Crowd Crawling: Towards Collaborative Data Collection for Large-scale Online Social Networks

Cong Ding[†], Yang Chen[‡], and Xiaoming Fu[†]
[†]Institute of Computer Science, University of Göttingen, 37077 Göttingen, Germany
[‡]Department of Computer Science, Duke University, Durham, NC 27708, USA
{cong, fu}@cs.uni-goettingen.de, ychen@cs.duke.edu

ABSTRACT

The emerging research for online social networks (OSNs) requires a huge amount of data. However, OSN sites typically enforce restrictions for data crawling, such as request rate limiting on a per-IP basis. It becomes challenging for an individual research group to collect sufficient data by using its own network resources. In this paper, we introduce and motivate *crowd crawling*, which allows multiple research groups to efficiently crawl data in a collaborative way. Crowd crawling is carefully designed by addressing several practical challenges including resource diversity of different partners, strict request rate limiting from OSN providers, and data fidelity. We implemented and deployed a crowd crawling prototype on PlanetLab, and demonstrated its performance through evaluations. We have made the datasets crawled in our evaluation publicly available.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Online Social Network; Data Collection; Crowd Crawling

1. INTRODUCTION

Online social networks (OSNs) have witnessed an evolutionary development and deployment, which attract a lot of attention all over the world. Recently, they have become an indispensable component in people's daily life, and a hot research topic involving many professionals from a variety of fields [11]. Given the huge amount of users, most research on this field needs a high volume of data from popular OSN sites, such as Facebook [6,7], Twitter [12,16], Renren [10,13], Weibo [5,8], and Foursquare [9]. Most of the datasets used in OSN research were crawled by individual groups using their own network resources and computing clusters. However, as most of the OSN sites have very strict request rate limiting on IP addresses and user accounts, it becomes challenging for an individual research group to crawl sufficient data covering a significant portion of an entire OSN site. A main reason is the number of IP addresses a research group owns is typically limited, especially for universities outside the United States. For example, the computer networks group at University of Göttingen owns only 14 public IPv4 addresses. It is difficult for a single group to compete against the increasing crawling requirement and unavoidable request rate limiting. For example, Weibo allows up to 1000 API requests per hour per IP address, and 150 requests per hour per user account. Moreover, different research groups might be interested in the same OSN site, while they crawl data independently nowadays. Such uncoordinated crawling may (1) waste computing and network resources as well as researchers' programming effort; (2) introduce unnecessary overhead (and extra cost) for OSN service providers; (3) lead to even stricter request rate limiting policies from OSN service providers; (4) not be able to provide universal/standard datasets as ground truth for the research community to develop and evaluate different research methods on social network analysis.

In this paper, we introduce and motivate *crowd crawling*, which leverages resources from various research groups to achieve large-scale distributed crawling. Crowd crawling allows several groups, so-called *partners*, to work in a collaborative way. In the collaboration, we require every partner to contribute crawled data beyond a pre-defined base line, if they want to benefit from the contents crawled by other partners. This collaboration framework has been used in PlanetLab in the past 10 years and proved working well [1].

We design and implement a crowd crawling prototype, and deploy it on PlanetLab to demonstrate its performance. Crowd crawling can achieve efficient data collection, while obeying OSNs' request rate limiting policies. There are three main challenges for building a crowd crawling system.

- **Resource diversity of different partners.** Different partners might allocate different amount of resources for crawling. Such diversity might include different number of crawlers, different hardware configurations of crawlers, and diverse bandwidths. These factors will lead to different data fetching efficiency. Therefore, the crawling system needs to be able to efficiently assign crawling tasks to different crawlers. Also, the system should be robust to several dynamic issues including node churning and bandwidth instability.

COSN'13, October 7–8, 2013, Boston, Massachusetts, USA.

http://dx.doi.org/10.1145/2512938.2512958.

