УДК 004.055

P. Zahozhenko, O. Syniavsky

BLATOCOL: DISTRIBUTED MICROBLOGGING SERVICE FOR EARLY ADOPTERS

Having recently become mainstream, microblogging services face several challenges due to their implementation constraints. First, their centralised architecture means each service has scalability and reliability issues; second, signal to noise ratio tends to degrade as more people are joining the service and it becomes exploited by spammers and marketing experts; third, public microblogging services, particularly Twitter, is effectively a form of mass media, which raises the question whether such an influential news source should be allowed to be controlled by a single for-profit corporation.

This paper describes the architecture of a niche distributed microblogging service, targeting early adopters, which addresses all of the major shortcomings of current microblogging solutions. This service is fully distributed, so it doesn't have a single point of failure, nor does it need investments in its infrastructure in order to scale. Its niche nature and features allow it to maintain high signal-to noise ratio throughout all stages of its lifecycle. The proposed service is also open-source, has documented specification and open data format, that nullifies the possibility of its monopolic control by a single organisation.

Keywords: microblogging, signal to noise ratio, scalability, community, open-source, distributed systems.

Introduction

Having appeared less than 5 years ago [9], microblogging already became mainstream with the leading microblogging service Twitter amassing almost 200 million users [5]. Microblogging also made an encroachment into enterprise with Yammer used by more than 100 000 businesses worldwide [14].

Let us list the main reasons of such enormous popularity.

1. 140 symbols message length limitation. Due to message length limitation microblog post usually requires less time and effort to write, thus making users likely to post microblog messages much more frequently than blog posts. Additionally, short message format forces to express one's thoughts clearly, and thus microblog messages are not only easy to write, but easy to read.

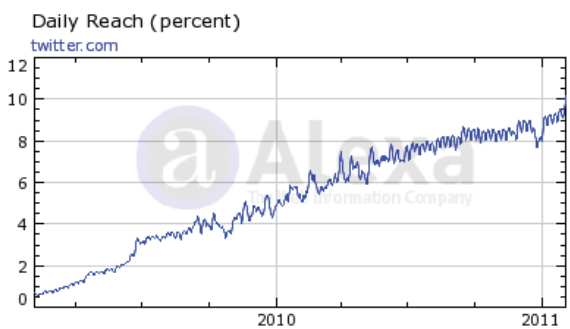

Fig 1. Twitter Alexa rating. Daily reach (percent of global Internet users who visit twitter.com)

2. Asymmetric directed social graph. Traditional social network services, such as Facebook, use sym-

metrical directed social graph, where each graph edge, or "friendship", is bidirectional. For all practical purposes such a graph may be considered equivalent to a simple undirected graph. Contrary to that, microblogging services use directed social graph, where each edge, or "follow" relation, doesn't need to be have the corresponding inverted edge. This feature greatly improves noise filtering capabilities of microblogging services: user doesn't need to follow all his followers, but choose to follow only those whos tweets he finds interesting.

3. Ubiquitous availability. From their very beginning, microblogging services, particularly Twitter, could have been accessed via both web interface and SMS. (SMS is used by 4 billion people worldwide [11]). As time passed, native Twitter clients were developed for every major desktop and mobile operating system. This was made possible by Twitter's simple and clean REST API.

4. Broadcasting vs narrowcasting. Microblogging allows both broadcasting and narrowcasting of messages. When a user tweets, he "narrowcasts" a message to his followers. If they find that the narrowcasted message might be of interest to their followers, they "retweet" it either by using in-built functionality or by simply copying the message adding the reference to original author. Thus, important message quickly propagates through the social graph. There are examples of extremely important messages, such as early reports of earthquakes or revolutions, propagating through the social graph very quickly, having hundreds of retweets per minute, reaching the other end of the world within minutes. For example, during May 2008 earthquake in China Robert Scoble spotted earthquake-related tweets from China and retweeted them to his followers before earthquake report appeared on USGS and an hour before CNN breaking news [12]. On the other hand, messages of local importance became propagated only through certain area, and messages of little-to-no importance aren't propagated at all, thus not spreading the noise throughout the system.

