Retrieval of Highly Dynamic Information in an Unstructured Peer-to-Peer Network

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

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

ABSTRACT

We present a framework for the retrieval of highly dynamic information in an unstructured peer-to-peer network. Non-exhaustive search in an unstructured network is necessarily probabilistic, and we utilize the probably approximately correct (PAC) search architecture to determine the required replication rate for a document in order to guarantee a high probability of retrieval. Once this replication rate is determined, the problem becomes how to replicate a new document across the network to meet this requirement, without overloading the communication capacity of the network. To solve this, we model the problem as rumour spreading, and use techniques from this field to propagate new documents. Our document spreading algorithm is designed such that a document has a very high probability of being replicated to the required number of nodes, but the probability of spreading to fewer or more nodes is small. Apart from facilitating rapid and restrained dissemination, our proposed method also withstands sudden spikes in the data creation rate. We illustrate the utility of the framework in the context of a micro-blogging social network. However it could also be used to index dynamic web pages in a distributed search engine or for a system which indexes newly created BitTorrents in a de-centralized environment. Simulations performed on network of 100,000 nodes validate our proposed framework.

1. 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 posts each day, usually consisting of personal status updates, current events, news etc. [1, 2].

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. Furthermore, 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]. Finally, there have also been several outages of the centralized system, which has led to the entire service becoming unavailable[3]. Such concerns of cost, security, fault tolerance and privacy provide motivation for a decentralized distributed P2P architecture.

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 [3], 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 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.

Previous research into information retrieval (IR) in peer-to-peer networks has generally assumed that the information on the network is relatively static. The main contributions of this paper is to present a framework for a micro-blogging social network in an unstructured peer-to-peer network and in particular

- a framework for micro-blogging in an unstructured P2P network by decomposition of the problem into probabilistic search and restrained dissemination
- use of *transfer buffers* to enable restrained and rapid dissemination
- simulation results confirming our theoretical models

2. PRIOR WORK

P2P networks can be generally categorized into two classes, namely structured and unstructured networks. Structured networks, typically based on distributed hash tables (DHTs), bind data to designated locations within the network. The advantage of a structured architecture is that the query latency, proportional to the number of nodes a query must visit, is $O(\log n)$ where n is the number of nodes in the network. However this binding makes structured networks particularly susceptible to adversarial attack [4], and to churn. Furthermore, since the DHT needs to be kept up-to-date, structured networks are generally unsuitable for data which is highly dynamic.

Unstructured networks exhibit no such binding between data and nodes. As such, they are much less affected by churn, and are generally more resistant to adversarial attack. However, since a particular document being sought by a user can be anywhere in the network, the only way to

[1]http://en.wikipedia.org/wiki/Microblog

[2]http://en.wikipedia.org/wiki/Censorship_of_Twitter
[3]http://royal.pingdom.com/2007/12/19/twitter-growing-pains-cause-lots-of-downtime-in-2007/

LSDS-IR 2013 Italy

guarantee searching the entire collection is to exhaustively query all nodes in the network. This is, of course, impractical. To be practical, any search must only query a relatively small subset of nodes in the network. Thus, search in an unstructured P2P network is necessarily probabilistic.

Whilst there has been research on P2P information retrieval, it has mainly focused on structured networks [5, 6] or super-node networks [7] where a few stable high-capacity nodes are chosen to federate the traffic. Whilst these approaches provide various strategies for text retrieval, they do not address how a P2P network would cope with highly dynamic data. To our knowledge there has been no prior study of highly dynamic information creation and retrieval in unstructured P2P networks.

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 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. The probabilistic model can be extended to encompass other information retrieval requirements, as discussed in Section 2.0.2.

2.0.1 Search in Unstructured Networks

Early work on probabilistic search in unstructured P2P networks has its origins in the study of probabilistic quorum systems [8] to improve the availability and efficiency of replicated systems. Ferreira *et al.* [9] proposed the use of probabilistic quorum model to describe search in an unstructured P2P network. Given n nodes in the network, an object is replicated $\gamma\sqrt{n}$ times onto a random subset of nodes. A query is also sent to a random subset of $\gamma\sqrt{n}$ nodes. It can then be shown that the probability of finding the desired object of the query is at least $1 - e^{-\gamma^2}$.