- **Different rate limiting policies from OSN providers.** Most OSN sites enforce request rate limiting policies. To date, mainstream OSN sites, such as Facebook, Twitter and Weibo, use IP- and account-based policies, which require careful consideration in our design. Also, for new rate limiting proposals such as link-based rate limiting [13], we introduce a compatible way to achieve efficient crawling.
- **Data fidelity.** The distributed nature of crowd crawling introduces data fidelity issue, which might be caused by malicious attacks. A straightforward motivation of such attacks is free-riding. Some partners might just want to benefit from other partners' results without contributing their own resources accordingly. Therefore, they pretend to provide enough crawled results, while in fact they just submit fake results. We propose a lightweight, flexible, and easy-to-configure solution to prevent such attacks.

The main contributions of this paper are three-fold. First, we propose and motivate crowd crawling, an integrated distributed data collection framework for online social networks. Second, we design a practical crowd crawling system, which works in an efficient and reliable way. Third, we implement a proof-of-concept prototype of crowd crawling, and deploy it on PlanetLab. We evaluate the crowd crawling prototype using a viable Weibo crawling example.

The reset of this paper is organized as follows. We introduce crowd crawling system design in Section 2. In Section 3, we provide implementation details of the proof-of-concept prototype, and discuss the evaluation result on PlanetLab. We then present related work in Section 4. Finally, Section 5 discusses several open issues for further investigations in this field, and concludes this paper.

2. SYSTEM DESIGN

In this section, we first present the high-level framework of crowd crawling (Section 2.1), by introducing three key components and describing how they collaborate. Then we discuss several practical considerations in crowd crawling.

In mainstream OSN sites like Facebook, Twitter, Weibo, Renren, and Foursquare, every user is represented by a unique identifier (UID). For instance, in Facebook, UIDs are 64bit integers. The overall crawling project is divided into small tasks, and each task consists of a set of UIDs. To crawl OSNs, we have to get UIDs first (Section 2.2). With the UIDs in hand, TAM will assign tasks to crawlers. The design of crowd crawling has to deal with the three challenges: handling resource diversity of partners' computers (Section 2.3), bypassing strict rate limiting (Section 2.4), and data fidelity (Section 2.5).

2.1 Overview

As shown in Figure 1, there are three components in the system: crawlers, task assignment module (TAM), and result collection module (RCM). Every partner is required to contribute some computers as crawlers, and provide Internet connectivity for these crawlers. For every data crawling project, we need the partners to elect a *coordinator*, which is trustworthy and responsible for assigning tasks to other partners. This coordinator is also in charge of collecting crawled results from other partners, and preparing an aggregate dataset. Both TAM and RCM are maintained by

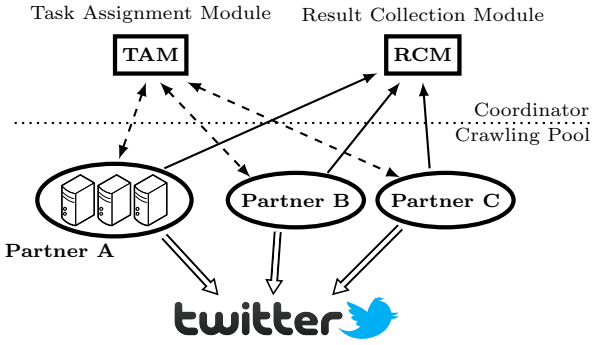

Figure 1: Crowd crawling architecture. ◄- - ► denotes task assignment, and ⟶ denotes crawled data delivery. The task assignment module (TAM) and result collection module (RCM) are controlled by the coordinator. Crawlers are owned by partners, and crawl OSNs collaboratively.

the coordinator. TAM is built based on a distributed hash table (DHT); and RCM is installed in a cluster of distributed servers to ensure scalability. TAM and RCM synchronize results periodically to ensure all the assigned tasks are finished properly and to detect prospective malicious behaviors from crawlers.

Crawlers. After getting a task from TAM, the crawler collects data of each UID in the task one by one. It sends HTTP request to the OSN site and collects data via parsing HTML documents or decoding JSON objects. Finally, the crawler sends the crawled data to RCM, and then repeats all the steps for the next task. We set every crawler's default crawling speed at a legitimate request rate (e.g., 80% of the request rate limiting), to avoid violating the request rate limiting policies.