These features allowed microblogging services to become the fastest news source on the planet (Twitter), or within the boundaries of particular organization (Yammer), as well as valuable mean of communication. However, current microblogging solutions have a number of problems, listed in the next chapter.

Shortcomings of current microblogging services

1. Centralized architecture. Current microblogging services are centralized and thus sometimes suffer from outages due to performance bottlenecks and single point of failure (as shown by Twitter frequent outages in 2008-2009). They are also vulnerable to DoS attacks and can be relatively easily blocked by a single corporation or government. As described in [8], services such as Twitter waste lots of traffic and other resources because clients are constantly polling the central server. Although we traditionally listed this limitation as the first one, it can be solved by existing microblogging services without radical changes to their underlying principles. As shown by Google, web service scalability is practically limitless if one abandons classical architecture and uses specialized tools, e.g. abandoning RDBMS and using key-value datastore instead, such as Google's BigTable or its open-source equivalent Cassandra [4]. Nevertheless, even though scaling centralized web service is possible, it requires substantial investment into its infrastructure.

2. Signal to noise ratio decay throughout microblogging service's lifecycle (information overload). The more people are joining the service, the more messages they generate, the harder it becomes to find relevant information in message flow. The single most popular microblogging service - Twitter - suffers from this problem the most. Twitter has several tools to help its users increase the relevancy of their tweet streams. First of all, its asynchronous social graph itself makes it easier to follow only those users whose tweets are relevant. Second, lists functionality provides a way to group users by topic and subscribe to lists, created by other users. Third, hashtags allow one to follow all messages, containing a specific hashtag. Forth, some tweets contain geographical information which allows to limit searches by a certain geographical location; however, this feature is not wide-spread, as not all Twitter clients are location-aware and only a fraction of users post their location from location-aware clients due to privacy concerns. Fifth, "trending topics" functionality provides an overview of current globally popular hashtags. Although somewhat effective, all these means are unable to deal with the global trend of decaying signal to noise ratio.

3. The need to earn revenue. As stated in microblogging problem #1, Twitter requires a huge and expensive infrastructure, and thus a stable revenue stream to support itself. To reach its revenue goals, Twitter already uses promoted trending topics. To meet the increased infrastructure expenses, it might introduce sponsored tweets or some form of contextual ads, thus further decreasing its signal to noise ratio.

4. Twitter long-term plan is to become "the pulse of the planet" . They also plan to become the first web service to reach 1 billion registered users (this confidential information was leaked by TechCrunch, [13]). Given their current growth (see fig. 1), such scenario is not impossible. If Twitter plan becomes reality, they will have the tremendous power, which they might exploit in order to increase their revenue.

Blatocol

In this article we present the service that solves all of the above-mentioned problems. This service is not general-purpose microblogging solution, as it is not intended for everybody. We believe that "one tool fits all" approach will not work for microblogging. Instead, we propose a number of interoperable services, each of them serving the needs of a particular niche. This article focuses on the first of these services: distributed microblogging service for early adopters, codenamed "Blatocol".

Distributed architecture

Many alternative microblogging solutions focus on centralization as the main problem of current offerings and propose various distributed architectures. Status.net offers a model in line with that of Wordpress, with both an open-source microblogging software, which can be installed on any LAMP server, and a cloud solution running similar software (Identi.ca) [7]. SMOB project [2] proposes a different approach: the service consists of a number of hubs, that communicate with each other to exchange microblog posts and follower notifications [3]. Cuckoo project demonstrates another approach: peer-to-peer architecture backed by a set of centralized servers [8]. The system we describe in this paper is also peer-to-peer, although it lacks any centralized servers and thus is purely peer-to-peer. However, in the article we are calling our peers "servers" to distinguish them from "clients" - tools, used to interact with peers, as each peer ("server") does not have any sort of interface beside its REST API.