BubbleStorm [10] provides the underlying network overlay required to implement a system where both data and query are replicated onto nodes, such as the one described by Ferreira *et al* [9]. The previous analysis assumed that an object/document is uniformly randomly replicated across nodes in the network. Other replication strategies are also possible [11].

2.0.2 PAC Search

The previous work on randomized search 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 [3, 12, 13].

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. If we are searching for a single, specific document d_i, then the probability of retrieving this document is given by

$$P(d_i) = 1 - (1 - \frac{\rho}{m})^z \quad (1)$$

where ρ is the capacity (number of documents indexed) on each node and m is the number of unique documents in the network.

In IR, it is more common to be interested in the top-k retrieved documents. In this case, the correctness of a PAC search is measured by *retrieval accuracy*. Let $\mathcal{D}$ denote the set of top-k documents retrieved when searching the full index, i.e. an exhaustive search, and $\mathcal{D}'$ denote the set of top-k documents retrieved when querying z nodes. Then the retrieval accuracy, a, is defined as $a = \frac{|\mathcal{D} \cap \mathcal{D}'|}{|\mathcal{D}|} = \frac{k'}{k}$ where k' denotes the size of the overlap of the two sets, i.e. $|\mathcal{D} \cap \mathcal{D}'|$.

The size of the overlap in the result sets, k' is a random variable drawn from a binomial distribution, and is given by

$$P(k') = \binom{k}{k'} P(d_i)^{k'} (1 - P(d_i))^{k-k'} \quad (2)$$

Since Equation (2) is a binomial distribution, the expected value of k' is $E(k') = kP(d_i)$ and the expected retrieval accuracy μ is

$$\mu = \frac{\mu_{k'}}{k} = \frac{k \times P(d_i)}{k} = 1 - \left(1 - \frac{\rho}{m}\right)^z = 1 - \left(1 - \frac{r}{n}\right)^z \quad (3)$$

where r is the number of nodes a file is replicated on, and n in the number of nodes in the network. The ratio, $\frac{r}{n}$ is referred to as the replication rate. Note that $\frac{\rho}{m} \equiv \frac{r}{n}$ holds only for *uniform* replication.

Equation (3) approximates to $1 - e^{-zr/n}$ [3]. The product of the replication rate, $\frac{r}{n}$, and the number of nodes queried, z, equivalent to $\frac{z\rho}{m}$, is the *sample index*, which is the ratio of the number of documents in a sample of z nodes, i.e. $z\rho$ to the number of documents in the global collection, m. We can utilize different combinations of the replication rate, $\frac{r}{n}$, and nodes queried, z, to arrive at the same accuracy. A more detailed analysis can be found in [3].

2.1 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. The theory of rumour spreading algorithms originates from the mathematical modelling of the spread of infectious diseases within a community [14].

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 [14] 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.* [15] 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*.

The study of information diffusion is not limited to computer networks; it has been studied in various different fields such as P2P recommendation [16] and viral marketing [17].

3. INFORMATION DISSEMINATION AND RETRIEVAL

Our goals are two-fold. First, a user should be able to retrieve the micro-blog posts[4] 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

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

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. Of this disk space, a proportion is utilized for storage of the posts, and the remaining proportion for indexing the stored posts.

When a post is published, it must be given some time to spread through the network before it can be retrieved. Since all nodes perform a *request* every s seconds, an appropriate value for the time a post is allowed to spread before becoming available for retrieval is s. Note, this is somewhat arbitrary.

The load on individual nodes can also be calculated as follows: Assuming that roughly the same number of peers $\left(\frac{n}{s}\right)$ issue a *request* at any given second, and each peer selects z random nodes for its request, the probability of being picked is simply $(\frac{z}{n})$, and therefore the number of requests any random node in the network is expected to receive per second is $\frac{n}{s} \times \frac{z}{n} = \frac{z}{s}$.

