Talash : Friend Finding In Federated Social Networks [*]

Ruturaj Dhekane[†]
Indian Institute Of Technology, kanpur
ruturaj@cse.iitk.ac.in

Brion Vibber[‡]
StatusNet
brion@status.net

ABSTRACT

In large online social networks, Friend Recommendation has evolved into an interesting problem. We try to find known acquaintances and new interesting friends on a Federated Social Network (FSN) , using StatusNet as our platform. FSNs are decentralized networks on the internet which can interoperate using the OStatus Suite of protocols. Friend finding on these networks is hard because we do not know the existence of other social networks or Users. We show how Linked Data representation like FOAF can solve this problem.

We devise a model for the Federated Network centered around a User and use it to define the problem of Friend Finding. The solution uses two phases, first known as *Quick Connect* which tries to find old acquaintances. The second phase, *Delayed Connect* uses the Social Graph of Users to find prospective friends. We show how the FSN information centered around a User can be extracted from FOAF entries and generate new recommendations. We shall illustrate the working of *Talash* as a part of StatusNet. We experimented on the existing FSN and collected feedback from its Users. The results are encouraging and open new avenues for Friend Finding on the Internet.

To the best of our knowledge, this is the first study of Federated Social Networks and the problem of Friend Finding in them.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and networks; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Design, Performance

[*]Part of this work was done as Google Summer Of Code 2010 project for StatusNet Inc. Part of this was also done at Indian Institue of Technology, Kanpur, as a part of authors masters program.

[†]Student of Computer Science And Engineering at Indian Institute Of Technology, Kanpur, India.

[‡]Senior Software Architect at StatusNet.

Keywords

Friend Recommendation, Federation, Social Networks

1. INTRODUCTION

Online social networks have evolved over the past few years and have been interest of research for the community. Social graph analysis has opened new avenues for better user experience and expansion of these networks. The term social networking is now synonymous with the various activities such as commenting, replying, direct messaging, like/faving, updating status etc.. Typically an Online Social Network(OSN) for a particular User has three parts to it [4] . Every User owns a Public or semi public profile within a bounded system. Secondly, it displays a list of Users with whom they share a connection, either one way or two way. Thirdly, the dynamism in the OSN is due to the actions of the User to view, traverse and interact with this list of connections.

Popularity of OSN sites like Orkut, Facebook, Twitter has generated huge amount of data about the various interaction of Users on the Internet. Social network analysis deals with study of these networks and their properties[13]. Social networks fit a scale free model, have small world properties and show a community structure [14] [15].

Many forms of social networking have existed before OSN's. Interaction on email within an organization has been an area of interest[3]. A drawback with all these OSN's is the bounded environment in which people need to interact. A public profile on one OSN means that he can only interact with other User's on that OSN and cannot interact with Users from external OSN without explicitly registering a new profile on it. The Linked data initiative however envisage a more open and interconnected social networking experience.

1.1 Federated Social Networks

A Federated Social Network (FSN) tries to break the boundaries of the specified system in which the User interacts. A User owns an online profile page (website) on which he describes himself and his connections. The connections, or friends list is described in a machine readable format [25], hence any change in his connections can be made easily[28] [20][29]. The advantage of Federated Social Network is that each User controls his presence on the internet and can interact with other Users using a set of protocols such as the OStatus Suite [21][27]. It allows interoperability between different OSN's and give a distributed control to their owners. This does not bind any User to fixed rules like those in Online Social Networking Services (OSNS). StatusNet[10] is an open source microblogging [18] software that provides such a facility to the Users. StatusNet was chosen for experimentation over other nascent projects like Thimbl, Appleseed, Diaspora, OneSocialWeb and Elgg.

1.2 Contributions

In such a federated scenario we study the problem of friend recommendation. Since each individual exists on different websites, it is very difficult to find a specific individual on the Internet that you already know, unless you know the exact website on which he exists. Once we know the website that profile can be accessed.

