GhostPost: Seamless Restoration of Censored Social Media Posts

Frederick Douglas
University of Illinois Urbana-Champaign

Matthew Caesar
University of Illinois Urbana-Champaign

Abstract

The control of voices within a country is as important to a censor as blocking information from outside. This control must extend to social media. Screening every post prior to publication is not practical; instead, censors find and delete objectionable content after it has been posted. This paper presents GhostPost, a distributed system that conveniently and safely restores deleted posts on any social media platform, with an implementation for Sina Weibo. Our simulations show that even if the censor deletes most posts within two hours (roughly the capability of Sina Weibo's censor), it cannot prevent a well established GhostPost deployment from preserving a majority of the posts our users would want to see.

1 Introduction

Governments have always sought some degree of control over what their citizens say, regardless of the means of communication involved. Censorship has followed civilization onto the Internet, with governments blocking traffic to undesirable web sites and services based in freer countries. China, the country this paper focuses on, is perhaps the most notable example; its Internet censorship is well enough known to have gained a nickname in common parlance: "The Great Firewall."

China's Internet censorship includes simple techniques such as disruption of DNS resolutions of banned domain names[6] and blocking of flows to blacklisted IP addresses, as well as more sophisticated techniques, such as killing flows when deep packet inspection detects a banned keyword[1], and identifying Tor connections with traffic pattern fingerprinting[7, 8]. Since sites and services hosted within the censored country could generally be compelled to remove the offending content, this form of censorship is, broadly speaking, an attempt to prevent the inward flow of unwanted ideas from other countries.

The other main goal of censorship is to prevent people from expressing their own ideas. Social media represents a unique challenge to a censor aiming for this goal. At a traditional newspaper, a handful of known employees write articles that can be carefully monitored, perhaps even requiring explicit approval from the censor to publish. With social media, a significant fraction of the country's entire population is speaking its mind, spontaneously and constantly. Punishing everyone who says anything in any way "wrong" is infeasible, and requiring posts to be pre-approved by the censor would be practically equivalent to shutting the platform down. Censored social media platforms will therefore always allow dissent to be (temporarily) expressed.

Conventional wisdom has come to view social media as particularly well suited to facilitating social unrest. Accordingly, the Chinese government is very careful in its control of social media sites, such as Sina Weibo. A study[9] of how long controversial posts survive before deletion estimated a lower bound of 4,200 full-time human employees monitoring Sina Weibo. Whatever the exact number, the Chinese censor clearly devotes a significant amount of resources to policing social media.

Although this army of censors keeps a lid on objectionable content in the long run, each undesirable post does manage to be visible for a short time. Over 90% of posts fated to be deleted survive less than 24 hours, but the median deletion time is over two hours[9]. These short periods of availability turn out to be sufficient for thwarting censorship.

This paper describes GhostPost, a functioning system for effectively, conveniently, and safely preserving and disseminating posts deleted from Sina Weibo. GhostPost relies on a community of Weibo users who wish to circumvent Weibo's censorship. These users record all Weibo posts they see, and report those they see deleted to the rest of the GhostPost system. Deleted posts are automatically inserted back into weibo.com's HTML during users' ordinary Weibo use.

GhostPost differs from similar projects[2, 3] in its use of distributed monitoring. In addition to the ability to scale to more accounts, by monitoring the accounts that its users follow, GhostPost's attention is focused exactly where it should be. A centralized monitoring approach must accept limitations; *e.g.*, Weiboscope [2] only monitors accounts with at least 1,000 followers. Fundamentally, GhostPost's differences from these projects are the result of differing goals. Its purpose is to restore deleted posts back into the feeds of users who were supposed to see them, which requires distributed monitoring (and selection of accounts to be monitored). The previous systems are intended for archival of censored posts and analysis of the censor's behavior, for which a centralized approach is sufficient.

2 GhostPost

We assume the censor is a branch of a large national government. Some level of day-to-day censorship might be carried out by the company that operates the platform, to avoid the government taking more direct control.

We assume that for a sufficiently popular GhostPost deployment, the censor will attempt to block its operation, and/or dissuade citizens from using it. In particular, the censor will attempt to identify and punish GhostPost users. We assume the censor can see every post posted on Weibo, and, as will become important in our evaluation of GhostPost's safety, can track which Weibo users ever *observed* a given post. The censor can join GhostPost, and see which posts it resurrects.

