2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops

A Framework for Peer-to-Peer Micro-blogging

H. Asthana and Ingemar J. Cox
h.asthana, ingemar@cs.ucl.ac.uk
Department of Computer Science,
University College London,
Gower St., London WC1E 6BT, UK

***Abstract*—The recent rise in the popularity of micro-blogging has been accompanied by concerns over censorship and anonymity in centralized systems. Thus, we consider the problem of implementing a micro-blogging social network over an unstructured Peer-to-Peer network. The problem can be decomposed into two sub-problems, dissemination (also known as replication) and retrieval, which are coupled. For example, the more a blog post is disseminated, the fewer nodes need to be queried in order to retrieve it with a high probability. Both dissemination and retrieval incur bandwidth costs. In this paper, we investigate the optimal replication of data, in the sense of minimizing bandwidth, and the balance between the number of nodes a micro-blog post is replicated to and the number of nodes that must be queried. Minimizing the system bandwidth is critical if our proposed system is to scale from small to larger networks. Our theoretical, probabilistic analysis predicts that micro-blog posts should be replicated onto approximately 20% and 6% of nodes in networks of 10,000 and 100,000 nodes respectively in order to minimize the overall bandwidth of the system.**

I. Introduction

In general, a micro-blog can be defined as short sentences, small images, or video links[1]. Whereas regular blog postings may contain multiple paragraphs and are usually published once a week or at a lower frequency, the short nature of a micro-blog encourages users to publish multiple micro-blog posts each day, usually consisting of personal status updates, current events, news etc. [1].

Over the last few years, micro-blogging social networks have proved extremely popular with Twitter gaining approximately 200 million users. Internet giants such as Google, Amazon, Microsoft, Facebook, and Twitter spend hundreds of millions of dollars each year maintaining vast data centres to support their centralized services. This high cost may discourage competition and innovation. Since a micro-blogging social network provides a convenient way to report and spread news and opinion, Twitter has been a target for censorship by authoritarian governments[2]. Twitter itself has announced its own intention of censoring tweets by country[3]. Finally, Twitter has also released information about users to law enforcement authorities who presumed they were blogging anonymously[4] [5]. Such concerns of cost, censorship, and lack of anonymity provide motivation for a decentralized distributed P2P architecture.

Our proposed framework is based on an unstructured network. We assume search of an unstructured network is based on a probabilistic search that queries a fixed number of nodes. This form of search is known as probably approximately correct (PAC) search [2], and a strong theoretical framework exists to predict the probability of a successful search based on the distribution of documents in the network. Retrieval and search in a dynamic environment such as a decentralized micro-blogging network is then determined by how quickly new information can be randomly distributed through the network. To achieve this rapid replication, we use a modified rumour spreading algorithm.

Utilizing the PAC search architecture, we can derive a relationship between the replication of a document, and the number of nodes we need to query in order to retrieve the document with a given high probability. We can then apply this knowledge to the unique characteristics of a micro-blogging social network to derive the optimal replication which minimizes the total bandwidth generated by the system. The problem is then transformed to that of replicating new information to this number of nodes given the constraints of (i) time (we wish to propagate the information as quickly as possible), and (ii) bandwidth (we wish to propagate the information using as little bandwidth as possible, otherwise our solution will not be scalable). We refer to this as *rapid-yet-restrained dissemination*. We emphasize that this replication must occur in the absence of any centralized coordination.

II. Prior Work

The distributed social network, Diaspora[6] addresses the issue of ownership of user data. Users publish data on servers known as *pods* which form the network. However, if the pod is forced off-line, the data is no longer available. In the last few years, DHT-based P2P social networks systems such as PeerSoN [3], eXO [4], Cuckoo [5], and My3 [6] have been proposed to address issues of privacy and data availability. However, these papers do not address how the DHT would facilitate the index required for keyword search of micro-blog posts, which are very high in frequency of creation. Previous research in Information Retrieval (IR) of large scale data has deemed DHT-based P2P networks infeasible due to bandwidth requirements [7].

