Enabling Decentralized Microblogging Through P2PVPNs

Pierre St Juste, Heungsik Eom, Kyungyong Lee, Renato J. Figueiredo
Advanced Computing and Information Systems Lab,
University of Florida,
Gainesville, FL, 32611, USA
Email: {ptony82,hseom,klee,renato}@acis.ufl.edu

***Abstract*—In the past few years, many peer-to-peer microblogging solutions have been proposed and/or implemented utilizing various technologies such as DHTs, multicast trees, and/or gossip protocols. These previous works address the issue of privacy and performance in a variety of ways including the use of session keys for message encryption or direct connections for low latency communication. We propose a decentralized microblogging service which takes advantage of available peer-to-peer virtual private networking (P2PVPN) technologies which provide privacy and low-latency communication in the common case of P2P messaging among social peers. Leveraging the private IP connectivity of P2PVPNs, our design utilizes both IP multicasting and random walks to ensure that peers are able to publish messages with varying degree of scope (i.e. friends, friends of friends, and/or the public). We study the implications of our data dissemination mechanism for a decentralized microblogging service through simulation-based analysis based on synthetic social graphs. Overall, our experimental results show that peers can effectively follow each other's updates with acceptable overhead. Through the use of our pseudo-random-walk algorithm, we estimate that, in a 900K social graph, with a TTL of 100, a user can retreive updates from anyone in the social graph 55% of the time, but by increasing the TTL to 400 that hit rate increases to 95%.**

***Index Terms*—microblogging, peer-to-peer, multicast, message distribution, social network**

I. Introduction

Twitter's rise in popularity, followed by its censorship in parts of the world, have motivated the research community to tackle the challenges of building a decentralized microblogging service. For example, during the Egyptian uprising, the country was disconnected from the global Internet while the internal networking infrastructure was still operational. A peer-to-peer microblogging service resilient to such events could still provide the people with the ability to communicate privately. Hence, in the past few years, many peer-to-peer microblogging solutions have been proposed and/or implemented utilizing various technologies such as structured DHT [1], multicast trees [2], [3], and/or gossip protocols [4]. To address message privacy, some approaches use session keys, or asymmetric keys, which brings the challenge of managing cryptographic keys. For efficient data dissemination, direct connections are used in many cases, or data storage is done through a DHT; hence, these approaches have to deal with user connectivity through NATs and firewalls, and/or the complexity of bootstrapping and maintaining a structured overlay for a DHT.

We propose a decentralized microblogging service which takes advantage of available peer-to-peer virtual private networking (P2PVPN) technologies, which provide private, low-latency communication in the common case of messages sent among social peers. With private connections and multicast message delivery already available in P2PVPNs, we are able to focus on other key areas of building a private and decentralized microblogging service, such as message scope and data availability. Our design utilizes both IP multicasting and random walks to ensure that peers are able to publish messages with varying scopes (i.e. friends, friends of friends, and/or the public). We study the implications of our data dissemination mechanism for a decentralized microblogging service through simulation-based analysis based on synthetic social graphs. We also developed a prototype implementation to demonstrate the feasibility of our design choices. Overall, our experimental results show that peers can effectively follow each other's updates with acceptable overhead.

The primary goal of our design is to enable fast, private updates among friends, rather than the Twitter model which focuses more on posting messages publicly on the Web. By utilizing current P2PVPN technologies such as Hamachi [5], or SocialVPN [6], we build upon a layer where users have private IP connectivity to each other along with IP multicast support thus enabling trusted social communication on the Internet (even through NATs and firewalls). P2PVPNs can naturally map to social graphs and therefore serve as a foundation for building decentralized social services because by providing low latency, private communication network links among friends.

In this paper, we focus on understanding the characteristics of the social overlay formed by P2PVPNs – for example, the ability to reach friends with a one-hop multicast IP packet, and the impact of the high clustering coefficient on replication and data availability. With this underlying social overlay, we build a microblogging service that lets the user control the propagation of a post in the social overlay through the use of IP multicast packets. We also leverage the properties of the social graph in our data replication heuristics in order to efficiently disseminate public posts throughout the social overlay. By designing with privacy as a starting point, we are able to build a system that makes it more difficult to censor, disrupt, and infiltrate which ultimately creates a more robust

978-1-4673-3133-3/13/$31.00 ©2013 IEEE

microblogging service.

