Understanding Churn in Peer-to-Peer Networks

Daniel Stutzbach, Reza Rejaie
University of Oregon
{agthorr,reza}@cs.uoregon.edu

ABSTRACT

The user-driven dynamics of peer participation, or *churn*, are an inherent property of Peer-to-Peer (P2P) systems that should be taken into account in both the design and evaluation of any P2P application. However, accurately characterizing churn requires fine-grained and unbiased information about the arrival and departure of peers which is challenging to acquire in practice. Prior studies on specific peer-to-peer systems show that peer participation is highly dynamic but provide conflicting characteristics for these dynamics. Therefore, churn remains poorly understood, despite its significance.

In this paper, we identify several common pitfalls that lead to measurement error when studying peer churn. We carefully address these difficulties and present a detailed study of churn in three types of widely-deployed P2P systems: an unstructured file-sharing system (Gnutella), a content-distribution system (BitTorrent), and a Distributed Hash Table (Kad). Our analysis reveals several interesting properties of churn: (i) the overall dynamics of peer participation is surprisingly similar across different systems, (ii) peer session times are not exponential, (iii) a large portion of participating peers at any point of time are highly stable while the remaining peers turnover very quickly, (iv) the Weibull distribution models peer inter-arrival times well while the exponential distribution model was not consistent with our observations, and (v) peer session times across consecutive appearances are correlated. In summary, this paper advances our understanding of churn by improving the accuracy of characterizations, comparing different P2P systems, and exploring new aspects of churn.

1 Introduction

During recent years, the Internet has witnessed a significant increase in the popularity of Peer-to-Peer (P2P) applications ranging from file-sharing (*e.g.*, Gnutella and FastTrack) to conferencing (*e.g.*, End System Multicast) and content distribution (*e.g.*, BitTorrent). A peer joins the system when a user starts the application, contributes some resources while making use of the resources provided by others, and leaves the system when the user exits the application. We define one such join-participate-leave cycle as a *session*. The independent arrival and departure by thousands–or millions–of peers creates the collective effect we call *churn*. The user-driven dynamics of peer participation must be taken into account in both the design and evaluation of any P2P application. For example, the distribution of session length can affect the overlay structure [27], the resiliency of the overlay [14], and the selection of key design parameters [15]. Towards this end, researchers and developers require a reliable, representative, and relatively simple model of churn.

However, accurately characterizing churn requires fine-grained and unbiased information about the arrival and departure of peers which is challenging to acquire in practice, primarily due to the large size and highly dynamic nature of these systems. Therefore, the characteristics of churn in large-scale P2P systems are not currently well understood. Several measurement studies [10, 24, 25] have presented a high level view (*e.g.*, the CDF of session lengths [23] or median session length [10]) of churn as part of broader characterizations of P2P systems. While these studies have revealed that peer participation is highly dynamic, their findings are dramatically different. For example, the reported median session lengths varies from one minute to one hour [22]. In the absence of a reliable model for churn, researchers must make assumptions about the distribution of arrival times and session lengths that may be incorrect.

This study takes a major step towards increasing our understanding of churn by conducting deeper analysis and relying on more accurate measurements. One of our contributions is to identify a number of key challenges in characterizing churn that arise from factors such as measurement limitations, network conditions and P2P dynamics. We then develop techniques to address these difficulties or at least bound the resulting error. As a result, our measurements are significantly more accurate and representative.

Our second contribution is an examination of churn at two levels: *(i) Group-level characteristics* that capture the behavior of the all participating peers collectively, and *(ii) Peer-Level characteristics* that capture the behavior of specific peers across multiple appearances in the system over time. Furthermore, we examine some of the underlying causes and implications of our findings on the design of P2P systems. To ensure the broad applicability of our results, we study churn in three types of widely deployed P2P systems: Gnutella, an unstructured file-sharing system; Kad, a Distributed Hash Table (DHT); and BitTorrent, a content-distribution system. Examining multiple systems allows us

to explore the similarities and differences in churn behavior between different types of P2P systems. Our main results can be summarized as follows:

- Aggregate, group-level properties of churn exhibit similar behavior across all three applications, but per-peer properties in BitTorrent are significantly different.
- Session lengths are fit by Weibull or log-normal distributions, but not by the exponential or Pareto.
- The Weibull distribution is consistent with the observed peer inter-arrival time in BitTorrent, while the exponential distribution is not.
- Past session length is a good predictor of the next session length in Gnutella and Kad, but not BitTorrent.
- The availability of individual peers exhibits a strong correlation across consecutive days.
- In BitTorrent, peers frequently remain in the system long after their downloads complete.

The remainder of the paper is laid out as follows. Section 2 describes the three systems we examine in more detail. Section 3 identifies common pitfalls in studying churn, and how we address them in our study. Sections 4 and 5 present our results, while Section 6 describes some key implications of our results on the design of P2P applications. Section 7 presents the related work on characterizing churn. Finally, Section 8 concludes the paper and describes our future plans.

2 Background

Before we discuss our methodology, we need to introduce the systems we examine. The Gnutella and Kad overlays provide a keyword-search function that allows users to locate files that have the keyword in the filename. Gnutella accomplishes this using an unstructured topology, while Kad uses a DHT. BitTorrent, on the other hand, forms an overlay to facilitate the rapid transfer of very large files (100+ MB) to a large number of peers. In the following subsections, we briefly describe the relevant aspects of these systems and introduce our datasets.

Overall, our goal is to measure the arrival and departure time of peers, so that we can compute characteristics such as session lengths and inter-arrival intervals. Each system provides slightly different hooks for measurement, each with advantages and disadvantages. The two most important properties are the *precision* in measuring arrival and departure times and the ability to capture a *representative* set of sessions. We address the issues of possible inaccuracies or bias in our data in Section 3.

Our datasets consist of two types: *(i)* centralized logs that capture the arrival and departure time of each peer, and *(ii)* sequences of snapshots that record the peers present at a particular time. By comparing snapshots, we can deduce when a peer arrived and departed. The length of time required to capture a snapshot, Δ, determines the precision in determining the arrival or departure time. All of the snapshots were captured with Cruiser [26], a fast, distributed P2P crawler that features a plug-in architecture allowing it to capture different P2P systems. Cruiser operates by progressively exploring an overlay topology, querying peers for a list of their neighbors, and adding newly discovered peers to its queue until the queue is exhausted. The start time and length of our datasets are summarized in Table 1.

Dataset	Start Date	Duration	Kad Zone
Gnutella 1	Oct. 14, 2004	2 days	
Gnutella 2	Oct. 21, 2004	2 days	
Gnutella 3	Nov. 25, 2004	2 days	
Gnutella 4	Dec. 21, 2004	2 days	
Gnutella 5	Dec. 27, 2004	2 days	
Kad 1	Apr. 13, 2005	2 days	0xab0/10
Kad 2	Apr. 16, 2005	2 days	0x594/10
Kad 3	Apr. 18, 2005	2 days	0xe14/10
Kad 4	Apr. 21, 2005	2 days	0x734/12
BitTorrent Red Hat	Mar. 21, 2003	5 months	
BitTorrent Debian	Feb. 22, 2005	2 months	
BitTorrent FlatOut	Nov. 11, 2004	2 months	

Table 1: Measurement collections

Gnutella: Gnutella is currently one of the most popular P2P systems, with more than 1 million simultaneous users [2]. It uses a two-tier overlay structure, similar to FastTrack and eDonkey. Most peers are *leaf peers*, while a small fraction of peers act as *ultrapeers*. The leaf peers connect to a handful of ultrapeers, which index the content of the leaves. Searches spread out from ultrapeers using a modified expanding-ring search.

Our Gnutella data consists of snapshots of the entire Gnutella network taken with Cruiser by Stutzbach *et al.* [27]. Using a special crawler hook provided by modern Gnutella [19], Cruiser can capture 1.3 million peers within 7 minutes. Since the snapshots capture all peers in the system, they are necessarily representative. The datasets consist of five sets of 48-hour periods of back-to-back snapshots, with crawl durations ranging from 4 to 10 minutes.

