Noname manuscript No.
(will be inserted by the editor)

twister: the development of a peer-to-peer microblogging platform

Miguel Freitas

Received: date / Accepted: date

Abstract This paper proposes a new microblogging architecture based on peer-to-peer networks overlays. The proposed platform is comprised of three mostly independent overlay networks. The first provides distributed user registration and authentication and is based on the Bitcoin protocol. The second one is a Distributed Hash Table (DHT) overlay network providing key/value storage for user resources and tracker location for the third network. The last network is a collection of possibly disjoint "swarms" of followers, based on the BitTorrent protocol, which can be used for efficient near-instant notification delivery to many users. By leveraging existing and proven technologies, twister provides a new microblogging platform offering security, scalability and privacy features. A built-in mechanism provides incentive for entities that contribute processing time to run the user registration network, rewarding such entities with the privilege of sending a single unsolicited ("promoted") message to the entire network. The number of unsolicited messages per day is defined in order to not upset users.

Keywords peer-to-peer · microblogging

1 Introduction

Microblogging platforms are one of the most versatile and empowering technologies on the internet today. Recent events have shown the important role of these tools for news coverage [23] and also for political movements, like in Middle East's "Arab Spring". Although their role in social revolutions should not be overstated [13], it is telling to learn how dictatorships frequently resort

Miguel Freitas
Centro de Pesquisa em Tecnologia de Inspeção
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, RJ, 22453-900, Brazil.
E-mail: miguel@cpti.cetuc.puc-rio.br

to shutting down the internet in trying to control such potentially destabilizing movements [10, 28]. Blocking internet access, however, is never fully effective against social movements, as some people always find ways to circumvent such blockage [4].

The possibility that the service providers themselves could be convinced to participate of a social media blockage [12] would affect people's ability to communicate in a much more dramatic way than just disrupting a few network backbones. As our society's dependence on these services increases, the single point of failure on such basic communication platforms (at the provider's own discretion) is not only unacceptable but also directly opposes the Internet's key design feature of providing redundancy for information transmission [32].

Reports of widescale internet wiretapping with the cooperation of large corporations [11] reveal the danger the present platforms pose to user's privacy. The fact that a single entity is able to access private communication and personal data at their will should raise serious concerns. A recent House of Lords (UK) report openly recognizes the dangers of this massive surveillance[1].

All these facts point to an obvious direction: there is an urgent need for open, secure and distributed personal communication platforms. This is where the present peer-to-peer (P2P) microblogging proposal fits in.

Of course, to be sucessful, such P2P microblogging cannot just provide resilience and security, but it must also be user-friendly. This is a key point to the adoption of any new software or web service. Some current P2P message proposals offer good examples of what not to do in terms of user-friendliness, like requiring the user to know a cryptic address composed of 36 case sensitive characters [29].

The ability to provide easy to remember usernames must be considered a fundamental requirement. While users must be free to choose their usernames, thus providing anonymity to whoever needs to express himself freely without fear of retaliation, it is important to realize that a web of trust is built on these microblogging infrastructure based upon real existing and fully identifiable people. This issue can be appreciated in Hudson plane crash coverage [23] where trusted aggregators helped separating the reliable information from noise. These people tend to function as hubs for information distribution and are often defined as "influential". Any serious P2P microblogging proposal must foster this kind of organization.

This paper presents a proposal of a new P2P microblogging platform that is scalable, resilient to failures and attacks, does not depend on any central authority for user registration, and provides both easy-to-use encrypted private communication and public posts. The architecture tries to leverage existing and proven P2P technologies as much possible. Privacy is one of the primary design concerns, providing protection of the user's online presence and IP address as well as his reading habits.

[1] "Mass surveillance has the potential to erode privacy. As privacy is an essential prerequisite to the exercise of individual freedom, its erosion weakens the constitutional foundations on which democracy and good governance have traditionally been based in this country." [15]

2 Decentralized microblogging issues

A completely decentralized microblogging proposal must address several technical difficulties that are not applicable to a standard centralized approach.

The first challenge is to obtain a globally unique username mapping that everybody will agree upon. In a conventional microblogging platform a single centralized database trivially does the job, so this is effectively a non-issue. In a fully decentralized system, however, there is no authority hierarchy so any peer may possibly hold a different database. The agreement between peers must therefore be algorithmically resolved and embeded into the P2P protocol. twister's proposal to this issue leverages Bitcoin's distributed ledger solution [18], adapting the mechanism to a non-financial application. This is further described in section 6.