GhostPost's purpose is to resurrect social media posts deleted by a censor. This goal of resurrecting deleted posts requires GhostPost to be able to carry out two fundamental tasks: 1) *observing posts being deleted*, and 2) *disseminating stored resurrected posts*. If consideration is not given to how deleted posts will be collected and disseminated, users could claim to have seen any account post any message. Therefore, 3) *users must be able to verify the source of a resurrected post*. Finally, these goals must be balanced against safety: a censor might seek to punish GhostPost users, in order to deter use of the system. If all resurrected posts are visible to all users, a censor could obtain information indicating that some Weibo users were more probably GhostPost users than others. Therefore, 4) *users must be able to control who receives the posts they resurrect*.

2.1 Design

1) Observe posts being deleted: Sina Weibo is a huge social media platform, with hundreds of millions of users posting on the order of 70,000 posts per minute[9]. Any of these posts could be deleted at any time. We do not want to miss any deleted posts, but Weibo's scale makes it inevitable. A more reasonable goal is to preserve just the posts that GhostPost users are likely to be interested in, as indicated by the lists of Weibo accounts that our GhostPost users follow. Having GhostPost users themselves watch for deleted posts is then a natural choice.

2) Disseminate resurrected posts: After resurrecting a deleted post, the GhostPost system must be able to distribute it to users. All users learn of resurrected posts from the central server. Clients' interactions with the server take place via RESTful HTTPS. These queries do not require anything approaching high performance: with only metadata and handfuls of censored text transferred, they are not bandwidth intensive, and because the client queries do not block the weibo.com page load, they are not latency sensitive. Therefore, practically any means of communication, including low-performance covert channels, are adequate.

3) Verify resurrected posts: Resurrected posts are claims that someone posted something that is no longer visible. Users can hopefully apply common sense to fight misinformation, but the best way to trust a resurrected post is to know that it comes from a trusted friend. The central server knows which users reported which resurrected posts, and can provide that information to a retrieving user (provided the sources are willing to let that user know they use GhostPost). Adding a web of trust approach, to expand post verification beyond immediate friends, might be a good direction for future work.

To guard against a malicious central server, users sign posts they resurrect. The GhostPost extension generates a key pair. Users can retrieve their friends' public keys from the server for convenience, but are encouraged to verify them separately.

4) Limit recipients of resurrected posts: As we discuss in §5, a censor tracking which accounts viewed which posts, and which posts GhostPost resurrected, can gain information about how likely various Weibo users are to be GhostPost users. A user *A* who is worried about this possibility can protect themselves by choosing to allow posts they resurrect to be viewed only by users they trust. If *A* makes this choice, a user *B* that *A* does not trust will not receive a post resurrected by *A* until some other user—one who either trusts *B*, or has not enacted this restriction—has also resurrected that post.

3 Implementation

GhostPost is implemented on the client side as a Google Chrome extension, and on the server side as a straightforward REST service. The implementation must not reveal to an observer (such as weibo.com, or the user's ISP) that the user runs GhostPost.

3.1 Usability

Because the Chrome Web Store is blocked in China, we must distribute the extension ourselves. This means that rather than clicking an "install this extension" link, users must instead drag and drop the extension package from their file system onto Chrome's extension management page. Our download page includes detailed instructions about this process.

Once the extension has been installed, GhostPost requires the user to register in the system by proving ownership of their Weibo account. They are prompted to do so by a pop-up message box the first time they visit weibo.com with GhostPost installed. The proof-of-ownership process is simple: the user must correctly predict the account's next post.

After registration, GhostPost functions without any user interaction. Other than an icon in the extensions section of the Chrome browser, the user's browser is completely unchanged, including while visiting weibo.com—unless GhostPost has a resurrected post that belongs on the page that the user is currently viewing. Resurrected posts are printed in red, and include the time when they were detected deleted, as well as whether they have been verified by someone the user trusts.

With Weibo's 140 Unicode character limit, a user whose followed accounts posted 1,000 posts per day would require a (loose) upper bound of 273KiB per day, or 97.5MiB per year, to save them all. Ideally we should save posts forever, but users could forget posts older than a few weeks without doing much damage: recall that over 90% of posts are deleted within 24 hours.

