Evaluating the Accuracy of Captured Snapshots by Peer-to-Peer Crawlers

Daniel Stutzbach, Reza Rejaie
{*agthorr,reza*}*@cs.uoregon.edu*

I. Introduction

During recent years, the increasing popularity of peer-to-peer (P2P) networks has led to growing interest in characterizing dynamics in P2P systems, in particular the dynamics of peer participation (*i.e.*, churn) and its impact on the resulting overlay topology and resource availability (*e.g.*, [1], [2], [3]). These characterizations provide deeper insight into the behavior of P2P systems, essential for proper design and effective evaluation. A common technique to characterize the dynamics of a P2P system is to capture snapshots of the system using a crawler. Examination of individual snapshots reveals various properties of the system (*e.g.*, the size and diameter of the network and node degree distribution), while the comparison of sequential snapshots identifies dynamics of various properties as a function of time (*e.g.*, churn rate). The accuracy of the conducted analysis based on the above methodology directly depends on the accuracy of the captured snapshots of the target P2P system. However, to our knowledge, previous studies have not verified the accuracy of their captured snapshots. Therefore, these studies were unable to address the impact of snapshot accuracy on correctness of their presented characterizations. A perfect snapshot of a P2P system is captured if a crawl is complete and instantaneous. However, in practice neither of these conditions are met for the following reasons.

1) Progressive Nature of Crawling: Crawlers discover participating peers in a progressive fashion. Therefore, capturing a snapshot may take a long time depending on the speed of the crawler and its available resources (*i.e.*, access link bandwidth and processing power). Given that P2P systems are moving targets, as the duration of the crawl increases, the captured snapshot becomes more *distorted* because more nodes might arrive or depart during the crawl. Furthermore, the duration of the crawl determines the time granularity between comparing back to back snapshots. Previous studies typically crawled their target P2P systems in 30 minutes to two hours (*e.g.*, [4], [5]).

2) Unreachable Peers: Captured snapshots are often incomplete because a non-negligible portion of discovered peers are not reachable by the crawler. Previous studies often assumed that unreachable peers have departed the system and simply excluded them from the captured snapshots. However, many of these unreachable peers have not left the system. Instead, they are either located behind a firewall (*i.e.*, NATed) or, more interestingly, are receiving too many SYN packets. Therefore, ignoring these peers certainly introduces a non-negligible error in the captured snapshot.

In summary, the accuracy of captured snapshots by P2P crawlers can be significantly affected by both the duration of a crawl and the ratio of unreachable peers. Determining the accuracy of captured snapshots of a P2P system is fundamentally difficult because a more accurate reference snapshot for comparison is not available. There is also a tradeoff between the duration of a crawl and the completeness of the captured snapshot. Furthermore, the desired characterization of P2P systems determines the granularity and type of collected information in each snapshot. For example, a study on churn only requires information about participating peers and may not need to directly contact all peers. In contrast, to study the overlay topology, a captured snapshot should include all edges of the overlay which requires the crawler to directly contact every peer otherwise a connection between two unvisited peers would be missed.

In this paper, our main goal is to quantify the accuracy of captured snapshots by P2P crawlers. More specifically, we try to answer the following key questions:

- How do the speed of crawling and the ratio of unreachable peers affect accuracy?
- What is the fundamental tradeoff between the duration of the crawl and completeness of the snapshot?
- What is the impact of snapshot accuracy on the analysis of various characteristics of the system?

We focus on the Gnutella network as a representative P2P system because it is the largest, open P2P system on today's Internet. However, we believe that most of the raised issues and findings are generic and applicable to other P2P systems. To answer the above questions, we try to push the envelope and capture more accurate snapshots of the Gnutella network by increasing crawling speed and resolving the status of unreachable peers. We also introduce techniques for estimating the accuracy without having a perfect reference snapshot. These more accurate snapshots are used as reference to quantify the impact of crawling speed and unreachable peers on snapshot accuracy. These snapshots also enable us to examine several important tradeoffs during crawling. More importantly, we provide deeper insight into the dynamics of the Gnutella network over short timescales which was not feasible in previous studies with slow crawlers.

