Twittering by Cuckoo – Decentralized and Socio-Aware Online Microblogging Services

Tianyin Xu[*,‡], Yang Chen[*], Xiaoming Fu[*], Pan Hui[†]
[*] Institute of Computer Science, University of Goettingen, Goettingen, Germany
[‡] State Key Lab. for Novel Software and Technology, Nanjing University, Nanjing, China
[†] Deutsche Telekom Laboratories/TU-Berlin, Berlin, Germany
{tianyin.xu, yang.chen, fu}@cs.uni-goettingen.de, pan.hui@telekom.de

ABSTRACT

Online microblogging services, as exemplified by Twitter, have become immensely popular during the latest years. However, current microblogging systems severely suffer from performance bottlenecks and malicious attacks due to the centralized architecture. As a result, centralized microblogging systems may threaten the scalability, reliability as well as availability of the offered services, not to mention the high operational and maintenance cost.

This demo presents a decentralized, socio-aware microblogging system named Cuckoo. The key aspects of Cuckoo's design is to take advantage of the inherent social relations while leveraging peer-to-peer (P2P) techniques in order to provide scalable, reliable microblogging services. The demo will show these aspects of Cuckoo and provide insights on the performance gain that decentralization and socio-awareness can bring for microblogging systems.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems – Distributed Applications

General Terms: Design, Performance

Keywords: Microblogging Services, Online Social Networking, Peer-to-Peer Systems

1. INTRODUCTION

With the phenomenal success of Twitter-like Internet services, microblogging has emerged as a new, popular communication utility in the latest years. Twitter, the most successful microblogging service launched in October 2006, has attracted more than 41.7 million users as of July 2009 [3] and its userbase is still growing fast. Unlike the previous weblogging services, microblogging services commonly rely on the opt-in publish-subscribe (pub-sub) model. The basic operation of the pub-sub model is "follow". Being a follower means that the user will receive all the microblogs[1] from the one he follows (named *followee*). A user can follow anyone, and the one being followed need not follow back. Based on the pub-sub model, microblogging services do not require users to explicitly poll information sources, but automatically deliver micro-news to users.

On the other hand, microblogging is more than feed readers for its online social network features. The offered services include helping people to maintain personal profiles, to discover potential acquaintances, to find interesting topics, etc. Thus, social links in microblogging also reflect users' real-life social relations. According to [2], Twitter users can be classified into different social identities, such as *broadcasters* and *miscreants*. 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.

Current microblogging systems depend on centralized architectures, where user clients repeatedly request centralized servers for newest micro-contents no matter whether there is a new update. Such polling-style requests are blind, sticky and superfluous, placing heavy bandwidth burdens on both service providers and user clients. Currently, centralized microblogging services suffer severely from performance bottlenecks (e.g., Twitter's rate limit[2]) and single points of failure (e.g., Twitter's not rare "over capacity" errors[3]). Moreover, the centralized services are quite vulnerable to service blocking as well as malicious attacks (e.g., DoS attacks[4]). Thus, decentralized solutions are desired to relieve the burden while improving the robustness, so as to achieve high scalability, reliability and availability of the offered services.

In this demo, we present Cuckoo, a decentralized and socio-aware online microblogging system. Our objective is to demonstrate (1) how Cuckoo's decentralized architecture helps microblogging systems to save the bandwidth costs, to remove single points of failure, and to resist service blocking and malicious attacks, while performing equally well or even better than its conventional centralized ilks; (2) how Cuckoo's socio-awareness helps the whole system to balance the load, to assist micro-news dissemination, and to reduce the response latency. The demo will show the performance gain that the two key aspects of Cuckoo's design can bring.

2. CUCKOO IN A NUTSHELL

This section briefly introduces some relevant design rationales of Cuckoo. More details can be found in [6].

Decentralization. Cuckoo is built on a hybrid overlay structure, in that it utilizes P2P techniques to reduce bandwidth and storage consumption for the server side. The un-

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

[2] Twitter currently has the rate limit that only allows 150 requests per hour for normal users.

[3] In June 2008, as many as 3% of page requests in Twitter yielded "over capacity" errors [5].

[4] In August 2009, Twitter did be crippled by DDoS attack [1].

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

derlying P2P overlay network combines both structured and unstructured overlays. For providing location services and improving availability, Cuckoo organizes user clients into a structured overlay. We employ Pastry [4] as the structured overlay infrastructure. 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 $O(\log(N))$ steps. For disseminating micro-news, Cuckoo clients with same interests (i.e., following same celebrities or being interested in same topics) form unstructured overlays based on gossip protocol. With the unstructured overlays, micro-news can be propagated to all the subscribers with high probability. On the other hand, the dedicated servers from service providers still hold resources like user profiles and statuses as before, while serving as backup servers to guarantee availability. Thus, Cuckoo does not exclude service providers from their architectural framework.