In this work, we model the Federated Social Network of a User in the form of a graph. We give the problems faced by a system for Friend Finding in this setting. Initially we only consider finding those friends that a User already knows beforehand. Further we design an automated friend recommendation system for the Users. We show how the scale free nature of a social network affects this recommendation system. We then exploit the properties of the social graph centered around a User to reduce the overheads in recommending new and finding interesting friends for a User.

The remainder of paper is organized as follows. Section 2 contains a review of technologies used, a quick overview of the OStatus Suite of protocols and Social Graph API. Section 3 builds a model of social graph and we formally define the problem of Friend Finding. Section 4 describes our approach to finding friends in federated social networks. We give a simple method to find new acquaintances on the Internet, or those friends that, you did not know, already existed. We give the problems faced by this system and how the nature of graph centered around a User in Federated Social networks can be used to leverage the drawbacks. Finally we provide the experimental results to show the success of our method.

This system named *Talash* was built for the StatusNet software and the results presented here are feedbacks from the users of Identi.ca [9]. To the best of our knowledge, this is the first work that focuses on Federated Social Networks and the problem of Friend Finding in them.

2. PRELIMINARIES

2.1 Federated Social Network

To define a Federated Social Network we break the phrase into two parts and try to infer its collective meaning. The term Social Network here is synonymous to the interactions among individuals with each other. Since we are studying OSN's we call the participants of this network as Users. Every User has an online presence in the form of a web-page, also known as the Profile Page. That presence may be either public or access controlled. Many different systems have been developed for securing privacy issues on OSN's[34][5]. If it is public, any one can access and view the information available on it.

Typically, Users sign up on Online Social Networking Service (OSNS) e.g. Facebook, Orkut for networking with friends and family. All the mechanism of social network interactions are governed by and limited to the regulations of that online service. There are other Users signed up on the same OSNS with whom hhe can interact. Usually Users on one social networking service cannot interact with a User on another social networking service.

StatusNet is a microblogging platform that enables a User to set up his own public profile. The activities of this User are now governed by his own regulations. He is not under any central authority or social networking service. The StatusNet software provides a set of protocols to interact with other independent profiles on the Internet. Since each of these profiles is federated (individually controlled and not under a central authority) in its activities and regulated by the User's themselves, we call this system a **Federated Social Network**.

In this paper we study the task of Friend Finding on this federated social network. In the following sections we give a brief overview of the OStatus Protocol suite and a centralized index of the Social Network on the internet - Social Graph API.

2.1.1 OStatus Protocol Suite

Previously known as the OpenMicroBlogging [17] Protocol, OStatus is an open source specification for interoperability between various sites. This allows various Users on these sites to interact with each other. The interaction is in the form of subscriber-subscription, updating status, repeat updates of your connections and mark them as favorite. OStatus is a suite of protocols which give the specification for such interoperability. There are mainly four protocols in the suite as described below.

WebFinger: an open protocol to identify Users with their email addresses [32]. This allows OStatus Users to refer to other Users as someuser@example.com.

PubSubHubBub: an open protocol based upon Publish Subscribe architecture [23]. It allows a publisher to distribute content to its subscribers by using an intermediate hub. When a hub server receive updates, it multicasts the new or changed content to all registered subscribers.

Salmon: [22] an open protocol to let information like comments on a post to flow from subscribers to the publishers. When the information reaches the source of the post it is re-published by the original content creator so that it reaches back to all its subscribers.

Activity Streams: interactions of a User with his social network are published to his subscribers and appear as a stream of activities [30].

Any OSNS can independently develop a website which uses OStatus Protocols and can become a part of the FSN. Any User who has OStatus enabled on his profile page can subscribe to or be subscribed by other Users on the FSN and we call such profiles *OStatus Subscribable*.