Saving every image encountered on Weibo may be too much of a storage burden for many users; to be safe, our initial implementation does not store images. However, while ugly, extreme JPEG compression can reduce typical photographs to 10-20KiB, while keeping important features basically discernible. 75% of deleted posts have pictures attached (or are reblogs of posts with pictures)[9]; the censor clearly cares about the content of images. Images might therefore be a good future addition to the system.

3.2 Implementation Safety

The use of the GhostPost extension must be completely invisible to the censor, both in terms of its behavior when interacting with weibo.com, and in terms of Internet activity. Google Chrome extensions are built to be completely isolated from the web pages the browser loads, unless the extension explicitly designs in hooks for the page to attach to. Therefore, GhostPost's presence is invisible to weibo.com. On the other hand, communication with the central server would be easily detectable. However, beyond detectability, we assume the central server will be blocked, and so GhostPost users must employ some form of traditional censorship circumvention to begin with.

Because the censor has control over the content of weibo.com, it could complete this registration process on behalf of any user. If we returned an error when someone tried to register an already registered account, the censor could easily test whether a given Weibo user was a GhostPost user. Therefore, we allow multiple registrations; each new registration simply adds to the list of credentials that should be accepted for that account. The fact that the censor can "take control" of someone's account in this manner is unfortunate, but not particularly damaging: all the censor could do would be to issue resurrected posts under the user's name, but with a different key from the user's real key, which the users' friends would detect.

4 Evaluation

GhostPost has two quantifiable aspects, which are somewhat in conflict: coverage and safety. We want to minimize the fraction of deleted posts that we fail to resurrect. At the same time, we must prevent the censor from finding GhostPost users. We now analyze GhostPost's performance in these two areas.

4.1 Coverage Metric

To evaluate of GhostPost's ability to catch deleted posts, we must decide on the quantity to be measured. We only care about posts authored by users followed by GhostPost users. For coverage analysis, we lose no information by pretending that Weibo consists only of the GhostPost user base, plus the Weibo users they follow.

Having identified the users and posts that we are interested in, the natural quantity to study would seem to be the quantity of posts deleted by the censor before GhostPost could witness them. However, this measure is not always particularly meaningful. Imagine a set of 1,000 deleted posts, among which 990 were posted by a user with a single follower, and 10 were posted by a user with 10,000 followers. If GhostPost managed to resurrect all of the 990 and none of the 10, this measure would tell us that GhostPost had 99% coverage. We must find a more meaningful measure.

Rather than considering all posts equally valuable, we will consider a post posted by a user with n followers to be worth n *postviews*. One postview represents a single user getting the opportunity to read a single deleted post that they are interested in. By this metric, we would say that in the previous example, 990 out of a possible 100,990 postviews were preserved, or about 0.98%.

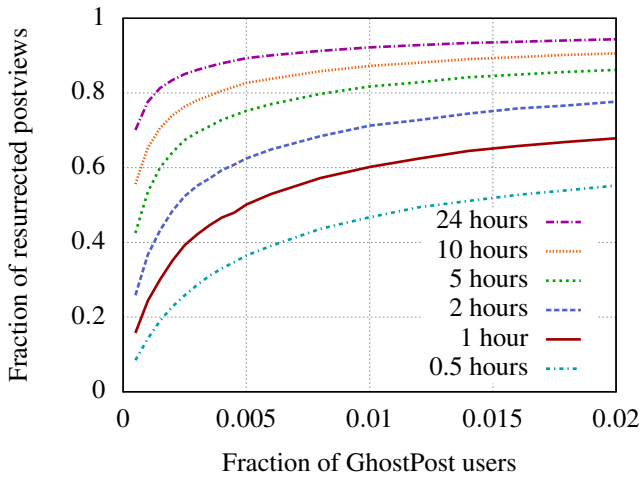

Figure 1: GhostPost's coverage against censors who manage to delete posts with various average speeds.

4.2 Coverage Simulation

We evaluated GhostPost's coverage using an event-based simulation framework[5] with a scale-free topology of 1,000,000 users. The events are 1) a user posts a new post, 2) a GhostPost user saves all visible posts from a random followed account, and 3) censor deletes a post. The simulation ends when all posts have been deleted by the censor[1]. We measure the fraction of postviews restored out of the total postviews in the system.

