Birds of a FETHR: Open, decentralized micropublishing

Daniel R. Sandler Dan S. Wallach
{dsandler,dwallach}@cs.rice.edu

Abstract

Microblogging, as exemplified by Twitter, is gaining popularity as a way to exchange short messages within social networks. However, the limitations of current microblog services—proprietary, centralized, and isolated—threaten the long-term viability of this new medium. In this work we investigate the current state of microblogging and envision an open, distributed *micropublishing* service that addresses the weaknesses of today's systems. We draw on traces taken from Twitter to characterize the microblogging workload. Our proposal, FETHR, connects micropublishers large and small in a single global network. New messages are gossiped among subscribers using a lightweight HTTP-based protocol. Cryptographic measures protect authenticity and continuity of updates and prove message ordering even across providers.

1 Introduction

Thanks to the phenomenal success of the Twitter service, microblogging has emerged as a significant new form of internet communication, yet it defies easy classification or even explanation. At its most essential, a microblog is a personal publication medium composed of very short, ephemeral messages. Unlike a conventional weblog, however, microblogs are tightly bound to one another via subscriptions: each user's view of the system interleaves his own published entries with the contributions of other microbloggers whom he has explicitly chosen to follow. This opt-in, social networking model effectively resists abuse while still allowing an off-hand comment to blossom into a large public discussion.

Like blogging before it, microblogging has the potential to become mainstream within just a few years—a "communication utility," in the words of one of Twitter's founders [1]. Unfortunately, as useful and indispensable as it has already become for early adopters, in its present form it is unlikely to earn a permanent place alongside email, instant messaging, and the Web. Today's fledgling microblogging offerings are centralized services, each a single point of control and failure for its own isolated user community. Twitter has earned some notoriety for its difficulties coping with its rapid growth, but its users are reluctant to move to more reliable competitors due to network effects. If this popular new communication medium is to survive and flourish, it must be rearchitected to be decentralized, secure, reliable, and open to new service providers.

In this paper, we begin by detailing the properties and limitations of current microblogging systems, including some early analysis of the microblogging workload derived from several weeks of Twitter traces (Section 2). This helps shape our proposed design for FETHR, a simple but powerful architecture for general *micropublishing* distributed across the Internet (Section 3). Finally, we describe our prototype implementations and sketch our future plans for the protocol (Section 4).

2 Background

The Twitter service has been the subject of countless news stories, blog posts—and, yes, Twitter updates—since its launch in 2006. The model is deceptively simple: a user may publish short messages—up to 140 bytes, just enough text to fit in an SMS—to others who have explicitly subscribed to that user's updates ("tweets" in Twitter jargon). Subscribers will receive those updates via the Web, on their cell phones, or via dedicated client software, as they prefer. Each user's "timeline" of messages is published on the Web for bulk perusal, but updates are more commonly consumed live, aggregated with the latest news from each reader's own idiosyncratic set of subscriptions.

Conceptually, the microblog falls somewhere in the space between a public weblog, a cloistered chat room or mailing list, and a private instant message exchange. It is perhaps most akin to a more corporeal mode of socialization: casual hallway conversation. Indeed, Twitter's main interface is characterized by a single, simple text entry box whose accompanying prompt, "What are you doing?", sounds like the canonical beginning of an exchange by the watercooler.

Even so, it has found use far beyond idle chatter. Twitter has become an important tool for interacting with broad groups of people, from companies using Twitter for customer service [1] to musicians (large and small) connecting with fans, to political candidates getting the message out to supporters and voters. It has become a source of immediate information on breaking news; Twitter was awash in reports of the major 2008 earthquake in China within seconds, before the shaking stopped—well before information appeared on major news and geology sites [10].

As Twitter's userbase has grown (by a factor of 14 in the last year [7]), dozens of competitors have quietly germinated. They differentiate themselves by offering additonal features (e.g., the unique graphical timeline view of plurk.com) or targeting different users (e.g., yammer.com, which caters to corporate users). Identi.ca distinguishes itself by its open-source codebase, allowing easy creation of Twitter clones. The combined userbase of these services, however, hardly approaches that of the Twitter juggernaut.