2.1.2 FOAF and Social Graph

FOAF [28] [20] stands for Friend Of A Friend is specification to describe a User and a list of his connections. The FOAF is described using the Resource Description Framework [25] and stored in a machine readable format. The machine readability feature powers the automation of friend finding exercisee in this work. Users can define their own connections by describing their links with other Users and define a relationship between them.

The social interaction of individuals can be modeled as a graph, with individuals being nodes and the interactions being edges if one User interacts with the other. For our problem we look at the social graph generated by the social interactions of Users on the internet. There are millions of Users on the internet that are part of various Online Social

Networks. We study the graph whose nodes are public profiles and those that declare their connections publicly. Former work [26] has described how FOAF data can be used to develop Social Graphs of OSN. FOAF can be used in a decentralized manner to declare a Users social connections. Publicly declared FOAF's can be parsed to create a social graph. Thus FOAF creates a base structure for the Federated Social Network.

Google has indexed the FOAF data on the internet and made it available as the Social Graph API [2]. We can query a User by his public profile page (URL) to discover all his public existences on the Internet as well as his publicly declared connections. We directly use this API for our Friend Finding algorithm.

2.2 Notations And Definitions

We now define certain notations which we shall use often in the paper.

- Users: an individual who uses and interacts on the Federated Social Network.
- Connections: the set of other Users with whom one User interacts. The interaction may be one way or two way.Connection are neighbors of a User on the social graph.
- Public Profile: an online webpage which describes the User and lists his publicly described connections.
- Status Update: an update posted by the User on his Profile Page that can be seen by everyone else.
- Subscriptions: list of connections whose activities User follows and declares on his Profile.
- Subscribers: list of connections that follow a Users activities.
- Friends: a common word to describe the connection. It might be an acquaintance, known individual, a subscriber or a subscription.

3. NETWORK MODEL

There already exist many systems on the web that behave in a decentralized mechanism.TCP/IP, Email, DNS operate in a distributed manner on the Internet [8]. Here we define the Federated Social Network centered around a User. Each User (U1) has a profile page which is publicly displayed. This is a web page on the Internet and can be visited by navigating to the URL of the webpage. He keeps a list of his subscriptions and subscribers that follow his activities on the social network. Every subscriber or subscription is another User (U2) on the internet who is part of the federated social network. This User also own a profile page, which is completely under his control. The User (U1) can visit his connection by using the URL of the profile page of other User(U2).

3.1 Mathematical Model

Consider the federated social network existing on the Internet. Let there be N Users, each denoted by $U_i \in U$ where U is the set of all Users and $i \in 1...N$. Every User U_i has a label which is a unique URL on the Internet. If any two

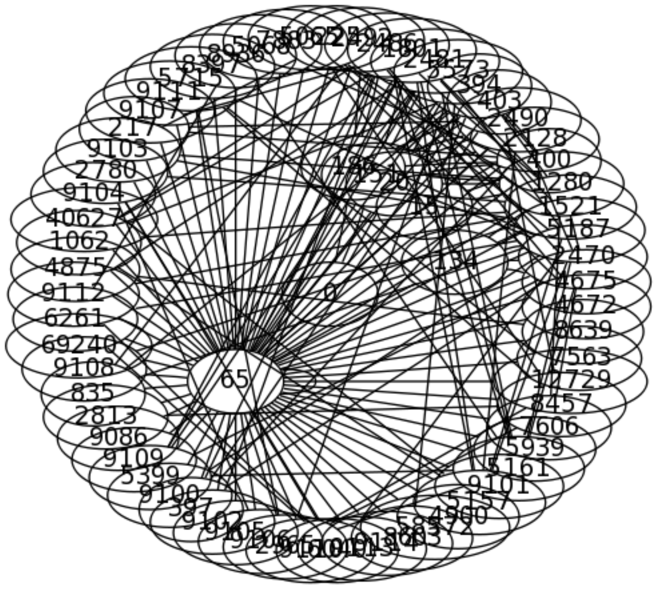

Figure 1: A Social Network Graph Of A User. The User number 65 is centered.