3.1 Node Behaviour

When a node queries other nodes, or is queried by another node, it transfers its most recent post, thereby spreading the post into the network. Unfortunately, sending only the most recent post to contacted nodes does not spread the post into the network at an acceptable rate. To increase the rate of spreading, we introduce a *transfer buffer*, $\mathcal{T}$, which stores a small fraction of the most recent posts that the peer has encountered.

To retrieve the posts Node $\mathcal{A}$ is *following*, it makes a *request* to z nodes every s seconds. A request contains

- the list, $\mathcal{L}_A$, of the user IDs Node $\mathcal{A}$ follows
- the most recent post created at Node $\mathcal{A}$
- the posts contained in the *transfer buffer* $\mathcal{T}_A$ of Node $\mathcal{A}$

When Node $\mathcal{B}$ receives a *request*, it sends back a *response*, which contains

- the most recent posts of all the peers specified in $\mathcal{L}_A$, which are found in its storage area
- the most recent post created at Node $\mathcal{B}$
- the posts contained in its *transfer buffer* $\mathcal{T}_B$

In addition to the *request* and *response*, when a post is created at a node, it immediately performs a *request* to z nodes. By performing a *request* at the time of post creation, the newly created post immediately starts to spread through the network.

To perform a keyword search, the list $\mathcal{L}$ is replaced by the keyword query in the *request*, and the *response* contains the posts matching the query, which are then merged and re-ranked at the originating node.

At the completion of a *request*, the responders store the contents of the requester's transfer buffer in the storage area. Similarly, the requester stores the contents of the responders' buffers in its storage area. After responding to a request or processing all the responses from the z queried nodes, each node reconstructs its transfer buffer.

3.2 Transfer Buffer

The goal of selecting posts for the transfer buffer is to continue spreading newer posts at the expense of older ones. Since each post is allowed s iterations to spread in the network, we consider only the posts which are s seconds old or younger. Posts are selected into the buffer with an exponentially decreasing probability which is a function of the age of the post: $P(sel) = e^{-t/\alpha}$, where t is is the number of seconds a post has been in existence and α is the dissemination parameter. The higher the value of the dissemination parameter, α, the higher the probability that an older post will be selected into the buffer and hence the larger the size of the node's transfer buffer, $|\mathcal{T}|$. 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 s seconds. Another way of describing this probabilistic selection process, is that at every iteration a node *stifles* a post with a probability $1 - e^{-t/\alpha}$.

4. ASSUMPTIONS

We assume that the interval between requests, s, is 30 seconds, which allows for a user experience that is similar to Twitter[5]. Equation (3) 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 2.0.2) is 95% and that the number of nodes queried is fixed at 25. 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. Both assumptions are arbitrary. However, an accuracy of 95% seems sufficiently high that most users will be satisfied with the performance. Note that 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.* The decision to query 25 nodes is also arbitrary and reflects a compromise. Querying a larger number of nodes will increase the probability of finding a post, but at the expense of network bandwidth and latency. We can rearrange Equation (3) to extract the replication ratio:

$$R = \frac{r}{n} = 1 - \exp\left(\frac{\log(1-\mu)}{z}\right) \tag{4}$$

Setting $\mu = 0.95$ and $z = 25$, we get $\frac{r}{n} = 0.12$, i.e. a post needs to be replicated to 12% of the network. The accuracy measure applies to both the retrieval of the *followed* posts as well as to searching the contents of all the posts in the network. Note, since each node makes a request every $s = 30$ seconds, 95% accuracy does not imply that a user will miss 1 in 20 of his/her followed posts. The accuracy is indeed 95% at the first request, but increases to 99.83% by the second request, and 99.99% by the third request, since making two requests to two sets of z random nodes is the same as making one request to $2z$ random nodes as long as the same information is requested.

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[6] encoding of a post, and, on average, 2 bytes per character. Thus each post requires 1 KB of 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. The allocation of 90% of the storage area (900 MB) to the storage of posts allows 9 million posts to be stored on each node, assuming a compressed post requires 0.1 KB [7]. Since the replication rate is 12%, the total

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

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

[7]Assuming 90% text compression when multiple posts are stored together in the storage area.