Additionally, other web services with "social" features (viz., subscribing to updates from other users) exhibit traits of micropublishing; examples include the list of saved pages on delicious.com or Google Reader, or a flickr.com user's recent photos. The FriendFeed aggregator attempts to unify these various streams of public data, allowing users to consolidate updates from their various social networking accounts into a single (copious) view. Finally, Facebook—reigning champion of social networks as of this writing—has, since the introduction of the News Feed in 2006, presented each user with an aggregate view of status updates and other recent changes made by his friends. The News Feed feature was revised in March 2009 to highlight and expand this feature [3]; the result, including a prominent text entry box (asking the user, "What's on your mind?"), is strikingly similar to the Twitter experience.

Common to micropublishing services is their opt-in subscription model. Unlike email, users receive messages only from others whom they've previously explicitly decided to follow (or "friend" in Facebook). A user's circle of contacts becomes his own unique view on the world, a cross-section of hallway discussions and idle commentary, all of it from sources of interest to that user.

2.1 Users and uses of Twitter

It is our goal to design a micropublishing *infrastructure* that integrates many different microblogging services, large and small. Real data taken from Twitter's global public timeline helps us understand the microblogging community and workload that must be accommodated.

We collected trace data over a three-week period (3–25 Sep 2008) by polling, once per minute, a special URL[1] that returns the 600 most recent public messages from all Twitter users. Our data set contains 4,917,042 public messages from 472,735 distinct Twitter users.

We first consider popularity in Twitter as measured by the number of subscribers to a given user's updates. As is typical of popularity contests, the data resemble a Zipf or power-law distribution. Figure 1 shows the cumulative distribution of subscribership size across users observed in our traces. We observe that roughly half of Twitter's users have ten or fewer subscribers; only ten per cent have 100 or more. A very few "Twitter celebrities" (led by cnnbrk,[2] according to twittercounter.com) have tens or hundreds of thousands of subscribers.

[1]We thank Alex Payne (@al3x), API Lead at Twitter, for access to this resource.

[2] @cnnbrk: a Twitter account used to post CNN breaking news, often before it reaches the cnn.com homepage. As of this writing (March 2009), cnnbrk has in excess of 622,000 subscribers.

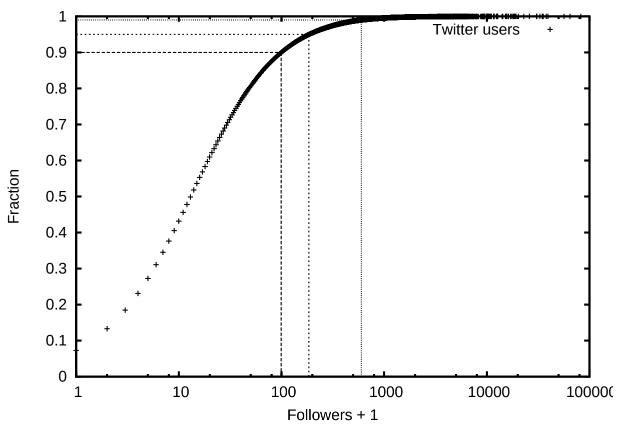

Figure 1: CDF of Twitter users' followers. 10% of users have 100 followers or more; 5% have 186; 1% have 598.

To understand the kind of traffic generated by this wide spectrum of users, we added a second dimension: message volume. Figure 2 plots messages vs. followers for each individual Twitter user. The average user has about 100 subscribers and sent about the same number of messages. While there are anomalously high-output users (usually robots such as RSS feeds) and the aforementioned highly-subscribed celebrities, there appear to be no users who fall into both categories. (Verbosity appears somewhat anathema to popularity.) We note finally that a few users have a high message-follower product; they generate the greatest per-capita proportion of Twitter traffic, and we must take steps in our design to accommodate these high-impact users (and their subscribers) even though the lion's share of Twitter users make much smaller demands on the microblogging service.