Probabilistic storage and search in an unstructured P2P network can be modelled as follows. Given a set of n nodes in the network, we assume that the object of interest is stored on a random subset of r nodes. A query is issued to a random subset of z nodes. We are interested in the probability that the two subsets have a non-empty intersection, as this

[1]http://en.wikipedia.org/wiki/Microblog
[2]http://en.wikipedia.org/wiki/Censorship_of_Twitter
[3]http://www.bbc.co.uk/news/world-us-canada-16753729
[4]http://www.bbc.co.uk/news/uk-england-tyne-13588284
[5]http://www.bbc.co.uk/news/technology-18990253
[6]https://joindiaspora.com/

978-0-7695-5023-7/13 $25.00 © 2013 IEEE
DOI 10.1109/ICDCSW.2013.63

implies a successful search for that object. In the context of information retrieval, this abstract model is equivalent to *known-item* search. In general, IR is broader than this, and it is therefore necessary to provide a set of documents, usually ranked by relevance.

A. PAC Search

The previous work on randomized search [8], [9], [10] looked at the expected search length necessary to find a specific document. Assuming a query is sent to a constant number of nodes, z, we can also ask what the probability of finding a document is. This, and related questions, are addressed in recent papers on PAC search [2], [11].

In the PAC search architecture, a query is sent to z machines, and the results returned by the different machines are consolidated and then displayed to the user. This functionality is applicable for both *known-item* search and top-k retrieved documents for a keyword search. The correctness of a PAC search is measured by *retrieval accuracy*, which is defined as the overlap of the documents retrieved by querying z nodes to that of querying the entire network. The accuracy (μ), is

$$\mu = 1 - \left(1 - \frac{r}{n}\right)^z \tag{1}$$

and the ratio, $\frac{r}{n}$ is referred to as the replication rate.

Equation (1) approximates to $1 - e^{-zr/n}$ [2]. The product of the replication rate, $\frac{r}{n}$, and the number of nodes queried, z, $\frac{zr}{n}$, is referred to as the *sample index*. We can utilize different combinations of the replication rate, $\frac{r}{n}$, and number of nodes queried, z, to arrive at the same accuracy.

PAC search can also be implemented for proportional and square-root replication [11], and in these cases, the accuracy can be substantially improved, albeit to the detriment of unpopular documents.

The work described in this paper extends the preliminary work by Asthana and Cox [12]. This work proposed the basic architecture used here for micro-blogging in an unstructured P2P network. However, the proposed implementation in [12] coupled replication with querying, i.e. replication occurred periodically and as part of the query - each node sought to replicate a portion of its recent blog posts when querying random nodes. This makes the system extremely inflexible. For example, increasing the interval between queries adversely affects the replication. Under the proposed system in [12], it is also impossible to arrive at an optimal replication which minimizes the total bandwidth. In this paper, we decouple querying from replication. Consequently, we are now able to analyze and minimize the overall bandwidth usage of the system. This is accomplished for both uniform and non-uniform replication policies. We further extend the system reported in [12] by (i) reporting results for keyword search as well as followed blog post retrieval and (ii) performance under churn.

B. Information Dissemination in P2P networks

Rumour spreading algorithms, also known as gossip spreading protocols or epidemic protocols, provide an efficient way to rapidly spread information within a network. Suppose we have a group of n individuals, and at $t = 0$, only one individual knows the rumour. At time t, let x denote the individuals who do not know the rumour (*susceptibles*), and y denote the number of individuals who know the rumour (*infectives*), so that $x + y = n$. Bailey [13] showed that the number of individuals who have received the rumour, y, is given by

$$y = \frac{n}{1 + ne^{-nt\eta}}$$

where η is the contact rate. The contact rate is analogous to the number of nodes each peer communicates with. Clearly, as $t \to \infty$, $y \to n$ and the entire network is infected. However, we desire information to be spread only to a portion of the network.

One of the earliest uses of rumour spreading was by Demers *et al.* [14] to synchronize replicated databases. Demers *et al.* introduce a new class of nodes which know the rumour but do not participate in spreading it (*stiflers*). An *infective* becomes a *stifler* with a probability θ when it is contacted by another *infective*.

III. System Design

For simplicity of the analysis and without loss of generality, we assume a synchronous model with a discrete global clock, zero processing times, and message latencies of a single time unit. We assume each iteration is equivalent of one second and we use the two interchangeably.

Our goals are two-fold. First, a user should be able to retrieve the micro-blog posts[7] of other users he/she is *following*, with a sufficiently high accuracy by querying z random nodes in the network. Second, a user should be able to perform a keyword search, akin to a search engine, by sending the query to, again, z random nodes. To retrieve the required posts, a node in the network makes a *request* to z other nodes every s seconds. By randomizing the request time at each node, we can ensure that the number of nodes making a *request*, at any given moment, is roughly equal.