The main contributions of this paper are:

- A novel decentralized microblogging system which uses IP multicasting and random walks to propagate updates through private links of a P2PVPN social overlay.
- A simulation-based analysis which estimates bandwidth costs and compares data dissemination strategies for three well-known synthetic social graph models.

II. Motivation

Microblogging services provide users with an intuitive and widely-available method for communicating ideas. Recent events have also shown them to be useful tools to organize revolts. As a result, they have also become targets of governments seeking to curtail communication among citizens by blocking access to micro-blogging services. In many extreme cases, whole nations have been disconnected from the global Internet in order to deny dissidents access to any service that may facilitate their revolution [7]. Although it is possible for government to disconnect a nation from the Internet by disabling a major links to the outside world, in most cases, it requires more effort (and causes more disruption to local infrastructure and economy) to shutdown a country's entire internal Internet apparatus. This was the case in Egypt; after disconnection from the Internet, services running internally where still accessible [8]. This occurrence motivates the need for a microblogging service that is decentralized and would continue to function despite a disconnection from the global Internet.

Our goal is to create a service which makes it possible for users to send messages to their followers directly in a peer-to-peer manner without having to depend on a centralized service. Hence, disabling such a service would require governments to shut down their internal Internet infrastructure, which may be a more costly and difficult endeavor. By leveraging existing P2PVPN technologies, we develop a service that allows communication among peers through direct, encrypted IP tunnels without the need of a middleman. With private messaging to trusted peers as the primary feature of a P2PVPN, we eliminate many of the shortcomings of a centralized approach by making it harder to aggregate all messages in a centralized database.

III. Related Works

FeedTree [2], and Megaphone [3] are peer-to-peer micronews/RSS services which are built on top of Scribe [9], where followers join a multicast tree through the use of the Pastry [10] structured overlay. By sending updates to the root node of the tree, the messages are propagated through all members of the tree. The major difference between these approaches is that FeedTree uses the RSS feed model to disseminate messages while Megaphone uses the microblogging model of delivering short 160-character messages. In our system, we explore the feasibility of designing a microblogging service without depending on a structured DHT.

FETHR [11] is a more recent approach aimed at a fully decentralized HTTP-based microblogging service. Peers subscribe to each other by exchanging canonical URLs and use the HTTP GET and POST methods to pull or push updates to each other. This approach also recommends using gossip-based dissemination for users with a high number of followers. In Cuckoo [1], peers use a DHT to discover each other's endpoints and send follow requests directly to each other. Publishers are then able to send updates directly to a subset of followers, depending on bandwidth availability, while the remaining followers use a gossip protocol to propagate updates among themselves. To handle churn, the Cuckoo approach relies on a centralized backend which stores all updates and follow requests from all users in case a publisher is not online. Unlike Cuckoo, our approach does not use a DHT nor centralized backend.

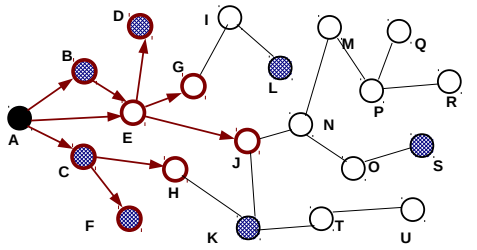
A decentralized microblogging service can also be considered as a decentralized online social network (OSN), specialized for efficient delivery of short text messages. PeerSoN [12] is a fully decentralized OSN. Similar to Cuckoo, peers use a DHT as a lookup service to locate each other's endpoints, then exchange encrypted information directly with each other. Safebook [13] is another DHT-dependent OSN which focuses on anonymity by adding an additional layer of indirection to message requests by routing them through multiple layers of friends and friends of friends. Neither of these approaches mention a replication mechanism to ensure content availability when users are offline. Vis-a-vis [14] is yet another decentralized OSN proposal which relies on DHT-based multicast for message propagation. In this system, users run a virtual individual server (Vis) which stores and serves all of the content on behalf of the user. To ensure content availability, the user can run a Vis on a cloud provider such as Amazon EC2. LifeSocial [15] is a totally DHT-dependent OSN; everything a user publishes is stored in the DHT which ensures data availability even in the case of churn.