Table 1: Accuracy as a function of the dissemination parameter, α in a network of 100,000 nodes. Note that the keyword search is performed using BM25 with parameters set using a node's local statistics

α	Blog Post Retrieval		Keyword Search					
			@ 10		@ 20		@ 30	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
14.6	94.885	3.145	84.414	3.336	86.697	3.419	87.828	3.099
14.7	95.032	2.919	84.296	2.760	86.566	3.195	87.809	2.720
14.8	**95.457**	**3.230**	84.970	3.213	87.277	2.922	88.361	3.509
14.9	95.552	2.919	85.036	2.692	87.381	3.064	88.684	2.915
15.0	95.963	2.921	85.486	2.805	87.767	2.657	89.085	2.888

Table 2: Accuracy as a function of the size of the dissemination parameter, α, in a network of 100,000 nodes, when each node has access to global BM25 parameter values

α	Blog Post Retrieval		Keyword Search					
			@ 10		@ 20		@ 30	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
14.7	95.032	2.919	91.990	2.655	92.454	2.695	92.645	2.769
14.8	95.457	3.230	92.796	3.362	93.284	3.274	93.338	3.105
14.9	95.552	2.919	92.838	3.047	93.365	3.003	93.628	2.981

number of *unique* posts (m) that can be stored is 75 million[8]. Note that this number can be increased by reducing the replication rate and correspondingly increasing the number of nodes queried. Once full capacity has been reached, older posts will start to get replaced by newer ones in the nodes' storage area. This is probably an acceptable storage capability, and we note that search for historical tweets on Twitter is highly variable depending on the popularity of the tweet[9].

For the selected values of our parameters, on average any random node in the network would have to answer $\frac{z}{s} = 0.83$ requests per second or 50 requests per minute.

The system parameters described in this section allow for various combinations which are under the control of the designer. For instance, we could envisage a network where the required accuracy is 95%, with $s = 60$, and $z = 50$. In this case, the load on the nodes would be the same as our previous configuration, but the posts would only need to be replicated to 6% of the network.

5. SIMULATIONS

Our framework describes the network at a higher level (overlay) than the network topology. At the abstract level it can be considered to be a fully connected graph where only z random connection are active when a node makes a request, and where the messages travel directly to the target nodes. We implemented our framework at this abstract level, i.e. the network of nodes is fully connected. The network topology is outside the scope of this paper, and we refer the reader to [18, 19] for an extensive overview of information dissemination in various types of networks as well as to [20, 21] for efficient ways to perform P2P bootstrapping, membership, and random node sampling.

Note that since the number of nodes queried, $z = 25$, is small, we assume that each iteration is one second in magnitude, i.e. the latency is less than 1 second, and that iteration and second can be used interchangeably.

At each iteration, 25 new posts are created at random nodes in the network. This is the normalized rate of post creation for a network of 1M nodes based upon Twitter's average of 2,300 tweets per second[10] and a 100 million *active* user base[11]. We also introduced a spike in the post creation rate where 125 posts are created at random nodes temporarily. 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 5 posts per second. A blog post creation rate of 25 posts per second in a network of 100,000 nodes corresponds 10 times Twitter's average.

We used the TREC Micro-blog Tweets2011 corpus[12] 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 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. Experiments are reported for the cases where the parameters of the BM25 algorithm are set based on (i) the statistics of a node's local collection, and (ii) the statistics of the global collection of tweets. The querying node received the ranked lists from each of the z queried nodes and merged them based on the BM25 scores each node provided. The baseline search of the centralized database is performed using BM25 and the statistics of the global collection.

We assumed each node follows 10 randomly chosen user ID's. This is a rather arbitrary number but seems reasonable based on the following three Twitter statistics[13]: 1) Only half the registered users follow 2 or more people, 2) Only about 10% of accounts follow more than 50 people, and 3) 92.4% follow less than 100 people. To evaluate the

[8]Since, as previously stated, the replication rate ($\frac{r}{n} = \frac{\rho}{m}$), where ρ is the number of documents stored on a node and m is the total number of unique documents in the network. Thus, $m = 75$ million.