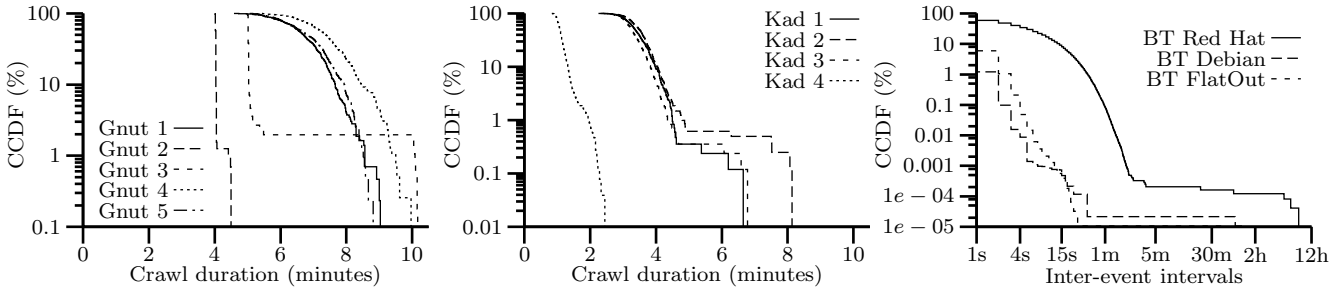
Kad: Kad is a Kademlia-based [20] P2P search network used by the eMule P2P file-sharing software [1]. To our knowledge, Kad is the largest deployed DHT, with more than 1 million simultaneous users. Similar to other DHTs each peer has a persistent, globally-unique identifier of length b bits (in Kad's case $b = 128$ bits). Keywords are hashed to b bits and stored on the peer with the closest matching identifier. Each peer stores a structured routing table pointing to other nodes in the network such that the expected number of overlay hops to perform any lookup is $O(\log n)$, where n is the population size.

We used Cruiser to capture a subset of the DHT identifier space, called a *zone*. As an input parameter, Cruiser accepts a zone, specified as a Kad ID address and mask, analogous to an IP subnet address. We use the "slash notation" to specify Kad ID zones. For example "0x594/10" specifies all Kad identifiers where the high-order 10 bits match the high-order 10 bits in the hexadecimal number 0x594. Monitoring a larger zone includes more sessions, but requires more time to crawl, and thus decreases the precision.

Since each peer selects its ID uniformly at random[1], each zone is a uniformly random sample of the Kad network as a whole and therefore representative. We used the Kad version of Cruiser to collect 48-hour slices of back-to-back snapshots of 4 different zones within the Kad overlay. The zone addresses and crawl durations (Δ) are given in Table 1 and Figure 1(b).

BitTorrent: BitTorrent is a popular P2P application for distributing very large files (100+ MB) to a large group of users. Unlike most P2P systems, which form one large overlay, BitTorrent has a distinct overlay for each file. To download a file, peers exchange different blocks of the content until each peer has the entire file. The peers locate one

[1]Technically, the identifiers are selected using a pseudo-random number generator seeded with entropy collected by the operating system. Because the identifiers are 128 bits, the probability of collisions occurring is extremely small.



(a) Gnutella crawl times (b) Kad crawl times (c) BitTorrent inter-event times

Figure 1: Inter-event gaps

another using a rendezvous point, called a *tracker*, whose address is provided to new members out of band. Each new peer contacts the tracker via HTTP, periodically sends an update of its progress to the tracker, and informs the tracker when it departs. Each peer may receive the entire file across multiple sessions, *i.e.*, it may obtain only a subset of blocks in one session and resume the download later. Many peers may give up without downloading the whole file [11]. The tracker logs its interactions with peers, providing the arrival and departure times of peers with one second resolution. While the tracker records the arrival time of all peers, the departure time is only captured for peers which depart gracefully. We have obtained tracker logs from three long BitTorrent networks: Debian ISO images, a Red Hat ISO image, and a demo of the game FlatOut[2].

3 Pitfalls in Characterizing Churn

In the previous section, we described how our data was collected and the granularity of the measurements. However, we have not yet addressed how to cope with limitations of the data, such as peers missed during a crawl or sessions longer than the measurement period. While we may not hope to address every possible event that might introduce error into the measurement and analysis of churn, we discuss certain pitfalls which are the ones most likely to cause significant error. In the following subsections, we describe and address these issues. In many cases, we are able to overcome the problem entirely. In some cases, we must settle for bounding or estimating the error.

3.1 Missing Data

One of the challenges in collecting data for studying churn is that it requires continuous measurement. If the observation software crashes or loses network connectivity, data will be missing, essentially breaking the dataset into two pieces. If the gap is unnoticed and the data is processed as one piece, it may introduce considerable error.

In Gnutella and Kad, our data is composed of a series of snapshots captured by a crawler. Each snapshot contains the start time of the crawl. Examining the sequence of these times, we did not find any significant gaps in the data. Within each dataset, the time from the start of one crawl to the start of the next is reasonably close to the mean. The distributions are show in Figures 1(a) and 1(b).

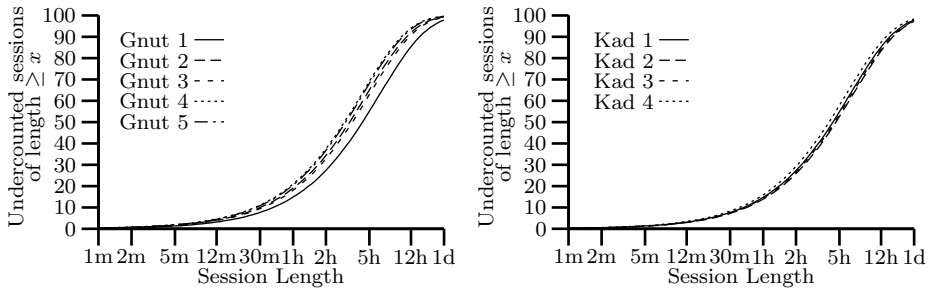
For BitTorrent, our data consists of logs from BitTorrent trackers which record each event with 1-second granularity. In our data, typically there are just a few seconds between each event. However, as shown in Figure 1(c) there are a few outlying points with large gaps between one event and the next. In total, we found five significant gaps in the Red Hat log, two in the Debian log, and one in the Flat Out log. The shortest of these gaps is twenty minutes while the longest is eight hours. All other inter-event intervals are less than 4 minutes. Although the Red Hat log has been studied by other researches [4, 11, 13], to our knowledge these gaps have not been previously reported. We also checked if the clock ever ran backward, but it did not. It seems very unlikely that an otherwise busy tracker receiving several hits per minute would abruptly have an eight-hour period of silence. Therefore, we conclude that during these gaps the tracker was not running or was experiencing network connectivity problems. An alternative explanation is that the clock time on the tracker was changed. Instead of using the entire logs, we restrict ourselves to the largest contiguous portion between gaps.

3.2 Biased Peer Selection

The most common approach [6, 17, 23] for measuring churn has been to select a set of peers and ping them at regular intervals to determine when their sessions begin and end. However, polling a closed population of peers repeatedly captures sessions from a small fraction of all peers (those who return regularly). This introduces bias in two ways: *(i)* the potential for bias in selecting the peers and *(ii)* consecutive sessions of the same peer are correlated (as we will show in Section 5.2). Selecting peers based on criteria that are correlated with session length will lead to biased results. Selecting based on query responses [6, 23] may be biased if there is a correlation between session length and number of files shared. Similarly, conducting a partial crawl of the topology [23] may be biased due to a correlation between session length and a peer's degree [27]. One way around these difficulties is to monitor the entire system, implicitly detecting new peers.

In BitTorrent, the tracker is a central monitoring point that captures the arrival time of all peers and the departure time of all peers that depart gracefully. Sessions that ended ungracefully were eliminated from our analysis (22% in Red Hat, 70% in Debian, 27% in FlatOut). While tracker logs provide a comprehensive account for a particular tracker, it is possible that different trackers see different behavior. To account for this, we present results from three different trackers, serving two different types of files: two Linux ISO images and one game demo.

[2]We would like to thank Ernst Biersack from the Institut Eurecom for kindly sharing the Red Had tracker logs [11], the Debian organization for their tracker logs, and 3D Gamers for the FlatOut logs.



(a) Gnutella (b) Kad

Figure 2: Upper-bound on the percentage of missed observations due to false negatives

For Gnutella, we use snapshots of all the peers in the system collected with Cruiser at approximately 7 minute intervals. For Kad, we use Cruiser's Kad module to capture all the peers within a particular zone of the DHT. Since each peer selects its Kad ID uniformly at random, there is no causal relationship between a peer's zone and its session length. Unlike sampling a set of addresses and probing them, zones allow us to monitor the arrival of new peers and therefore does not suffer the drawbacks of closed populations. For these reasons, zones are a representative way of characterizing the overall system behavior with respect to churn. In Section 3.4, we address the possibility that some peers may be missing from the snapshots (*i.e.*, false negatives).