IV. Design

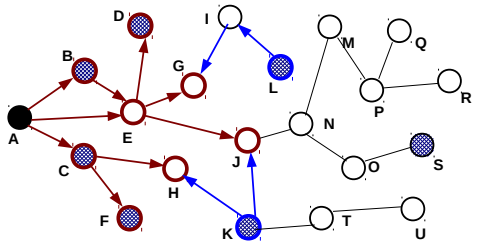
Our microblogging service is built on two basic IP layer mechanisms: 1) IP multicasting to propagate messages to two-hop neighbors in the social graph, and 2) UDP datagrams for random-walking through the social graph to disseminate updates to socially distant peers. In order to enable this service, the assumption is that users run a P2PVPN service that provides peer-to-peer, encrypted IP tunnels to friends, even those behind NATs and firewalls; and the P2PVPN service supports IP multicast.

A. P2PVPNs and IP Multicasting

Although it is common knowledge that the Internet does not support IP multicasting and UDP traffic is firewalled in many cases, there still exist methods that currently allow peers to have direct, unrestricted IP connectivity to one another, including IP multicasting support. Virtual private networking is the typical solution to providing unblocked IP connections to nodes that are geographically dispersed. A peer-to-peer virtual



(a) Multicast Push



(b) Multicast Pull

Fig. 1. **Multicast Push.** Publisher (filled in black): A, Followers (partial filled in blue): B, C, D, F, K, L, S. Publisher node A pushes posts to nodes B, C, D, E, F, G, H, J (since they are two-hops away). Some nodes see duplicate posts (e.g. node E from both nodes A and B). **Multicast Pull.** Nodes K and L are three and four hops away and they are able to receive updates from publisher node A through nodes G, H, and J. When node L does a two-hop multicast request, it reaches node G which sends the posts back to node L.

private network (P2PVPN), such as [5], [6], is a practical decentralized alternative to centralized VPNs. In a P2PVPN, peers leverage P2P technologies to tunnel IP traffic directly to each other instead of having to rely on a centralized VPN gateway. P2PVPNs provide a means for trusted and social peers to communicate with each other at the IP layer in order to collaborate (e.g. multiplayer gaming, screen sharing, media sharing, among others). Since connections are created with diligence and mainly with trusted, social peers, the resulting peer-to-peer network is a social overlay. P2PVPNs also provide IP multicast support by tunneling the multicast packets to each friend that they currently have an encrypted P2P connection with.

B. Multicast Push to Followers

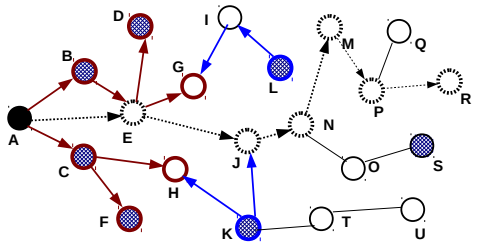
Message Format. Every post generated in the system contains the following information: a creator UID, a destination, a TTL, a timestamp, a post ID, a permission flag, a message, and a signature. The creator UID helps the system keep track of the source of the posts. The destination field is set to *all*, meaning that the message is broadcasted to all friends through IP multicast. The TTL field lets the system control the scope of the message; if a user would like to post messages to only friends, then the TTL is set to 1. If a user would like to reach friends, as well as friends of friends, then the TTL is set to 2. The timestamp helps keep track of the creation time of the message. The post id is a number which increments by one for each post a user generates. The permission flag helps control the privacy of the message; peers are only allowed to share updates from pull requests if the message is set to P (or public). The message is the actual content of the post; currently this is limited to a maximum of 140 characters. The signature field is created by hashing the creator UID, the timestamp, the post ID, and the message and signed with the creator's private key. Therefore the signature ensures the integrity of the message and verifies its creator, assuming recipients have access to the creator's public key.

A multicast push is the most basic message type in our system. Since a P2PVPN gives IP tunnels among friends, and multicast support which makes it very easy to contact all friends privately, implementing this feature simply involves sending a UDP packet to an IP multicast address. However, it is important to note that despite its simplicity, it is a vital first step in allowing peers to send updates privately. This multicast push, which we view as the common case, is efficient because messages are sent directly to friends, and they are private by default because they go through encrypted IP tunnels. For example, suppose user Alice wants to advertise an event to her friends. She uses our service to quickly broadcast that message to her one-hop friends who are currently online. In this example, Alice would limit the scope of her message to only friends, which means that the TTL will be set to 1. Unlike many previous works, we believe that pushing messages to close friends complements polling for new messages because it improves the interactivity of the system by delivering messages with low latency.