To achieve these goals, we developed a fast and efficient Gnutella crawler, called *Cruiser*, that is able to capture a complete snapshot of the Gnutella network in around 5 minutes with 6 off-the-shelf desktop PCs. Cruiser achieves this significant reduction in crawl time as follows: *(i)* it leverages several features of modern Gnutella including its semi-structured topology, efficient new handshake mechanism, and high degree of node connectivity among top-level peers. *(ii)* it substantially increases the degree of concurrency during the crawling process by deploying a master-slave architecture and allowing each slave crawler to contact hundreds of peers simultaneously. We also address several systems issues and performance bottlenecks in the design of P2P crawlers. Our preliminary results show that Cruiser can capture accurate snapshots of the Gnutella network. Furthermore, we quantify the impact of crawl duration and crawler speed on the accuracy of captured snapshots. The rest of this extended abstract is organized as follows: In Section II, we briefly present key features of the modern Gnutella protocol. Section III provides a short overview of Cruiser. Finally, we present some of our preliminary results in Section IV. More details on the design and evaluation of Cruiser, along with further results will be presented in the final version of the paper.

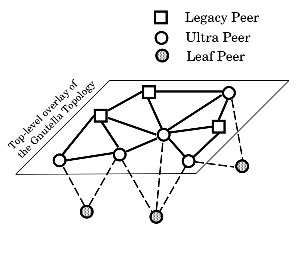

Fig. 1. Semi-Structured Topology of Modern Gnutella

II. Modern Gnutella

In this section, we briefly describe a few key features of modern Gnutella [6], [7] that are used by Cruiser. The original Gnutella protocol had limited scalability due to its flat overlay. To address this limitation, most modern Gnutella clients implement a two-tiered network structure by dividing peers into two groups: *ultrapeers* (or super-peers) and *leaf* peers. As shown in Figure 1, each ultrapeer neighbors with several other ultrapeers within a top-level overlay. The majority of the peers are leaves that are connected to the overlay through a few ultrapeers. Furthermore, modern Gnutella clients implement a mechanism that allows high-bandwidth, un-firewalled leaf peers to become ultrapeers in order to maintain a proper ultrapeer-to-leaf ratio in the overlay. Those peers that do not implement the ultrapeer feature can only reside in the top-level overlay and do not accept any leaves. We refer to these peers as *legacy* peers. We also refer to the legacy peers and ultrapeers collectively as the *top-level* peers. Our recent measurements [8] reveal that the degree of connectivity among top-level peers is much higher than that in flat original Gnutella.

Finally, modern Gnutella clients implement a special handshaking feature that enables the crawler to quickly query a peer for a list of its current neighbors. Previous crawlers relied on other features of the Gnutella protocol, namely Ping-Pong messages, to retrieve this information. These techniques were less efficient and potentially less reliable.

III. The Gnutella Cruiser

Our primary goal in the design of Cruiser is to significantly improve crawling speed compared to previously reported crawlers in order to improve the accuracy of captured snapshots. We deploy several basic techniques and features to achieve this design goal: First, the handshaking mechanism in modern Gnutella enables Cruiser to quickly obtain a fresh list of current neighbors from each peer. Second, Cruiser leverages the two-tier structure of the Gnutella network by only crawling the top-level peers. Since each leaf must be connected to an ultrapeer, this approach enables us to capture all the nodes and links of the overlay by contacting a relatively small fraction of all peers. Furthermore, the high degree of peer connectivity within the top level overlay substantially increases the rate of discovery for new ultrapeers. Overall, this strategy leads to a major reduction in the duration of a crawl without loss of information.

Third, Cruiser employs a master-slave architecture in order to achieve a high degree of concurrency and to effectively utilize available resources on multiple desktop PCs. A master process coordinates among multiple slave processes where each slave acts as a virtually independent crawler and crawls the network in parallel. To further improve the degree of concurrency, each slave process uses asynchronous communications to open hundreds of connections in parallel. Although Cruiser uses considerable local resources, its offered load on individual Gnutella peers is minimal since each top-level peer is contacted only once per snapshot.

