Cuckoo: Towards Decentralized, Socio-Aware Online Microblogging Services and Data Measurements

Tianyin Xu[1,2], Yang Chen[1], Jin Zhao[1,3], Xiaoming Fu[1]
[1] Institute of Computer Science, University of Goettingen, Goettingen, Germany
[2] State Key Lab. for Novel Software and Technology, Nanjing University, Nanjing, China
[3] School of Computer Science, Fudan University, Shanghai, China
{tianyin.xu, yang.chen, fu}@cs.uni-goettingen.de, jzhao@fudan.edu.cn

ABSTRACT

Online microblogging services, as exemplified by Twitter [9] and Yammer [12], have become immensely popular during the latest three years. Twitter, the most successful microblogging service, has attracted more than 41.7 million users as of July 2009 [25] and is still growing fast. However, current microblogging systems severely suffer from performance bottlenecks and central points of failure due to their centralized architecture. Thus, centralized microblogging systems may threaten the scalability, reliability, as well as availability of the offered services, not to mention the extremely high operational and maintenance cost.

However, it is not trivial to decentralize microblogging services in a peer-to-peer fashion. The challenges first derive from the heterogeneity of the inherent online social network (OSN) features. The non-reciprocation feature of microblogging services also increases the heterogeneity. Moreover, different from traditional approaches used in centralized server-based systems, an efficient, robust and scalable approach for data collection and dissemination in such distributed heterogeneous environments is desirable.

In this paper, we present a decentralized, socio-aware microblogging system named Cuckoo. The design takes advantages of the inherent social relationships while leverages P2P techniques towards scalable, reliable microblogging services. Besides, Cuckoo provides a flexible interface for data collection while circumventing unnecessary traffic on the server. We discuss the benefits that our system may bring for both service providers and end users. We also discuss the technical aspects to be considered and report our work in progress.

Categories and Subject Descriptors

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

General Terms

Design, Performance

Keywords

Microblogging Services, Online Social Networking

HotPlanet'10 , June 15, 2010, San Francisco, CA, USA

1. INTRODUCTION

With the phenomenal success of Twitter-like Internet services, microblogging (e.g., Twitter [9], Yammer [12], Buzz [5], etc) has emerged as a new, popular communication utility in the latest years. Microblogging has become popular quite quickly. For example, Twitter, the most successful microblogging service launched in October 2006, has attracted more than 41.7 million users as of July 2009 and its userbase is still growing incredibly fast. Nowadays, microblogging services play important roles in many aspects of people's social life [21, 35]: from releasing emotional stress, to keeping in touch with friends, to collaborating with colleagues at work, etc. Moreover, the functionalities of microblogging services are even beyond human social networks. Microblogging is becoming an important media for spreading news to large groups of people [25, 31, 35], from new product marketing, to political campaign publicity, to immediate news breaking, etc. One impressive story is that Twitter broke the news about China's massive earthquake well before the mainstream media [11].

Current microblogging services depend on centralized architectures, where user clients repeatedly request centralized servers (e.g., Twitter Server) for newest micro-contents no matter whether there is a new update[1]. Such polling-style requests are blind, superfluous and sticky. The centralized microblogging systems suffer severely from performance bottleneck (e.g., Twitter's rate limiting, 150 requests per hour is allowed) and central point of failure (e.g., Twitter's not rare "over capacity" errors and "database maintenance" errors). Although Twitter has made several performance improvements, such as removing the upper limit on the subscription number in 2009, by increasing the number of dedicated servers, it still can hardly catch up with the steadily-growing user demands. As a result, centralized microblogging systems may threaten the scalability, reliability, as well as availability of the offered services, not to mention the extremely high operational and maintenance cost.

One important aspect is data collection. Data collection is critical for researchers and third-party developers to understand user behavior and access patterns of microblogging services. So far, data collection is performed via data crawling on centralized servers [21, 25], which may cause traffic storms and is not scalable. Even worse, such methods may fail to retrieve statistics from departed peers. More efficient and scalable data collection schemes are required to let users directly exchanging vital statistics in a decentralized fashion.