To participate in the network, each node contributes some disk space. We assume that each node in the network contributes, on average, 1GB of disk space to support the services. Of this contributed disk space, we assume that 90% is utilized for storage of the posts, and the remaining 10% for indexing the stored posts.

To complete our network, we must estimate the number of followers (f) a node is likely to have. If Node $\mathcal{A}$ *follows* Node $\mathcal{B}$, then in our graph there exists a uni-directional link from $\mathcal{A}$ to $\mathcal{B}$. We set the total number of *follower-links* to $35n$ [1]. Though the total number of follower-links in the Twitter graph is not known, Kwak *et al.* [1] reported 1.47 billion links for 41.7 million users. For our initial calculations we set $f = 35$ uniformly such that each node *follows* 35 others. We relax this restraint when we consider non-uniform replication in Section III-B, and also in our simulations.

Next, we define a micro-blog post as text limited to 500 characters including white-space. This is approximately 3.5 times larger than a Twitter message and is large enough for a small paragraph of text or a couple of sentences with a URL. We assume UTF-8[8] encoding of a post, and, on average, 2 bytes per character. Thus each post requires 1 KB of disk space.

[7]Henceforth, *blog* is used interchangeably with *micro-blog* and *post(s)* is used as an abbreviation for *micro-blog post(s)*

[8]http://tools.ietf.org/html/rfc3629

For dissemination, we allow a blog post 30 seconds to replicate onto the desired number of nodes before it becomes available for retrieval. The choice of 30 seconds is somewhat arbitrary but it allows our proposed system to mirror real life centralized micro-blogging systems, such as Twitter. We test our proposed framework with a blog post creation rate of $\alpha = 2.5 \times 10^{-4}$ posts per node per iteration. This is 10 times higher than Twitter's average of 2,300 tweets per second[9] with a 100 million *active* user base[10].

A. Bandwidth

We now consider the total bandwidth generated by our proposed framework at any given iteration. When a post is created by a node, it is instantly replicated onto r other random nodes. (This is of-course idealized, and in Section IV we describe our rapid-yet-restrained method of dissemination). Also, at any given iteration approximately $\frac{n}{s}$ nodes contact z other random nodes to retrieve the posts of the peers they follow.

The proposed framework can be thought of as an overlay to an actual network topology, where one node can contact another random node directly. For a particular network topology (e.g. homogeneous, random graph, power law), the bandwidth needs to be multiplied by the average number of hops required for one random node to contact another. The network topology is outside the scope of this paper, and we refer the reader to [16], [17], [18] for an extensive overview of information dissemination in various types of networks, P2P bootstrapping, membership, and random node sampling.

To calculate the bandwidth we assume the following:

- $c_t = 64$ – the cost, in bytes, when one node contacts another. This is the size of an empty TCP/IP packet, and the cost must be incurred in all requests and responses.
- $c_n = 8$ – the cost, in bytes, of specifying a peer in a request. When a Node $\mathcal{A}$ makes a request to Node $\mathcal{B}$, for each peer Node $\mathcal{A}$ follows, it must specify - in the request to $\mathcal{B}$ - the peer number, and the number of the last post published by that peer which Node $\mathcal{A}$ has already retrieved.
- $c_b = 1000$ – the size, in bytes of a blog post, as described earlier.

We also assume the interval between requests, s, is 60 seconds, which allows for a user experience that is similar to Twitter[11]. Finally, Equation (1) provides the basis for determining the required replication rate of a document in order to meet a required level of accuracy. We assume that the required accuracy (as defined in Section II-A) is 95%. An accuracy of 95% implies that on average we will retrieve 95% of the posts we want by querying z nodes, as compared to searching the entire network. The accuracy is for one request, which consists of querying z randomly selected nodes. *Performing another request for the same information increases the accuracy further.* Note, this does not imply that a user will miss out on 1 of 20 of her required blog posts; any missed blog posts will, with very high probability, be retrieved when the node queries again s seconds later.

The accuracy μ approximates to $1 - e^{-zr/n}$ [2]. Let $\epsilon = \frac{zr}{n}$. For $\mu = 95\%, \epsilon = 2.996$. Since a uniform replication of documents in PAC lends itself to a binomial distribution, we can also conceptualize the retrieval process as follows: The accuracy gives us the probability of retrieving *at-least one* document and ϵ gives us the number of *expected* documents when we query z random nodes.