Task assignment module. Task assignment module (TAM) acts as a controller to moderate the crawling procedure. It arranges tasks to crawlers, and obtains the latest crawling progress information from RCM. To maintain the system state, it keeps three sets of UIDs: an unassigned UID set, an assigned-but-unfinished UID set, a finished UID set. Each of these three sets is stored in a DHT. To reflect the latest system state, TAM moves UIDs between the three sets based on the status of tasks and dynamics of crawlers (Section 2.3).

Result collection module. Result collection module (RCM) is responsible for aggregating crawling results. It stores received crawling results into a local database, and updates the crawling progress to TAM. On one hand, it discovers unknown UIDs from the crawling results, i.e., social connection data of the crawled users, and reports them to TAM for further crawling (Section 2.2). On the other hand, by inspecting the fetched results, it finds out prospective malicious partners to ensure data fidelity, and reports them to TAM in real-time (Section 2.5).

2.2 Fetching UIDs

Most of the mainstream OSN sites, except Foursquare [9], assign UIDs continuously. Therefore, we need an efficient algorithm to fetch UIDs of OSN users. Conventionally, we can leverage social connections among users. It can start from a small set of known users, and fetch the UIDs of their neighbors, and neighbors' neighbors in a recursive way. There are

several social graph traversal algorithms [6, 12, 17, 22], such as breadth first search (BFS), metropolis-hasting random walk (MHRW), and frontier sampling (FS). Alternatively, some OSN sites provide interfaces for keyword-based search or public global timeline. This establishes another way to fetch massive UIDs.

2.3 Handling Crawling Failure

The crawling of a task may fail due to many reasons, as the crawlers are owned by different partners and may crash unexpectedly. To keep an active view of the crawling, TAM applies a timeout policy to monitor all the crawlers. When a task has been assigned to a crawler, RCM will be notified to keep track of this task. If the crawling result of this task has not been received by RCM after the pre-defined timeout threshold, this task will be assigned to another crawler in the next task request. We use adaptive timeout here, which means the timeout threshold is adaptive to the history crawling time of this crawler. There are several adaptive timeout strategies proposed and used in Internet flow characterization [19], packet retransmission [4], etc. In our system, we extend the simple adaptive timeout algorithm used in STUN [18]. At the beginning, we set a large-enough timeout value for every crawler, which varies by the feature of target OSN sites. The coordinator can perform a crawling test to estimate the time cost for fetching each UID, to define the number of UIDs each task contains and select the initial timeout value. We adjust the timeout value based on the real crawling time of a certain crawler which decreases the failure handling cost. If the real crawling time of a task is less that 1/4 of the current timeout for the relevant crawler, we set the timeout to be 1/2 of the current timeout. If a timeout event happens, we double the current timeout while ensuring it is less than the initial timeout value.

2.4 Bypassing Request Rate Limiting

OSN sites enforce several request rate limiting policies to prevent aggressive crawling. Some are widely used, like IP-based rate limiting and account-based rate limiting. Some are newly proposed and emerging, like link-based rate limiting. In this section, we describe how to leverage crowd crawling to bypass them.

IP-based rate limiting. IP-based rate limiting is quite common for mainstream OSN sites such as Facebook, Twitter, and Weibo. Such limiting restricts the number of requests per hour per IP address. Crowd crawling is designed to address it in a straightforward way. As different partners own different IP blocks, they can form an aggregate pool of massive IP addresses to bypass IP-based rate limiting policies.

Account-based rate limiting. Account-based rate limiting is another common restriction. It is also used by Facebook, Twitter, and Weibo. Such limiting restricts the number of requests every hour in a per-authenticated-user basis. Account registrations are typically limited to IP addresses. For example, Weibo allows users to register up to 3 accounts per IP address per day. Benefiting from the large number of IP addresses all the partners have, it is able to create a large amount of OSN accounts.

Although we need to introduce some non-human accounts at this point, we strictly require these accounts to act properly. These accounts will never perform any harmful behaviors like spamming.

