1 Introduction

Social networks have been receiving increasing attention throughout recent years, specifically due to the success of sites like Facebook, Twitter, Youtube, Flickr, etc. As their popularity has increased, several issues have arisen revolving in general around centralized social networks as a type of social networks.

First and foremost is the issue of privacy. All centralized social network sites require the user to upload the content they will share to their site, which typically involves releasing of ownership of content from the user to the service provider. Furthermore, there are the privacy statements which in many cases are difficult to read and actually rarely read by users. Also, there are the laws in specific countries requiring service providers to keep user data for several years even after the user has deleted his account with obligation to provide the data to the authorities upon request.

Another question that arises with today’s social networks is scalability with profitability. One example in this case is the video service Youtube which started in 2005 and was acquired by Google in 2006. Official financial numbers have not yet been published by Google for Youtube’s operation, but statements from Google officials reveal that it was still not profitable until the end of 2010.[1]

Furthermore, there is the issue with availability. Hypotetically if Facebook or Youtube or any other social network were to be shutdown at any point in time, the social networks and the services provided will dissappear leaving their users the only option to migrate to a different service, which might be very difficult if at all possible.

Having these issues in mind, questions naturally arise about the possibility for introducing distributed decentralized social networks. Distributed social networks in general allow the users complete control over their content with respect of how and who they will share their content to. Furthermore the scalability and availability in this case does not depend on the profitability of the network, but will depend on the quality of the implementation of the system. Since some of the users can not afford to host a node by themselves, in most of the cases these social networks offer the possibility for a node to host content

for multiple users, which for the users might present a friend’s server, or trusted ISP, etc.

However, distributing the social network doesn’t come without a price. Many of the features that are straightforward in a centralized scenario are difficult to implement efficiently in a distributed network, if they are at all possible to be implemented. Some examples are detection and handling of misuse in the network, distributed authorization and authentication, aggregation and analysis of data, etc. Furthermore, there might appear problems with latencies or inconsistent performance as the network scales.

The paper [LNT11] discussed in this report introduces eXO, a decentralized autonomous scalable social network. eXO offers a fully decentralized social network with the ability to efficiently index and search globally for top-k users and content based on metadata information. The architecture of eXO may also provide content replication in a similar manner as P2P networks. eXO is primarily based on a distributed hash table (DHT), on top of which it adds methods for efficient indexing, search and retrieval of users and content in the social network scenario.

This report is organized as follows: first in section 2 we will give a short background on distributed hash tables and the implementation of it used in eXO. Later in section 3 we will lay out the most significant aspects of the architecture of eXO. Then in section 4 we will present the experiments and results obtained in the paper and finally we will conclude with general insight into the method and the paper and related work in sections 5 and 6.

## 2 Background

### 2.1 Distributed Hash Table (DHT)

As we have mentioned, at the core of eXO runs a distributed hash table [RD01], for which purpose they have used an existing open-sourced implementation [2].

The DHT in Freepastry allows for communication between any two nodes in the network based solely on the information of (unique) network node IDs. The routing in this case toward the destination node is based on the similarity of the node IDs. Each node keeps a routing table of nodes, such that given an ID $a$ the node can return a node whose ID is closer to $a$ than its own ID. The implementation of Freepastry is such that the size of the routing table for each node and the number of messages required for one communication are both in $O(\log(N))$, where $N$ is the total number of nodes in the network.

As the actual routing in Freepastry only depends on the similarity of the queried ID to the destination ID, this fact can be used to also route to a certain node only by knowing an ID that is closest to the node’s ID in the network. This feature is the backbone of the global search of eXO which will be discussed in section 3.

Another feature provided by DHTs is that content can be replicated across nodes with adjacent IDs, meaning that following a failure from a certain node, the nodes that are adjacent to it in the ID space can take over its responsibility.

---

1 [www.freepastry.org](http://www.freepastry.org)
3 Architecture

The most important aspect of eXO is that it is a fully decentralized autonomous social network, on top of which it offers an efficient search for users and content. eXO also offers increased availability of content through replication, which has significant impact on the usage of the system by the users as discussed later. Furthermore eXO provides the most important basic functionalities of creating friendships in the network and tagging other users’ content. The backbone of the architecture of eXO is comprised by the nodes that communicate in a fully decentralized manner, all of which have the following three roles in the network: resolving user requests - which is the actual web application and interface to the users, network storage - for greater availability of content and finally the role of a storage for global index of users and content and resolution of the respective global queries.