However, it is not trivial to decentralize microblogging ser-

[1]In this paper, we use "microblog", "micro-content", "micro-news", "status", "update" interchangeably.

vices in a peer-to-peer fashion. The challenges mainly derive from the heterogeneity of the inherent online social network (OSN) features. Furthermore, the non-reciprocation feature of microblogging's opt-in subscription model [31] increases the heterogeneity. The basic operation of the subscription model is "follow". Being a follower means that the user will receive all the micro-contents from the one he follows. A user can follow anyone, and the one being followed need not follow back. We summarize the main challenges as follows:

1. Support users with heterogenous social relationships, such as *broadcasters* and *miscreants* [23]. Broadcasters include celebrities and news media that have huge numbers of followers. For example, **cnnbrk** (CNN Breaking News) in Twitter has over 2.97 million followers but only follows 28 users. Miscreants try to contact everyone and hope that someone can follow back. The design should be able to support different user social patterns with different purpose of use.

2. Support users with heterogenous access patterns as well as social behaviors. Unlike other P2P application such as file downloading and video streaming, microblogging users have highly dynamic access patterns. In case of Twitter, various communication channels are supported including SMS, RSS feeds, Web, dedicated software, etc. Moreover, the users' behaviors are quite different. Such heterogeneity causes significant dynamics to decentralized systems (e.g., peer churn).

3. Support users with efficient, robust and scalable data collection. Since the social profiles/attributes and other interaction information are very important aspects for system designers and socialists to understand the user behaviors, restore from failures or peer churns as well as improve the system design, it is crucial to collect related information in an efficient, robust and scalable manner, whilst protecting the users' privacy.

In this position paper, we present a decentralized, socio-aware online microblogging system named Cuckoo. Our objective is to design, implement, and deploy a decentralized socio-aware microblogging service, which is fully compatible to the current Twitter architecture. In other words, we expect to study how the decentralized Cuckoo system helps microblogging service to improve the performance, to save the bandwidth costs, and to reduce the sluggishness and downtime, while performing equally well or even better than the conventional centralized system. For this objective, the design of Cuckoo conserves valuable bandwidth and storage resources on Twitter server while taking advantage of peer assistance in a complementary fashion. In response to the challenges, Cuckoo takes advantages of social relationship while leverages P2P techniques towards scalable, reliable microblogging services. Different from traditional approaches used in centralized server-based systems [21,25], data collection and dissemination in such distributed heterogeneous environments are challenged by the dynamics, heterogeneity and distributed nature of the system.

The rest of the paper is organized as follows. Sec. 2 reviews the background and related works. Sec. 3 presents the design rationale of Cuckoo and summarizes the incentives of Cuckoo adoption which may brings for both the service provider and the end users. We discuss the technical aspects to be considered and report our work in progress in Sec. 4. Sec. 5 concludes the paper and poses the future work.

2. BACKGROUND AND RELATED WORK

Thanks to the success of Twitter [9], a number of microblogging services are developed and deployed on the Internet [5,6,8,12]. Although each of them has some additional features than original Twitter, common to these microblogging services is the opt-in subscription model based on the "follow" operation. The subscription model is deceptively simple. A user can publish microblogs with a length limit of viewable text (up to 140 characters in Twitter). The other users who have explicitly follow that user will receive this microblog, i.e., being a follower means that the user will receive all the micro-contents from the one he follows. A user can follow anyone, and the one being followed need not follow back. Note that the relationship of following and being followed requires no reciprocation, i.e., "follow" is a unidirectional operation. This opt-in, social networking model effectively resists spamming so that users only receive the microblogs from who they are concerning. On the other hand, the "small-world" social feature [28] makes it possible for a free comment to blossom into a worldwide discussion.

The popularity of microblogging services has already attracted some research interests in the computer networking perspective [21,23,25,31]. Most of these works are measurement studies on Twitter including Twitter's social graph, its topological and geographical characteristics, the popularity of the micro-contents, etc. Java et al. [21] report some early observations of Twitter in 2007. They study the topological and geographical properties and show user clusters formed by users with similar intention. Krishnamurthy et al. [23] identify distinct classes of Twitter users as well as their behaviors according to the number of followings and followers. Kwak et al. [25] crawl the entire Twittersphere based on which a comprehensive measurement study is performed. In the following-follower analysis, they find a no-power-law follower distribution, a short effective diameter, and low reciprocity. The influentials, trending topics, and impact of retweet are also studied.