Target audience

Blatocol is designed for early adopters, even for a specific niche of early adopters: programmers, computer scientists, and system administrators.

Being a part of this specific niche ourselves, we understand its needs. The question arises, how it is better to limit service users to its target audience. In this case, it cannot be solved by the specific thematics of the service, cause microblogging is in itself general-purpose, not bound to any specific topic. We believe that the best solution is to make the service function in a way that it will be unusable by someone not from our intended target audience. In this case, the task is relatively easy: make it accessible only by programmers. Programmers are very specific people, they use tools no one else is able to use.

The whole architecture of our service is designed in a way that it can only be used by the target audience. Each registered user is required to run his own software server. This server resources consumption is very low, so it can be easily run on a typical personal computer or an old server, or a single Heroku Dyno, using only a fracture of its resources. However, setting up such a server requires certain degree of technical skill and a real IP address (otherwise, one's communication will be limited to one's LAN). Such a requirement serves 2 purposes: 1) it allows the system to be distributed, with each user storing all of the messages on his own server and pushing newly posted messages to all of his followers' servers 2) it greatly limits service's target audience, as only a tiny fracture of global Internet users understand what software server is, let along know how to set it up.

The service architecture is such that each user's server is responsible for pushing his newly posted message to all his followers' servers, which means that one has to maintain his server and guarantee its uptime in order to spread information. More so, the more followers user has, the more powerful server he has to use in order to cope with the load. Thus, user influence within the system is always proportional to his contribution to system's hardware resources. It is in line with meritocracy found in open-source communities: one's influence is proportional to one's contribution.

To make the service even more usable for its target audience (and unusable for everyone else) we decided that the tools (clients) to access the service should be either command line or as a library for some programming languages. Currently, the only way to communicate with the service is via curl, but we're adding Ruby library and command line tool, libraries and tools in other programming languages will follow.

By strictly limiting the service to its intended target audience we hope to achieve high signal to noise ratio throughout the whole service lifecycle. Should this service ever become popular outside the boundaries of its intended target audience, which we highly doubt, we will consider it a failure. The ideal state of this service is permanent early-adopter stage.

Current implementation and future work

Current service implementation is the work in progress. The first server is implemented in Ruby on Rails. Basic messaging and distributed architecture is ready, but there are still a number of open questions:

1. Security. We've got several different possible security schemes, but haven't yet decided which one to implement.

2. Software updates. Ideally, all future versions of our service's software should be backwards compatible on API level. However, new features may be introduced, so we either need a way to force software updates or leave this responsibility to users.

Service implementation is open-source, current stable version is available on Github: https://github.com/buru/blatocol

The focus is to make API as simple as possible, so that it can be easily implemented in any web framework. As soon as API stabilizes, we will release its detailed specification.

Summary

Summing up the main principles of the proposed service, it's main features are:

1. Distributed architecture. Each registered user runs his own server.
2. Unusable for technically illiterate users.
3. Simple REST API.
4. Heterogeneous system. Servers implemented in different programming languages and frameworks, running on different platforms.
5. Open-source implementation.
6. Uncontrollable system. No single organization is able to control such a service.
7. Potentially interoperable with other similar services if/when they appear.

