}. \tag{199}$$

Combining (196)-(199), we establish the result in (191).

Next, we derive $\varphi(x, z, \theta)$. For $L \leq x$, $R_{i,k}(x, \theta) = W_{i,k}(\theta)$ and thus $\varphi(x, z, \theta) = \omega(z, \theta)$, which is the first line in (192). For $L > x$, we count the number of points with offset θ in the interval $[A, A + L - x]$ and then subtract it from $W_{i,k}$, from which the second line in (192) follows. □

Now, we are ready to prove Theorem 11 using the above results.

Proof. Notice that from conditional expectation:

$$\begin{aligned} E[W_{i,k}(\theta)] &= \int_0^\infty E[W_{i,k}(\theta)|L=z]dF_L(z) \\ &= \int_0^\infty \omega(z,\theta)dF_L(z), \end{aligned} \tag{200}$$

and

$$\begin{aligned} E[R_{i,k}(x,\theta)] &= \int_0^\infty E[R_{i,k}(x,\theta)|L=z]dF_L(z) \\ &= \int_0^\infty \varphi(x,z,\theta)dF_L(z). \end{aligned} \tag{201}$$

Substituting (200)-(201) into (186), it follows that for sufficiently large t_0, residual distribution $H(x,t_0)$ is given by:

$$H(x,t_0) = \frac{\int_0^\infty \varphi(x,z,t_0^\star)dF_L(z)}{\int_0^\infty \omega(z,t_0^\star)dF_L(z)}. \tag{202}$$

Further substituting (191)-(192) into (202) yields the desired result. □

VITA

Xiaoming Wang received the B.S. degree in Computer Engineering and the M.S. degree in Computer Technology from Beijing University of Posts and Telecommunications, Beijing, China, in 1999 and 2002, respectively. He graduated with his Ph.D. degree in Computer Science from Texas A&M University in August 2009.

During 2002-2003, he worked for Samsung Advanced Institute of Technology, South Korea. His research interests include traffic monitoring, peer-to-peer systems, probabilistic analysis of computer networks, and topology modeling. He may be contacted at:

Xiaoming Wang

Harvey R. Bright Building (HRBB)

Department of Computer Science and Engineering

Texas A&M University

College Station, TX 77843-3112

U.S.A

Email: xmwang@gmail.com

The typist for this dissertation was Xiaoming Wang.