The only proposed decentralized system prototype focusing on microblogging so far is FETHR [31], which is most related to our work. FETHR [31] envisions the fully decentralized microblogging for the first time. The main idea of FETHR is to let users directly contact each other via HTTP and employ gossip for popular micro-content propagation. However, FETHR lacks of several practical considerations, e.g., decentralized user lookup, overlay maintenance, etc. In addition, FETHR cannot provide any guarantee on micro-content delivery and the delivery performance will be degraded under asynchronous user access patterns.

There are some decentralized OSN prototypes which are related to our work [15,17,24,32]. PeerSoN [15] is a project for P2P OSNs that proposes to use dedicated third-party DHTs for lookup meta-data. Based on the meta-data, direct information exchange between users can be achieved. Safebook [17] aims at protecting users' security and privacy. In Safebook, the "matryoshka" structures are designed to protect users data based on trust transitivity. Another related decentralized OSN system, Vis-à-Vis [32] is based on the concept of Virtual Individual Server (VIS), which is used for managing and storing user data. VISs self-organized into multi-tier DHTs that represent OSN groups, with one DHT per group. The top-tier meta-DHT is used to advertise and search for public OSN groups. uaOSN [24] proposes to distribute only the storage of heavy content while retain the business model and functionalities of OSN provides.

Table 1: Comparison with the related systems

System	Design Target	Socio-Aware	Compatible	Additional Infr.	Struc. Overlay
Cuckoo	microblogging	√	√	×	√
FETHR	microblogging	×	×	×	×
PeerSoN	OSN	×	×	√	√
Safebook	OSN	√	×	×	√
Vis-à-Vis	OSN	×	×	√	√
uaOSN	OSN	×	√	√	×

These works give good inspiration to the design of Cuckoo. However, since previous works do not consider the features of microblogging service, they cannot fit the new service well. We summarize these works in comparison with Cuckoo in Table 1. The system features listed for comparison include the target service of design (Design Target), whether taking advantage of social relationship (Socio-Aware), whether compatible for current system (Compatible), whether requiring additional infrastructure (e.g., third-party DHT [15]), and whether based on structured overlay.

3. DESIGN RATIONALE

In typical microblogging services, a user has one or several of the following social relations: *friend*, *neighbor*, *follower*, and *following*. Friend is a reciprocate social link between two users, which indicates that two users are acquaint with each other and willing to help each other. Microblogging (e.g., Twitter) shows a quite low level of such reciprocity compared with conventional OSNs [25]. Neighbor refers to the relationship between users with common interests. For example, two users sharing a same following are neighbors. In microblogging, follower and following are the most common one-way connections. Our design takes advantage of these social relationship to construct the decentralized service.

For providing location service and improving availability, Cuckoo organizes user clients into a structured P2P overlay. Without loss of generality, we utilize Pastry [30] as the underlying P2P overlay. The 128-bit nodeId can be generated according to the sequential userId (e.g., Twitter userId). In this case, a user can find any online user in less than $\lceil log_{2^b} N \rceil$ steps on average (N is the number of online nodes and b is a configuration parameter with typical value 4). In our current design, we do not let peers store irrelevant micro-contents for security and fairness considerations.

3.1 Node State

Besides the Pastry routing table, each user maintains 4 lists for friends, neighbors, followings, and followers. A user keeps connection with m online friends. We set $m = max(min(f, T), log_{2^b} f)$, where f is the user's number of friends and T is a threshold. The user itself and his f friends make up the *virtual node* (VN) via request redirection, i.e., friends help each other to balance load and improve availability. Fig. 1 gives an example of the VN mechanism. The VN **d46a1c** consists of the physical node **d46a1c** and its 3 friends (**001ab0**, **420a1b**, **64fb26**). The following list is a static config file containing nodeId of all the following users. No real-time connection is required. Like the friend list, a user also maintains n random followers where $n = max(min(l, T'), log_{2^b} l)$, ($l$ is the number of followers and T' is a threshold). The idea of the calculation of m and n is that if a user has 10 or fewer followers/friends (it is reported in [31] that half of Twitter users have 10 or fewer followers), he keeps connection with all his followers. Otherwise for a user has huge numbers of followers, he keeps connections with a logarithmic subset of all the followers. Finally, the neighbor list is initialized during the bootstrapping stage. In boostrapping, the user updates its following's recent statuses and initializes the neighbor list in this process.