Fourth, Cruiser implements an adaptive load management mechanism to ensure that slaves processes remain busy but do not become overwhelmed. This is important for the steady progress of the crawl especially when different slave nodes have heterogeneous processing capabilities. Toward this end, Cruiser enables each slave process to adjust its own load (*i.e.*, number of open connections) using an AIMD algorithm similar to TCP's congestion control mechanism.

Unreachable Peers: A non-negligible subset of contacted peers in each crawl timeout or refuse TCP connections. Peers are unreachable when they have already left the system (*i.e.*, departed), they are located behind a firewall (*i.e.*, NATed), or they receive too many SYN packets (*i.e.*, overloaded). While the problem with departed and NATed peers were already raised in previous studies, we discovered unreachable peers that were overloaded, and refused and then accepted TCP connections sporadically over a short period of time (*i.e.*, within a single minute they would alternate repeatedly between accepting and refusing connections [8]). Unreachable ultrapeers can introduce the following errors in a captured snapshot: *(i)* including unreachable peers that were departed, *(ii)* missing branches between unreachable ultrapeers and their leaves, and *(iii)* missing branches between two unreachable top-level peers. To minimize these errors, it is important to quantify what portion of unreachable peers were departed versus firewalled or overloaded. Unfortunately, there is no reliable test to firmly verify the status of an unreachable peers among the three possible scenarios, since both overloaded, NATed, and departed peers may or may not reply to SYN packets.

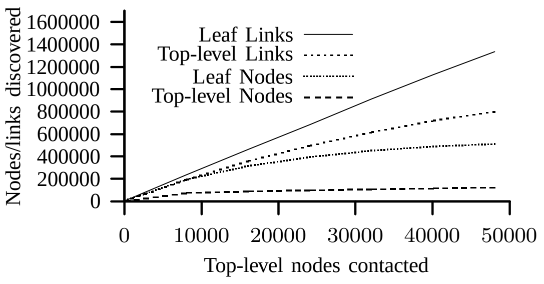

Fig. 2. Cumulative information per contacted ultrapeer

We used the following approach to identify the status of unreachable peers. We performed back-to-back crawls to capture two snapshots. Then, the unreachable peers in the first snapshot that were missing from the second snapshot, were considered "departed peers" during the first snapshot. This is feasible because the duration of each crawl is very short (around 5 minutes). To distinguish overloaded peers from NATed peers, we created a "profile" of Gnutella peers that contains various information about each discovered peer (*e.g.*, the number of times each peer was successfully contacted). The profile is updated by Cruiser after each crawl. After excluding departed peers from a snapshot, the status of remaining unreachable peers in each crawl is quickly determined as follows: if a peer was successfully contacted at least once in previous crawls, it is considered to be overloaded. Otherwise, it is likely to be NATed. Clearly, the accuracy of this approach increases with the number of previous crawls (*i.e.*, the number of attempted contacts for each unreachable peer). We also group the NATed peers in the profile that have the same prefix in order to identify those ISPs that deploy a firewall for their clients. Currently, our Gnutella profile consists of more than one million peers, and is growing.

IV. Results

We have been running Cruiser on six 1Ghz Pentium III PCs in our lab during the past couple of months and have captured several hundred snapshots of the Gnutella network. In this section, we present a preview of our results to demonstrate the ability of Cruiser to capture accurate snapshots and examine several key tradeoffs. Note that the profile is not used to reduce error in the presented results. We will report the benefits of the profile along with the impact of unreachable peers in the final revision of the paper. **Completeness of Snapshots:** To examine the completeness of captured snapshots by Cruiser, we kept track of the following variables during each single crawl: number of discovered top-level peers, number of leaves, number of links between ultrapeers, and number of links to leaves. Figure 2 presents variations of these four variables as a function of number of contacted peers thus far in a sample crawl. Note that the number of discovered top-level peers as well as leaves curve off which is evidence that Cruiser has captured a majority of the participating peers. Links between top-level peers curves off somewhat. Finally, links to leaves is linearly increasing with the number of top-level peers because each top-level peers provide a unique set of links between itself and its leaves.