Three factors affect GhostPost's coverage: the time it takes the censor to delete a post, the fraction of Weibo users who use GhostPost, and the interval at which GhostPost users save an account's current posts. Our implementation only has control over the last of those three. We have fixed this frequency at an average of 5 per hour, as a compromise between frequent coverage, and avoiding making the user's activity look suspicious.

A study[9] of the time it takes for Weibo posts to be deleted found that post deletion times are not very evenly distributed. First, the distribution of deletion times is long tailed; while they observed the majority of posts being deleted within 24 hours, a non-negligible fraction lived for days or weeks. More interestingly, grouping post lifetimes by the time of day they were posted at (in 24 1-hour bins) yields two clusters. Namely, fated-for-deletion posts that are posted during the hours between 3 and 9 A.M. have median lifetimes of 8 or 9 hours, while posts posted from 10 A.M. until midnight have medians around 2 hours. The study's authors hypothesize that the censor's post-examining workforce is greatly reduced late at night, causing a backlog to build up, which is not cleared until late morning.

Because identifiable classes of posts are being monitored by censors of different capacity, we evaluated GhostPost against different censors: posts have an expected lifetime of $\{0.5, 1, 2, 5, 10, 24\}$ hours. The 2 and 10 hour censors roughly correspond to the median performance of the real world daytime and nighttime censors. The 24 hour censor is an upper bound for over 90% of real world posts[9].

Because the censor is a group of employees processing a queue of posts, we model deletion time with the exponential distribution. The exponential distribution lacks real post deletion times' heavy tail, but this makes the exponential distribution a conservative choice: our results indicate that even the tiniest GhostPost deployment is extremely likely to resurrect posts that live multiple days.

When simulating GhostPost, we varied the fraction of GhostPost users in the system from 0.05% to 2%. The results of these simulations are plotted in figure 1. Encouragingly, GhostPost reaches nearly its full potential very quickly; steep gains are made until about 0.5% of Weibo users are using GhostPost.

Whether a post can be resurrected is ultimately a race between the censor's agents and the poster's GhostPost followers: whichever side examines the post first wins. For a given author, GhostPost can run this race faster by either having more of its users following that author, or by having its users checking current Weibo posts more frequently. The censor can reach posts faster by hiring more employees to examine posts. However, even a highly aggressive and capable censor, one that deletes posts on average within 30 minutes after they are posted, fails to hide the majority of postviews from a GhostPost system with around 1.5% Weibo users. Sina Weibo's nighttime censor would allow a 1% GhostPost deployment to resurrect nearly 90% of postviews; the daytime censorship would allow over 70%.

4.3 Per-user Coverage

The previous analysis shows how likely a deleted post of interest to a GhostPost user is to be saved. In addition to that evaluation of GhostPost's ability to serve post *consumers*, we should also consider how well GhostPost serves post *producers*. A user with more followers is more likely to have more GhostPost followers, providing better coverage. We therefore examined the distribution of preserved posts for users with varying follower counts, shown in figure 2.

With 1% GhostPost users and a 10-hour censor, nearly 90% of all postviews are resurrected (see figure 1). Most of those resurrected postviews come from users with huge amounts of followers. Users with typical numbers

[1]Posts that are never deleted do not need to be restored, and so are irrelevant to GhostPost's degree of coverage. For purposes of evaluating coverage, we can focus only on posts that are fated to be deleted.

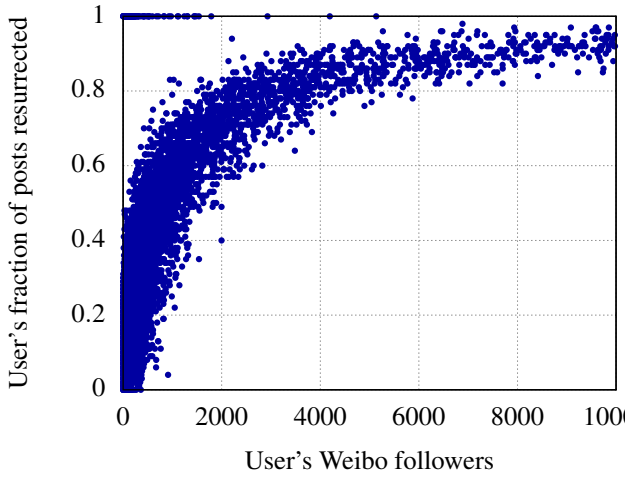

Figure 2: Each point is a user. The censor deletes posts after an average of 10 hours, and 1% of Weibo users are GhostPost users.

of followers (low hundreds) have over half of their censored posts missed by GhostPost.