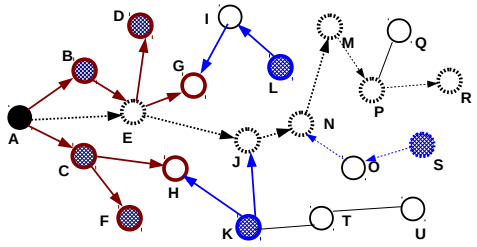
C. Multicast Pull by Followers

Message Format. Users do not only rely on the publishers to push messages to them; they can also proactively pull for new updates and retrieve them either from the publisher or from others nodes in the P2PVPN. A pull request contains the following information: a requester UID, a destination, a TTL, a request ID, a timestamp, the list of followers, and a signature. The requester UID helps the system identify who should receive responses for a particular request. The destination field is set to *all* to indicate the use of multicasting. The TTL can be set to 1 or 2 depending if the follower would like to pull updates from friends and/or friends of friends as well. Each request has a unique ID, which allows the system to keep track of where a request originated so that replies can be sent along the right path. The timestamp helps determine the freshness of a request; requests beyond a certain time window can be ignored. The list of followers includes the publishers that the requester is following, along with the last post ID for each follower. That information is used by each node that receives the request to determine if a requester is missing the latest posts from a particular publisher. The signature in this case is the signed hash of the requester UID, request ID, the timestamp, and the list of followers. The signature makes it possible to verify the legitimacy of the request and helps guard against spoofed requests.

Multicast pulls are used by followers because sometimes they do not receive pushed posts due to packet drops in the P2PVPN, or due to the follower being offline. Currently in our prototype, the pull requests are generated every five minutes,



(a) Random Push



(b) Random Pull

Fig. 2. **Random Push to Distant Followers.** Publisher (filled in black): A, Followers (partial filled in blue): B, C, D, F, K, L, S. Node A pushes posts through a random path to extend the reach of the posts beyond four-hops. As a result, node A's post is stored at nodes E, J, N, M, P, and R. **Random Pull by Distant Followers.** Node S does a random-walk and gets node A's posts through node N which has cached it due to node A's random walk push.

but this period is configurable by the user. The multicast pull also serves another important task: it makes it possible for followers who are four hops away to receive public updates from a publisher. As shown in Figure 1(b), if node A pushes a public post to friends and friends of friends, node L is able to retrieve those updates through node G, even though nodes A and L are separated by four social hops. Therefore, when node A pushes a public post to all friends and friends of friends, the two-hop requests make it possible for followers who are four hops away to receive updates through common subset of friends of friends.

D. Random-walk Push to Distant Followers

Message Format. The message format for the random-walk push is very similar to the multicast push, but with two main differences. First, the destination field is set to *any* instead of *all*. When set to any, the system selects a random peer to forward the message. This is the most basic form of a blind random-walk. Second, the TTL value is set to a large value (e.g. 100, 200, or 400) depending on the user's preference. The TTL serves as the replication factor because the message is stored at each node it reaches. Selecting a higher TTL increases the chances that distant followers will be able to receive the updates of the publisher. Our analysis discusses the impact of choosing different TTL values.

The need for a random-walk push is necessary to ensure that any peer in the network could follow each other. Although this form of interaction is not expected to be the common case, it is important to provide a controllable mechanism in which a publisher could expand the reach of their posts. By replicating posts throughout the social overlay, a publisher can control the availability of his/her updates. However, randomly replicating the messages does not ensure that it is stored at follower nodes; followers also need to request updates from peers in the network to increase their chances of obtaining messages. For example in Figure 2(a), the random push method replicates posts at nodes E, J, N, M, P, and R. The nodes are not follower nodes, but replicate posts such that they will make it easier for followers to retrieve updates from node A. Finally, it is possible that a node is visited twice in our random-walk; we discuss this further in our analysis section.

E. Random-walk Pull by Followers

Message Format. The message format for the random-pull is quite similar to the multicast pull requests. However, the destination field is changed to *any* in order to cause a random-walk in the social overlay. Also, the TTL is set to a large TTL (e.g. 100, 200, or 400) to increase the search path for updates.