The bandwidth ($\mathbf{B}$) has two components: replication and retrieval. The replication bandwidth is a function of the number of blog posts created, αn, and the number of nodes, r, they are replicated onto, and for one iteration is

$$\mathbf{B}_r = \alpha n r \left(c_t + c_b\right) \tag{2}$$

TABLE I: The nodes queried, z, the optimal replication, and the associated minimum bandwidth for various values of query interval, s, for an accuracy of 95% in a network of 10,000 and 100,000 nodes where each node follows $f = 35$ other nodes.

$n = 10,000$					
		Optimal	Bandwidth (MB)		
s	z	Replication	Replication	Retrieval	Total
30	15	21.05%	5.599	5.599	11.199
60	16	19.45%	5.174	5.175	10.349
90	16	18.90%	5.027	5.026	10.053
120	16	18.61%	4.950	4.949	9.899
$n = 100,000$					
		Optimal	Bandwidth (MB)		
s	z	Replication	Replication	Retrieval	Total
30	45	6.658%	177.103	177.079	354.182
60	49	6.153%	163.670	163.693	327.363
90	50	5.976%	158.962	158.971	317.932
120	51	5.885%	156.541	156.561	313.102

The retrieval bandwidth has two elements due to the query ($\mathbf{B}_q$) and the response ($\mathbf{B}_e$). Since $\frac{n}{s}$ nodes peform a query at any given iteration, the bandwidth for querying is:

$$\mathbf{B}_q = \frac{n}{s} \times z \times \left(c_t + f c_n\right) = \frac{\epsilon n^2 \left(c_t + f c_n\right)}{sr} \tag{3}$$

To calculate the response bandwidth, we need to know the number of expected blog posts that are retrieved by the requests to z random nodes made by a given node, say, Node $\mathcal{A}$. The probability that one of the peers has published a post at a given iteration is simply α, and therefore the expected number of posts published by the peers that Node $\mathcal{A}$ follows is $fs\alpha$, and the expected number of retrieved posts by making a request to z random nodes is $\epsilon fs\alpha$. Thus the bandwidth for replying at any given iteration is given by:

$$\mathbf{B}_e = \frac{n}{s} \times z \times \epsilon fs\alpha \left(c_t + c_b\right) = \frac{n^2 \epsilon^2 f \alpha \left(c_t + c_b\right)}{r} \tag{4}$$

The total bandwidth at any given iteration, $\mathbf{B} = \mathbf{B}_r + \mathbf{B}_q + \mathbf{B}_e$. The total bandwidth, $\mathbf{B}$, is a function of the replication rate, r. We can therefore determine the value of r that minimizes $\mathbf{B}$, and then use Equation (1) to determine the corresponding z for a given accuracy. Using equations (3)-(6), for an interval of $s = 60$ seconds, an accuracy of 95%, and a network sizes

[9] http://news.cnet.com/8301-13506_3-20076022-17/twitter-tallies-200-million-tweets-per-day/

[10] Twitter's user base is estimated to be around 200 million users. However a fraction of the user accounts are dormant (i.e. user who signed up but did not participate) and another fraction are fake accounts created only to boost another user's popularity. The *active* user base is estimated to be only half of the total user base [15].

[11] https://dev.twitter.com/docs/rate-limiting

TABLE II: The minimum bandwidth ($\mathbf{B}_{\min}(h)$ and $\mathbf{B}_{\min}(g)$) achieved via the optimal replication (r_h and r_g) for high and regular popularity peers respectively in a network of 10,000 and 100,000 nodes for a $s = 60$, and $\mu = 95\%$

$n = 10,000$								$n = 100,000$							
$p(h)$	High			Regular			Total	$p(h)$	High			Regular			Total
($\times\alpha$)	r_h (%)	z_h	$\mathbf{B}_{\min}(h)$	r_g (%)	z_g	$\mathbf{B}_{\min}(g)$	$\mathbf{B}_{\min}$	($\times\alpha$)	r_h (%)	z_h	$\mathbf{B}_{\min}(h)$	r_g (%)	z_g	$\mathbf{B}_{\min}(g)$	$\mathbf{B}_{\min}$
0.01	48.02	6	0.255	17.48	17	9.208	9.464	0.01	15.213	20	8.093	5.531	54	291.281	299.375
0.1	16.95	18	0.902	17.64	17	8.445	9.347	0.1	5.366	56	28.549	5.580	54	267.153	295.702
0.2	13.23	23	1.408	17.85	17	7.597	9.004	0.2	4.188	72	44.561	5.646	53	240.315	284.876
0.3	11.73	26	1.872	18.12	17	6.747	8.619	0.3	3.713	81	59.261	5.731	52	213.434	272.695
0.4	10.90	28	2.320	18.47	16	5.895	8.215	0.4	3.451	87	73.442	5.842	51	186.492	259.934
0.5	10.38	29	2.760	18.95	16	5.040	7.800	0.5	3.284	91	87.356	5.995	50	159.457	246.814