URL's are same, then they represent the same User, hence we assume that all Users U_i are distinct and hence unique.

For every User U_j, he keeps a list of connections in two separate lists. $S_{j_{in}} \subseteq U \setminus U_j$ is the set of all Users that *subscribe* to User U_j. $S_{j_{out}} \subseteq U \setminus U_j$ is the set of all User's to whom U_j subscribes to and hence known as *subscriptions* of U_j. All the User's $U_k \in U \setminus (S_{j_{in}} \cup S_{j_{out}} \cup U_j)$ are unknown to the User U_j and is denoted by the set H.

The interactions in this social networks are modeled as a graph. The $G = (V, E)$ where V is set of vertices's and E is a set of edges. The V here is a set of User's in the social network, also equal to U. Each User (node) is uniquely defined by is URL of its public profile. The edges are directed and E_{ij} shows an edge from User U_i to User U_j.

We define a directed edge E_{ji} if U_i belongs to the subscriptions list of the User U_j, that is $U_j \in S_{i_{in}}$, and directed edge E_{ij} if U_j belongs to the subscription list of User U_i, that is $U_j \in S_{i_{out}}$.

Since the model is federated, any User U_i can never view the complete social network. He can only see or operate upon the network with U_i as the center and intereact with $S_{i_{in}} \cup S_{i_{out}}$. *This is the FSN centered around a User.* If he needs to query the social graph of any User U_k , he sends a request to the profile of U_k and is returned with the social graph of User U_k as its center. Figure 1 shows a FSN User labeled 65, and all his connections. All the friends are connected to the User 65, and may have interconnections between themselves.

Problem Statement: *For every User U_i we choose a set $R_i \subseteq H$ such that for every $R_{ij} \in R_i$*
1. U_i ***knows*** *R_{ij}*
2. U_i will find R_{ij} ***interesting****, if they know each other.*

The notion of *knows* is defined by the fact that, one User appears in the others address book or has communicated with the other atleast once over email or on other public forums or online social networks. The relationship strength between two Users can be modeled by the interaction activity and the notion of *knows* can be made stronger [33].

The definition of *interestingness* of one User to another is relative and we say that one User will be interesting to another if they share some common attribute. Tags for people and status updates can also determine intrestingness of a User [11].

4. FRIEND FINDING

Consider a new User on the Internet, a new addition to the FSN. Initially he does not have any subscriptions and subscribers and his profile URL is unknown to the rest of the nodes in FSN. We have no social graph associated with this User at the center that can be analyzed to find the first set of subscriptions for our User. Its a *Cold Start* - where we don't have apriori knowledge about the User's friends and interests. Thus we divide the problem of finding friends in two parts. The first deals with the cold start to provide the User first set of subscriptions to interact with. We call this *Quick Connect*. The second approach known as Delayed Connect deals with analysis of User's social graph and finding prospective friends.

4.1 Quick Connect

The idea of *Quick Connect* is to allow a User to generate his first set of connections as fast as possible. One of the largest data stores of information about the different contacts of a User is the Address Book. An address book associated with online email services stores the frequently contacted individuals. Email data can also be parsed to find the most contacted friends of a User and to mine their Social Network [7]. Email contact lists can be optionally be stored in a flat file as comma separated values. A vCard [31] format can also be used to store information about an individual.

Address books generally have a particular format which can be parsed to get the information about the contacts. They have various fields which can be of interest to our system. We extract the email address field from the User's address book for all his contacts and use the Social Graph API to search for the contacts OStatus Subscribable public profiles.

Since most of the address books are easily available from email service providers, its easy to access such contact lists and generate a list of email addresses. OAuth [16] is used to authorize the StatusNet software to access the User's address books. We could have used other methods of authorization to access this data but we chose OAuth so that the User's password is never stored in the software. Large number of email service providers provide OAuth end points for accessing this data which makes our method more flexible to suit to any service. These services also offer a wide variety of data format in which they provide the contact lists. When ever available we chose the PoCo [6] address book format.