The random pull requests are handled in a similar manner as the multicast pull request; if a peer notices that the requester does not contain the latest updates, it replies with these updates. As shown in Figure 2(b), node S is able to pull node A's updates through node N that holds a replica of node A's updates from the random walk. Node N replies to node S and forwards the random pull request to the next node in the system. The replies take the reverse path of the random-walk. The reverse path is known because each node builds a routing table that maps request IDs to IP addresses. That information makes it possible to send a reply along its reverse path in the system as long as the reply has the same ID as the request. Also, a random pull request continues through the network until the TTL reaches zero. This ensures that the requester can control the number of nodes to sample for new messages from his/her followers. Overall, the combination of our random push and pull mechanism makes it possible for any peer in the system to follow another peer as long as both the publisher and the follower use the appropriate TTL values in their request, while the common case of private communication among social peers is handled with P2PVPN multicast push and pull.

V. Analysis

In this section, we discuss the bandwidth costs of our multicast messages, and study the impact of different TTL values on the propagation of updates through the social overlay. Our analysis is based on the assumption that P2PVPNs form social overlays; therefore, by analyzing our design on top of various social graphs, we are able to gain some insight on the performance of our system.

A. Implementation

We also created a prototype implementation to demonstrate the feasibility of our design. Coding simplicity is one of the strengths of our approach. By leveraging P2PVPNs and reusing Berkeley sockets API, HTTP and SQLite for user interface and data storage, we designed the whole system with less than 1500 lines of code. Our implementation is written in

TABLE I
SOCIAL GRAPH STATISTICS

Method	Nodes	Edges	CC	ND	BW (KB)
Random	97,134	291402	0.000	6	7
Random	905,668	9,056,680	0.000	20	21
Barabasi-Albert	97,134	291,393	0.001	6	22
Barabasi-Albert	905,668	10,867,872	0.001	24	84
Nearest Neighbor	97,134	279,694	0.17	6	13
Nearest Neighbor	905,668	12,302,767	0.15	27	86

ND - average node degree, CC - average clustering coefficient, BW - average number of packets per node for 2-hop multicast

Python and uses its corresponding built-in modules to enable the following capabilities: network communication, object serialization, data encryption, data storage, and web interface. To communicate, the service first joins an IP multicast group and then listens on the appropriate UDP port for incoming posts and pull requests. To send a multicast post or a pull request, the payload is encapsulated in a UDP datagram that is sent to the multicast IP address through the virtual networking interface (NIC) of the P2PVPN. The P2PVPN then sends the multicast IP packet to all connected friends, so that the listening nodes can receive the multicast UDP message. The P2PVPN handles multicast group management as well as key management, and encryption.

B. Social Models for Synthetic Graph Generation

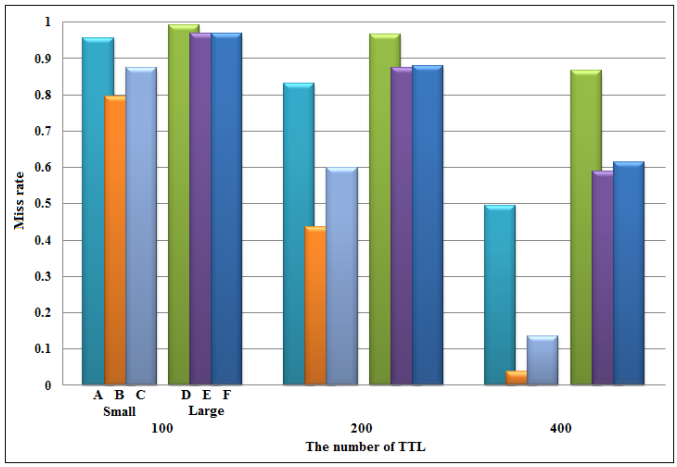
There is active research on generating representative graphs that maintain the fidelity of social networking graphs [16]. For this paper, we utilized two recently studied synthetic social graph techniques presented in [16]: Barabasi-Albert and nearest neighbor. To have a basis for comparison, we also used a regular random graph. Using the parameters provided in [16], we generated a 90K-node graph, and a 900K-node graph for each graph generation technique (see Table I for graph statistics). These graphs are generated and analyzed with the NetworkX graphing library [17], which is a Python package for examining complex networks.