However, this imbalance only applies to authors who do not care about it. Any Weibo user who wants to ensure that their posts will be preserved in GhostPost can simply become a GhostPost user, and preserve their own posts. This option is reflected in figure 2: points with resurrected fraction 1 are GhostPost users.

Users whose posts are deleted unusually quickly are also less covered. Users who have already had many posts deleted have lower median post lifespans than users who have only had a few deleted. The median post lifetime was about twice as long for users with fewer than 10 deleted posts than for users with 100 or more[9]. In this case as well, users who want their posts to reach the GhostPost audience should become GhostPost users.

5 Safety

We must hide GhostPost users' identities from the censor. GhostPost is not directly visible to weibo.com or the user's ISP, but observing which posts are (not) resurrected might yield information about who uses GhostPost. For this analysis, we assume that all GhostPost users allow all other users to see the posts they resurrect. The following vulnerability is eliminated if GhostPost users restrict their resurrected posts to their friends.

Imagine a Weibo account A with only two followers, F_1 and F_2, both of whom are also GhostPost users. F_1 is a legitimate user, but F_2 is an agent of the censor. If the censor deletes one of A's posts, and then F_2 receives a resurrected copy of that post, it can be sure that F_1 is a GhostPost user. Realistically, the censor will not immediately pinpoint a GhostPost user with a single post in this manner; real, active Weibo users have more than one follower.

The censor must monitor two classes of information: whether a deleted post was resurrected, and which users saw a post before its deletion. Let $\mathscr{D}$ be the set of all posts that the censor has deleted, and $\mathscr{D}_R \subset \mathscr{D}$ be the subset of resurrected posts. For each $d \in \mathscr{D}$, we will define $O(d)$ to be the set of users who observed d, *i.e.*, all those users who loaded a Weibo page including d before it was deleted. Although the censor is only interested in $O(d)$ for $d \in \mathscr{D}$, it must track $O(p)$ for the entire lifetime of all posts: it does not know whether $p \in \mathscr{D}$ until it has decided to delete p.

To maintain $\mathscr{D}_R$, for every post $d \in \mathscr{D}$ the censor deletes, it must determine whether GhostPost can resurrect d. This requires the censor to monitor GhostPost from within: the censor must make a Weibo account, follow all accounts whose posts will be deleted, and register with GhostPost. Although a single GhostPost user claiming to follow every account would look suspicious, the censor could easily spread across multiple accounts.

Based its on observations ($\mathscr{D}$ and $\mathscr{D}_R$), the censor tracks the likelihood that each Weibo user is a GhostPost user: $\Pr[GP_u]$. The censor does not know which users are GhostPost users: that is what it is trying to learn. Therefore, when a post r is resurrected, the censor must assume that each user $u \in O(r)$ is equally likely to be responsible for resurrecting the post. Then for every new post r to be added to $\mathscr{D}_R$, for each user $u \in O(r)$

$$\Pr\left[GP_u \mid \mathscr{D}_R \bigcup \{r\}\right] = 1 - \frac{|O(r)|-1}{|O(r)|}(1 - \Pr[GP_u \mid \mathscr{D}_R])$$

Additionally, for every deleted post $d \in \mathscr{D} \setminus \mathscr{D}_R$, *i.e.* not resurrected, every user $u \in O(d)$ has

$$\Pr\left[GP_u | \mathscr{D} \bigcup \{d\}\right] = 0$$

As posts are deleted and not resurrected, the censor will establish with certainty that more and more users are not GhostPost users (at the times of the deletions, at least). A non-GhostPost user remains under suspicion as long as every deleted post they observe is also observed by a GhostPost user. In other words, if a user has nothing to offer GhostPost, it is impossible to conclude from resurrected posts that they do not use GhostPost. Because almost all of GhostPost's coverage comes from the first 1 or 2% of its users, more widespread GhostPost use would allow the censor to rule out fewer non-GhostPost users. Furthermore, if some GhostPost users (perhaps those with a history of attention from the Weibo censor) are restricting their resurrections to trusted friends only, then "ruled out" users will include false negatives.