The centralized microblogging plataform's main activity is to operate as a store and forward facility for user's posts. Because every user authenticates with the server when they log in, their posts are guaranted by the server to be authentic. Forgery produced by directly modifying the server's database is often not considered a threat in this model. Another crime, the unlawful wiretapping of personal comunications, is also not perceived as a threat by most people even though there are known cases of provider's employees who have been caught spying into user's chats [1].

In contrast, the P2P microblogging platform has to allow every peer to operate as store and forward node. Unlike the centralized approach, these relaying facilities must be assumed individually unreliable or even malicious. There are two main aspects to be considered when communicating through untrustworthy peers: redundancy plus diversity of routes, and end-to-end authentication plus encryption. Peers trying to forge posts of other users must not succeed because all twister's posts are signed by the sender. Private messages must be encrypted so intermediate peers won't be able to inspect their contents or metadata. Besides these basic premises, twister does not require any single specific cryptographic algorithm to be used, so generic asymmetric cryptography operations are described in sections 6-8.

Section 4 explores some additional network implications of the untrustworthy peers model. General attacks against P2P networks [27] however (those which are not specific to the microblogging application are deliberately not discussed) except for privacy-related node addressing P2P issues in section 7.

Scalable storage and retrieval of arbitrary resources in P2P networks are provided by a structured network overlay organized as a Distributed Hash Table (DHT) [16]. Most of the DHT usage in a microblogging scenario is straightforward, like using predefined hash entries in order to retrieve user's profiles and avatars. The twister's DHT secure primitives are defined in section 7 while specific proposals which adapt the DHT infrastructure to implement common microblogging features, such as linking posts, thread navigation and hashtags, are discussed in section 8.

The microblogging decentralization based on P2P further requires that peers contribute some storage capacity back to the network to provide the

overall storage capacity. In order to be scalable and also robust against denial-of-service attacks, the amount of data to be stored by each peer has to be constrained. Appropriate limits have been implemented to address these concerns (see section 9) while providing an experience which is compatible with the known usage profile of the existing centralized platforms in terms of the content produced [25]. The scalability assumption is that the number of online peers (the ones that maintain the distributed total storage) preserves proportionality to the amount of data to be persisted. In other words, some proportion is assumed to exist between number of online peers and the total number of users and between the number of users and the total amount of posts to keep.

The last remaining issue of paramount importance to successfully deploy a decentralized microblogging is to ensure efficient and near-instant propagation of posts from producers to followers. The simplest DHT-based solution of periodically checking a specific hash entry for updates (where the last status is posted by each user) is neither scalable nor efficient. This is referred to as the "lame, repeated polling" issue by the developers of the PubSubHubbub protocol [7] because it would require the user to keep polling everybody he follows. twister's proposal to the near-instant notification issue is to leverage BitTorrent's protocol [2]. BitTorrent is an unstructured P2P overlay network for distributing files by sparing the producer (the "seeder" in BitTorrent nomenclature [31]) from using his bandwidth to serve all the interested peers (the "swarm"). twister implements small changes to the BitTorrent protocol so the shared data is replaced by a list of posts produced by a given user. New posts added to the swarm are automatically and efficiently distributed among the other peers. More details of this mechanism are presented in section 9.

To sum up, the the proposed platform is comprised of three mostly independent overlay networks, two unstructured and one structured. The first unstructured overlay network is based on the Bitcoin protocol and provides distributed user registration and authentication. The second is a structured DHT network, providing key/value storage for user resources and tracker location for the third network. The last, also unstructured, overlay network is a collection of possibly disjoint "swarms" of followers, based on the BitTorrent protocol, which is used to provide efficient near-instant notification delivery to many users.

3 Related work

Existing social networks like Diaspora [8], StatusNet [24] and identi.ca [21] are frequently cited as free, open and distributed alternatives to Facebook or Twitter. These platforms are based on the concept of "federated social networks" [30] where users may join the social websites of their choice and these sites communicate with each other using open protocols. While technically superior to a single, closed platform in terms of achieving better privacy control, the user still needs to delegate his own data to a third party (unless he

wants to setup his own server to federate). So, conceptually, twister and these federated plataforms are very different things.