of 10,000 nodes, the minimum bandwidth is observed when a post is replicated onto 19.45% of the network and nodes query $z = 16$ other randomly selected peers. For a network of 100,000 nodes, the minimum bandwidth is observed at 6.153% replication and nodes query $z = 49$ other randomly selected peers. As the network size increases from 10,000 to 100,000 nodes, the cost of replicating documents to the same network percentage increases linearly, and the minimal bandwidth is found at a lower replication.

As we can see from Equation (3), if we increase the interval between requests, s, we can expect the bandwidth to reduce as well. Table I shows the optimal replication which gives the minimum bandwidth for various values of s.

B. Non-Uniform Replication

We now consider whether replicating and retrieving the posts of high-popularity peers separately would decrease bandwidth significantly. Social networks such as Twitter are generally scale-free networks which exhibit power law characteristics. In a power law, an entity x is distributed as $P(x) = cx^{-\gamma}$, where c is a normalization constant. In our system, the total number of other peers a node follows is $f = 35$, and thus the total number of edges in our network is $nf = 35n$. Instead of these edges being distributed uniformly, we can construct a non-uniform distribution, such that the top-1% of most popular nodes, cumulatively, account for 20% of the followers. Using this distribution, in a network of 10,000 and 100,000 nodes the most popular node has approximately 6,560 and 26,300 followers respectively. We denote the top-1% most popular nodes as *high popularity* (h), and the rest as *regular popularity* (g) respectively. We then distribute each nodes' followers randomly across the network such that the *following* distribution is similar to the follower distribution (i.e. a small number of nodes follow a large of peers, but most follow a small number of peers). The result of this transformation of our network is that while the number of peers a node follows is now a non-uniform distribution, 20% of the followed peers are high popularity nodes and the rest are regular popularity nodes.

Another feature of power-law social networks is that a small number of peers account for most of the information created. We therefore investigate this by setting the probability of blog post creation by high-popularity peers to various fractions of α. The system now consists of n/s nodes making two sets of requests to z_h and z_g nodes to retrieve blog posts created by high and regular popularity peers respectively. The high and regular popularity blog posts are replicated onto r_h and r_g nodes respectively, and the minimum bandwidths for high and regular popularity data can be obtained using Equations (2)-(4), with

$$\mathbf{B}_{\min}(h) \propto p(h) r_h^* \text{ , } \mathbf{B}_{\min}(g) \propto p(g) r_g^*$$

where $p(h)$ and $p(g)$ are the probabilities of blog posts being created by high and regular popularity nodes respectively, with $p(h) + p(g) = 1$. r_h^* and r_g^* denote the optimal replication for minimizing the total bandwidth for high and regular popularity post respectively. The total minimal bandwidth for the entire system is then given by: $\mathbf{B_{min}} = \mathbf{B_{min}}(h) + \mathbf{B_{min}}(g)$

Table II shows the minimum bandwidth ($\mathbf{B}_{\min}$) for various values of $p(h)$ in networks of 10,000 and 100,000 nodes for a query interval of $s = 60$ and an accuracy of 95%. The gains in bandwidth are, unfortunately, small. This is due to the nature of the a system in which the data is highly dynamic. Unlike distributed web search where non-uniform distribution of documents can give large reductions in bandwidth [11], peers in a micro-blogging social network *periodically* query for both high and low popularity documents. Furthermore, unlike a relatively static system, we periodically incur the replication and retrieval cost. This erodes any advantage of replicating high popularity nodes to a larger number of nodes.

IV. Data Dissemination

Previous work on rumour spreading usually has the goal of ensuring that the rumour spreads to the entire network at the fastest possible rate. Our requirement is subtly different; we require that a document be replicated rapidly only onto a fraction of the network.