3.3 Handling Long Sessions

Because we can only make observations for a finite length of time, we must carefully account for long sessions. Given a measurement window of length τ, we can compute the length of each session that begins and ends within the window. However, this would lead to a significant bias towards shorter sessions. For example, we could only observe a session of length τ if it started exactly at the beginning of our measurements, whereas we would have τ opportunities to observe a session of unit length. Worse still, we are not able to measure any sessions longer than τ.

We use the "create-based method" employed by Saroiu *et al.* [23] to overcome this dilemma. We divide the measurement window into two halves, and only consider sessions that begin during the first half. This provides equal opportunity to observe session lengths less than $\frac{\tau}{2}$. We cannot make unbiased measurements of the length of longer sessions using this method. However, we can observe how many sessions started with a length greater than $\frac{\tau}{2}$, allowing us to *count* sessions that continue past the end of our measurement window, even though we cannot measure them. In summary, our sample includes *all* the sessions which *begin* in the time period $[0, \frac{\tau}{2}]$, either by measuring their session length or by noting that the session length is greater than $\frac{\tau}{2}$.

Our initial measurements, as well as previous studies [6], showed fluctuations in network size correlated with the time of day. In this initial study, we are interested in studying the average behavior and reserve exploring the influence of time of day for future work. Therefore, we must capture all times of day roughly equally by either choosing $\tau = k \cdot 2$ days, where k is a positive integer, or chosing $\tau \gg 2$ days.

3.4 False Negatives

Unlike BitTorrent, in Gnutella and Kad there is the possibility that a peer is actually online even though it is missing from a snapshot. This could occur, for example, due to significant network congestion between the observation point and the peer. Even when the chance of false negatives is small, the impact on measurements of long sessions is significant. For example, polling every 5 minutes with a 1% chance of a false negative per peer can decrease the observed 1-day sessions by 94%. Because false negatives are inaccuracies in the measurements themselves, we cannot directly measure how often they occur. However, we can draw some inferences from the data. We consider two types of false negatives that could theoretically occur:

Systematic failures: Some peers may be very prone to being missed by the crawler, perhaps due to routing instability between the observation point and the peer. If the failure is permanent and the peer is never captured, no sessions are observed and the peer is simply omitted from the data and does not significantly impact the results. If the failure is intermittent, the peer will appear to flap, frequently going up and down. In Section 5.3, we show that few peers exhibit flapping behavior.

Random failures: Random failures are cases where the crawler randomly misses some fraction, λ, of peers during each crawl. Even if λ is quite low, the cumulative probability of a false negative grows exponentially with the session length. The observed session length distribution is therefore limited by the exponential distribution with rate parameter λ. We can leverage this fact to compute an upper-bound on λ as follows, using our Gnutella 1 dataset as an example.

We observed that 2.2% of sessions had a length of 1 day or more. Using the mean crawl duration of $\Delta = 6.7$ min, we compute the predicted decrease in these sessions according to the exponential distribution: $\exp\left(-\lambda \cdot \frac{1 \text{ day}}{\Delta}\right)$. For example, if $\lambda = 10\%$, then we would expect to observe only 1 in every 3.1 trillion sessions that are 1 day or longer. Since 2.2% is substantially more than 1 in 3.1 trillion, λ must significanlty smaller than 10%. We can compute an upper-bound on λ by solving $2.2\% = \exp\left(-\lambda \cdot \frac{1 \text{ day}}{6.7 \text{ min}}\right)$ which yields $\lambda = 1.8\%$. In fact, λ must be significantly lower, otherwise the observation of 2.2% would only be possible if nearly all sessions were at least 1 day in length.

Given an upper-bound on λ, we can compute an upper-bound on the number of reduced observations as a function of session length, shown in Figure 2. Short and moderate length sessions (up to an hour) are not significantly affected; however, long sessions (5 hours and longer) may be dra-

matically undercounted. We factor this limitation into our conclusions as we analyze our results.

3.5 Handling Brief Events

The granularity of polling can lead to measurement oversights. If a brief departure is missed, two sessions may look like one longer session. On the other hand, very short sessions may not be observed at all.

In BitTorrent, a centralized tracker records all events, so brief events are not a problem. Even if the events are shorter than the logs' granularity (1 second), they are still recorded in order.

For Gnutella and Kad, we are limited by the snapshot granularity of around 7 minutes and 4 minutes, respectively. Very short sessions will be missing from the data, but this does not adversely impact our observations of other sessions. Brief departures pose a larger problem, as they may artificially inflate the number of observed long sessions. However, in Section 5.3 we present evidence that most peers do not repeatedly come and go, suggesting that the number of errors of this type is small. Moreover, it is outweighed by the problem of false negatives, described in the previous subsection, which causes the number of long sessions to be under-counted.

3.6 NAT

Network Address Translation (NAT) devices present an obstacle to observing churn in two ways. First, they can make it difficult to observe a peer at all, since it is not possible to send the peer unsolicited packets to check its status. This could prevent any NATed or firewalled peers from being included, which may mean 73% of all peers [9]. Second, if there are several peers behind a single NAT device, they may all look like one peer with the NAT device's external IP address. As a result, as long as any of the actual peers is up, the single peer will appear to be up, creating one artificially long session. Typical home users with a NAT and only one peer behind it do not pose a problem; however, erroneous data points will be collected if a large organization has many users all connecting to the same P2P network through a NAT device.

In BitTorrent, we don't need to contact peers, since they contact the tracker, overcoming the first difficulty. To overcome the second difficulty, we use a heuristic to identify IP addresses which appear to be NAT devices with multiple peers and eliminate those IP addresses from our dataset. BitTorrent clients generate a random identifier on start-up, which is transmitted to the tracker and stored in the tracker logs. If an IP address is downloading the same file using multiple identifiers simultaneously, we classify it as a NAT device with multiple peers. Table 2 shows the number of IP addresses observed and eliminated from each log.

As a DHT, Kad requires all participating peers to be able to directly receive unsolicited TCP and UDP packets. Therefore, none of the Kad peers are behind NAT devices, eliminating the NAT problem. While there are NATed peers that make use of the Kad overlay, they do not participate in the overlay itself and are not part of our study.

Log	Total IPs	NAT (Eliminated)
BT Red Hat	115,016	9,022 (8%)
BT Debian	23,880	3,964 (17%)
BT FlatOut	1,250	64 (5%)

Table 2: Total IP addresses in BitTorrent logs and the fraction eliminated due to NAT

For Gnutella, the snapshots captured by Cruiser contain all the peers directly contacted as well as all of their neighbors, including NATed peers. Therefore, contacting NATed peers is not necessary to establish their presence. However, we lack a good heuristic for Gnutella to detect when there are multiple peers behind one NAT device. This introduces a potential bias in our Gnutella data towards long sessions. Again, the typical home user with a NAT device does not pose a problem; only large organizations with multiple P2P users all behind one NAT device will cause measurement errors.

3.7 Dynamic Addresses

DHCP and PPP dynamically assign IP addresses to hosts. Under normal circumstances, as long as the host remains up, it's IP address will not change. Therefore, dynamic address assignment is not a problem for measuring the length of sessions. However, it can become problematic for measuring the gap between sessions or correlations between consecutive sessions. When the peer is not continuously participating in the P2P network, the peer's host may have gone down and later returned with a different IP address.

Dynamic address assignment can make it difficult to make conclusions about user behavior. However, it is sometimes useful to know whether *any* peer is likely to be available at a particular IP address. For example, imagine designing the bootstrapping mechanism that helps integrate the peer into the overlay when the application starts. When the application exits, it stores to disk a cache of IP addresses of other peers so that when it starts again it can attempt to contact them. Ideally, the application wants to store IP addresses that have a high probability of being up later. It doesn't matter if the IP address represents the same user; the important part is whether there is a peer at that IP address. For these reasons, dynamic addresses are not a serious problem as long as only appropriate conclusions are drawn from the data.

Kad uses persistent unique identifiers, allowing us to precisely identify particular users across sessions even when their IP address changes. While BitTorrent peers also use unique identifiers, these identifiers are *not* persistent across sessions; the BitTorrent application selects a new identifier each time it starts[3]. Characterizing the behavior using these persistent identifiers is useful for the design of P2P applications that store persistent state across sessions. Therefore, when studying behavior across sessions in Section 5, we examine the behavior of users in Kad and the behavior of IP addresses in Gnutella and BitTorrent.