C. Bandwidth Cost

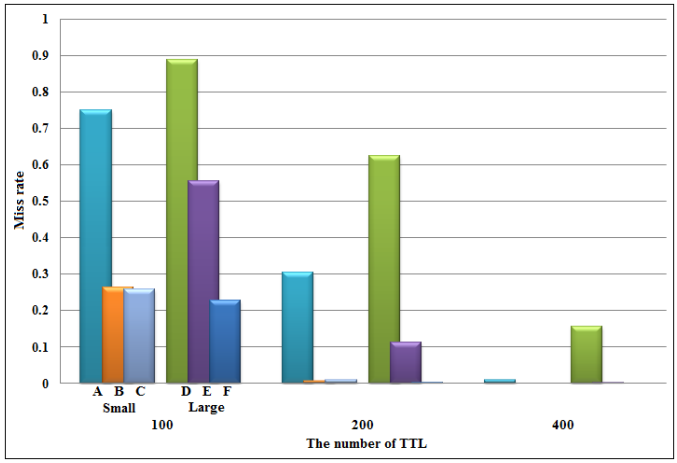
Multicasting is our primary method for pushing and pulling messages to friends and friends of friends; therefore it is important to have an understanding of the amount of bandwidth these types of messages consume. In the Table I, we show the average bandwidth cost of a two-hop push or pull for the different types of social graphs (assuming a message size of 1 KB). For the 90K-node graphs, the average bandwidth per node is between 20KB to 30KB, and for the 900K-node graphs, the average bandwidth is between 80KB to 90KB. Therefore, the use of the 2-hop multicast mechanism is practical in the case of a 1 million node network since it is not too costly in terms of bandwidth.

D. Random-walk Replication

We conducted various simulations on the six social graphs in order to determine the impact of different TTL values.



(a) Random-walk



(b) Pseudo-Random Walk

Fig. 3. **Miss rates 10K push/pull messages.** Graphs types: A - Random 90K, B - Barabasi-Albert 90K, C - Nearest-neighbor 90K, D - Random 900K, E - Barabasi-Albert 900K, F - Nearest-neighbor 900K. In graph (a), the blind random-walk is only practical with a TTL of 400 where the probability of pulling updates from a follower is between 50% to 95%. In graph (b), the pseudo-random-walk performs better with a success rate of 70% to 100% at a TTL of 200. Followers that are more than 4-hops away in the social graph can follow updates from any user in the network with high likelihood.

In our simulations, we implemented two types of replication strategies. The first is a blind random-walk where posts are replicated at nodes along the random path. The query for these nodes also follows a random path as well. Since our random-walk does not remember past visited nodes, it is likely that some nodes are visited twice. With the 90K-node graphs, the effective replication is about 75% of the TTL because about 25% of the nodes are revisited in the random-walk. For the 900K-node graphs, the effective number of replicas is 95% of the TTL.

E. The Impact of the TTL on Random Push/Pull

The graphs in Figure 3 show the miss rate of a random push and pull mechanism. The miss rate indicates the probability that a random node in the social overlay would miss updates from a particular peer using the random push/pull model.

As shown in Figure 3(a), a blind random walk has a poor performance in all of the graphs, with miss rates higher than 80% for a TTL of 100. When the TTL is set to 400, the miss rate is at a maximum of 22% for the social graphs with 90K nodes; however, the miss rate jumps to 60% for the social graphs with 900K nodes. An important observation is the fact that the social graphs have lower miss rates than the random graphs in all cases, especially in the 400-TTL case. This indicates that the high clustering and power-law node degree distribution in social graphs make it easier to lookup data [18].

We also tested another replication heuristic which we call a pseudo-random walk. Instead of replicating the updates at random nodes, the message takes two random hops without replication, then finds the node with the closest identifier to the ID of the message as the target node for replication. For example, assuming the random path in Figure 2(a), the post starts a node A, goes through nodes E and J randomly without storage; at node J the message is forwarded and replicated at node N only if node N has the node ID closest to the message ID (in comparison to nodes E, and K). From node N, the post travels to nodes M and P randomly without replication again, and goes through the same process of selecting the closest node again. In summary, the heuristic is to take 2 random hops, store the post at the node with closest ID, and repeat.