To attain our goal of rapid-yet-restrained dissemination, when a node publishes a blog post, it immediately replicates its post to z_s nodes. This is the initial seeding of the post. At every iteration, a fraction of the network is selected as *Replicators* ($\mathcal{R}$). A node is selected into set $\mathcal{R}$ with probability $e^{-(t-1)/\delta}$ where t is the age of the most recent blog post found in its storage area. I.e. if a node has a blog in its storage which is one iteration old, the node is selected into $\mathcal{R}$. If a node does not have a blog post which is one iteration old, but *does* have a post which is 2 iterations old, it is selected into $\mathcal{R}$ with probability $e^{-1/\delta}$ and so on.

Each replicator must select blog posts to replicate. At every iteration, each node in the network constructs a *transfer buffer* ($\mathcal{T}$), which consists of a small number of blog posts it has recently come across. Since each post is allowed 30 iterations to spread in the network, we only consider posts which are younger than 30 seconds. Posts are selected into the buffer with an exponentially decreasing probability which is a function of the age of the post: $P(sel) = e^{-t/\beta}$, where t is the number of seconds a blog post has been in existence and

TABLE III: The bandwidth, replication, and retrieval accuracies in networks of 10,000 and 100,000 nodes with blogs seeded to $z_s = 50$ nodes, replicator selection parameter $\delta = 0.66$, and a maximum transfer buffer size of $|\mathcal{T}|_{max} = 100$

Network size (n) = 10,000, transfer buffer parameter $\beta = 6.9$												
z_r	Bandwidth (MB)		Replication (% of network)		Blog Post Retrieval		Keyword Search @ 10		Keyword Search @ 20		Keyword Search @ 30	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
8	8.320	0.841	18.879	4.692	93.851	3.736	93.009	4.088	93.558	3.721	94.094	3.911
8.5	**9.209**	**1.003**	**21.284**	**4.978**	**95.542**	**2.555**	94.143	2.751	94.882	2.617	95.274	2.505
9	9.758	1.090	24.704	5.685	96.654	3.319	95.557	3.091	96.018	2.994	96.200	3.473
Network size (n) = 100,000												
Network size (n) = 100,000, transfer buffer parameter $\beta = 15.9$												
z_r	Bandwidth (MB)		Replication (% of network)		Blog Post Retrieval		Keyword Search @ 10		Keyword Search @ 20		Keyword Search @ 30	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
7	267.288	6.752	5.564	1.708	91.992	1.236	90.748	1.289	91.052	1.143	91.310	1.207
7.5	**307.320**	**7.043**	**6.571**	**1.961**	**95.845**	**1.009**	94.506	1.010	94.943	1.063	95.417	1.030
8	349.106	8.665	7.309	2.097	97.924	0.603	96.399	0.598	96.512	0.549	96.955	0.629

β is a system parameter. The transfer buffer has a maximum size, $|\mathcal{T}|_{max}$.

The higher the value of β, the higher the probability that an older post will be selected into the node's transfer buffer and hence the larger the expected size of $\mathcal{T}$. Note, once a post is not selected into the transfer buffer of a node, it is permanently set for no further selection at that node even if its age is less than 30 seconds. Another way of describing this probabilistic selection process, is that at every iteration a node *stifles* a post with a probability $1 - e^{-t/\beta}$. At each iteration, every replicator disseminates the posts in its transfer buffer to z_r randomly chosen nodes in the network.

The goal of selecting replicators with parameter δ and constructing the transfer buffer with parameter β is to spread new blog posts at the expense of older ones. By *stifling* older blogs, we can achieve restrained dissemination. Parameter β keeps older blogs from being selected into $\mathcal{T}$. Parameters z_s (initial seeding) and δ control the number of replicators active at any iteration and z_r controls the amount of dissemination the replicators are able to accomplish. Many combinations of z_s, δ, and z_r can give us our required replication; we can have fewer replicators contacting a larger number of nodes (z_r) or a larger number of replicators contacting fewer nodes. A detailed analysis of the combination of parameters is outside the scope of this paper.