5.1 Post Alteration

Despite the previous point, a censor could still rule out many users in a smaller GhostPost deployment. Another countermeasure is possible. The censor must track the users who see each Weibo post ($O(p)$), and then observe whether GhostPost resurrects p once it has been deleted ($\mathscr{D}_R$). However, if the censor cannot reliably track $\mathscr{D}_R$, then it cannot be sure that $O(p)$ are not GhostPost users for $p \notin \mathscr{D}_R$.

The challenge, then, is to distribute to our users some version of the post that the censor cannot easily recognize. Weibo users evade keyword bans by using homophones, puns, characters whose components include the intended characters, and similar linguistic tricks. These tricks interfere with automated processes, while preserving the original meaning for an intelligent human reader. In this way, a user concerned about being identified as having observed a resurrected post can more safely share their resurrected posts with users outside of their friends, benefitting the system's coverage. We should also add a random offset to the post's time, and remove the author. We could prompt our users to make these modifications. There is also an automated technique[4] for generating usable Chinese homophone substitutions.

Although the homophone substitution technique would interfere with automated processes, we should also consider a censor that is investigating whether a few specific Weibo users are GhostPost users. When targeting user A, who follows some other user B, such a censor could have weibo.com display (only to A) a fake post "written by" B. Then, shortly after the next time A loads B's wall, the censor could query the GhostPost system for any new resurrected posts authored by B. A is a GhostPost user if and only if the fake post is included in the response. If a human is manually carrying out this attack, then the post alteration technique would not help much. As with other attacks, sharing resurrected posts with trusted friends only would keep the user safe.

6 Conclusion

GhostPost preserves deleted Sina Weibo posts and presents them in their original context: displayed in between other posts on weibo.com, where they would be had they never been deleted. The fundamental technique would also be applicable to any similar site, *e.g.*, Twitter. Our implementation of the GhostPost Chrome extension is available at `https://salmon.cs.illinois.edu/ghostpost.html`. Users monitor for deleted posts, allowing GhostPost's coverage of Weibo to scale with the number of GhostPost users, and focuses the monitoring exactly on the accounts users care about.

We simulated GhostPost against a censor modeled after observations of the real Weibo censor. GhostPost performs well even when only a tiny fraction of Weibo users use GhostPost: *e.g.*, with 0.5% of Weibo users, against the daytime censor, GhostPost users would be able to see nearly two thirds of the deleted posts of interest to them.

There is tension between coverage and safety. In smaller GhostPost deployments, the censor could eliminate many Weibo users from suspicion of using GhostPost. This attack can be mitigated if some users only allow their friends to see their resurrections, and can be completely shut down if all users do so. Automated alterations to deleted posts could also help, and would not hurt coverage. We therefore believe that Weibo users can be reasonably safe while using GhostPost.

References

[1] Daniel Anderson. Splinternet behind the Great Firewall of China. *Queue*, 10(11):40, 2012.

[2] King-wa Fu, Chung-hong Chan, and Marie Chau. Assessing censorship on microblogs in china: Discriminatory keyword analysis and the real-name registration policy. *Internet Computing, IEEE*, 17(3):42–50, 2013.

[3] Greatfire.org. https://en.greatfire.org/.

[4] C. Hiruncharoenvate, Z. Lin, and E. Gilbert. Algorithmically bypassing censorship on Sina Weibo with nondeterministic homophone substitutions. In *9th Intl. AAAI Conf. on Web and Social Media*, 2015.

[5] P. L'Ecuyer, L. Meliani, and J. Vaucher. SSJ: A framework for stochastic simulation in Java. In *Proceedings of the 2002 Winter Simulation Conference*, pages 234–242. IEEE Press, 2002. http://simul.iro.umontreal.ca/ssj/indexe.html.

[6] Neo, Sparks, Tank, Dozer, and Smith. The collateral damage of internet censorship by DNS injection. In *SIGCOMM Computer Communication Review (CCR July 2012)*, pages 21–27. ACM, 2012.

[7] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *10th annual WPES*, pages 103–114. ACM, 2011.

[8] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. *Free and Open Communications on the Internet*, 2012.

[9] Tao Zhu, David Phipps, Adam Pridgen, Jedidiah R Crandall, and Dan S Wallach. The velocity of censorship: High-fidelity detection of microblog post deletions. *arXiv preprint arXiv:1303.0597*, 2013.