4.1.1 Anatomy Of Quick Connect

The User is given a list of email service providers from which the address book information will be retrieved. The User is redirected to the OAuth Authorization page on the service providers OAuth endpoint. After the User delegates authorization, the service provider returns an OAuth Access Token and and OAuth Secret Token. This pair can be used in future to access the address book data without

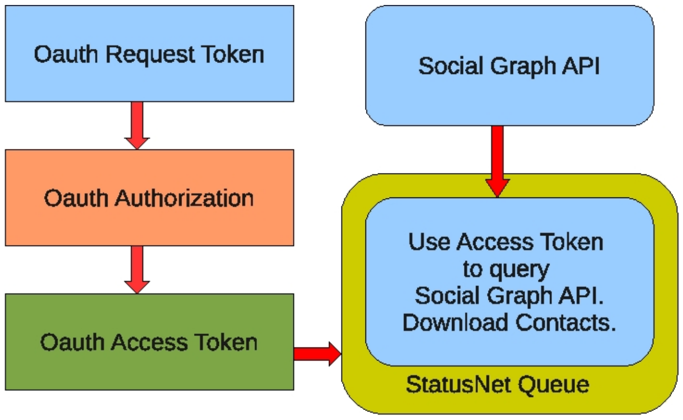

Figure 2: OAuth Dance and Social Graph API Request.

Users intervention. The User can alternatively revoke an Authorization to stop access.

When returned from delegating authorization, the User is shown a list of his contacts page by page. Here he is given an option to email the contact to invite him to create his own presence on the Federated Social Network. Simultaneously, the request is en-queued in the StatusNet software's Queue [24] to update the list of contacts in background. If the User navigates away from the list of contacts, the Queue Daemon ensures safe download of all email contacts.

The background process of download the address book additionally queries each email address on the Social Graph API. This returns a list of public profiles of the contact. If these are OStatus Subscribable then the User is notified about it. This way User's can directly subscribe to FSN User's whom they already know. Figure 2 shows the OAuth Authorization followed by handing over of OAuth tokens to the StatusNet Queue Handler. The address book is downloaded and Social Graph is queried to check whether it is OStatus Subscribable.

Quick Connect is used partially in many social networking websites. This is the first instance in which we search for User's not on the same domain. In case a centralized database of social graph as provided by the Social Graph API does not exist, WebFinger protocol can be used to query the contact email address. The Linked Data representation of friends list in the form of FOAF can be parsed to obtain this information.

The method has a disadvantage of many HTTP GET requests to download the address book and to query the profile page of each contact through the Social Graph API. It is a long process for address books containing thousands of contacts. The use of StatusNet background queues coupled with OAuth access token to a User's data allow download of all the contacts without the User's intervention.

4.2 Delayed Connect

In an online social network, a User tries to expand his social boundaries by connecting to more friends. These maybe his long lost classmates, new friends based on similar interests or connecting to new communities. The graph expands slowly and the *Quick Connect* provides enough fuel to give him the initial set of connections to interact with. Once all the contacts in the address book are analyzed for their pres-

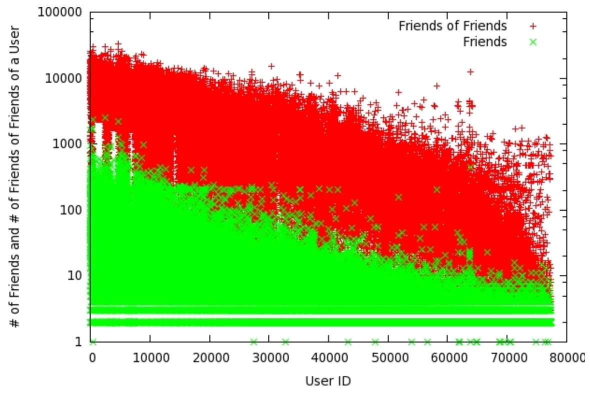

Figure 3: Plot showing number of Friends and number of Friends of Friends on logarithmic scale, of Users of an OSNS. It can be noticed that Users with higher ID, are newer, hence have lesser Friends, but high value of Friends of Friends

ence on the Internet, the responsibility of further expansion of the User's social graph solely lies with him. This is the section where we illustrate the friend finding algorithm of *Talash*.