Previous P2P microblogging proposals do exist however, like Cucko [33] and Megaphone [20]. Neither of these addresses the problem of the decentralized user registration. Privacy is also not one of Cucko's objectives since it is explicitly designed to know about the online presence of anyone. One similarity of twister and Cucko is that both share the idea of using an unstructured overlay network for dissemination of user posts, unlike Megaphone where all followers must register to the sender, forming a multicast tree for post propagation.

Another solution to the near-instant notification issue is the already mentioned PubSubHubbub protocol [7], which is also decentralized to a certain degree but still relies on servers ("hubs") to operate, so it is also conceptually closer to federation than P2P.

A more advanced social network proposal Safebook [3] addresses several privacy issues by implementing different levels ("shells") of access to the published data. While Safebook's scope goes far beyond twister's, it still relies on a centralized "Trusted Identification Service" for user registration.

At the present time, except for twister, no public implementation seems to be available to any of these other P2P proposals. In contrast, twister network [9] has been operating since December 31 2013 with tens of thousands of registered usernames to date.

4 Threat model

The basic threat model for twister is that any other peer is assumed to be individually untrustworthy. This untrustworthy peer may try to deceive users by providing forged posts from other genuine users or refuse to store and forward some posts. In order to prevent these two scenarios, the twister client must (1) always check if each post is properly signed by the sender and (2) propagate the new posts to multiple random clients.

Requirement #1 demands security from the underlying cryptography to ensure the authenticity of posts and the inviolability of private messages.

Requirement #2 further assumes that the twister client is able to connect to other truly random peers who would not collude to undermine the privacy of the user's IP address (online presence), or even attack and isolate the user into a separate network. The current threat model, therefore, clearly assumes the attacker is not a government controlling all the network links but rather another common user with limited resources. While being a clear limitation of twister, this is a deliberate choice as to focus on solving the other aspects of the decentralized microblogging technology. twister's proposal leaves the problem of truly secure anonymizing technology implementation to the experts of the field, namely the Tor Project [22].

As of the date of writing, twister's current implementation does not fully support operating on top of a Tor proxy. The two unstructured networks are

already Tor-aware, but DHT overlay network uses UDP which is not supported by Tor. In the near future, twister will implement routing DHT traffic through other peers in order to be fully compatible with Tor, thus allowing a far more interesting threat model where the adversary could be a censoring government.

5 Notation

Tuples (concatenation): $[a, b, c, ...]$
Apply function f to payload x: $y = f(x)$
One-way hash function: $H(x)$
Username of user j: U_j
Asymmetric key pair of user j: $PUBK_j; PRIVK_j$
Encryption/decryption operations: $PUBK_j(PRIVK_j(x)) = x$
and $PRIVK_j(PUBK_j(y)) = y$
Signed content x from user j: $SIG_j(x) = [PRIVK_j(H(x)), x]$

6 User registration P2P network

Decentralized yet secure user registration is achieved by means of the Block Chain mechanism, which is used in Bitcoin [18] to avoid the double-spending problem without the need for a central authority. In the proposed system the mechanism is used to guarantee the uniqueness of users, again with no need for a central authority. New registrations must be "notarized" by a number of Blocks before they can be considered granted to a given user. Each Block is defined as:

$Block_i = [i, H(Block_{i-1}), nonce_i, SpamMsg_i, [UserReg_j, UserReg_{j+1}, ...]]$

where i is the block number, $H(Block_{i-1})$ is the hash of the previous block, $nonce_i$ is an arbitrary number wisely chosen, $SpamMsg_i$ is a promoted message and $UserReg$'s are the user registrations. The choice of $nonce_i$ will be made in such a way that $H(Block_i)$ presents a partial hash collision. More specifically, $H(Block_i)$ must start with a certain number of leading zeros so this quantity defines the difficulty of the brute force searching challenge. This is the Proof-Of-Work (POW) mechanism of Bitcoin. The number of leading zeros is automatically set by the network in order to maintain the average number of blocks per hour.

The Block Chain provides a public dictionary from U_j to $PUBK_j$ by collecting all user registrations. A new user j registering to the network must broadcast $UserReg_j$, which is defined as

$UserReg_j = [U_j, PUBK_j, nonce_j]$

where $nonce_j$ allows a POW to be set upon $H(UserReg_j)$. Other nodes, upon receiving $UserReg_j$, must check this POW before the request can be retransmited/accepted. It prevents DoS attacks due to flooding of bogus registrations and also counterfeiting $UserReg_j$ by replacing the $PUBK_j$ while keeping the same username. POW of $UserReg_j$ is much smaller than POW of

block chain, typically just a few minutes of an average computer time (difficulty may be hardcoded in software and changes only with protocol versions).