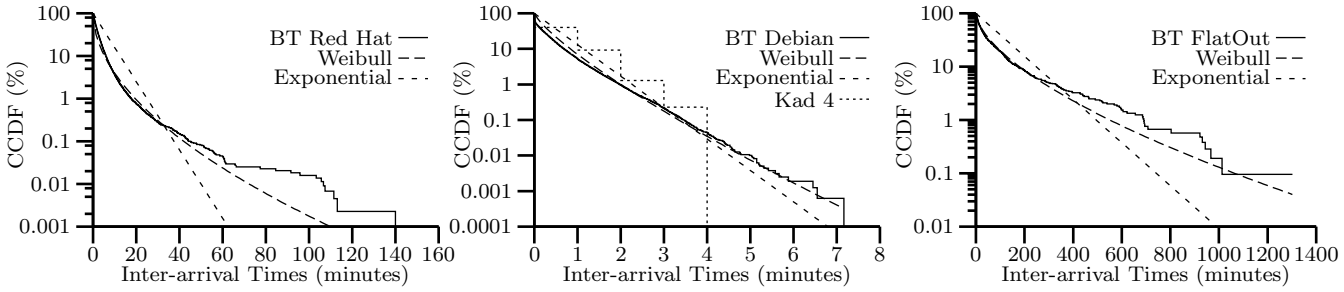
4 Group-Level Characterization

This section explores two fundamental properties of churn which do not rely on maintaining peer identity across sessions as follows: *(i)* the inter-arrival time is the time that passes from the start of one session to the start of the next session (not necessarily by the same peer) and *(ii)* the session length is the time that passes from the start of a session until the end of that session. In other words, the inter-arrival distribution captures when peers arrive and the session length distribution captures how long they stay in the system once they have arrived. Prior simulation and analysis

[3]This is true of the official BitTorrent application at the time our traces were collected. More recent versions of BitTorrent or third-party implementations may have different behavior.

Dataset	Fig. 1	Fig. 3,12	Fig. 4,5(a)	Fig. 5(b),5(c)	Fig. 6	Fig. 9	Fig. 10	Fig. 11
Gnutella 1	430	N/A	5,624,972	N/A	154,804,229	5,670,886	2,850,712	3,118,797
Gnutella 2	714	N/A	11,713,861	N/A	263,595,662	11,700,224	8,697,797	3,182,355
Gnutella 3	561	N/A	9,743,605	N/A	221,638,443	9,749,129	6,573,694	3,368,625
Gnutella 4	390	N/A	13,232,322	N/A	154,978,617	13,256,624	9,579,572	3,834,708
Gnutella 5	423	N/A	12,117,124	N/A	155,671,531	12,116,071	8,550,825	3,737,315
Kad 1	844	N/A	7,104	N/A	404,751	7,136	5,037	2,299
Kad 2	810	N/A	6,326	N/A	390,874	6,303	4,348	2,254
Kad 3	845	N/A	6,161	N/A	397,301	6,175	4,280	2,135
Kad 4	2,573	5,676	2,723	N/A	337,722	2,713	2,205	540
BitTorrent Red Hat	2,457,473	43,956	25,782	4,058	804,296	5,627	2,294	147,536
BitTorrent Debian	9,386,976	160,020	28,047	7,250	5,856,787	13,590	16,729	137,382
BitTorrent FlatOut	9,387,684	1,050	648	398	8,409	129	52	5,542

Table 3: Number of observations for data presented in figures



(a) BitTorrent Red Hat (b) BitTorrent Debian and Kad (c) BitTorrent FlatOut

Figure 3: Inter-arrival Time Distribution CCDFs

Dataset	Observations	1 Hour	2 Hours	8 Hours
Gnutella 1	5,624,972	2,307,729	1,493,194	456,830
Gnutella 2	11,713,861	3,137,153	1,691,327	412,641
Gnutella 3	9,743,605	3,307,803	1,891,606	474,570
Gnutella 4	13,232,322	4,028,592	2,046,154	322,369
Gnutella 5	12,117,124	3,758,452	1,927,332	350,549
Kad 1	7,104	2,681	1,968	810
Kad 2	6,326	2,719	1,974	805
Kad 3	6,161	2,566	1,918	814
Kad 4	2,723	791	557	205
BT Red Hat	23,266	4,526	3,727	1,853
BT Debian	24,335	7,660	5,886	2,504
BT FlatOut	600	170	67	15

Table 4: Number of observations for Fig. 7 and 8

studies typically assume both distributions to be exponential [15,18,22], though some studies model the session length distribution as Pareto [14,16] (*i.e.*, heavy-tailed). We found that for session lengths neither the exponential nor Pareto distributions fit our data. For inter-arrival times, the exponential distribution may be close enough for some purposes, but the Weibull distribution fits significantly better.

Additionally, to provide greater insight into the implications of the session length distribution, Section 4.4 empirically examines how long peers in the system have been up (the *uptime*) and Section 4.5 examines the power of uptime as a predictor for how much longer peers will remain in the system (the *remaining uptime)*.

Throughout Sections 4 and 5, we present many complementary cumulative distribution functions (CCDFs). Tables 3 and 4 list the number of observations from each dataset used in constructing these CCDFs. The values vary from one figure to another depending on what type of behavior we are observing.

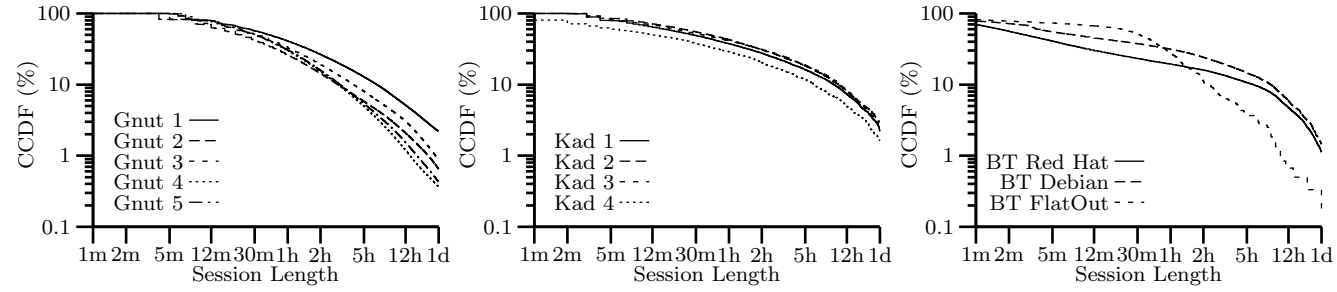
4.1 Distribution of Inter-Arrival Time

The distribution of peer inter-arrival times is an important distribution for understanding churn because it captures the pattern of how peers arrive. To measure inter-arrival times, we must be able to observe individual arrival events. In Gnutella, this is not possible since tens of thousands of new peers arrive between two consecutive snapshots. In contrast, measuring inter-arrival time in BitTorrent is straightforward since tracker logs capture arrival times with one-second granularity. To examine inter-arrival time in Kad, we use our Kad 4 trace which monitors a small fraction of the network with 1 minute snapshots. However, even at 1 minute, the granularity is poor. Therefore, we rely primarily on our BitTorrent data.

Figure 3 presents log-linear plots of the distribution of peer inter-arrival time for four datasets: BT Red Hat, BT Debian, BT FlatOut, and Kad 4. All four datasets appear roughly linear on the log-linear scale, which might suggest that the exponential distribution is a good fit. However, for the FlatOut and Red Hat datasets, there is slight curvature on the left side of the graphs, making a good fit impossible. For the Debian datasets, there is also some curvature, but it is much less pronounced. The Kad dataset, shown in Figure 3(b), has poor granularity but does serve to illustrate that the inter-arrival behavior in Kad is at least roughly similar to that found in BitTorrent.