2.2 Limitations of modern microblogging

Twitter, and all other existing microblogging websites, are centralized systems. They may use distributed systems internally to manage load, but from the standpoint of the user, they are monolithic black boxes with a concomitant set of drawbacks and limitations:

The service is a performance bottleneck and central point of failure. Twitter has a history of sluggishness or outright downtime, particularly during times of exceptional load. For example, as many as 3% of page requests in June 2008 yielded "over capacity" errors (Twitter's now-famous "fail whale" image [11]). To mitigate performance issues, Twitter has been forced to take drastic measures, including rate-limiting, retraction of features that have proven difficult to scale (e.g., XMPP service), and controlled partial outages (i.e., turning off certain features). As of late 2008, Twitter continues to improve its performance, but its steadily-growing userbase seems to hungrily consume the additional capacity. While

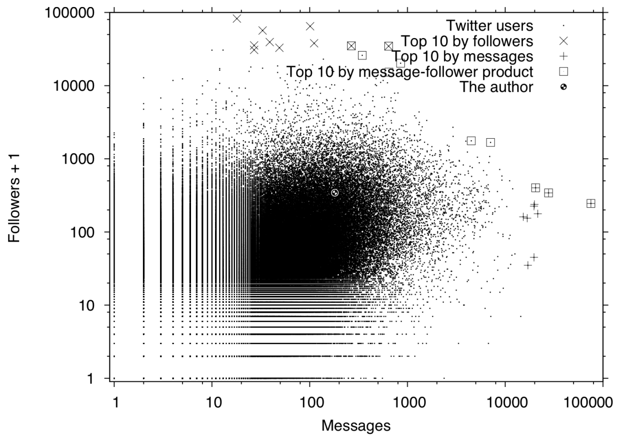

Figure 2: Twitter users in message-follower space. A rough centroid exists: most users have about 100 followers and sent about 100 messages during this three-week trace.

such growing pains are not uncommon for new web apps, users will benefit greatly from a system that does not depend completely on a single service provider. Similarly, security breaches (such as the recent compromise of an administrative account that allowed several dozen high-profile Twitter accounts to be hijacked [12]) threaten all users of such a centralized system.

Poor design decisions are forced on everyone. Centralized and distributed systems alike will enforce architectural decisions for all users, but Twitter also applies higher-level limitations that are not strictly necessary but users are powerless to change. For example:

- To shorten messages, Twitter automatically replaces URLS of a certain length with TinyURL contractions, whether or not users want this behavior (and associated dependency on tinyurl.com).
- Twitter has a public API for third-party clients, but it has constraints (rate limiting, for example) that hinder external development and make some extensions difficult or impossible to realize.
- Twitter has a simple "reply" syntax: if Alice prefixes a message with the text @Bob, that message is assumed to be a reply to Bob's most recent message at that time. The Twitter API is slightly more flexible here, as it allows client software to supply an arbitrary message ID as the replied-to message, enabling a "reply to this tweet" feature in te GUI. However, only one parent is allowed, and it is not possible to walk the links in the opposite direction (looking for replies).

Network effects prevent users from leaving. As mentioned earlier, Twitter now has many contemporaries: competitors offering additional features, better uptime, faster page loads, or simply an alternative to the dominant service. Unfortunately, these systems are unable to talk to one another, and so *network effects* become powerful influencing factors in user behavior. The intrinsic value of joining Twitter—that is, the benefit it offers a single user in a vacuum—is small, and likely similar to other services. But that value is massively amplified by the presence of other users with whom each new user may interact. Once dominant players are established, new competitors find it impossible to gain ground. The likely result is something akin to the current IM landscape: a few large services that do not interoperate, forcing users to maintain accounts on each to ensure connectivity to everyone.

Alternatively, modern internet email has escaped this Balkanized fate by inheriting a venerable set of protocols for interoperation. There are indeed a small number of quite large providers of email (Yahoo!, Hotmail, and so on), but thanks to SMTP, RFC 2822 and MIME messaging, users need not acquire separate email accounts on each service to communicate with users of those services. Furthermore, the burden of handling email is distributed across the world; small servers are suitable for individuals or small organizations, while major webmail providers may apply datacenter-scale techniques to satisfy a giant userbase. Users are free to migrate from one provider to another in search of better performance or more features; local failures only impact local users. We wish to achieve similar properties in the evolution of microblogging.