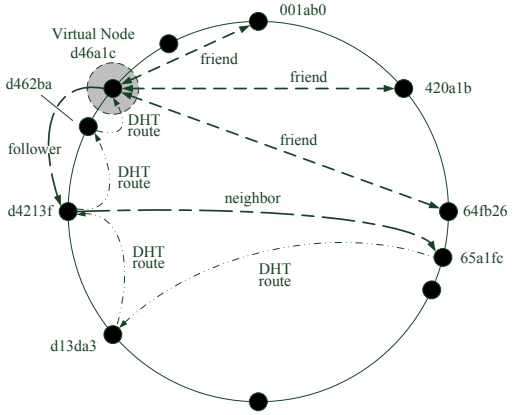

Figure 1: Network Model

3.2 Update Recent Statuses

When joining the system, a new node setups its Pastry routing table and then utilizes the underlying infrastructure to fetch the updated statuses of its followings. Note that if the following user is popular[2], it is likely that the routing path contains the new node's neighbors. In this case, the routing process will not continue. The new neighbor directly sends the recent statuses of the following to the new node while help to initialize the neighbor list. Fig. 1 gives an example for this process, in which node **65a1fc** gets the statuses of node **d46a1c** directly from node **d4213f**. Even if no neighbor is met, the request message will reach the destination and both recent statuses and neighbor list can be updated. Assume the popularity of the following user is δ, the expected steps for fetching the updated statuses is approximately $\sum_{i=1}^{K}(1-\delta)^{i-1} \cdot \delta \cdot i$ ($K = \lceil log_{2^b} N \rceil$).

Note that DHT is efficient for locating rare items but incurs higher overheads. On the other side, flooding-based searching is more efficient for locating highly popular items but poorly suited for locating rare items [27, 34]. Thus, our design chooses a hybrid searching methods, i.e., use flooding to fetch statuses from influentials like **cnnbrk** and use

[2]Here popularity refers to the ratio between the number of followers and the total number of users.

DHT to fetch statuses from normal users. The flooding is through "near" nodes according to some proximity metric. Fortunately, Pastry have already provided these near nodes in its "neighborhood set".

3.3 Micro-content Propagation

When an online user publish a new micro-content, the microblogging service should disseminate it to all the user's followers. We use push method instead of conventional pull method to achieve the dissemination. The push method is much more efficient than the pull method. It does not require meta-data exchanging or periodical requests. Moreover, by replicating micro-contents among followers, the followers can help each other in a cache-and-relay style, in the case the original publisher later become unavailable or the publisher cannot afford sending updates to all his followers.

For normal users, directly pushing messages from an original publisher to his followers seems to be enough (it is reported that 90% users in Twitter have less than 100 followers [31]). For the broadcasters or influentials like **cnnbrk**, the publisher cannot afford sending updates to all his followers. In this case, our design uses gossip-based push between neighbors to propagate the micro-news. Gossip (also called epidemic, rumor) protocol has been proved to be a robust and scalable way of propagating information in dynamic and heterogeneous networks [19]. The theoretical support provided in [22] proves if there are n nodes and each node gossips to $\log(n)+k$ other nodes on average, the probability that everyone gets the message converges to $e^{-e^{-k}}$, very close to 1.0 without considering the bandwidth constraint, latency, failure, etc. This result provides a guideline for the design of membership management, i.e., maintain the neighbor list to be logarithmic in the size of the number of all the followers.

3.4 Functionalities of Service Providers

In our system, if an unpopular node is not online nor its friends, its profile and recent statuses cannot be found in the overlay network. In this case, the dedicated servers from the service provider (e.g., Twitter server) work as backup servers. The omnipotent servers are always online and guarantee high availability of the service. Note that the users fetch profiles and statuses from servers only when they cannot find them successfully in the overlay network. In most cases, it is mainly due to the inactive and unpopular users, and thus requesting from servers causes acceptable overhead.

When publishing new microblogs, the publisher directly uploads the digest of the microblog to the server in the same time of gossiping. In our design, we allow long blog content (with more characters than 140) exchanging within the overlay network. On the other hand, the users only upload fixed-length (e.g., 140 characters) digests to the server, with the consideration of the micro features that is suitable for SMS transmission.

Fig. 2 demonstrates the system architecture as well as the relationship between service provider and end users. Our design does not exclude the service provider from the picture. Instead of decentralizing the full system, we propose to enrich the users' quality of experience in the overlay network while let the servers provide high availability and reliability. In other words, our objective is to help the service providers (i.e., the business companies) but no to bury them.