The existing social graph of the User is now analyzed and uses the social graph API to recommend prospective friends. We also define the problem of finding connections that might be interesting to the User. The algorithm will try to uncover the prospective social network as seen by a User.

This method is named *Delayed Connect* because the execution of this mechanism is not instantaneous and is heavy on resources e.g. Bandwidth. We can run this mechanism for any User whenever we have available resources and suspend the completion in its absence. Delayed Connect believes in expanding the social graph of the User very slowly, so as to give him enough time to share his thoughts and interact with his existing network.

4.2.1 Interesting Connections

A User might be interested in interacting with friends from same community, geographical location or similar interests. Interestingness of a profile is defined with respect to our Users profile information. Interestingness can also be defined in terms of Similarity between two profiles. If the two profiles display content on similar topics, if the biographies of both the User's declare a geographic location close by, or they are both connected to same set of friends, then we say that the profiles are similar and they might be connected for interaction on the social network. Many different similarity functions can be defined given the various attributes of each User [12].

We augment the model given in Section 3.1 to include Profile attributes. Profile attributes is a set of attributes declared by the profile owner as being his own. Formally we augment every node of the FSN by a feature vector $F = a_1, a_2, ..., a_n$ where a_i is a profile attribute denoted by a key value pair of {Attribute Name : Attribute Value}. Since each User on FSN controls his profile, the number of attributes he declares is his wish. Hence different User's may have different set of attributes and different in number. We assume that Attribute Names are are unique and uniformly used. Hence an attribute of name 'Location' on two different profiles will denote the same attribute.

A Similarity function is defined which takes two profiles P1 and P2 and returns a similarity score between them. The function returns a score between the two attribute vectors provided by each profile. Various metrics such as common friends, graph distance, Adamic-Adar [1] can be used to find the similarity score[12]. The score is normalized to the range [0,1] with higher value signifying higher similarity between Users. We claim that a User finds those profiles interesting with whom it has higher similarity.

Since each profile is not centrally controlled and Attributes can be newly added by installing plugins [19] on the StatusNet software, the problem of defining a Similarity function gets complicated. We solve it by defining an endpoint in the StatusNet software, where these plugins can register a handler function. This handler function is designed by the creator of the plugin, whose responsibility is to find the similarity score between attribute values of two different profile.

4.2.2 Searching Friends

We now turn our attention to identifying new connections whom we can recommend our User to subscribe to. We return to the mathematical model described in Section 3.1 and remind that a User does not have holistic view of the FSN. The User can only see and access his connections, all his subscriptions and subscribers, and the publicly displayed connections of his friends. This idea is captured in the FSN centered around a User. User does not know anything beyond two hops from the him.

The Social Graph API is used to identify new connections for the User (say U_i). For every Subscription $S_{ij} \in S_{i_{out}}$ for User U_i we query the Social Graph API for his publicly declared connections. We parse the response from the API to generate a list of Users which are at two hop distance from U_i, let this denote a set $Foaf_i$.

For every profile $Foaf_{ik}$ in $Foaf_i$ we use two criterion on which we recommend the profile to the User.