Link-based rate limiting. Different from the two typical strategies described previously, link-based rating limiting strategy is proposed in Genie [13]. The intuition behind Genie is that a legitimate user tends to visits users who are connected with or close to the user in the social graph. In contrast, Genie assumes that a crawler is not easy to create enough social links to be close to all the users whose data the crawler wants to fetch. Therefore, Genie launches rate limiting for requesting distant users' data. To bypass this prospective restriction, crowd crawling has to utilize some user accounts which are well connected in the social graph. According to our tests in Facebook and Weibo, creating social connections with legitimate users can be done quickly in an automatic way. This helps us easy to get short distances with users we are going to crawl and bypass link-based rate limiting policies. Note that we respect to users' privacy. Although the crawlers' accounts will establish some social links with legitimate users, crowd crawling would never collect the data which are configured as "visible to friends".

Facebook, the largest online social network in the world, who has the global Alexa traffic rank of 1 in August 2013, applies an undirectional friendship system. When A is B's friend, B must be in A's friend list as well. Every user can send friend requests to any other users. In January 2013, we performed two tests in Facebook: (1) sending friend requests to a random set of Facebook users; (2) sending friend requests to a tightly connected community, which means a random user with its friends, and its friends' friends. For each of the tests, we send friend requests to 1000 selected Facebook users. These requests were sent from a Facebook account manually created with a fictional profile, including an attractive picture, complete personal information, and some posted statuses. 24 hours after sending these requests, we examined how many users had accepted the friend requests. The test result shows that the acceptance rate is 4.3% for random friend requests, and 6.3% for community. The median number of friends of a Facebook user is about 100 [21]. As a result, in order to create an account having 100 friends, we can send friend requests to 2325 randomly selected users, or to 1588 users in the aforementioned tightly connected community. We can perform this automatically by leveraging Facebook's API.

Launched in August 2009, Weibo has become the largest microblogging service in China, which has the global Alexa rank of 33 in August 2013. Weibo has attracted more than 500 million registered users by 2012 [15]. Same as most microblogging services such as Twitter, the follow relationship in Weibo is directional, which means you do not have to follow your followers. Similar to what we did in Facebook, we created a fictional account with an attractive picture and appealing personal information. We also posted some tweets to this account. In January 2013, we performed three tests in Weibo: (1) following a random set of Weibo users; (2) following a selected random set of Weibo users, all of which have *#followers/#followings* between 0.5 and 2; (3) following a tightly connected community, which means a random user and other users who have bi-directional friendship with the random user (i.e. followed by this random user and following this user). For each of the tests, we followed 1000 Weibo users. We check the number of users following back in 24 hours. The follow-back rates are 1.9%, 3.1%, and 5.8%, respectively, for the three cases. Similar to Facebook, we can easily create a Weibo account with hundreds of followers.

2.5 Data Fidelity

As a collaborative crawling solution, we need a reliable scheme to verify the fidelity of the crawled data provided by different partners. As crawling data is a resource-intensive task, some partners might want to get benefits from other partners' results, while contributing less than what they should. For some tasks, a malicious partner might let its crawlers submit some fake results obtained by random generators, instead of performing real crawling. Therefore, we need a strategy to ensure the aggregate crawling results trustworthy. We assume that a reliable partner always tries its best to provide accurate results. Also, a malicious partner might try to escape from being detected, i.e., it might occasionally act properly.

Our solution is based on the accountability of every partner. For every completed task, the coordinator knows which partner it was assigned to. Based on this, we introduce a *redundant crawling* scheme to detect prospective malicious crawlers. This scheme is lightweight and easy to configure.

Besides TAM and RCM, we require the coordinator to have one or multiple *redundant crawler(s)*. According to the information provided by RCM, the coordinator maintains a set of recently obtained results and corresponding UIDs. According to a pre-defined percentage p% (e.g., 1%), the coordinator picks p% of these UIDs for redundant crawling, i.e., let the redundant crawler(s) perform crawling according to these randomly selected UIDs. After a round of redundant crawling, the coordinator is able to verify malicious behaviors, by comparing the results of the same UID submitted by the redundant crawlers and by a partner. For each partner i, the coordinator maintains a credit score S_i. For every round, we set an initial credit score K to each partner. During the verification procedure, once a mismatching is found, we decrease S_i by 1. Once S_i falls below zero, partner i will be regarded as malicious, and all results submitted by this partner will be treated as unreliable. Accordingly, this partner will be excluded in the final stage of result sharing.