3.5 Security Issues

We would be remiss if not discuss the security issues of

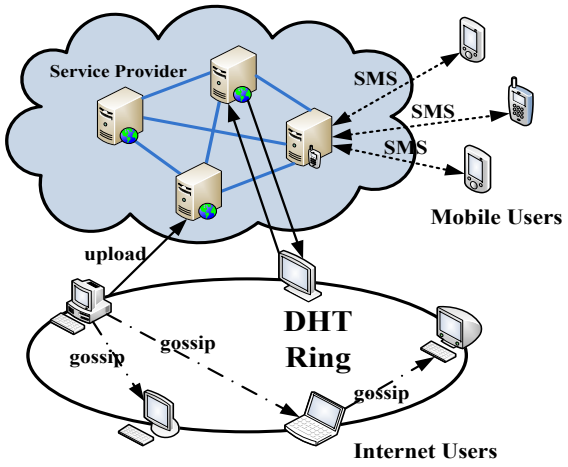

Figure 2: System Architecture

Cuckoo in today's Internet environment, although our main focus is not on security. The presence of malicious users and malwares in the overlay network requires additional security mechanism to safeguard against attack.

One important thing is authentication since some malware may impersonate some normal users to distribute spam. To defend against the forged mico-contents, asymmetric key cryptography and digital signature can be employed, i.e., publishers include digital signatures with each microblogs or even encrypt the micro-contents. To defend against the forged users, some sophisticated authentication infrastructure should be used, e.g., challenge-response protocol.

Another problem is suppression, i.e., malicious intermediate nodes refuse to forward micro-contents or filter some important micro-news. FETHR [31] proposes to link a user's microblogs together into a hash chain according to the timeline, based on which the integrity can be checked.

The third problem is privacy preserving. Although current microblogging services like Twitter and Identi.ca seem quite open, there are still some users who do not want their profiles and microblogs to be in public. In this case, data encryption and control access by appropriate key sharing and distribution is necessary There are already some great effort to protect privacy in OSNs including the PKI-based (public key infrastructure) approach [15] and attribute-based encryption (ABE) [14].

3.6 Data Collection

One important aspect of microblogging systems is data collection which is critical for researchers and third-party developers to understand user behaviors as well as user access patterns. So far, data collection is performed via crawling on centralized servers [21, 25], which may cause traffic storms and is not scalable. Even worse, such methods fail to retrieve statistics from departed peers. In Cuckoo, data collectors are able to collect vital statistics based on the underlying overlay network so as to circumvent server bottleneck. The basic idea is to let peers directly exchange statistics in a decentralized fashion. Advanced techniques such as network coding [29] and data aggregation [33] can be integrated.

It is worth noticing that in microblogging services as well as OSNs, the developers should collect and measure social

information to improve system design, in addition to the traditional network metrics (e.g., end-to-end delay [13], available bandwidth [20], transmission error [16]). The social information includes user popularity, user social behavior, user access pattern, etc. Cuckoo includes an inherent data collection module which allows the efficient, robust and scalable exchange, look up and storage of such social data in a privacy protected manner. Further details are currently investigated and extended for potentially other OSNs.

3.7 Incentives

We summarize the incentives of adoption of Cuckoo in two sides: the service provider side and the end user side. For the service provider, the incentives of adopting the Cuckoo system is straightforward:

1. **Lower Bandwidth Cost.** Servers can get rid of superfluous, sticky and repeated HTTP requests from huge numbers of user clients even there is no new update. Bandwidth is mainly used for collecting users' profiles and microblogs.

2. **High Scalability.** The decentralized architecture is highly scalable that moderates service provider's growing pain in the face of steadily growing userbase. Less infrastructures are demanded compared with the centralized equivalents.

3. **Higher Security.** The decentralized nature makes it more robust to malicious attacks, e.g., DoS attacks (Twitter did be a victim of DDoS attack [10]).

4. **Better Quality of Service.** Better QoS can be achieved by removing the performance bottleneck and single point of failure.

Most important, the business company will not lose any functionality nor the user community. For the end users, the incentives include:

1. **High Reliability.** Since both user profiles and microblogs are distributed and replicated in the distributed overlay network in addition to the central server, information losses in a single location can be easily restored. Besides, the failure of central server (e.g., outrage [3]) will not paralyze the whole system.