V. Simulations

As described in Section III-B, the total number of *follower* links in the network is $35n$ and is distributed via a non-uniform distribution such that the top-1% most popular nodes have 20% of all the followers. These top-1% of nodes also account for 20% of all blog posts created. This is motivated by the fact that in Twitter a small proportion of users are responsible for a disproportionately large amount of Tweets and the larger the number of followers a user has, the more they are likely to Tweet [19]. We tested our simulations with a blog post creation rate 10 times larger than Twitter. This corresponds to 2.5 and 25 posts per iteration in a network of 10,000 and 100,000 nodes respectively. Nodes query every $s = 60$ seconds to retrieve the blog posts they follow by querying $z = 16$ and $z = 49$ random nodes in networks of 10,000 and 100,000 nodes (Table I). Since we randomize the request time at each node, there are roughly $\frac{n}{s}$ nodes making a request at any given moment.

Apart from measuring the blog post retrieval accuracy, we also measured the keyword search accuracy.[12] We used the TREC Micro-blog Tweets2011 corpus[13] as the source of posts for each node, i.e. when a node created a post, it sampled, without replacement, a random post from the collection. To evaluate the accuracy of keyword search, we generated a selection of keywords from the collection. In order to measure the accuracy of keyword search, each time a node publishes a post, it is also stored in a central database. When a node performs a keyword search, the results obtained by the node are compared to the results obtained by searching the centralized database, the latter serving as a gold standard, being equivalent to performing an exhaustive search of all nodes in the network. A node that received a keyword query performed a search of its local collection using the BM25 ranking algorithm. The querying node received the ranked lists from each of the z queried nodes and merged them based on the BM25 scores that each node provided.

We tested various values of seeding (z_s), replicator selection (δ), and transfer buffer construction (β). Our aim is to replicate the blog posts to 19.45% and 6.153% of the nodes in networks of 10,000 and 100,000 nodes respectively and to compare the predicted bandwidth and retrieval accuracies against their observed values. We performed 10 simulations for each value of the parameters and network size (10,000 and 100,000 nodes). Each simulation lasted for 1200 iterations, so that each node in the network would make at-least 20 *requests*. At each iteration of each simulation, we recorded the total bandwidth generated, and the posts retrieved by the querying nodes. In our simulations, when a node made a *request* to retrieve the posts it is following, it also included a keyword search within the request. At each iteration, for every node which made a *request*, we recorded the accuracy, as defined in Section II-A for both post retrieval and keyword search.

Table III summarizes the simulations for network sizes of 10,000 and 100,000 nodes. The values are obtained by seeding each blog to $z_s = 50$ nodes, setting the replicator selection parameter to $\delta = 0.66$. The table shows the effect of increasing the value of the nodes contacted by the replicators (z_r) on the dissemination and consequently the retrieval accuracies.

[12]Note, the bandwidth generated by the keyword search is not taken into account in Table III. The key design components of the system are the replication of posts and their retrieval by the nodes following their publishers. Keyword search is included in the simulations to demonstrate that PAC-based system can facilitate both post retrieval and keyword search.

[13]http://trec.nist.gov/data/tweets/

As expected, as the replicators contact more nodes z_r, the blog posts are replicated onto more of the network. As the replication increases, nodes are able to retrieve the blog posts they follow with higher accuracy by querying the same number of random nodes. Other combinations of these parameter values can also yield the same mean replication but with slightly different standard deviations. We omit the full results due to space limitations.

The post retrieval accuracy is the proportion of posts retrieved by querying z nodes relative to performing an exhaustive search of all nodes. The keyword accuracy @ k is the proportion of posts in the top-k of the merged results of querying z nodes in comparison with the top-k results of performing the identical search on the centralized database.

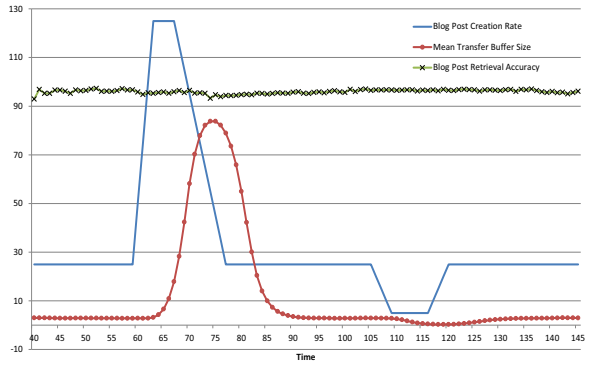

Fig. 1: The blog post retrieval accuracy and mean values of the buffer, $\mathcal{T}$, in a network of 100,000 nodes, with $\beta = 15.9$, $|\mathcal{T}|_{max} = 100$, $z_s = 50$, $\delta = 0.66$, and $z_r = 7.5$.