Nodes must enforce the uniqueness of U_j before including $UserReg_j$ into a new Block. The only exception to this rule is the key replacement case, where the new public key is signed by the previously known key pair. The enforcement of uniqueness of U_j and POW of $UserReg_j$ is also applied when receiving new Blocks, since all registrations included therein must be checked. U_j is also subject to additional text rules, such as maximum size and allowed characters.

$SpamMsg_i$ is an unsolicited message (commonly and euphemistically called "promoted") that must be shown by all clients and provides incentive for joining the Block generation effort. If the same Bitcoin's block creation rate is maintained (6 per hour), a display probability factor may be implementated in order to not upset the users with too much spam. The current twister implementation will display one promoted post every eight hours (noncumulative). Developers should not implement hiding of spam messages as a "feature" of their clients since this incentive is important to the security of the entire network. Omitting unsolicited messages from clients would only hurt the users in the long run. The display probability also priorizes localization (by giving higher probability to messages of the same language of the user) to improve effectiveness and also user's experience.

7 Secure DHT network primitives

The second P2P network is a structured Distributed Hash Table (DHT) overlay network based on Kademlia [17]. The single most important feature of this network is to allow arbitrary resource storage and retrieval by users, including profiles, avatars and posts. Mapping of resources into specific peers is based on an one-way hash function, which provides deterministic location of resources while evenly distributing the content over the network. Special DHT entries are also used for BitTorrent tracker purposes as discussed in section 9. Direct delivery of notification between users can be thought of as a secondary usage for DHT (see section 8.3).

Peers joining the DHT network must do so by providing an address ID which then becomes a possible destination for other peers' requests. It would be tempting to use $H(U_j)$ directly as the ID of the peer joining the DHT network, as it would permit simple challenge-response authentication, possibly preventing ID forgery. Forged ID addresses is arguably the most serious security issue on P2P and DHT networks (see Sybil and Eclipse attacks [27, 34]). Using $H(U_j)$ for DHT addressing, however, would greatly compromise privacy since it is a fundamental characteristic of such a network to know the IDs' of the other peers in order to create optimized routing tables. This would not just permit easy detection of online user presence, but also immediately reveals the user's IP address to a potential attacker.

Therefore, instead of $H(U_j)$ as ID, twister follows the standard procedure of hashing IP address and port number to obtain the ID_{node_j} used to join the DHT network:

$ID_{node\_j} = H([IP_j, port])$

In [5] it is shown that a secure mapping of external IPs to ID is Sybil-proof when limited per participant.

Packets on this DHT network sent from source ID_s to ID_d are defined as:

$Packet_{s \to d} = [ID_d, ID_s, SIG_j(Payload), U_j)]$

The payload is signed by a given user U_j which is possibly unrelated to the sender's ID_s. In other words, a signed payload may be retransmited by another node who did not sign it. ID_d and ID_{node} both belong to the same address space, thus allowing for some distance metric to be used to determine the proximity between resources and nodes. These characteristics comprise the basic "layer 3" functionality offered by this overlay network.

Going up in the conceptual model for the proposed DHT overlay network there is an "application layer" where a data storage primitive (PUT) is defined with the following payload:

$Payload_{PUT} = [target, value, time, seq]$ where

$target = [owner, resource, restype]$ and $ID_d = H(target)$

Some simple rules must be checked by the destination node in order to accept the storage request:

1. $ID_d = H(target)$: ensures the destination address was properly computed.
2. ID_d is neighbor of ID_{node} that actually received this request. [2]
3. $U_j = owner$, only enforced for $restype =$"single".
4. seq is greater than previously stored seq_{old}, only enforced for $restype =$ "single".
5. $time$ is a valid time (ie, not in future).

The two possible $restype$ values are "single" and "multi". These two types provide, respectivelly, resources which may only be updated by the owner of this key (like an avatar image) and resources which collect multiple responses from different users (like replies to a certain post). In case of the "single" type, the node stores just a single $value$ associated with this key ID_d. For "multi", however, new PUT requests are appended to a list of $value$'s. This kind of storage provides no guarantees, as old values may be discarded following some expiration policy (based on the $time$ field) or Least Recently Used (LRU) cache strategy. Authenticated ("single") storage takes precedence over any previously "multi" value.