Impact of Crawling Duration: To examine the impact of crawl duration on the accuracy of captured snapshots, we modified Cruiser to stop the crawl after a specified period. Then, we performed two back-to-back crawls and repeated this process for different durations. We define δ_+ and δ_- as the number of new and missing peers in the second snapshot compared to the first one, respectively (normalized by the total number of peers in the first crawl). Figures 3 presents the sum $\delta = \delta_+ + \delta_-$ as well as the total number of discovered peers as a function of the crawl duration for all participating peers (both top-level and leaves). During short crawls (left side of the graph), δ is high because the captured snapshot is incomplete, and each crawl captures a different subset. As the duration of crawl increases, δ decreases which indicates that the captured snapshot becomes more complete. Increasing the crawl length beyond two minute does not decrease δ any further, and achieves marginal increase in number of discovered peers. This figure reveals a few important points. First, there exists a "sweet spot" for crawl duration beyond which crawling has diminishing returns if the goal is simply to capture the population. Second, for sufficiently long crawls, Cruiser can capture a relatively un-stretched snapshot. Third, the change of $\delta = 0.08$ is an upper-bound on the distortion due to the passage of time as Cruiser runs. The relatively flat delta on the right suggest that a small but significant fraction of the network is unstable and turns over quickly.

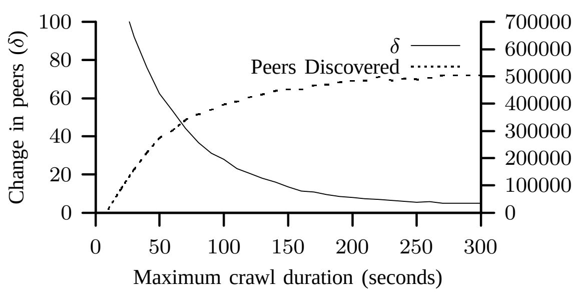

Fig. 3. Error as a function of maximum crawl duration, generated by running two crawls back-to-back for each x-value and compute the δ. Mean of 8 runs.

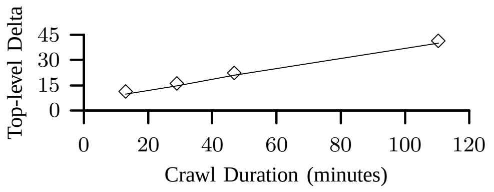

Fig. 4. Effects of crawling speed

Impact of Crawling Speed: To examine the impact of crawling speed on the accuracy of captured snapshots, we decreased the speed of Cruiser by reducing the number of parallel connections that each slave process can open. Figure 4 depicts the error in top-level peers between snapshots from back-to-back crawls as a function of crawl duration. The first snapshot was captured with the maximum speed and serves as a reference whereas the speed (and thus duration) of the second snapshot has changed. The duration of the second snapshot is shown as x value. This figure clearly demonstrates that the accuracy of snapshots decreases significantly for longer crawls.

References

[1] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *International Workshop on Peer-to-Peer Systems*, 2003.

[2] Stefan Saroiu, P. Krishna Gummadi, , and Steven D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Multimedia Systems Journal*, vol. 8, no. 5, Nov. 2002.

[3] David Liben-Nowell, Hari Balakrishnan, , and David Karger, "Analysis of the Evolution of Peer-to-Peer Systems," in *Principles of Distributed Computing*, Monterey, CA, July 2002.

[4] clip2.com, "Gnutella: To the Bandwidth Barrier and Beyond," Nov. 2000.

[5] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design," *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.

[6] Anurag Singla and Christopher Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability," Gnutella Developer's Forum, Nov. 2002.

[7] Lime Wire LLC, "Crawler Compatability," Gnutella Developer's Forum, Jan. 2003.

[8] Daniel Stutzbach and Reza Rejaie, "Characterizing Today's Semi-Structured Gnutella Network," Tech. Rep. CIS-TR-04-02, University of Oregon, 2004.