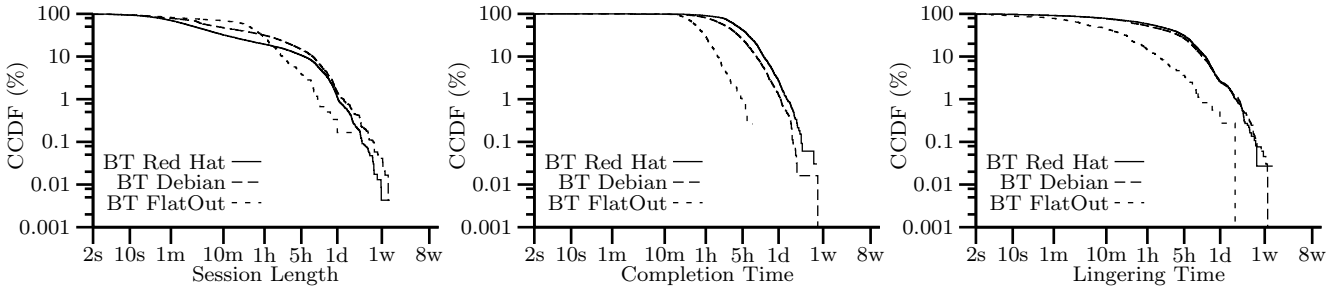
In addition to the data, Figure 3 shows a fit of the exponential distribution, indicating that there are more extreme values than are captured by that model. For example, in the FlatOut data, 33% of inter-arrival times are less than 10 minutes, while the exponential model predicts 9%. On the other hand, in the FlatOut data, 1.5% of inter-arrival times are longer than 10 hours, while the exponential model predicts 0.38%.

The Weibull distribution provides a much better fit, as shown in Figures 3(a) and 3(c), with scale parameter $k = 0.79$ for Debian, $k = 0.53$ for Red Hat, and $k = 0.62$ for FlatOut. We found the parameters of the distributions using the non-linear least-squares method on the log-linear transform of the CCDF. The exponential distribution may still be acceptable model for many purposes. While the exponential exhibits considerable deviation from our datasets,



(a) Gnutella (b) Kad (c) BitTorrent

Figure 4: Session Length CCDFs



(a) BitTorrent session times (b) Completion times (c) Lingering after download completion

Figure 5: Interactions with Transfer Time

it is a simpler model[4] and using simpler models is sometimes desirable. However, it's important to know when one is simplifying, in case it does matter. *In summary, the measurements are not consistent with the commonly used exponential distribution for the inter-arrival time of peers. The Weibull distribution (with* $0.53 \leq k \leq 0.79$*) provides a more accurate model.*

4.2 Distribution of Session Length

One of the most basic properties of churn is the session length distribution, which captures how long peers remain in the system each time they appear. Figure 4 presents the distribution of session length across different datasets for each system. Figure 5(a) presents a different angle on the session lengths in BitTorrent using a wider x-axis scale. These results demonstrate the following points: First, within a single system, the distribution of session length from different datasets are very similar. This implies that the distribution of session length does not significantly change over time. Second, and more importantly, the distributions in different systems are similar, suggesting that user behavior, which determines the session length distribution, is consistent across different P2P systems. Third, session lengths do not follow an exponential distribution.

Prior studies [5,10,16,24] have reported that peer sessions lengths are heavy-tailed. A heavy-tailed distribution is one with the following property [8]:

$$P[X > x] \propto x^{-\alpha}, \text{ as } x \to \infty, \ 0 < \alpha < 2$$

The parameter α is called the *tail index* and is equal to the "slope" of the tail on a log-log plot. As a result, if a distribution is heavy-tailed, on a log-log plot of the CCDF the tail will appear linear with a "slope" between 0 and 2. Examining the tails in our BitTorrent data, we find that this is not the case. By fitting a line to the log-log transform of the tail of the data, we found $\alpha = 2.5$ for Red Hat, $\alpha = 2.7$ for Debian, and $\alpha = 2.1$ for FlatOut. Therefore, we must conclude that *session times in BitTorrent are not heavy-tailed.*

Since neither the exponential nor heavy-tailed distributions proved consistent with our observations, we investigated two other distributions that might provide a good fit: the log-normal distribution and the Weibull distribution. We found that the log-normal provided a decent fit for all three BitTorrent datasets, but significantly overestimated the number of sessions longer than one day in the Red Hat and Debian datasets. The Weibull distribution was able to provide a tighter fit with shape parameters $k = 0.34$ for Red Hat, $k = 0.38$ or Debian, and $k = 0.59$ for FlatOut.

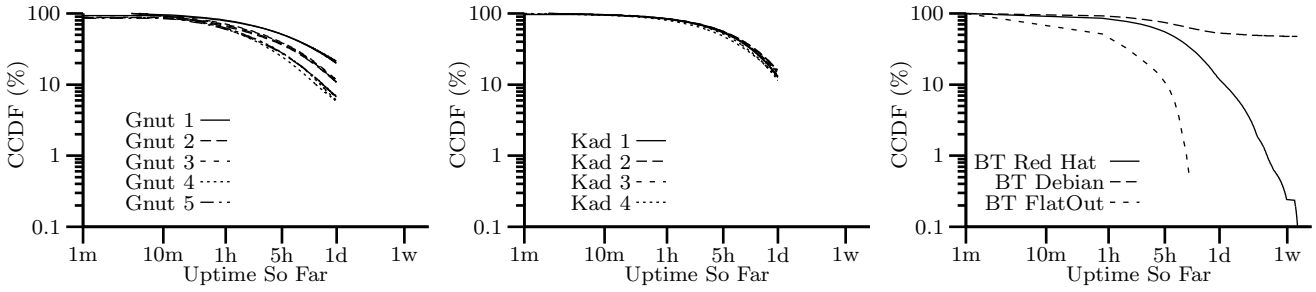
The Gnutella and Kad CCDFs (Figures 4(a) and 4(b)) exhibit slight downward curvature, very similar to that seen in BitTorrent over the same scale (Figure 4(c)). This suggests that they, too, are not heavy-tailed. However, due to the potential for significantly under-counting the number of long sessions in Gnutella and Kad (as described in Section 3.4), we cannot rule out the possibility that this curvature is a measurement artifact. We found that we could fit a log-normal distribution tightly to any of the Gnutella and Kad datasets. The Weibull distribution could also be fit reasonably well.

In summary, while most sessions are short (minutes), some sessions are very long (days or weeks). This differs from the exponential distribution, which exhibits values closer together in length, and heavy-tailed distributions, which have more pronounced extremes (years). We will explore the consequences of this further in the following subsections.

4.3 Lingering after Download Completion

We found the similarity between BitTorrent and the other systems surprising. While Gnutella and Kad applications

[4]In fact, the exponential distribution is a special case of the Weibull distribution where the shape parameter is $k = 1$.



(a) Gnutella (b) Kad (c) BitTorrent

Figure 6: CCDF of the uptime of peers that are in the system at any given moment

provide a keyword-search feature and manage the downloading of many files, BitTorrent is designed to download a single large file[5]. Intuitively, we would expect users to exit BitTorrent soon after the file transfer completes, which would tightly couple the session length with the transfer time. To investigate this issue, we divided the BitTorrent sessions into two parts: *(i)* the time to completely download the file and *(ii)* the additional time the peer lingers in the system. We call these the *completion time* and the *lingering time*, respectively. Figures 5(b) and 5(c) depicts the distribution of these two values, for sessions that download the entire file in a single session (16–61% of all sessions, see Table 3). Interestingly, many peers linger for a few hours after their download is complete, and a few peers linger for days or weeks, particularly in the Red Hat and Debian traces. By comparing Figures 5(a), 5(b), and 5(c), we can see that the tail of the session length distribution is dominated by peers who are lingering long after their download has completed. Long-lingering peers explain the fact that in steady-state more than 20% of active BitTorrent peers have completed their download and are continuing to participate, as shown by prior studies [7, 11]. In fact, *the good performance of BitTorrent may be largely due to user habits leading to the large number of active peers who have finished downloading but continue to contribute resources.*

4.4 Distribution of Peer Uptime

The previous subsections presented the distribution of session length across all sessions. However, they did not illustrate what combination of these peers might coexist in the system at any point of time. To address this issue, we turn our attention to how long the peers currently in the system have been present (their *uptime*).

To compute the uptime distribution, we slightly alter the methodology described in Section 3.3 to avoid bias towards short-lived peers as follows. Again, we divide each measurement window of length τ into two halves, A and B. For peers in B, we can either compute their uptime or we know that their uptime is at least $\frac{\tau}{2}$. For Gnutella and Kad, for each snapshot in B, we observe the uptime for each peer. For BitTorrent, where we have tracker logs instead of snapshots, we observe the state of the system once per minute during B.