3 FETHR

3.1 Objectives

We seek a robust decentralized infrastructure for micropublishing in order to better support the remarkable interest in this new medium. It is imperative that we connect today's isolated services in a distributed, global, interoperating ecosystem—a model that has been demonstrated to be effective and enduring by the Web and email. We are emphatically not trying to improve the performance of any one microblogging service (e.g., Twitter); rather, we wish to admit new providers, freeing users to choose any available service (or perhaps start one of their own) without abandoning the dominant user community.

A successful design will have the following properties:

Decentralized. It must not rely on a single service for authentication, publication, or subscription to updates.

Robust. It must continue to function when some of its participants are unavailable or malfunctioning.

Secure. The authenticity, integrity, and completeness of any publisher's updates must be preserved by the system and independently verifiable by subscribers.

Abuse-resistant. The system must not allow bad actors to significantly decrease the quality of service offered to other users. The nuisance and spamming capability of attackers must be minimized or eliminated.

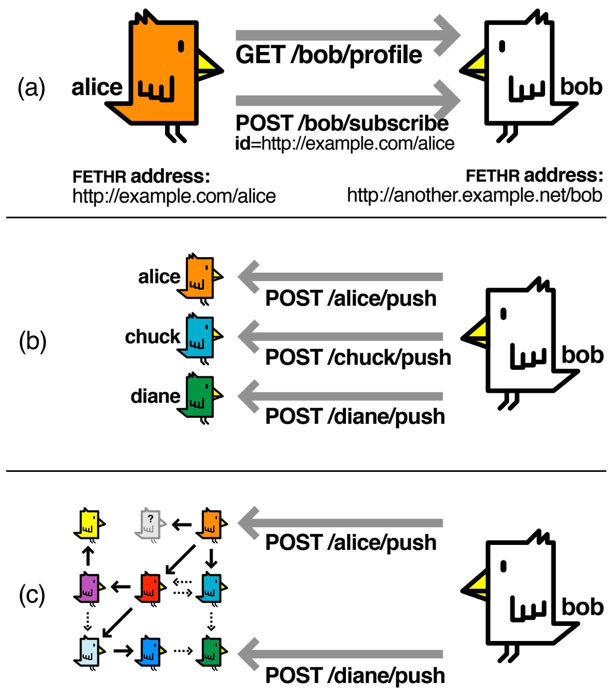

Figure 3: FETHR publishing. In (a), Alice collects information about Bob by requesting his profile, and then subscribes by posting a subscription request containing her canonical URL. When Bob wishes to publish updates (b), he pushes new signed messages to his subscribers using HTTP POST. To control the costs of publishing to many recipients (c), Bob may push new messages to a subset of his subscribers who then gossip those updates.

Timely. Updates must be distributed to subscribers promptly; hard real-time guarantees are unnecessary, but delays of more than a few minutes will severely impact the utility of a micropublishing system.

Interoperable. The protocol design must allow existing micropublishing and microblogging services to interoperate, allowing users of service A to subscribe to any number of users of service B.

Flexible. While current microblogging services tend to contain a single 140-byte payload, other content may be desirable.

3.2 Functional description

FETHR (Featherweight Entangled Timelines over HTTP Requests) is our protocol design for distributed micropublishing. Users of web applications that understand the FETHR protocol are able to subscribe to one another's updates as if they were members of the same conventional centralized microblogging service. We term this global federated service the "FETHR publishing network." At a high level, the FETHR network involves the following entities and functions [illustrated in Figure 3 (a) and (b)]:

- Individual users participating in FETHR may fill the role of a *publisher*, *subscriber*, or both. (Note that many publishers may in fact be users of the same multi-user service; for example, a FETHR-enabled Twitter would host many thousands of publishers.)
- A FETHR publisher maintains a *local timeline* of *entries*. The content of these entries is unspecified; for a microblog they would likely include fragments of plain text or HTML. Whatever the payload, the publisher must wrap it in a digital signature to allow subscribers to validate the entry's authenticity.
- Each publisher provides a *canonical* URL where those entries and other peripheral information can be found in human-readable (viz., HTML) format.
- This URL also serves as the publisher's "address"; it is a rendezvous point for FETHR clients to consume entries and other metadata in a machine-readable format (detailed below).
- One or more FETHR participants, given the publisher's canonical URL, can ask the publisher to be *subscribed* to (viz., "follow") her updates.
- The publisher is responsible for *distributing* new entries to each subscriber.