A data retrieval primitive (GET) may operate on both types of resources indistinctively. Some special non-storage resources associated with dynamic content may also be implemented using the same primitives, thus sharing the same API.

[2] Kademlia [17] defines a XOR metric to measure the distance between two IDs. In this paper, a "neighbor" is defined as a node which ID distance to the target is ranked within the smallest ones among all the online nodes.

8 Microblogging data structures and mechanisms

8.1 User posts

The k-th message of user j is defined as:

$Post_{jk} = SIG_j([U_j, k, type, MSG_k, REPLY_k])$

where MSG_k is the content (140 characters limited), k is a monotonic increasing number and *type* may define if it is a new post, a reply, retransmission (RT) or Direct Message (DM). $REPLY_k$ is an optional field which provides a reference pointer to the original message, in case of a reply/RT (see section 8.4) and is defined as tuple $REPLY_k = [U_{j'}, k']$, where original post is the k'-th message of the user j'.

The posts are distributed simultaneously in two overlay networks: (1) as a stored value, possibly short lived, in DHT network and (2) in a file-like archive pertaining to a modified BitTorrent swarm. Whenever a new post is created, the client must send two PUT requests to the following addresses:

$ID_{Post\_jk} = H([U_j, \text{“post”} + k, \text{“single”}])$ and

$ID_{swarm\_j} = H([U_j, \text{“swarm”}, \text{“single”}])$.

The ID_{Post_jk} is the address of a storage target defined in section 7 and provides arbitrary post retrieval capabilities.

The ID_{swarm_j} is a special gateway address to reach a BitTorrent swarm (section 9), which contains all posts from a given user j and helps propagate them independently of the DHT network. The neighbors of ID_{swarm_j} are required to join this swarm, as much as the neighbors of ID_{Post_jk} are required to store the value.

8.2 Direct Messages

User posts may also be used to send a Direct Message (DM) from user U_j to user U_l, provided that recipient U_l is a follower of U_j (same requirement as Twitter).

$DM(j \rightarrow l)_k = Post_{jk} = SIG_j([\text{“”}, k, \text{“dm”}, [PUBK_l(M_k), H(M_k)]])$

One should note that DM is equivalent to a normal post except that

$MSG_k = [PUBK_l(M_k), H(M_k)]$, as defined in section 8.1 for public posts.

DM is only received by destination user U_l by checking for sucessful decryption of M_k. No other user will know for which recipient the DM was sent to (metadata monitoring), although the encrypted message will be seen by all of his followers and more.

This naive description of DM encryption mechanism is only meant to explain the concept as the actual implementation may differ. Currently, the working twister prototype is based on a public ECIS (Elliptic Curve Integrated Encryption Scheme) implementation by Ladar Levison [14], formerly the owner of Lavabit encrypted email service, following the SECG SEC1 standard [6].

8.3 Mentioning

If a post mentions the user U_j by using the syntax "@username", the client must also send a notification to $ID_{mention\_j}$, by including the full message. Notification is routed to a "multi" resource

$ID_{mention\_j} = H([U_j, \text{"mention"}, \text{"multi"}])$

which is set to receive and accumulate mentions and is kept by nodes which are neighbors of $ID_{mention\_j}$ as usual.

There are two possible issues here. The first one is again the "lame, repeated polling" as the mentioned user would need to periodically poll such key (although in a much more limited scale than a hashtag, for example). The second one is the intrinsic lack of a delivery guarantee for "multi" resources.

A way to prevent polling of user mentions to achieve faster delivery while preserving some degree of privacy is to declare $ID_{mention\_j}$ as a special resource, with more storage capacity, and enlisting their neighbors as "listeners" for this user's mentions. The listener's job is to immediately forward the mentions to the final user whenever the user is online. The idea is partially based in SASON [26], although not as secure since an additional anonymizing network is not used.

The system would work like this: the recipient U_j first uses the DHT network to find nodes near $ID_{mention_j}$. The user then asks them directly to forward new mentions to ID_{node_j}, therefore revealing his real IP address to a small group of listeners. Listeners may perform a challenge validation to make sure the user is really U_j by asking for SIG_j(random number). Since the other node has access to the full directory of public keys, he can easily authenticate.

Mentioning, like other mechanisms described here, requires the cooperation of the client software in order to work. If a given user does not send the notification packet to $ID_{mention\_j}$ (along with his own post) the mentioned user would never know.