Figure 6 shows the CCDF of the uptime for co-existing peers within a snapshot and reveals several interesting points. First, the uptime distribution exhibits very similar behavior across different systems, except for the Debian dataset. The uptime distribution in that dataset is heavily influenced by a large number of "seed" peers run by the Debian organization. Second, a significant fraction of peers have an uptime longer than half our measurement period (shown by a gap between the rightmost data-point and the x-axis). More specifically, roughly 10%–20% of peers per snapshot in Gnutella and Kad have an uptime longer than a day, and around 1–3% of BitTorrent peers have an uptime longer than 2 weeks for the Red Hat and Debian traces. Third, and most importantly, these distributions are heavily weighted towards uptimes longer than a couple of hours, *i.e.*, they show that the majority of peers in each snapshot are long-lived peers. For example, if we randomly select a peer from these systems, the probability that the selected peer has been up for more than five hours is roughly 40% in Gnutella, 55% in Kad, and 60% in BitTorrent for the Linux ISO images. This effect is significantly less pronounced, but still present, in the FlatOut trace, where 15% of active peers have been up for more than five hours.

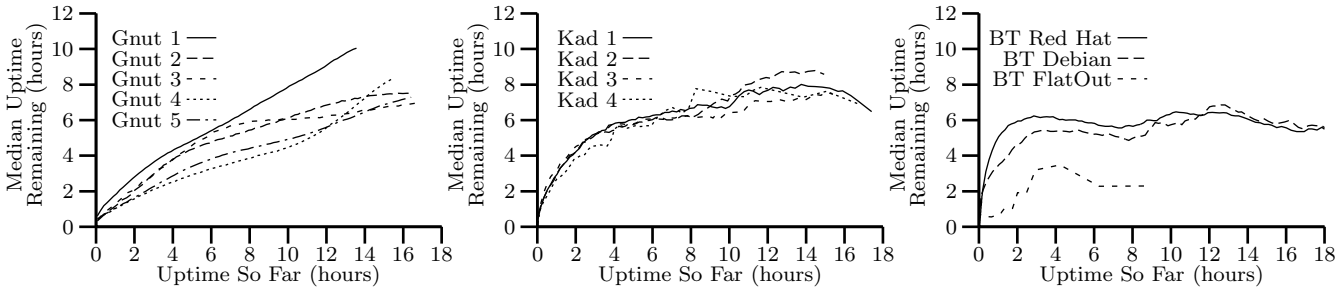
The combination of the uptime distribution (Figure 6) and the session length distribution (Figure 4) present an enlightening view of churn in P2P system as follows: *At any point of time, a majority of participating peers in the system are long-lived peers. However, the remaining small portion of short-lived peers join and leave the system at such a high rate that they constitute a relatively large portion of sessions.* Describing this from a different angle, the session length of a randomly selected session (Figure 4) is likely to be short whereas the uptime of a randomly selected active peer from the system (Figure 6) is likely to be long.

4.5 Uptime Predictability

One interesting question is whether a peer's uptime is a good predictor of it's remaining uptime. Although Figure 4 suggests this is the case, to empirically explore this property, we examine the correlation between uptime and remaining uptime at two levels. Figure 7 depicts the CCDF of the median[6] remaining uptime as a function of a peer's current uptime. It shows that while uptime is in general a good predictor of remaining uptime, its strength is different among candidate systems and for different uptime values. More specifically, peer uptime in Gnutella is a good indicator of remaining uptime regardless of uptime value; the median peer has a remaining uptime between 50% and

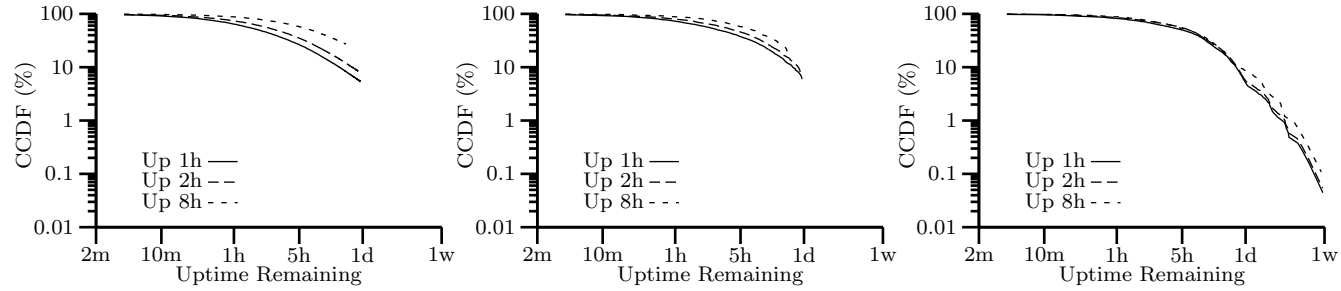
[5]Modern versions of BitTorrent can manage several file downloads within one instance of the application, but at the time of our data each file required a separate instance.

[6]Since we cannot measure the length of very long sessions as discussed in Section 3.3, we cannot compute the mean. However, we do know how *many* of these sessions there are and can thus find the median.



(a) Gnutella (b) Kad (c) BitTorrent

Figure 7: Median remaining uptime as a function of current uptime



(a) Gnutella 1 (b) Kad 1 (c) BitTorrent Red Hat

Figure 8: CCDF of Remaining Uptime for peers already up 1 hour, 2 hours, and 8 hours

100% of its uptime so far. However, the uptime of Kad peers is a stronger predictor of remaining uptime up to around 4 hours. Beyond that, the median peer's remaining uptime increases only slowly. Nevertheless, at $x = 16$ h, Kad peers have approximately the same remaining uptime as their Gnutella counterparts. In the BitTorrent datasets, we see a rapid increase in the median remaining uptime as the uptime approaches one hour, but peers up for at least two hours all have approximately the same remaining uptime. The FlatOut trace appears to wobble, which is likely noise given the relatively small number of sessions in this dataset.

While Figure 7 presents the median remaining uptime, we'd also like to understand the variance of this predictor. If the variance is low, that would make the predictor more useful. To explore this, Figure 8 shows the CCDF of the remaining uptime for peers currently up for 1 hour, 2 hours, and 8 hours, from one trace of each system. The CCDFs show that the reliability of the predictions is highly variable, covering a broad range of times. For example, in Gnutella around 50% of peers up for 8 hours will be up for at least another 8 hours. However, the bottom 20% of the peers will be up for less than 2 more hours, while the top 30% will be up for more than 16 hours! *In summary, our results show that while uptime is on average a good indicator of remaining uptime, it exhibits high variance. Therefore it should be used when a bad prediction does not have a major cost but collectively making better choices improves overall performance.*

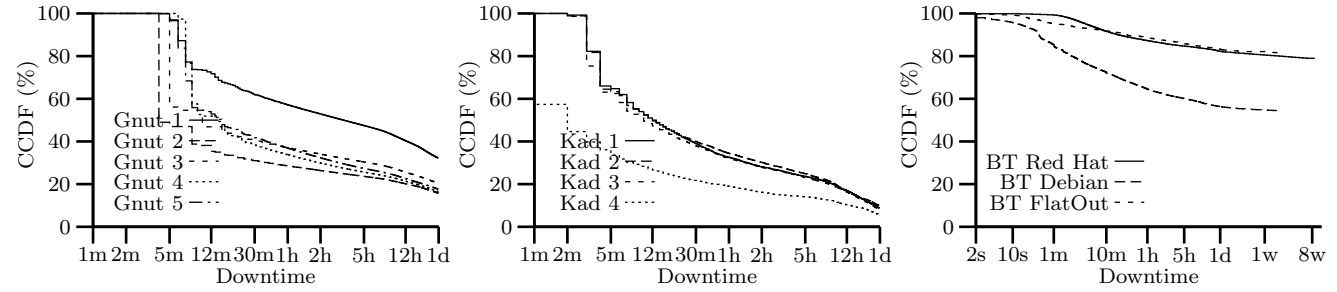
5 Peer-Level Characterization

In this section, we characterize the behavior of a peer across multiple appearances in the system. These characterizations are useful for both the design and evaluation of peer-to-peer systems. For example, in the previous section we showed that the current uptime of a peer is a good predictor for the remaining uptime. However, a peer that has just arrived can't use this fact to make much of a prediction. By examining multiple appearances, we can see if previous session lengths are good predictors of future session lengths. We examine the following characteristics: *(i)* the distribution of downtime, *(ii)* the correlation between consecutive session lengths, and *(iii)* the correlation of availability on consecutive days.