This pseudo-random-walk lowers the miss rate significantly. As seen in Figure 3(b), for a TTL of 100, the miss rate is below 55%, in contrast to the 80% of the random-walk. By increasing the TTL to 200, the miss rate decreases to 10% and below, and at a TTL of 400, the miss rate goes down to 1% or less. Once again, the random graph has a much worse performance with miss rates of 30% for the 90K random graph and 60% for the 900K random graph for a TTL of 200. In this case, a TTL of 100 has a poor performance with miss rates as high as 75% for the social graphs; however, with a TTL of 400, the miss rate decrease to less 5% for all of the social graphs. By placing data at nodes with the closest ID to the ID of the message, we are able to gain a much better hit rate indicating that it is feasible for a user to push public messages in the social overlay with a TTL of 400.

VI. Conclusion

We have demonstrated the possibility to leverage the private IP links and the social overlay formed by P2PVPNs to create a decentralized microblogging service. The design starts with private message at the core of the protocol which differentiates it from other approaches. The approach also minimizes coding complexity by allowing the use of Berkeley sockets API for communication instead of a customized P2P library interface. With IP multicasting and random-walks, the system is able to push private updates to social peers with low latency, publish updates beyond friends (and friends of friends) through the use of a pseudo-random replication throughout the social overlay, and allow followers to retrieve posts with multicast and random-walk pulls. Our analysis shows that we can achieve fairly good coverage with a TTL of 200 in a 900K-node social overlay.

References

[1] T. Xu, Y. Chen, J. Zhao, and X. Fu, "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*, ser. HotPlanet '10. New York, NY, USA: ACM, 2010, pp. 4:1–4:6.

[2] D. Sandler, A. Mislove, A. Post, and P. Druschel, "Feedtree: sharing web micronews with peer-to-peer event notification," in *Proceedings of the 4th international conference on Peer-to-Peer Systems*, ser. IPTPS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 141–151.

[3] T. Perfitt and B. Englert, "Megaphone: Fault tolerant, scalable, and trustworthy p2p microblogging," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, may 2010, pp. 469 –477.

[4] G. Mega, A. Montresor, and G. Picco, "Efficient dissemination in decentralized social networks," in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, 31 2011-sept. 2 2011, pp. 338 –347.

[5] "Hamachi - instant, zero configuration vpn," http://secure.logmein.com/products/hamachi/vpn.asp.

[6] "Socialvpn - a free and open-source p2p vpn that connects you to your friends," http://socialvpn.org.

[7] I. van Beijnum, "How egypt did (and your government could) shut down the internet," January 2011. [Online]. Available: http://arstechnica.com/tech-policy/news/2011/01/how-egypt-or-how-your-government-could-shut-down-the-internet.ars

[8] J. Glanz, "Egypt leaders found off switch for internet," February 2011. [Online]. Available: http://www.nytimes.com/2011/02/16/technology/16internet.html

[9] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489 – 1499, oct 2002.

[10] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware '01. London, UK, UK: Springer-Verlag, 2001, pp. 329–350.

[11] D. R. Sandler and D. S. Wallach, "Birds of a fethr: open, decentralized micropublishing," in *Proceedings of the 8th international conference on Peer-to-peer systems*, ser. IPTPS'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 1–1.

[12] S. Buchegger and A. Datta, "A case for p2p infrastructure for social networks - opportunities & challenges," in *Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, ser. WONS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 149–156.

[13] L. Cutillo, R. Molva, and T. Strufe, "Privacy preserving social networking through decentralization," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, feb. 2009, pp. 145 –152.

[14] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, and A. Varshavsky, "Vis-a-vis: Privacy-preserving online social networking via virtual individual servers," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, jan. 2011, pp. 1 –10.

[15] K. Graffi, C. Gross, P. Mukherjee, A. Kovacevic, and R. Steinmetz, "Lifesocial.kom: A p2p-based platform for secure online social networks," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, aug. 2010, pp. 1 –2.

[16] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao, "Measurement-calibrated graph models for social network experiments," in *Proceedings of the 19th international conference on World wide web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 861–870.

[17] "Networkx - high productivity software for complex networks," http://networkx.lanl.gov/index.html.

[18] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *CoRR*, vol. cs.NI/0103016, 2001.