2. **Better Quality of Experience.** Corresponding to better QoS offered by the service provider, end users can receive better QoE including quick response time, higher searching efficiency, longer contents allowed, etc.

3. **Enrichment of Additional Functions.** Cuckoo's decentralized overlay network is an open, powerful infrastructure for third-party developers to enrich additional functions for end users which are not yet provided by service providers, e.g., photo and multimedia contents in microblogs.

4. IMPLEMENTATION

We are now working on to implement the proposed microblogging system prototype. We choose FreePastry [4] as our overlay infrastructure for Pastry's good properties (e.g., locality aware) as well as FreePastry's platform independence (Java source code). Cuckoo can be directly applied by any structured overlay network that supports the key-based routing (KBR) API defined in [18]. Vis-à-Vis [32] and uaOSN [24] propose to use some more stable external storage facilities (Amazon EC2 [1] and Amazon S3 [2] in Vis-à-Vis, set-top boxes/residential routers with hard drives [26] in uaOSN) to achieve high availability. However, the former is too expensive for end users (running a virtual machine for one month costs close to US$75), while the latter is restricted by the deployment difficulties. Our implementation just employs end users' desktop machines to construct the decentralized system. We are confident to provide good performance in the face of peer churn. On the other hand, we can expect a much higher performance when the expensive external storage facilities are available.

In the next stage, we would like to validate our design and study the performance gain by deploying the system on PlanetLab [7] and testing it by real microblogging traces and dataset, e.g., the dataset provided in [25]. We also plan to implement a Twitter client software that is based on the design of Cuckoo and is fully compatible to current Twitter's microblogging service. Note that Cuckoo does not require any functionality and modification on the server side. We expect to collect large volumes of microblogging user data via this Cuckoo-based Twitter client.

5. CONCLUSIONS AND FUTURE WORK

Although in the infancy, microblogging services are having the golden era in the latest years. Twitter, the pioneer microblogging service provider has attracted more than 41.7 million users in less than three years and its userbase is still growing incredibly fast. However, current microblogging systems severely suffer from performance bottlenecks and central points of failure due to the unscalable centralized architecture. In this position paper, we present a decentralized, socio-aware microblogging system, named Cuckoo. We present the design rationale of Cuckoo that takes advantages of the inherent social relationships while leverages P2P techniques towards scalable, reliable microblogging services. Besides, Cuckoo provides a flexible interface for data collection while circumventing unnecessary traffic on the server. We discuss the technical aspects to be considered and show that Cuckoo can bring good incentives of adoption for both service providers and end users.

In the future, we plan to improve the work in the following aspects. First, we expect to support "topic trend" function in Cuckoo like the "#" operation in Twitter. A quite common use for microblogging services these days is looking at particular topics (e.g., UK general election). Second, we are interested in supporting user mobility in Cuckoo. Currently, Cuckoo is designed only for desktop Internet users (the SMS is provided by dedicate servers). User mobility via wireless Internet access implies high churn and should be designed carefully. Third, we hope to enrich more third-party functions in Cuckoo such as group communication, real-time chatting in addition to the basic "follow" operation. Fourth, we would like to drive the service providers to add some functions on the server side in a complementary manner that can benefit the whole system. Finally, the details about data collection module of Cuckoo will be explored, which is expected to be applicable for other OSNs.

6. ACKNOWLEDGEMENT

The authors would like to thank Pan Hui from Deutsche Telekom Laboratories and Tristan Henderson from University of St Andrews for their helpful comments.

7. REFERENCES

[1] Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/.

[2] Amazon Simple Storage Service (S3). http://aws.amazon.com/s3/.

[3] Facebook database outrage cut off about 150,000. http://news.cnet.com/8301-13577_3-10373349-36.html/.

[4] FreePastry. http://www.freepastry.org/.

[5] Google Buzz. http://www.google.com/buzz/.

[6] Identi.ca – public timeline. http://identi.ca/.

[7] PlanetLab. http://www.planetlab.org/.

[8] Plurk is a social journal for your life. http://www.plurk.com/.

[9] Twitter. http://www.twitter.com/.

[10] Twitter, Facebook attack targeted one user. http://news.cnet.com/8301-27080_3-10305200-245.html?tag=mncol.

[11] 'Twitters' beat media in reporting china earthquake. http://afp.google.com/article/ALeqM5hUZGbUECMKx_P5H3lRXPnGo6cPjw/.