[9]Tweetreach (http://help.tweetreach.com/entries/140464-how-far-back-can-tweetreach-search) estimates that the oldest retrievable tweets are between 4 and 7 days old.

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

[11]Twitter's user base is estimated to be around 200 million users. However a fraction of the user accounts are dormant and another fraction are fake accounts. The *active* user base is estimated to be only half of the total user base [22].

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

[13]http://www.sysomos.com/insidetwitter/

Table 3: Replication rate (% of network) as a function of the dissemination parameter, α in a network of 100,000 nodes.

α	Overall		Fraction of posts replicated to between x-y% of the network						
	Mean	Std Dev	0-5	5-10	10-15	15-20	20-25	25-30	>30
14.6	12.139	5.862	9.729	30.792	30.147	18.864	7.632	2.355	0.481
14.7	12.269	5.946	9.540	30.666	29.755	19.029	8.036	2.398	0.576
14.8	**12.622**	**6.051**	8.405	30.013	29.678	19.519	8.887	2.725	0.774
14.9	12.899	6.118	8.038	28.069	30.322	20.478	9.259	2.957	0.877
15.0	13.251	6.223	7.159	26.970	30.305	20.705	10.503	3.257	1.100

Table 4: Replication rate (% of network), blog post retrieval accuracy, and keyword search accuracy as a function of dissemination parameter value of $\alpha = 15.7$ and various maximum transfer buffer sizes, $|\mathcal{T}|_{max}$.

$\|\mathcal{T}\|_{max}$	Replication		Blog Post Retrieval		Keyword Search					
					@ 10		@ 20		@ 30	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
∞	15.124	7.003	96.910	2.580	87.010	2.778	89.442	2.558	90.664	2.323
20	12.233	6.342	94.490	2.947	84.069	3.032	86.052	2.805	87.090	2.705
25	**12.882**	**6.546**	**95.126**	**2.731**	84.982	2.505	86.897	2.948	87.921	2.748
30	13.190	6.568	95.421	2.650	85.396	2.467	87.397	2.510	88.600	2.410

effect of the dissemination parameter, α, we performed 10 simulations for each value of α. Each simulation lasted for 600 iterations, so that each node in the network would make at-least 20 *requests*. At each iteration of each simulation, we recorded the number of posts in each node's transfer buffer ($|\mathcal{T}|$). In our simulations, when a node made a *request* to retrieve its required posts, it also included a keyword search within the request. For every node which made a *request*, at an iteration, we recorded the accuracy, as defined in Section 2.0.2 for both post retrieval and keyword search. The accuracy values reported are averaged over all iterations of all simulations of each dissemination parameter (α) value. To calculate the dissemination of posts, we recorded each post's replication at the end of each simulation.

Table 1 illustrates the accuracy for retrieval of *followed* posts and retrieval based on keyword search. 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.

We note that the accuracy of keyword search is lower than post retrieval (following) accuracy. This is due to the fact that the BM25 ranking algorithm requires parameters (e.g. the number of documents in the collection, the number of documents matching a query term) that are calculated from each node's local collection. These values differ from the corresponding values of the global collection. To confirm this, we re-ran our simulations and made the global values of the BM25 parameters available to all nodes. Table 2 illustrates the keyword search accuracies with global BM25 parameter values. The values are much closer to the expected 95% accuracy. We utilized BM25 to demonstrate the viability of keyword search within the PAC search architecture. However, BM25 may, on its own, not be the best method for ranking very short documents, such as tweets, and query independent features, such as user influence, need to be implemented.

In our network of 100,000 nodes, a dissemination parameter, α, value of 14.8 provides the required 95% retrieval accuracy. Table 3 shows the replication rate of the posts. As the value of α increases, the percentage of posts which spread to a higher percentage of the network increases, as expected.