Socio-Awareness. In microblogging services, a user has one or several of the following social relations: *friend*, *neighbor*, *follower*, and *followee*. Friend is the reciprocate social link between two users, which indicates that the two users are acquaint with each other and willing to help each other. Neighbor refers to the relationship between users with same interests. Follower and followee are the most common one-way connections. A Cuckoo client maintains these social information and takes advantage of social relations. The basic idea is (1) each node and its friends make up the *virtual node* via request redirection, i.e., friends help each other to balance load and improve availability; (2) normal users directly push microblogs to their followers[5]; (3) for propagating micro-news from broadcasters or celebrities like **cnnbrk**, gossip-based push between neighbors are put into use.

3. DEMO SCENARIOS

Demo Setup. In this demo, we run Cuckoo in the site network connected to the Internet. We deploy Cuckoo using 3 netbooks, each of which runs several Cuckoo applications. Each Cuckoo application represents a user of Twitter. These Cuckoo applications make up the Cuckoo microblogging system. We require that each Cuckoo application can access Twitter via Internet. Here, we choose Twitter just to demonstrate that Cuckoo is compatible for most current microblogging systems because Cuckoo does not require any functionality or modification on the server side. One design guideline of Cuckoo is to help microblogging service providers but not bury them (i.e., losing user community).

Cuckoo Use Cases. Cuckoo is easy to use. It provides the basic functions of microblogging services including "tweet" (i.e., publishing a microblog) and "follow". The main user interface can be seen in Fig. 1. After login with username and password, the user can receive the microblogs from all his followees in time ordered by the timeline.

Bandwidth and Traffic Saved. We demonstrate that Cuckoo's decentralized architecture significantly saves the bandwidth and traffic consumed by the bandwidth & traffic analyzer. The analyzer measures the real-time traffic from/to other peers compared with the traffic from/to the server for fetching/disseminating micro-contents. Fig. 1 shows the interface of the incoming traffic analyzer, the red curve and blue curve represent the traffic from peers and servers respectively. We will see that most micro-news can

[5]It is reported in [5] that 90% users in Twitter have less than 100 followers and half of Twitter users have 10 or fewer followers.

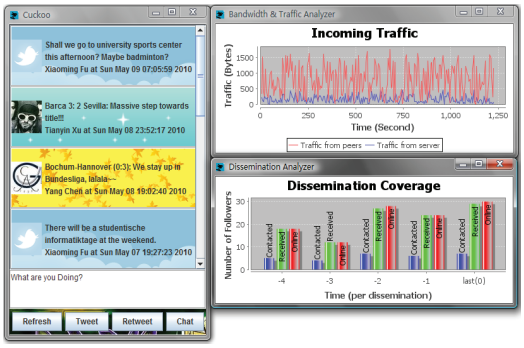

Figure 1: Cuckoo's user interface and analyzers: the left part is the user's timeline-based wall, the upper right part is the bandwidth & traffic analyzer and the lower right one is the dissemination analyzer.

be fetched via peer collaboration, while only some inactive users' microblogs have to be retrieved from the server. Note that the sum of the two kinds of traffic is still much less than the corresponding traffic in centralized architecture, because user clients in Cuckoo no longer need to request the server blindly and frequently.

Dissemination Effect. We also demonstrate Cuckoo's socio-aware dissemination in terms of propagation coverage, i.e., the percentage of online followers that successfully receive the disseminated micro-news. Fig. 1 shows the dissemination analyzer that uses bar chat to illustrate the percentage of contacted followers, followers that receive the microblogs, current online followers over all the user's followers for each micro-content dissemination process. Through this scenario, we show how Cuckoo applications disseminate their microblogs according to their social relations (i.e., follower/followee relations and neighbor relations). We also show that high propagation coverage can be achieved via neighbors' mutual assistance.

4. CONCLUSION

Cuckoo is a decentralized, socio-aware microblogging system that is more scalable and reliable than current centralized equivalents. The motivation of Cuckoo derives from the fact that current microblogging systems severely suffer from performance bottlenecks and single points of failure, and is quite vulnerable to service blocking and malicious attacks.

The demo illustrates Cuckoo's design rationale, architecture as well as some technical and practical considerations. At the same time, we will demonstrate how to use Cuckoo and most importantly, the performance gain that the design of Cuckoo could bring for microblogging services.

5. REFERENCES

[1] CNET News. Twitter crippled by denial-of-service attack.

[2] B. Krishnamurthy, P. Gill, and M. Arlitt. A Few Chirps about Twitter. In *Proc. of WOSN*, 2008.

[3] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *Proc. of WWW*, 2010.

[4] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, 2001.

[5] D. R. Sandler and D. S. Wallach. Birds of a FETHR: Open, decentralized micropublishing. In *Proc. of IPTPS*, 2009.

[6] T. Xu, Y. Chen, J. Zhao, and X. Fu. Cuckoo: Towards Decentralized Socio-Aware Online Microblogging Services and Data Measurements. In *Proc. of HotPlanet*, 2010.