As described in Section 3.7, Kad is the only of our systems in which individual peers have a *persistent* unique identifier. Thus, our characterizations of Kad reflect the behavior of individual users, while our characterizations of Gnutella and BitTorrent reflect the behavior of IP address. Both types of characterization are useful. The behavior of users is important when persistent state is stored across sessions (such as the set of files that a user shares), while the behavior of IP addresses is important when we want to know if *any* peer will be available at the address (such as when bootstrapping).

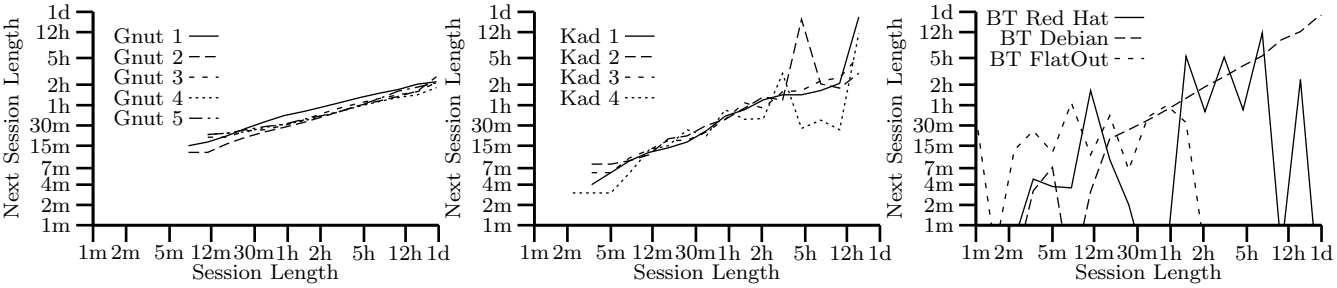
5.1 Distribution of Downtime

We define downtime as the interval between the moment a peer departs and its next arrival. To ensure an unbiased selection of downtime measurements, we again apply the window-halving methodology described in Section 3.3. The possibility in Gnutella and Kad that the crawler misses some peers, described in Section 3.4, has the opposite effect on downtime observations. Rather than increasing the number of long events, it may significantly increase the number of short events. Figure 9 presents the distribution of downtime for Kad (based on ID) as well as Gnutella and BitTorrent (based on IP address) and Table 3 lists the number of observations from each dataset. The gaps between the rightmost data-point and the x-axis represent the fraction of downtime events which were too long to measure without bias as well as instances where the peer never returns (*i.e.*, in-



(a) Gnutella (based on IP address) (b) Kad (based on node ID) (c) BitTorrent (based on IP address)

Figure 9: Downtime Distribution CCDFs



(a) Gnutella (based on IP address) (b) Kad (based on node ID) (c) BitTorrent (based on IP address)

Figure 10: Correlation between Consecutive Session Length

finite downtime). One interesting feature of these figures is that the behavior is approximately the same in Gnutella and Kad, despite one of the measurements being based on ID while the other is based on IP.

These CCDFs reveal a number of interesting qualities. First, the behavior in BitTorrent is significantly different. In Gnutella and Kad, most peers that depart return within 1 day (70–85% in Gnutella and 90% in Kad), while in BitTorrent most peers do not appear to return at all. Second, in Gnutella and Kad peers that will return have a strong tendency to return sooner rather than later. For example, departed peers return within 1 hour 40–70% of the time in Gnutella and 65–80% of the time in Kad. In BitTorrent, a departed peer is somewhat more likely to reappear withing a few minutes, but after that has a relatively equal probability of returning at any point between 10 minutes and 1 week, if it returns at all. *In summary, our results show that in a content-distribution system, departed peers are unlikely to return. However, those that return are equally likely to return at any time. In contrast, most peers in file-sharing systems do return, and there is a strong tendency to return sooner rather than later.*

5.2 Correlation in Session Length

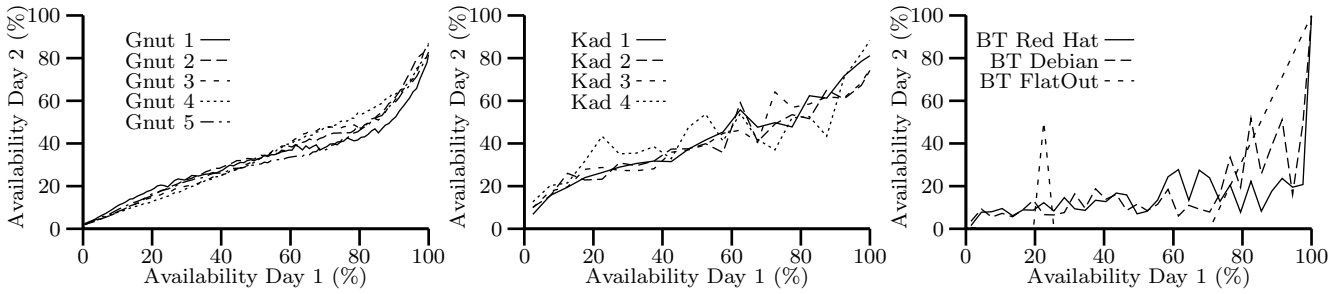
Another interesting question is "How correlated are session lengths across different appearance of a single peer?" Characterizing such an effect illustrates whether past session lengths of a peer are good predictor for its future session lengths. For example, in Gnutella, a peer can promote itself to an Ultrapeer earlier if it can reliable estimate that its remaining uptime is likely long. For a peer that appears n times during our measurement, there are $(n-1)$ pairs of consecutive sessions. Given all pairs of consecutive session lengths in our measurement window, Figure 10 presents the distribution of the median[6] of the second session's length for first sessions with length x. These figures show that there is a strong correlation between consecutive session lengths in Kad (based on ID) and in Gnutella (based on IP address). However, session lengths in BitTorrent do not exhibit a clear correlation. This result is not surprising because the pattern of participation in BitTorrent is likely to be different from Kad or Gnutella. In summary, *past session length of a peer is a good predictor of its next session length in both structured and unstructured file-sharing applications, but not in content-distribution systems such as BitTorrent.*

5.3 Correlation in Availability

In this section we examine correlation in the availability of peers across consecutive windows of time, which is a coarse measure of the peer's overall participation regardless of its pattern of appearance. The availability of a peer is the fraction of time the peer is available in the system. For example, a node with 50% availability during one day, might appear just once for 12 hours, or appear 4 times and stay 3 hours during each appearance.

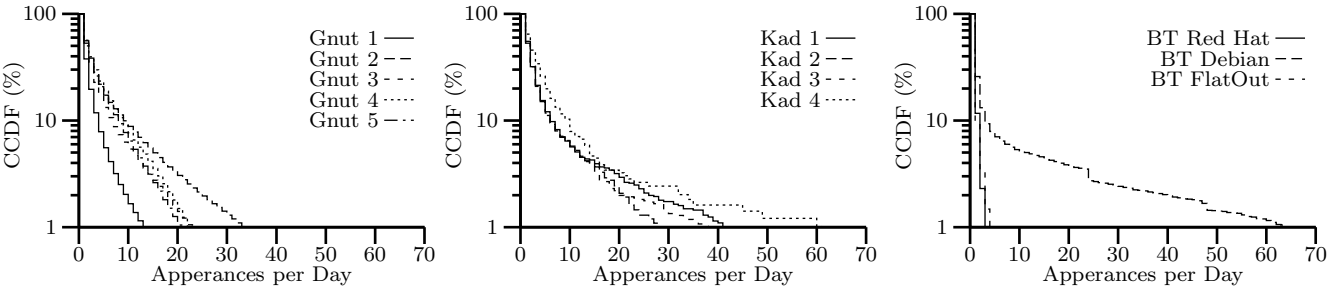
To study peer availability, we divide each two-day dataset into two windows of one-day length, and examine the correlation between the availability of peers in two consecutive days. For the longer BitTorrent datasets, we use each pair of consecutive days. Figure 11 depicts the median availability in the second day as a function of the availability on the first day. These results show that a strong correlation exists between availability in Gnutella and Kad. However, BitTorrent does not exhibit such a clear correlation, except for one case: if a peer is up for a full 24 hours, it is very likely to be up for the next full 24 hours as well.