3.3 Message distribution

We decide in FETHR to adopt the strategy of messaging systems (email & IM) in pushing the entire contents of updates to subscribers rather than merely notifying subscribers of new information. Micropublications are often short, so a notification and the full message text are of roughly the same size; furthermore, by replicating updates among subscribers, we allow that data to be reviewed (or relayed to new subscribers who did not originally hear it) if the original publisher later becomes unavailable. Currently, when a microblog service goes offline, users are unable to even read old messages without the help of some local cache, such as a desktop client.

Even with small microblog updates, there exist scenarios in which distribution load is of concern. For small readerships—a few hundred subscribers—serial unicast is quite tolerable. However, while the traces presented in Section 2.1 show that this accounts for the vast majority of Twitter users, there are outliers with readerships in the tens or hundreds of thousands. Subscriber-assisted redistribution of messages is therefore an essential feature for these users who find themselves so popular that pushing updates directly to every subscriber becomes intractable. (Worse still would be a polling-based architecture such as RSS, in which popularity brings with it many redundant update requests even when no update is available [8].)

FETHR subscribers may therefore propagate public updates via gossip. Gossip is well-established as a lightweight mechanism for achieving eventual consistency in distributed systems, and is quite resilient in case of network failures and asymmetries. A FETHR publisher may elect to send a message only to a subset of her subscribers, with the instruction (embedded in the signed message) that the message should be gossiped to other subscribers. By limiting her fanout, the publisher controls publishing load and bandwidth requirements while still ensuring that all subscribers receive the message with very high probability. (The necessary fanout has been shown to be logarithmic in the size of the overall subscribership; the probability of success may be tuned by adding a small constant number of additional recipients to the fanout [4].) Subscribers who *never* receive an update via gossip can detect this situation (by observing a cessation of updates) and take action, such as contacting the publisher directly.

We note that gossip is not the only point in the design space. Application-level multicast trees guarantee that updates reach all subscribers in a logarithmic number of steps without unnecessary traffic, but additional complexity is typically necessary to maintain the system (e.g., multicast tree repair; upkeep of a structured routing overlay). Furthermore, redundancy must be reintroduced in order to survive failure and attack. However, in low-churn situations with very many subscribers, extremely high update rates, or more real-time requirements, this may be a reasonable approach, and such a propagation strategy could easily be layered atop FETHR.

3.4 Security features

The presence of untrusted third parties in the update distribution path requires additional measures to safeguard against attack. To guarantee authenticity, publishers include digital signatures with each update. This is not sufficient to defend against suppression of individual updates, so FETHR also links a publisher's entries together in a hash chain.

Each entry published in FETHR includes the cryptographic hash of one or more of that publisher's prior entries, establishing a provable partial ordering over those entries (and a total ordering if every message succeeds exactly one other, save the first). A publisher's timeline is therefore a kind of secure log [2, 9] in which the most recent entry serves as an authenticator for her entire history. These chains allow subscribers to identify gaps in a publisher's timeline (introduced by the intentional or accidental failure of intermediaries to pass along updates).

Although a simpler mechanism such as sequence numbers would achieve the same effect, hash chains offer additional useful properties. First, authenticators allow us to verify that a publisher has not retroactively modified or elided any entry from his timeline. While current Twitter users may not be overly concerned with the historical integrity of timelines, we believe a more general micropublishing service will benefit from such a property, particularly if we create dependencies between timelines from different publishers.

Therefore, a FETHR publisher includes hashes of prior events from *other* participants in addition to his own, weaving many individual timelines into a single DAG spanning the entire FETHR network. With their timelines thus *entangled* [6], different publishers can now establish a provable order between their messages in the manner of Lamport's logical clocks [5], giving our platform timestamping and notarizing properties.