Literature

1. Ahonen Tomi. Insider's Guide to Mobile, Free Edition.
2. Alexandre Passant, Tuukka Hastrup, Uldis Bojars, John Breslin, Tuukka Hastrup, Uldis Bojars, John Breslin. Microblogging : A Semantic and Distributed Approach // Proceedings of the 4th Workshop on Scripting for the Semantic Web, Jun 2008.
3. A. Passant, U. Bojars, J. G. Breslin, T. Hastrup, M. Stankovic, and P. Laublet. An Overview of SMOB 2: Open, Semantic and Distributed Microblogging // 4th International Conference on Weblogs and Social Media, ICWSM 2010. AAAI, 2010.
4. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable // ACM Transactions on Computer Systems. – 2008. – 26. – 2:26.
5. Meaningful Growth // Twitter Blog. Web. – Режим доступу: <http://blog.twitter.com/2010/12/stocking-stuffer.html>. – Назва з екрана.
6. Scott John. Social Network Analysis: a Handbook. – London : SAGE Publications, 2000.
7. StatusNet | Your Network. Web. – Режим доступу: <http://status.net>. – Назва з екрана.
8. Tianyin Xu, Yang Chen, Lei Jiao, Ben Y. Zhao, Pan Hui, Xiaoming Fu. "Cuckoo: Decentralized and Socio-Aware Online Microblogging Services" Technical Report No. IFI-TB-2011-01, Institute of Computer Science, University of Goettingen, Goettingen, Germany. Jan 2011. Print.
9. "Twitter | CrunchBase Profile". CrunchBase, The Free Tech Company Database. Web. – Режим доступу: <http://www.crunchbase.com/company/twitter>. – Назва з екрана.
10. "Twitter Is Censoring the Discussion of #Wikileaks | Safety First." Safety First | This Site Is about Things Thought through. Web. – Режим доступу: <http://bubbloy.wordpress.com/2010/12/05/twitter-is-censoring-the-discussion-of-wikileaks/>. – Назва з екрана.
11. "Twitter.com Site Info". Alexa the Web Information Company. Web. – Режим доступу: <http://www.alexa.com/siteinfo/twitter.com>. – Назва з екрана.
12. "Twittering the Earthquake in China – Scobleizer." Scobleizer – Searching for World-changing Technology. Web. – Режим доступу: <http://scobleizer.com/2008/05/12/quake-in-china>. – Назва з екрана.
13. "Twitter's Internal Strategy Laid Bare: To Be "The Pulse Of The Planet"" TechCrunch. Web. – Режим доступу: <http://techcrunch.com/2009/07/16/twitters-internal-strategy-laid-bare-to-be-the-pulse-of-the-planet/>. – Назва з екрана.
14. Yammer : The Enterprise Social Network. Web. – Режим доступу: <http://yammer.com>. – Назва з екрана.

Захоженко П. О., Синявський О. Л.

BLATOCOL – РОЗПОДІЛЕНИЙ МІКРОБЛОҐҐІНГОВИЙ СЕРВІС ДЛЯ РАННІХ ПРИБІЧНИКІВ

Нещодавно ставши популярними, мікроблоґґінгові сервіси зіткнулися з певними складностями. По-перше, їхня централізована архітектура призводить до проблем із масштабованістю; по-друге, рівень сигнал-шум в системі зменшується разом із зростанням популярності сервісу; по-третє, публічні мікроблоґґінгові сервіси (зокрема, Твіттер) є засобами масової інформації, отже, небажано, щоб їх контролювала одна корпорація.

Ця робота розглядає архітектуру нішевого мікроблоґґінгового сервісу, призначеного в першу чергу для ранніх прибічників (найбільш прогресивних користувачів, які постійно шукають щось нове), що покликаний вирішити всі труднощі сучасних мікроблоґґінгових систем. Сервіс повністю розподілений, а отже, легше масштабується і не потребує значних інвестицій в інфраструктуру. Завдяки своїй нішевості та іншим особливостям сервіс здатний забезпечити стабільний рівень сигнал-шум впродовж всіх стадій свого життєвого циклу. Крім того, запропонований сервіс є продуктом із відкритим вихідним кодом, задокументованою специфікацією та відкритим форматом даних, що унеможливлює монопольний контроль даного сервіса будь-якою організацією.

Ключові слова: мікроблоґґінг, рівень сигнал-шум, масштабованість, спільнота, відкритий вихідний код, розподілені системи.

Матеріал надійшов 11 травня 2011 р.