3.1 Terminology

• user is an account or account holder in the social network who can upload content, add and accept friends, manage access rights, etc.
• node is a physical instance of a member in the distributed network which is responsible for a user. This report, like the paper itself, will assume that there is a one-to-one relation between users and nodes. Note, however, that there is no restriction in the eXO’s architecture that prevents a node from hosting multiple users.
• content is a text, image, video, sound, url or other type of data that can be shared by a user
• term is a user specified word that describes the user itself or a content shared by the user. Terms are used to index users and content
• user profile unlike its meaning in the popular social networks’ terminology, in eXO it represents a set of terms that describe a given user. Example: \{peter, johnson, newyork, usa, databases, phd\}
• content profile respectively represents a set of terms that describe a given content. Example: \{video, football, highlights, eurosport\}
• query is consisted of two sets of terms \( q = q_c \cup q_u \), such that \( q_c \) is the content part of the query and \( q_u \) is the user part of the query. Given that, there are two kinds of queries: content queries and user queries
• content query is a query meant for searching for content. \( q_c \) in this case describes the content to be searched for, and \( q_u \) is an optional set of terms describing the owner user of the searched content.
• user query is a query meant for searching for users. In this case \( q_c = \emptyset \), and \( q_u \) is set of terms describing the user to be searched for.
• tag is a word attached to a user or a content. Tags unlike terms can also be attached by other users in which case they are called socialtags. The other difference from terms is that tags are not used to index users and
Table 1: Example index data kept at the node responsible for the term *barcelona*

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Owner UID</th>
<th>content profile</th>
<th>user profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>zs.38sv</td>
<td>25AC59</td>
<td>barcelona, sport, video</td>
<td>darko, makreshanski, skopje</td>
</tr>
<tr>
<td>al29zkd</td>
<td>234AD2</td>
<td>match, barcelona, zurich</td>
<td>john, doe, zurich</td>
</tr>
<tr>
<td>zlx,38s</td>
<td>913DEF</td>
<td>university, barcelona, logo</td>
<td>official, ub</td>
</tr>
</tbody>
</table>

User part:

<table>
<thead>
<tr>
<th>Owner UID</th>
<th>user profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>A48DC9</td>
<td>lionel, messi, barcelona, argentina</td>
</tr>
<tr>
<td>28DC27</td>
<td>juan, jose, carlos, barcelona</td>
</tr>
</tbody>
</table>

content due to the apparent overhead caused to the index by frequent tagging.

### 3.2 Global search for users and content

eXO provides a global decentralized search of users and content. This search is based on distributed key-value index, where keys are terms and values are user and content profiles together with node IDs of user and content owners. The index is distributed such that for every term in the global term space there is a responsible node in the network. A node with an ID $a$ is said to be responsible for a term $t$ if $hash(t)$ is closest to $a$ among all the other node IDs in the network.

A node responsible for a certain term $t$ will keep all the content and user profiles in the world that contain the term $t$. An example for the index data that is kept at a node is shown in Table 1. For the nodes that wish to update or query for a certain user or content, this means that they would have to contact all the nodes that are responsible for all the terms in the user and/or content profile. As a note, in the case of updates it also holds that the node has to contact nodes responsible for terms that were not part of the update, so that it updates the corresponding entries for its user and/or content profiles.

In the following part we will examine what happens in the system when a node issues a query into the system. We will explain the procedure only about content queries, as the user queries can be considered as a subtype of content queries.

When a node wants to query $q = q_c \cup q_u$ for obtaining the top-$k$ content results whose profiles contain terms $q_c = \{q_{c1}, ..., q_{cm}\}$ and whose owners’ profiles contain the terms $q_u = \{q_{u1}, ..., q_{un}\}$, it will:

1. send the same top-$k$ query $q$ to nodes responsible for all the terms in $q$
2. receive the $|q| \cdot k$ results, merge them, and obtain the top-$k$ from them if necessary
3. contact the respective content owner nodes, to retrieve the content data

When a node responsible for a certain term $t$ receives a top-$k$ content query $q.t \in q$ it does the following procedure
1. calculate the ranks for each entry in its index data for the term \( t \) with respect to the query \( q \)

2. order entries by the calculated ranks and return the top-\( k \) entries

The rank of a content with respect to a query is the cosine similarity between the term vector for the content and content’s owner user profiles and the term vector for the query. A term vector in this case is a vector that corresponds to a set of terms and holds values for all the terms in the global term space. Naturally the set of terms does not contain all the terms in the term space, but just contains a small subset of it and respectively most of the values in this term vector are zero. Consequently, all the non-zero values in the term vector are the values that correspond to the terms that are contained in the set of terms. In this case, the non-zero values depend on whether the query is a content query or user query. In the case of a content query, a value for a term \( t \) in the term vector for a content \( c \) is 1 if the term is contained both in the content profile of \( c \) and in the \( c \)'s owner’s user profile. If the term is contained in either the content profile or the user’s profile than the value is 0.5. The same procedure is also applied to the term vector for the query. In the case of user queries, the value is just 1 if the term is contained in the user profile (or in the query).

Notable thing with the rank calculation is that the rank calculated in this case is global, since the nodes do the calculation based on the complete content and user profiles. This implies two things. First is that even if a certain content appears in the results of two or more nodes for the same query, then the calculated rank for the content will be the same for all received results. Second the rank is final, meaning that it requires no further computations after it is received by the node that issued the query.

The paper also mentions that this global search can easily be extended to rank results of content that are owned by friends or users that are friends to our friends higher by including the list of friends along with the user and content profiles in the index entries. However, they don’t further investigate nor discuss the overhead and scalability of this feature.

### 3.3 Retrieval of content

Since the results obtained from the nodes responsible for the terms in the query can only return the user and content profiles and user IDs, the node that issued the query must subsequently contact all the nodes that are owners of the users or content in the obtained list of results.