[12] Yammer: Enterprise Microblogging. https://www.yammer.com/.

[13] S. Agarwal and J. R. Lorch. Matchmaking for Online Games and Other Latency-Sensitive P2P Systems. In *SIGCOMM'09: Proc. of the 15th ACM SIGCOMM*, Barcelona, Spain, August 2009.

[14] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An Online Social Network with User-Defined Privacy. In *SIGCOMM'09: Proc. of the 15th SIGCOMM*, Barcelona, Spain, August 2009.

[15] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. PeerSoN: P2P Social Networking – Early Experiences and Insights. In *SNS'09: Proc. of the 2nd ACM Workshop on Social Network Systems*, Nuremberg, Germany, March 2009.

[16] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *MobiCom'03: Proc. of the 10th ACM MobiCom*, San Diego, CA, September 2003.

[17] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: a Privacy Preserving Online Social Network Leveraging on Real-Life Trust. *IEEE Communication Magazine*, 47(12):94–101, 2009.

[18] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a Common API for Structured Peer-to-Peer Overlays. In *IPTPS'03: Proc. of the 3rd International Workshop on Peer-to-Peer Systems*, Berkeley, CA, USA, February 2003.

[19] P. T. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulié. From Epidemics to Distributed Computing. *IEEE Computer*, 37:60–67, 2004.

[20] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In *SIGCOMM'02: Proc. of the 8th ACM SIGCOMM*, Pittsburgh, PA, USA, August 2002.

[21] A. Java, X. Song, T. Finin, and B. Tseng. Why We Twitter: Understanding Microblogging Usage and Communities. In *Proc. of the Joint 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Social Network Analysis*, San Jose, CA, USA, August 2007.

[22] A. M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, 2003.

[23] B. Krishnamurthy, P. Gill, and M. Arlitt. A Few Chirps About Twitter. In *WOSN'08: Proc. of the 1st ACM SIGCOMM Workshop on Online Social Networks*, Seattle, WA, USA, August 2008.

[24] M. Kryczka, R. Cuevas, C. Guerrero, E. Yoneki, and A. Azcorra. A First Step Towards User Assisted Online Social Networks. In *SNS'10: Proc. of the 2nd ACM Workshop on Social Network Systems*, Paris, France, April 2010.

[25] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *WWW'10: Proc. of the 19th International World Wide Web Conference*, Raleigh, NC, USA, April 2010.

[26] N. Laoutaris, P. Rodriguez, and L. Massoulié. ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers. *ACM SIGCOMM Computer Communication Review*, 38(1):51–54, 2008.

[27] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In *IPTPS'04: Proc. of the 3rd International Workshop on Peer-to-Peer Systems*, San Diego, CA, USA, February 2004.

[28] S. Milgram. The Small-World Problem. *Psychology Today*, 1(1):61–67, 1967.

[29] D. Niu and B. Li. Circumventing Server Bottlenecks: Indirect Large-Scale P2P Data Collection. In *ICDCS'08: Proc. of the 28th International Conference on Distributed Computing Systems*, Beijing, China, June 2008.

[30] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Middleware'01: Proc. of the 18th IFIP/ACM International Conference on Distributed System Platforms*, Heidelberg, Germany, November 2001.

[31] D. R. Sandler and D. S. Wallach. Birds of a FETHR: Open, decentralized micropublishing. In *IPTPS'09: Proc. of the 8th International Workshop on Peer-to-Peer Systems*, Boston, MA, USA, April 2009.

[32] A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres. Privcy, Cost, and Availability Tradeoffs in Decentralized OSNs. In *WOSN'09: Proc. of the 2rd ACM SIGCOMM Workshop on Online Social Networks*, Barcelona, Spain, August 2009.

[33] R. van Rennesse, K. Birman, D. Dumitriu, and W. Vogels. Scalable Management and Data Mining using Astrolabe. In *IPTPS'02: Proc. of the 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA, USA, March 2002.

[34] M. Zaharia and S. Keshav. Gossip-based Search Selection in Hybrid Peer-to-Peer Networks. In *IPTPS'05: Proc. of the 4th International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA, February 2005.

[35] D. Zhao and M. B. Rosson. How and Why People Twitter: The Role that Micro-blogging Plays in Informal Communication at Work. In *GROUP'09: Proc. of the 7th International Conference on Supporting Group Work*, Sanibel Island, FL, USA, May 2009.