We observed that the mean and maximum buffer sizes were approximately 3.5 and 22 blog posts respectively when the system was generating 25 posts per second. However,

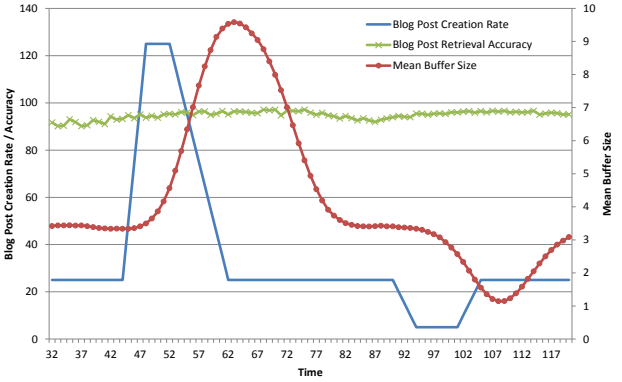

Figure 1: The accuracy and mean values of the transfer buffer, $\mathcal{T}$, in a network of 100,000 nodes and a dissemination parameter value of $\alpha = 15.7$ with $|\mathcal{T}|_{max} = 25$.

during a spike, when the blog post creation rate increases to 125 posts per second, the maximum buffer size increased to approximately 130 blog posts. This may overwhelm the few nodes which have very large buffer sizes during a spike. We, therefore, repeated our simulations with a set maximum transfer buffer size, $|\mathcal{T}|_{max}$, where the posts are selected into $\mathcal{T}$ from the storage area using the same probabilistic method until $|\mathcal{T}| = |\mathcal{T}|_{max}$.

Table 4 illustrates the effect of an upper-bound on the size of $\mathcal{T}$ on post replication, retrieval and keyword search. We can achieve the required 95% accuracy with $|\mathcal{T}|_{max} = 25$. However we need to increase the dissemination parameter, α from 14.8 to 15.7. Figure 1 shows the post creation rate, retrieval accuracy, and the mean buffer size for a network of 100,000 nodes, $\alpha = 15.7$, and $|\mathcal{T}|_{max} = 25$. The figure demonstrates that setting a maximum buffer size does not impede the ability of the system to respond to a spike or a lull in the post creation whilst maintaining the required 95% accuracy. The important point to note is that the accuracy remains at around 95% even when there is a spike or lull in the post creation rate, and our proposed system can automatically deal with spikes which are encountered in actual centralized systems such as Twitter.

6. CHURN

Churn is an important factor in any P2P application. We refer the reader to [23] for an overview of churn in P2P networks. Typically, a node may join the network for periods (mean session lengths) from 1 minute to 1 hour, depending on the type of P2P application, e.g. Gnutella, BitTorrent. Further, it has been observed that the network generally

Table 5: The transfer buffer size, $|\mathcal{T}|$, blog post retrieval accuracy, and keyword search accuracy @10 as a function of the dissemination parameter, α, for a network of 100,000 nodes under heavy churn and a maximum buffer size of $|\mathcal{T}|_{max} = 25$.

	$\|\mathcal{T}\|$		Blog Post		Keyword @ 10	
α	Mean	Std	Mean	Std	Mean	Std
15.7	4.757	0.356	92.526	2.682	84.204	2.008
17.2	6.310	0.407	95.367	2.739	87.378	2.526
17.3	**6.374**	**0.494**	**95.460**	**2.554**	**87.470**	**2.515**
17.4	6.490	0.266	95.524	2.655	87.601	2.474

consists of a small portion of highly stable nodes with the remaining peers exhibiting high turnover with session times following an exponential distribution.

We tested our framework with a high level of churn by setting the mean of the session time exponential distribution to just 60 seconds. We initiated a network of 100,000 nodes with a maximum transfer buffer size $|\mathcal{T}|_{max} = 25$, a creation rate of 25 posts per second, and 50,000 active nodes. Churn was then induced into the simulation. New nodes join with a Poisson arrival rate to balance out the exponential exit rate [10].

Table 5 illustrates the effect of churn on the accuracy of followed post retrieval and keyword search. As expected, the net effect of churn is to reduce the post retrieval accuracy as well as the the keyword search accuracy. The dissemination of the posts with $\alpha = 15.7$, which was previously sufficient to obtain our desired retrieval accuracy (Table 4), is now reduced by heavy churn. 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. We can observe from Table 5, that increasing the value of the dissemination parameter, α, from 15.7 to 17.3 restores the accuracy to desired levels.