Due to the dynamic nature of OSN sites, the desired information of a certain UID might change over time. For example, a new friend is added, or a new tweet is posted. Therefore, a mismatch might not be caused by fake crawling. On one hand, we keep the interval between two crawlings short to reduce the negative impact of the dynamic features. On the other hand, the selection of the initial credit score K provides some tolerance of mismatching.

For some partners which are found to be possibly unreliable due to historical record, the coordinator might verify a higher portion of its crawled results. However, we still ensure a minimal percentage of sampling for every partner.

3. SYSTEM IMPLEMENTATION AND PERFORMANCE EVALUATION

Our proof-of-concept prototype consists of 2,000 lines of Python code. In its TAM, we use Redis as its in-memory database benefiting from Redis's high throughput[1], and we have implemented a simple append-only database for RCM.

We deploy TAM and RCM on two servers located in Tsinghua University, China. Each of them has a 16-core 2.5GHz processor and 64 GB memory. We use PlanetLab to emulate a number of crawling partners. By trying to connect to each PlanetLab server on August 18th, 2013 from 11am to 12pm (if not explicitly stated otherwise, the time zone is CEST), we get 472 available PlanetLab servers distributed globally, which are from 236 research institutes. There are 202 servers located in North America, 237 located in Europe, and 33 located in Asia/Oceania. These servers act as crawlers. Every crawler runs one thread for crawling and sleeps a uniformly random time between 0 and 2 seconds after finishing each HTTP request.

[1]Benchmark in Linode 2048 instance shows that Redis can handle 195 thousand SET requests per second, or 250 thousand GET requests per second. More details are available in http://redis.io/topics/benchmarks.

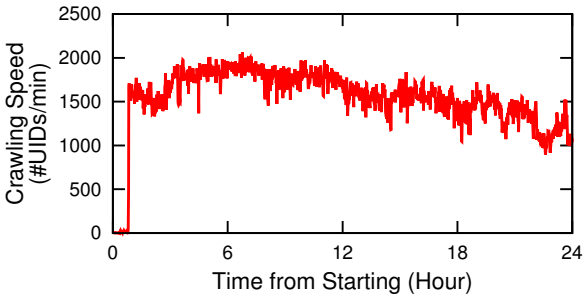

Figure 2: Overall crawling speed

We utilize crowd crawling to collect data from Weibo, the largest microblogging service in China. To obtain a large number of random UIDs, we leverage Weibo's API to access the latest posted tweets in the whole site, and get the UIDs of the tweet publishers. We used 5 servers for obtaining UIDs. Each server queried this API every 2 seconds for one day (from August 17th at 2pm to 18th at 2pm). In total, we obtained 3.95 million unique UIDs. Our distributed crawling is based on these UIDs. For each user, we crawled its profile, followings, followers, and all posted tweets. We performed an aggressive crawling using crowd crawling with the 472 PlanetLab servers as its crawlers. The crawling lasted 24 hours from August 18th at 4pm to 19th at 4pm. During the 24 hours, crowd crawling collected 2.22 million users' data with the total size of 1.86 terabytes. In the following, we describe details and observations of the crawling.

Overall crawling speed. As shown in Figure 2, the crawling speed keeps relatively stable. On average, crowd crawling crawls 1.54 thousand UIDs (1.29 gigabytes data) per minute. The relatively low crawling speed in the beginning is caused by the starting time of crawlers. Interestingly, the average number of crawled UIDs per minute during the night of China is 17.1% higher than daylight, which might be caused by the longer response time of Weibo during the daytime.

Diverse crawling efficiency. From the cumulative distribution function (CDF) of number of UIDs collected by each crawler, shown in Figure 3, we can see that more than half crawlers have crawled more than 5.23 thousand users' data, while only 4.87% crawlers have crawled data of less than 100 users. We divide our crawlers into three regions, i.e., America, Europe, and Asia/Oceania. We can see the overall distribution of crawlers in America and Europe follow a similar distribution. Also, we find that Asia/Oceania contributes most of the fastest crawlers. That might be because Weibo is located in this region, which results in a high throughput.