We reuse this notion of explicit pointers between events to represent and reconstruct threads of conversation, which abound on Twitter but (as noted in Section 2) are often inferred based on cues in the message text. FETHR makes these relationships explicit by also including in each message the authenticator of any number of related messages, permitting reconstruction of the entire conversation—something not easily achieved at present.

3.5 Adoption; incentives

While FETHR represents an entirely new protocol for micropublishing, it is one that maps neatly onto existing systems like Twitter, with good reason: it is Twitter's large community that most contributes to its continued success, and it is our explicit goal to include those users. We therefore intend for current microblogging systems to add support for the FETHR API to their offerings. Every existing user of a system like Twitter would become a FETHR publisher: Twitter would respond, for each user, to FETHR queries and push FETHR updates.

As the pioneering microblogging example, Twitter is unlikely to suffer much from the arrival of competitors; in fact, the profile of microblogs can only be raised by a larger group of users who can all talk to one another. But Twitter has an even more compelling incentive to participate: when, inevitably, a serious challenger appears, interoperability allows it to remain relevant and connected rather than becoming a ghost town (like so many other social networking sites now abandoned for newer offerings).

4 Conclusions and Future Work

Twitter and its ilk have discovered a previously unknown sweet spot in the social network design space: lightweight messaging via short updates, broadcast to the world, funneled along social links to interested subscribers. To support the evolution of this style of communication, we have proposed FETHR, a similarly lightweight distributed publishing system with Twitter-like subscription semantics.

Our future work includes flexible *group communication* (a frequently-requested but rarely found microblogging feature); multiple-level *privacy* (expanding on Twitter's simplistic "protected account" to allow timelines that mix public and private information, borrowing also from the successes and failures of Facebook's privacy system); and *attention management* (filtering a user's view to help deal with the increasing volume of data that micropublishing often creates). We believe that each of these can be built on top of the FETHR protocol using only local policy changes.

We intend also to expand our prototype, which currently includes a single-user FETHR microblog *Birdfeeder* and a Twitter gateway service. Together they comprise about 1500 lines of Python code, owing to the simplicity of the design and reliance upon existing technologies like HTTP. We are currently collecting data from the system, which has been in operation since June 2008 and has processed 120,000 messages (to and from the authors). Birdfeeder will be a useful platform as we explore the directions described above and identify necessary optimizations to the design; we plan to make our code and data available at http://brdfdr.com.

References

[1] S. Baker. Why Twitter matters. *Businessweek*, May 15 2008.

[2] M. Bellare and B. Yee. Forward integrity for secure audit logs. Technical report, UC at San Diego, Dept. of Computer Science and Engineering, Nov. 1997.

[3] P. X. Deng. Welcome to your new home page. The Facebook Blog, Mar. 11 2009. http://blog.facebook.com/blog.php?post=59195087130.

[4] 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.

[5] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[6] P. Maniatis and M. Baker. Secure history preservation through timeline entanglement. In *USENIX Security*, Aug. 2002.

[7] M. McGiboney. Twitter's tweet smell of success. Nielsen Wire, Mar. 18 2009. http://blog.nielsen.com/nielsenwire/online_mobile/twitters-tweet-smell-of-success.

[8] D. Sandler, A. Mislove, A. Post, and P. Druschel. FeedTree: Sharing Web micronews with peer-to-peer event notification. In *IPTPS '05*, Feb. 2005.

[9] B. Schneier and J. Kelsey. Cryptographic support for secure logs on untrusted machines. In *USENIX Security*, Jan. 1998.

[10] R. Scoble. Twittering the earthquake in China, May 12, 2008. http://scobleizer.com/2008/05/12/quake-in-china/ (accessed 8-Oct 2008).

[11] E. Williams. Measurable improvements, July 2008. http://status.twitter.com/post/41492128/measurable-improvements (accessed 25-Sep-2008).

[12] K. Zetter. Weak password brings 'happiness' to Twitter hacker. Wired News, Jan. 6, 2009. http://blog.wired.com/27bstroke6/2009/01/professed-twitt.html.