The current twister software prototype does not yet fully support this proposal since the listener forwarding mechanism is not implemented. The twister client is currently required to periodically check DHT resources for updates, thus causing a few seconds of lag in mentions delivery.

8.4 Explict message request

Any user may explicitly request a certain message from user U_j without joining the swarm. This is achieved by a simple value retrieval from address ID_{Post_jk}.

Because posts may include a reference to the original post they are replying to (the $REPLY_k$ field), this feature allows for "upward message thread navigation" like in Twitter and is not resource intensive.

8.5 Downward message thread navigation

Downward navigation (finding out about replies/retransmissions of a certain post) might be a difficult problem since there are many, possibly unlimited, answers to the question "what are the replies to this particular post?".

One possible solution is to send another notification to the special address of multi-value list storage:

$ID_{replies\_jk} = H([U_j, \text{“replies”} + k, \text{“multi”}])$

The values to store are copies of the replies themselves. Again, it is the cooperation of client posting the reply which allows this mechanism to work.

8.6 Hashtags

Just like mentioning, hashtags must be detected in the content of any new messages being posted to the network. A copy of the message is then sent to a special address of multi-value list storage:

$ID_{hashtag\_t} = H([hashtag\_t, \text{“hashtag”}, \text{“multi”}])$

This is pretty much the same mechanism as downward message thread navigation except for an additional feature: a hashtag may create a new unstructured P2P swarm to broadcast new posts among their members. This mechanism is meant to ensure scalability to high-traffic hashtags, but it is not yet implemented in twister.

8.7 Word search

Searching for arbitrary words may be achieved by extending the hashtag implementation concept to all words in every post. In order to reduce overhead and network traffic, certain limits may be imposed, like a minimum word size, exclude prepositions and so forth.

$ID_{word\_w} = H([word\_w, \text{“word”}, \text{“multi”}])$

The word search proposal is not yet implemented in twister.

9 A BitTorrent swarm of posts

The swarm mechanism for distributing new posts is meant to address the problem of efficient notification of new posts, sparing the followers the need to do polling on a certain address of the DHT network to check for updates. The swarm is a modified BitTorrent P2P unstructured overlay network. In the original BitTorrent, the shared data is divided into pieces of the same size which have their hashes written to the "torrent" file. Every peer must download the torrent file beforehand in order to check the integrity of the received pieces. twister's swarm, in contrast, maps each post into a piece by using the same monotonic increasing number k defined in section 8.1. Because

every post is user-signed, their integrity may be easily verified without external hashes, rendering the torrent file mostly unnecessary.

The other function of the torrent file in original BitTorrent is to serve a list of IPs of the current members of the swarm, allowing a new peer to join in by connecting to them. Keeping this members list up-to-date is what the centralized "tracker" server does and is the motivation that led to the DHT usage in BitTorrent in the first place, that is, to decentralize the online peers list.

twister's tracker is a special DHT resource addressed by
$ID_{tracker\_j} = H([U_j, \text{"tracker"}, \text{"multi"}])$

Instead of just collecting the IPs from other peers which want to inform they joined the swarm as BitTorrent does, the ID_j node that is neighbor of $ID_{tracker\_j}$ is also required to join the swarm himself. The reason for this new approach is twofold. First, it increases the difficulty of a poisoning attack, where fake peers would announce themselves to the peer list by just sending and replying to a few UDP packets. With this new mechanism, instead, a more complete BitTorrent client is required to actually join the swarm to be announced and, after joining the swarm, additional peers are obtained by exchanging data with other members. Secondly, it ensures a minimum set of swarm members that will persist the post's data even if the user has no followers at all.

At this point it should be clear that the bootstrap sequence for a user U_l to start following the user U_j, is to first send a DHT GET request to acquire an initial list of peers from $ID_{tracker\_j}$. Even the producer U_j himself will be required to join his swarm the same way.

The neighbors of $ID_{tracker\_j}$ may not be aware of this vicinity until they receive a DHT GET request for the special "tracker" address. Then, they should perform additional checks to prevent wastefully joining swarms they are not tracker of. Current twister implementation also rotates all the automatically joined swarms in an active list of limited size, therefore keeping network and computational resource usage limited. The auto-rotation of torrent swarms is one of the several distinguished twister underlying features that leaverages the excellent opensource libtorrent project [19].