We also tested the system by inducing a spike in the blog post creation rate. This is motivated by the fact that on Twitter, a five-fold increase in tweets-per-second is known to happen when important events occur. For completeness, we also introduced a lull period where the post creation rate drops to a fifth of the normal post creation rate. Figure 1 shows the blog post retrieval accuracy and the mean transfer buffer size for a network of 100,000 nodes when a spike and a lull is induced in the post creation rate. The figure demonstrates that the retrieval accuracy remains at around 95% regardless of the blog post creation rate. This also demonstrates the utility of the transfer buffers, as the method used to construct them allows them to grow and constrict according to the data creation rate.

A. Churn

Churn is an important factor in any P2P application. We refer the reader to [20] for an overview of churn in P2P networks. We tested our framework with a high level of churn by setting the mean of the session time exponential distribution for the nodes to just 60 seconds. New nodes join with a Poisson arrival rate to balance out the exponential exit rate [9]. Though the number of active nodes at any given moment does not vary by much, approximately 1,000 nodes exit and join the network at every second. We ran two sets of churn simulations - ones in which the high popularity nodes were designated as the stable nodes, and ones in which the stable nodes were chosen at random. The results were nearly identical.

As expected, the net effect of churn is to reduce the post retrieval accuracy as well as the the keyword search accuracy. Of course, we can compensate for the effects of churn by increasing the replication rate of posts or by querying more nodes in the network. Increasing the number of nodes which are contacted by the replicators, z_r, from 7.5 to 8.5, restores the accuracy to desired levels.

VI. Conclusions

This paper considered the design of a micro-blogging social network in an unstructured peer-to-peer network. Such a system could complement existing centralized systems, and provide a useful service to users concerned with censorship or anonymity. Our proposed system is based on the PAC search architecture and we designed the system for 95% accuracy. We decomposed the problem into two sub-problems - replication and retrieval, and then derived the optimal replication which would minimize the system bandwidth for a given network size. Our proposed method of replication achieves our required aim of rapid-yet-restrained dissemination and is also able to cope automatically with spikes in the data creation rate.

References

[1] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

[2] I. J. Cox, R. Fu, and L. K. Hansen, "Probably approximately correct search," in *ICTIR*, 2009.

[3] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "Peerson: P2p social networking: early experiences and insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, 2009.

[4] A. Loupasakis, N. Ntarmos, P. Triantafillou, and D. Makreshanski, "exo: Decentralized autonomous scalable social networking," in *CIDR*, 2011.

[5] T. Xu, Y. Chen, X. Fu, and P. Hui, "Twittering by cuckoo: decentralized and socio-aware online microblogging services," in *SIGCOMM*, 2010.

[6] R. Narendula, T. G. Papaioannou, and K. Aberer, "My3: A highly-available p2p-based online social network," in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*. IEEE, 2011.

[7] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, and D. R. Karger, "On the Feasibility of Peer-to-Peer Web Indexing and Search," in *IPTPS*, 2003.

[8] R. Ferreira, M. Ramanathan, A. Awan, A. Grama, and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," in *P2P*, 2005.

[9] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann, "Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search," in *SIGCOMM*, 2007.

[10] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2002.

[11] H. Asthana, R. Fu, and I. Cox, "On the feasibility of unstructured peer-to-peer information retrieval," *ICTIR*, 2011.

[12] H. Asthana and I. Cox, "Pac'npost: a framework for a micro-blogging social network in an unstructured p2p network," in *WWW*, 2012.

[13] N. Bailey, *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[14] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, and H. Sturgis, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th annual ACM Symposium on Principles of distributed computing*, 1987.

[15] "http://www.xinz.org/blog/how-big-is-twitter-some-stats/."

[16] M. Nekovee, Y. Moreno, G. Bianconi, and M. Marsili, "Theory of rumour spreading in complex social networks," *Physica A: Statistical Mechanics and its Applications*, vol. 374, 2007.

[17] M. Newman, "Spread of epidemic disease on networks," *Physical Review E*, vol. 66, no. 1, p. 016128, 2002.

[18] M. Jelasity, S. Voulgaris, R. Guerraoui, A. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, 2007.

[19] "http://www.sysomos.com/insidetwitter/."

[20] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *SIGCOMM*, 2006.