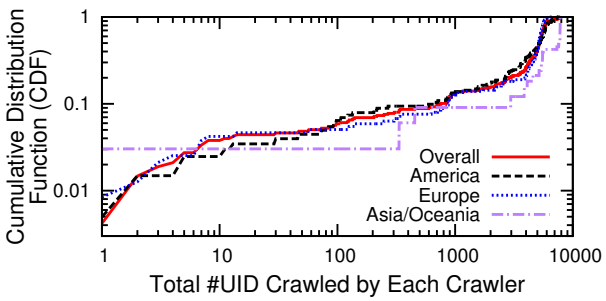

Figure 3: Crawling speed distribution

Collected data. After running crowd crawling for one day, we have obtained 2.22 million users' full content (profiles, social connections, and posted tweets). 49.2% of them are male users, while 50.8% of them are female users. The median number of a user's followings is 179, the median number of a user's followers is 140, and the median number of tweets is 333. There are 1.26% users registered in 2009, 16.8% in 2010, 27.0% in 2011, 27.4% in 2012, and 27.6% in 2013. From these users, we have obtained 1.04 billion tweets. To the best of our knowledge, the coverage of our data is significantly larger than previous work such as Fu *et al.*'s [5] (30 thousand UIDs crawled in 6 days) and Guo *et al.*'s [8] (1 million UIDs crawled in 1 month). This better coverage demonstrates the advantage of crowd crawling. The huge amount of data crawled in such a short period of time also provides a consistent snapshot of Weibo. We foresee that if we perform the crowd crawling prototype for a longer time, we will obtain the full content of tens of millions of users or even hundreds of millions. Analyzing the crawled Weibo data is left for future work.

4. RELATED WORK

Using crowdsourcing for online social network data collection is relatively new, while there is still some research on related areas.

Lots of work on distributed computing and measurement has been proposed in the past decade. They divide tasks to subtasks to achieve distributed computing and measurement, while crowd crawling utilizes this idea for distributed collaborative crawling. MapReduce [3] divides a complete computational task into subtasks, and assigns these subtasks to a number of nodes. By contributing the processor cycles, these nodes work in parallel to solve the original task. With the similar idea, crowd computing [14] provides a novel approach using mobile devices to achieve large-scale distributed computation. It combines social structure to improve performance. Dasu [20] is a distributed network measurement platform. Built as a BitTorrent extension, Dasu clients are hosted by numerous Internet end users. With Dasu, a number of network measurement activities can be conducted at the Internet's edge.

There has been some work on distributed crawling, for either web pages or OSNs. To our knowledge, all of them utilize a cluster of computers to achieve efficient crawling, without involving multiple partners. Cho *et al.* [2] present their parallel crawling system architecture. They consider several practical issues like huge amount of URLs management and partitioning function. However, this system is designed for web page crawling, i.e., getting pages from a number of linked websites. It is significantly different from OSN crawling, which typically focuses on one particular OSN site with strict rate limiting. Gjoka *et al.* [7] demonstrate a distributed crawling system with their own server cluster. They utilize 28 servers as crawlers to collect data from Facebook. Such solution is still restricted by the IP-based rate limiting, unless the crawling organization has a large number of IP addresses.

5. CONCLUSION AND DISCUSSION

In this paper, we introduce and motivate crowd crawling, a scalable framework to enable collaborative social data collection. Crowd crawling is robust to different rate limiting manners enforced by OSN sites, and can prevent free-riders. Our evaluations on PlanetLab has demonstrated the high efficiency of OSN data collection, i.e., we can finish the crawling of 2.22 million Weibo users in 24 hours, including their profiles, social connections, and posted tweets.

As the first step of our collaborative social data collection framework, crowd crawling has been demonstrated with several advantages. At the same time, we are aware of several open issues. We would not expect the discussions in this section could reach a conclusion at this point, while we hope that our discussion would provide some insights for further investigations in this field.

First of all, different OSN sites have different terms of service (ToS). Some are stricter, while some are more flexible. However, for research purposes, different groups of people from academia continuously perform aggressive crawling in different OSN sites. As many of these crawling are duplicate, we hope our crowd crawling could at least save the load of OSN sites, while improving the crawling efficiency of research groups. In short, crowd crawling might still violate the ToS of some OSN sites, but at least it is more OSN-friendly than today's independent aggressive crawling.