Any swarm member may announce the availability of a new piece, that is, a new post. This is also implemented by means of small changes to the BitTorrent protocol which then takes care of the rest (ie. propagating this piece among all peers). The fact that nothing in this scheme requires the peer announcing the new piece to be actually the producer of this post may be explored to achieve better privacy protection. By creating an additional DHT address ID_{swarm_j}, a simple DHT-to-swarm gateway can be implemented. The producer using the gateway mechanism is not required to join his own swarm in order to add new pieces, he just needs to send the post to some of the neighbors of ID_{swarm_j}. The gateway mechanism is not yet implemented in twister software.

Because swarm members only know each other by their IP address, it is not possible to construct a list of the followers' usernames. The act of following

an user is unilateral, the followed user doesn't have to authorize and is not notified. A consequence of the protocol described up to this point is that the "following list" is private (the list is only known by its owner).

However, accessing the following list is a widely used feature of microblogging systems because it helps increase the social connectivity of the network. In order to support this mechanism without losing privacy, twister introduces the concept of publicly following. It is up to the user, or its client software, to decide which usernames of the following list should be announced by using a set of public DHT entries, namely

$ID_{following\_jn} = H([U_j, \text{"following"} + n, \text{"single"}])$

These entries are lists of usernames, but stored in a resource of the type "single" since only the owner is allowed to update it. The requirement for several n entries is to overcome the maximum packet size limits of the UDP and underlying primitives.

In order to prevent the swarm members from being attacked with a huge number of pieces to store, the piece number k is constrained by

$k < 2 * (i_{current} - i_{UserReg_j}) + 20$

where $i_{current}$ is the last known Block number and $i_{UserReg_j}$ is the Block number the username was registered at. Any post violationing this rule will not be accepted. Considering that a new Block i is produced every 10 minutes, this limits the mean post rate of new users, for life, to a maximum of 288 posts/day. Average.

10 Conclusion

This paper has described the development of twister, a new peer-to-peer microblogging platform with security, scalability, usability and privacy features.

Several attack scenarios have been considered and the resulting platform is believed to be as resilient as some of the most sucessful P2P networks in use today, namely Bitcoin and BitTorrent.

Design decisions have been presented to support the scalability claims of the twister proposal. Given a small set of assumptions, the peer's computational and network resources utilization should not explode as the number of users and nodes increases. The number of users and nodes is also believed to not adversely affect the performance of the system.

Usability issues have been considered in twister's design. Supporting commonly used usernames, instead of long cryptographic hashes as seen in some other proposals, should make the system as user-friendly as the existing (centralized) microblogging platforms. The most common operations supported by these existing systems have been successfully replicated in a distributed architecture.

For privacy, the architecture makes the compromise to protect the user's online presence and IP address as much as possible by design without having to reinvent a new complete anonymizing network layer. The threat model for

that assumes an adversary which is not in possession of unlimited network resources or that is able to wiretap Internet links at a global scale. Users demanding further privacy should wait upcoming support in twister in order to use it on top of Tor [22].

The twister alpha software is available for download [9] and testing. The network already has tens of thousands of registered usernames. This architecture opens a wide range of future research topics, ranging from efficient distributed spam detection to the implementation of more advanced social features such as group communication.