One interesting question is how many appearances per day a peer makes. Figure 12 presents the distribution of the number of appearances per day. It shows that more than half the peers in both systems appear only once per day while a very small number of clients may return to the system very often (up to 60 times per day). Bhagwan *et*



(a) Gnutella (based on IP address) (b) Kad (based on node ID) (c) BitTorrent (based on IP address)

Figure 11: Correlation in Availability



(a) Gnutella (based on IP address) (b) Kad (based on node ID) (c) BitTorrent (based on IP address)

Figure 12: Appearances

al. [3] show the availability of different peers are mostly independent. Their results are complimentary to ours, since we examine the correlation in availability of a *single peer* across multiple measurement windows while they looked for correlations across multiple peers in the same measurement window. *In summary, the availability of individual peers, across two consecutive days are strongly correlated. Furthermore, most observed peers appear only once per day.*

6 Design Implications

In this section, we discuss a couple of key implications of our findings on the design of P2P applications. To gracefully cope with churn, P2P systems must be able to efficiently handle the significant fraction of peers who join the system for just a few minutes. In particular, churn could significantly affect the connectivity of P2P overlays. To improve resiliency against churn, each peer should prefer to maintain routing information about other stable long-lived peers. This approach implies that long-lived peers maintain state about each other and provide a backbone of connectivity among peers. For example, peers in a DHT should select long-lived peers as neighbors to ensure better connectivity and resiliency against churn (*e.g.*, [16, 20]).

Our results suggest two reliable strategies for identifying long-lived peers are as follows: *(i)* randomly selecting a set of peers that are all active at the same time is likely to capture long-lived peers since the distribution of uptime among peers up at any given moment (Figure 6) is weighted more heavily towards long-lived peers, or *(ii)* weight observed peers by the number of times they are observed. The key point is that observations made *over time* become skewed towards the larger number of short-lived peers if the observations are not weighted back in favor of the long-lived peers.

Our results motivate a scalable and low cost bootstrapping mechanism that does not require a central bootstrapping node. In particular, we showed that 20%–30% of peers at any moment have an uptime longer than one day. This implies that each peer can select and cache IP addresses of several long-lived peers using one of the strategies we describe above. To connect to the system at any later time, each peer can contact cached long-lived peers to locate a participating peer in the system. Maintaining a sufficiently large number of long-lived peers ensures that each peer can always successfully bootstrap to the system without the need to contact a well-known, centralized address. In contrast, selecting cached peers based on a first-in first-out strategy, as currently implemented by Gnutella clients, tends to generate a list of short-lived peers that are unlikely to be available.

Finally, the strong correlation in the length of consecutive sessions and availability implies that a P2P application can roughly estimate the duration of its session length (or availability per day) based on its last observed behavior. The advantage of this approach is that it does not require the application to wait for a while to measure uptime as a sign to predict user behavior for the rest of the session.

7 Related Work

We are not aware of any prior study that focuses primarily on churn in P2P networks. However, several studies present a passing investigation of session length as part of wider characterizations of P2P applications. We divide these studies into two groups based on their measurement technique as follows:

Passive Monitoring: As part of a study on P2P flows in a large ISP network, Sen *et al.* [24] use passive measurement at several routers to monitor flows in FastTrack, Gnutella, and Direct-Connect. They present a CDF of the duration an IP address is active (the *ontime*), based on a threshold, $\delta = 30$ minutes, of inactivity. They show that the ontime is heavy-tailed, but does not follow a Zipf distribution when

ranked. As part of a study on workload characterization in Kazaa, Gummadi *et al.* [10] present session lengths based on passive monitoring of a router at the University of Washington. They found that session lengths are heavy-tailed, with a median session length of 2.4 minutes while the 90th percentile is 28.25 minutes. In general, passive monitoring techniques to characterize churn are likely to underestimate session lengths because some peers may not be continuously generating traffic through the observation point. Difficulties in correctly identifying peer-to-peer flows [12] can also limit measurement accuracy. Furthermore, it is difficult to determine whether the subset of captured users is representative of the entire P2P user population, especially if data is collected at a small number of measurement points.

Active Probing: Several studies use active probing or crawling to characterize P2P networks and present the behavior of session length across peers. Chu *et al.* [6] present the session length distribution in Napster and Gnutella and fit it to a log-quadratic distribution. In their insightful characterization of peers in Napster and Gnutella, Saroiu *et al.* [23] present a CDF of session durations, showing session lengths are heavily skewed. They also present CDFs of peer availability in these systems. Bustamante and Qiao [5] monitored peers in the Gnutella network to motivate preferential neighbor selection based on uptime. Their tool measures the length of sessions for peers which return to the network during their measurement period, and they fit peer session lengths to the Pareto distribution. More recently, Liang *et al.* [17] similarly provide a CDF of session lengths for super-nodes in the Kazaa network, based on active probing. Finally, Bhagwan *et al.* [3] examine availability in the Overnet DHT using probing. They show there is little correlation between the availability of different peers. Stutzbach *et al.* used Cruiser to illustrate the distribution of peer uptime is not Poisson and show how churn leads to biased connectivity among long-lived peers [27]. Prior studies of BitTorrent [11, 21] have used tracker logs to show that session lengths are heavily skewed.

Compared to all the above studies, our work takes a significant step towards deepening our understanding of churn across different P2P systems. We identify common pitfalls in measuring churn and either address them or bound their effect. We believe that these pitfalls could be the main contributing factor to the conflicting results reported by previous studies. Leveraging our more accurate measurements, we examine different aspects of churn and compare these characteristics among different P2P systems.

8 Conclusions and Future Work

This paper took a major step towards increasing our understanding of churn by characterizing different aspects of peer dynamics in three different classes of P2P systems: Gnutella, Kad and BitTorrent. We identified an array of measurement pitfalls in characterizing churn and either addressed them or bound their resulting error. We characterize churn at both group-level as well as peer- and IP-level. Our main findings, listed in Section 1, present new insight into peer dynamics which can be used in the design and evaluation of churn-aware P2P applications.

We are pursuing this work in several directions. We are investigating potential heuristics to identify when a peer was missed during a snapshot versus a genuine departure, which will allow us to more closely study the behavior of long lived peers in these systems. Additionally, we plan to examine the underlying characteristics of churn by studying how user habits are correlated with time of day, geographical location, and file preferences.

9 References

[1] eMule. http://www.emule-project.net, 2005.
[2] slyck.com. http://www.slyck.com, 2005.
[3] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *IPTPS*, 2003.
[4] A. Bharambe and C. Herley. Understanding and Deconstructing BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, 2005.
[5] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *WCW*, 2003.
[6] J. Chu, K. Labonte, and B. N. Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. In *ITCom Conference*, 2002.
[7] B. Cohen. Incentives Build Robustness in BitTorrent. http://www.bittorrent.com/bittorrentecon.pdf, 2003.
[8] M. E. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *ToN*, 5(6), 1997.
[9] Free Peers, Inc. BearShare Network Statistics. http://www.bearshare.com/stats/, 2005.
[10] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *SOSP*, 2003.
[11] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, 2004.
[12] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P dying or just hiding? In *Globecom*, 2004.
[13] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *IMC*, 2005.
[14] D. Leonard, V. Rai, and D. Loguinov. On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks. In *SIGMETRICS*, 2005.
[15] J. Li, J. Stribling, F. Kaashoek, R. Morris, and T. Gil. A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs under Churn. In *INFOCOM*, 2005.
[16] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient Management of DHT Routing Tables. In *Networked Systems Design and Implementation*, 2005.
[17] J. Liang, R. Kumar, and K. W. Ross. The KaZaA Overlay: A Measurement Study. *Computer Networks Journal*, 2005.
[18] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Principles of Distributed Computing*, 2002.
[19] Lime Wire LLC. Crawler Compatability. Gnutella Developer's Forum, 2003.
[20] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *IPTPS*, 2002.
[21] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-sharing System: Measurements and Analysis. In *IPTPS*, 2005.
[22] S. Rhea, D. Geels, and J. Kubiatowicz. Handling Churn in a DHT. In *USENIX*, 2004.
[23] S. Saroiu, P. K. Gummadi, and S. D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems Journal*, 8(5), 2002.
[24] S. Sen and J. Wang. Analyzing Peer-To-Peer Traffic Across Large Networks. *IEEE/ACM ToN*, 12(2), 2004.
[25] K. Sripanidkulchai, B. Maggs, and H. Zhang. An Analysis of Live Streaming Workloads on the Internet. In *IMC*, 2004.
[26] D. Stutzbach and R. Rejaie. Capturing Accurate Snapshots of the Gnutella Network. In *Global Internet Symposium*, 2005.
[27] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *IMC*, 2005.