The paper states that the architecture of DHT can be utilized such that among the \( |q| \cdot k \) results obtained from the \( |q| \) nodes, only the \( k \) closest results with respect to the number of hops needed to reach the corresponding nodes. The paper, however does not explain this algorithm in detail nor it provides evaluation on it.

### 3.4 Tagging

Tagging is a procedure in eXO where a user can provide additional descriptive words to a user or a content. Tags are different from terms and respectively the user and content profiles, meaning that they are not indexed in the global index, and thus users and content are not searchable based on the tags they
possess. Tags are not included in the index because of efficiency reasons. The paper notes, however, that the owner can choose manually to include certain tags in the corresponding user or content profile.

Although not used for indexing purposes, the tags can be used for query expansion. This means that after a node has received the results from its query, it can contact the corresponding owner nodes and retrieve the respective tags for the users or content and issue a new query based on the tags.

Tags are kept always on the tagee’s side however node IDs of the tagger and tagee are kept between each other for cross-reference.

3.5 The public and private social networks
The paper discusses that as a social network system, eXO provides two types of social networks. The first is the global public network that is derived from the globally accessible index on users. The second is the private social network, which is comprised of the users that are in your friends list, or in your friends’ friends lists or in your friends taggers or taggees. Thus the private social network can be defined as the list of node IDs that you have access to, apart from the global index.

4 Evaluation
For evaluation purposes the paper uses two datasets. The first is a dataset from Flickr containing 49832 content items from 1000 users containing a total of 264379 terms, and the second dataset is from crawling Facebook where they reported to crawl 36261 user profiles containing collection of user interests and tastes. For the Facebook dataset, they have used 8000 users per run, and for both experiments there is one user per node in the network.

The paper reports three experiments. First they have tested the node hit load distribution among the nodes for randomized global searches. Second they have tested the scalability of the global search by measuring the number of messages exchanged with respect to the number of nodes in the network and the amount of terms in the query. Third, they have tested the average bytes per query result by varying the $k$ parameter in the top-$k$ queries for both datasets. In this report we will discuss and present only the first two experiments.

The results of the first experiment are shown in Figure 1 and Figure 2 for the Flickr and Facebook datasets respectively. Each figure consists of two images, where the image on the right shows the distribution of user and content profiles per term, and the left image shows the per node percentage of node hits out of total node hits for all the queries issues during the experiment. The paper does not provide quantitative analysis of the results, thus briefly stating that the load distribution between nodes is fairly balanced with the exception of certain peaks as it is shown in the graph.

The result of the second experiment is shown in Figure 3. Here, expectedly, the number of messages for a query grows logarithmically with the number of nodes in the network and linearly with the number of terms in the query.

While both experiments give some insight on the scalability of the system, we would expect a more thorough analysis of the system especially for the scalability part. For the first experiment, while the noad hits load distribution...
gives some picture, we would expect that it would be important to test the actual scalability of the global search itself, and specifically the increase of the computation costs for a node while computing the rank of the content items and users for a query. The reasoning for this is simple, while computing the rank for a term that has e.g. 10000 content profiles is not a problem for a normal computer, computing the rank of a term in a network of size of millions where a popular term might be contained in millions of content metadata would probably be a significant overhead for the node, and plus greatly increase the latency of the query.

5 Related work

As the privacy issues were raised for centralized social networks, the interest over distributed social networks has significantly increased. The first commer-
cial success story is Diaspora[3], a project that even in pre-production phase managed to raise over $200k in several weeks. Diaspora, has actually, developed into a distributed social network with only simple social network features, not including search. From the many others projects that currently exist and are being developed, we would distinguish Friendika[4], a distributed social network implementing the DFRN protocol[5], provides global search, though centralized, together with encryption, remote authentication, etc.

As for the academic side of related works, substantial amount of work exists on decentralized global search in DHTs[BMT+05], to which eXO extends by providing an efficient method for indexing and searching in social networks scenario. Furthermore, there are works related to searching in social networks based on gossiping protocol[BBG+10] and in general for distributed top-k query processing [CW04][MTW05].

6 Conclusion

eXO provides a decentralized social network based on distributed hash tables. The novelty of eXO is that it provides an efficient global search of users and content in a fully decentralized scenario. Like other applications based on DHTs eXO can also provide data replication, which can greatly influence the availability of content such that it might even be usable in cases where nodes correspond to personal computers of users, which are neither as available nor have the capacity to host content as specialized hosted servers. This, could even more

Figure 3: Number of messages versus network size and query size

3 join diaspora.com
4 project.friendika.com
5 dfrn.org
increase the level of control of the users over their content.

The paper, however, fails to provide a thorough analysis on the scalability of the method especially with respect to the computation overhead for the global search and the latency.

Furthermore while the paper notes that the content replication techniques can be utilized by eXO as well, they don’t provide any analysis on the eventual robustness of the method and more importantly does not discuss the issues and solutions that arise with privacy, authentication and authorization with the replication of the content across nodes. This is left as a future work along with other social networking features such as news feeds, friend groups, one-way relationships etc.

References