Secondly, unlimited data sharing might cause ethical issues. We believe that the crawled data should be owned by the groups who contributed resources to crawling. The motivation here is similar to PlanetLab testbed. Nevertheless, if all the groups in a crawling project have reached an agreement, they have the flexibility to make the data public, or release a sampled subset. There are two noteworthy issues in making data sets public. On one hand, to preserve users' privacy, only anonymized data should be released. On the other hand, some OSN sites might prohibit public sharing of its particular contents. For instance, it is not allowed to share data crawled from Twitter [23].

Thirdly, in our current design, every partner contributes crawled data more than the pre-defined criteria can obtain the entire aggregate data set. We plan to build an incentive framework as our prospective future work, with which the partners who contribute more will be rewarded. This is desirable because all the contributions made by different partners are recorded in crowd crawling.

Finally, some researchers might want to get private data from OSN sites, while different OSN sites have different user privacy configurations. For Twitter and Weibo, if we know a user's UID, we can retrieve her entire profile, social connections, and all posted contents. Differently, for Facebook, if we know a user's but are not in her friend list, we might be only able to obtain some fields in her profile and social con-

nections. Therefore, similar to existing individual crawlers, crowd crawling is restricted by the privacy configurations of OSN sites.

6. ACKNOWLEDGEMENTS

We are grateful to the anonymous reviewers, Ruichuan Chen and our shepherd, Anne-Marie Kermarrec, for their insightful comments.

7. REFERENCES

[1] Joining PlanetLab. http://www.planet-lab.org/joining.

[2] J. Cho and H. Garcia-Molina. Parallel crawlers. In *WWW*, 2002.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI*, 2008.

[4] S. W. Edge. An adaptive timeout algorithm for retransmission across a packet switching network. In *SIGCOMM*, 1984.

[5] K.-W. Fu and M. Chau. Reality check for the Chinese microblog space: a random sampling approach. *PloS ONE*, 2013.

[6] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: a case study of unbiased sampling of OSNs. In *INFOCOM*, 2010.

[7] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Practical recommendations on crawling online social networks. *JSAC*, 2011.

[8] Z. Guo, J. Huang, J. He, X. Hei, and D. Wu. Unveiling the patterns of video tweeting: a Sina Weibo-based measurement study. In *PAM*, 2013.

[9] W. He, X. Liu, and M. Ren. Location cheating: a security challenge to location-based social network services. In *ICDCS*, 2011.

[10] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao. Understanding latent interactions in online social networks. In *IMC*, 2010.

[11] L. Jin, Y. Chen, T. Wang, P. Hui, and A. V. Vasilakos. Understanding user behavior in online social networks: A survey. *IEEE Communications Magazine*, 2013.

[12] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW*, 2010.

[13] M. Mondal, B. Viswanath, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Defending against large-scale crawls in online social networks. In *CoNEXT*, 2012.

[14] D. G. Murray, E. Yoneki, J. Crowcroft, and S. Hand. The case for crowd computing. In *MobiHeld*, 2010.

[15] J. Ong. China's Sina Weibo grew 73% in 2012, passing 500 million registered accounts. *thenextweb.com*, 2013.

[16] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. In *SIGCOMM*, 2010.

[17] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *IMC*, 2010.

[18] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN–simple traversal of user datagram protocol (UDP) through network address translators (NATs). *RFC 3489*, 2003.

[19] B. Ryu, D. Cheney, and H.-W. Braun. Internet flow characterization: adaptive timeout strategy and statistical modeling. In *PAM*, 2001.

[20] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger. Dasu: pushing experiments to the Internet's edge. In *NSDI*, 2013.

[21] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the Facebook social graph. *arXiv:1111.4503*, 2011.

[22] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li. Understanding graph sampling algorithms for social network analysis. In *SIMPLEX*, 2011.

[23] A. Watters. How recent changes to Twitter's terms of service might hurt academic research. *readwrite.com*, 2011.