References

1. Chen, A.: Gcreep: Google engineer stalked teens, spied on chats (updated). http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats/all (2010). [Online; accessed 1-October-2013]
2. Cohen, B.: The bittorrent protocol specification (2008)
3. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A privacy-preserving online social network leveraging on real-life trust. Communications Magazine, IEEE **47**(12), 94–101 (2009)
4. Dachis, A.: How to foil a nationwide internet shutdown. lifehacker http://lifehacker.com/5746046/how-to-foil-a-nationwide-internet-shutdown (2011). [Online; accessed 23-July-2013]
5. Dinger, J., Hartenstein, H.: Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In: Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on, pp. 8–pp. IEEE (2006)
6. for Efcient Cryptography Group, S.: Sec1: Elliptic curve cryptography, ver. 2. http://www.secg.org/download/aid-780/sec1-v2.pdf (2009). [Online; accessed 1-October-2013]
7. Fitzpatrick, B.: pubsubhubbub - a simple, open, webhook based pubsub protocol and open source reference implementation. http://code.google.com/p/pubsubhubbub/ (2013). [Online; accessed 24-July-2013]
8. Foundation, D.: Diaspora*. https://diasporafoundation.org/ (2013). [Online; accessed 1-October-2013]
9. Freitas, M.: twister download site. http://twister.net.co (2013). [Online; accessed 4-April-2013]
10. Glanz, J.: How mubarak shut down egypt's internet. The Age World http://www.theage.com.au/world/how-mubarak-shut-down-egypts-internet-20110216-1awjj.html (2011). [Online; accessed 23-July-2013]
11. Greenwald, G.: How microsoft handed the nsa access to encrypted messages. The Guardian http://www.guardian.co.uk/world/2013/jul/11/microsoft-nsa-collaboration-user-data (2013). [Online; accessed 23-July-2013]
12. Halliday, J.: Facebook and twitter to oppose calls for social media blocks during riots. The Guardian http://www.guardian.co.uk/media/2011/aug/24/uk-riots-facebook-twitter-blackberry (2011). [Online; accessed 23-July-2013]
13. Khondker, H.H.: Role of the new media in the arab spring. Globalizations **8**(5), 675–679 (2011)
14. Levison, L.: Code for using ecies to protect data (ecc + aes + sha). http://openssl.6102.n7.nabble.com/Code-for-using-ECIES-to-protect-data-ECC-AES-SHA-td39269.html (2010). [Online; accessed 1-October-2013]
15. of Lords (UK), H.: Surveillance: Citizens and the state, volume i. http://www.publications.parliament.uk/pa/ld200809/ldselect/ldconst/18/18.pdf (2009). [Online; accessed 29-July-2013]
16. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S., et al.: A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys and Tutorials **7**(1-4), 72–93 (2005)

17. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Peer-to-Peer Systems, pp. 53–65. Springer (2002)
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted **1**, 2012 (2008)
19. Norberg, A.: The opensource libtorrent library. `http://libtorrent.org/` (2013). [Online; accessed 1-October-2013]
20. Perfitt, T., Englert, B.: Megaphone: Fault tolerant, scalable, and trustworthy p2p microblogging. In: Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on, pp. 469–477. IEEE (2010)
21. Prodromou, E.: Identi.ca. `http://identi.ca` (2013). [Online; accessed 1-October-2013]
22. Project, T.T.: Tor (the onion router). `https://www.torproject.org` (2013). [Online; accessed 23-July-2013]
23. Sklar, R.: Hudson plane crash on twitter: First reports, best coverage. MEDIAite `http://www.mediaite.com/online/hudson-plane-crash-on-twitter-first-reports-best-coverage/` (2009). [Online; accessed 23-July-2013]
24. StatusNet, I.: Statusnet. `http://status.net` (2013). [Online; accessed 1-October-2013]
25. Teutle, A.R.M.: Twitter: Network properties analysis. In: Electronics, Communications and Computer (CONIELECOMP), 2010 20th International Conference on, pp. 180–186. IEEE (2010)
26. Tsai, H., Harwood, A.: A scalable anonymous server overlay network. In: Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on, vol. 1, pp. 973–978. IEEE (2006)
27. Wang, L.: Attacks against peer-to-peer networks and countermeasures. In: T-110.5290 Seminar on Network Security (2006)
28. Warner, M.: Syria internet shutdown: A loser's strategy. PBS Newshour `http://www.pbs.org/newshour/rundown/2012/11/syria-internet-shutdown---a-losers-strategy.html` (2012). [Online; accessed 23-July-2013]
29. Warren, J.: Bitmessage: A peer-to-peer message authentication and delivery system (2012)
30. wikipedia: Distributed social network. `http://en.wikipedia.org/wiki/Distributed_social_network` (2013). [Online; accessed 29-July-2013]
31. Wikipedia: Glossary of bittorrent terms. `http://en.wikipedia.org/wiki/Glossary_of_BitTorrent_terms` (2013). [Online; accessed 23-July-2013]
32. Wikipedia: History of the internet. `http://en.wikipedia.org/wiki/History_of_the_Internet#Packet_switching` (2013). [Online; accessed 23-July-2013]
33. Xu, T., Chen, Y., Zhao, J., Fu, X.: Cuckoo: towards decentralized, socio-aware online microblogging services and data measurements. In: Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement, p. 4. ACM (2010)
34. Yang, Y., Yang, L.: A survey of peer-to-peer attacks and counter attacks. In: International Conference on Security & Management (SAM 2012), pp. 176–182 (2012)