7. CONCLUSIONS

This paper considered the design of a micro-blogging social network in an unstructured peer-to-peer network. The general problem is one of probabilistically retrieving highly dynamic information, and therefore the solution is applicable to indexing dynamic web pages in a distributed search engine or for a system which indexes newly created BitTorrents in a de-centralized environment.

We analyzed the problem as into one of rapid yet restrained dissemination and this problem is solved using our proposed transfer buffers. Simulations of a 100,000 node network, supporting a normalized post creation rate of magnitude 10 larger than Twitter, provided empirical support to the theoretical analysis and we were consistently able to achieve the required 95% accuracy. Importantly, our proposed system was able to adapt to spikes in the post creation rate. It is worthwhile emphasizing that 95% accuracy does not imply that a user will miss 1 in 20 of his/her followed posts. As each node makes a request every s seconds, the accuracy is indeed 95% at the first request, but increases to 99.83% by the second request, and 99.99% by the third request. Note, the second and subsequent requests are not redundant, but in fact necessary, because they facilitate the dissemination of posts via the buffers of the request and the response.

Apart from churn, P2P networks are also afflicted by the problem of malicious nodes which *censor, spam* or *poison* results or try to disable the network via *flooding*. Various solutions have been proposed to identify and eliminate malicious nodes [24]. Nodes can also be encouraged to behave responsibly and contribute fairly [25]. We plan to investigate these solutions within our proposed framework as part of our future work.

In our simulations, each user follows 10 other randomly chosen users and blogs are created randomly over the network. Analysis of Twitter shows that the blog creation rate of individual users is not constant, but follows a power law distribution, as do the *follower* and *followed by* statistics. We plan to incorporate these statistics into our framework as part of our future work.

Acknowledgment

We are thankful for the comments and suggestions provided by the referees, which has helped in improving this paper.

8. REFERENCES

[1] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD on Web mining and social network analysis*. ACM, 2007, pp. 56–65.

[2] 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.

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

[4] G. Urdaneta, G. Pierre, and M. Steen, "A survey of dht security techniques," *ACM Computing Surveys (CSUR)*, 2011.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer, "Minerva: collaborative p2p search," in *VLDB'05: Proceedings of the 31st International conference on Very Large Data Bases, Trondheim, Norway*, 2005.

[6] O. Papapetrou, W. Siberski, W. Balke, and W. Nejdl, "Dhts over peer clusters for distributed information retrieval," in *Advanced Information Networking and Applications.*, 2007.

[7] J. Lu, "Full-text federated search in peer-to-peer networks," in *PhD thesis, Carnegie Mellon University*, 2007.

[8] D. Malkhi, M. Reiter, and R. Wright, "Probabilistic quorum systems," in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, 1997.

[9] R. Ferreira, M. Ramanathan, A. Awan, A. Grama, and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," in *Proceedings of the Fifth IEEE Conference on P2P Computing*, 2005.

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

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

[12] I. J. Cox, J. Zhu, R. Fu, and L. K. Hansen, "Improving query correctness using centralized probably approximately correct (pac) search," in *ECIR*, 2010, pp. 265–280.

[13] H. Asthana, R. Fu, and I. Cox, "On the feasibility of unstructured peer-to-peer information retrieval," *Advances in Information Retrieval Theory*, pp. 125–138, 2011.

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

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

[16] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, and M. Reinders, "Tribler: a social-based peer-to-peer system," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, 2008.

[17] J. Leskovec, L. Adamic, and B. Huberman, "The dynamics of viral marketing," *ACM Transactions on the Web*, 2007.

[18] 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.

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

[20] 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.

[21] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

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

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

[24] D. Wallach, "A survey of peer-to-peer security issues," *Software Security-Theories and Systems*, pp. 253–258, 2003.

[25] I. A. Kash, J. K. Lai, H. Zhang, and A. Zohar, "Economics of bittorrent communities," in *WWW*, 2012.