Friends You Already Know : For every profile $Foaf_{ik}$ in $Foaf$ we re-query the Social Graph API to find all his public profiles and the publicly declared connections on that profile. If $Foaf_{ik}$ is connected to U_i on some other social networking service, it is possible that they know each other beforehand and its safe to recommend the profile to the User. Instances of such a recommendation is when two friends are connected on sites like Twitter or Flickr! and do know about each others OStatus Subscribable accounts. The social graph of prospective connection is queried and its checked whether the User U_i is connected to him on other OSNS. If yes, we can recommend the new connection's OStatus account to U_i for subscription.

Interesting Profiles: For every profile $Foaf_{ik}$ the Similarity function is provided with two profiles, $Foaf_{ik}$ and U_i. The function returns a similarity score based on the attributes of two profiles. A profile is called *Interesting* if the score is greater than 0. The score can be also used to rank each connection in $Foaf_{ik}$ in decreasing order of their similarity scores. The connection with higher similarity score is more *interesting* than the others.

4.2.3 Caching Techniques

The number of friends of friends found at two hop distance from a User in FSN grow exponentially. For example, consider a User U_i with 400 subscriptions. Each of the

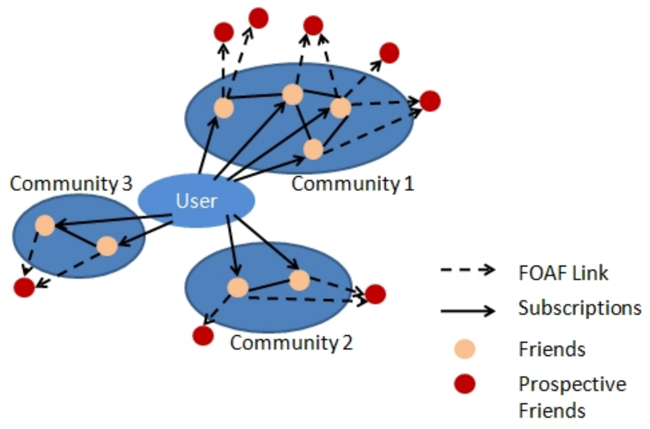

Figure 4: Showing a User and the communities to which he belongs. For each community we generate the FOAF entries and consider them as prospective recomendations.

subscription has at least 400 connections. The number of HTTP requests generated to find interesting friends for U_i is about 160000. Figure 3 shows how the number of Friends and number of Friends of Friends varies for every User. The X axis shows the Users of the FSN and places them in the order they joined the FSN. The Users with higher User ID have lesser number of Friends since they have joined recently. However they have a very high value for Friends of Friends.

We implemented a Cache to store the Social Graph API. For each connection S_{ij} for User U_i, we store its connections in a database with a time-stamp. The tuple stored is of the form $\{S_{ij}, S_{jk}, \text{TIMESTAMP}\}$, where $S_{ij} \in S_{i_{out}}$ is set of subscriptions User U_i and S_{jk} is the set of subscriptions of U_j. The cache served the twin purpose of skipping the HTTP request to Social Graph API in immediate future (Till the entry was invalidated) and allowing interruptions in the Delayed Connect due to network disruption.

4.3 Algorithm

We now outline the algorithm used to identify new connections which can be recommended to the User.

1. Consider the FSN centered at the User. Find the communities to which he belongs by removing the User and finding connected components in this graph. 4.2.3 shows 3 such communities formed.

2. For every friend in the community we can find his FOAF to find prospective connections. Top $\log(n)$ friends (ranked by number of subscribers) are chosen and their FOAFs are requested.

3. For every FOAF, the similarity function evaluates the score of interestingness between the User and FOAF.

4. Top $\log(|FOAF|)$ connections of the FOAF are returned as recommendations.

5. EXPERIMENTAL RESULTS

Experimenting on the system designed and discussed in this paper is a challenging task since the StatusNet software can be configured by any User of the FSN. User's can select what plugins to run and control the features displayed on their profile page such as their location, tags and biographies. The results of this system can be only seen when the algorithm runs on the Users StatusNet instance and the network evolves with the help of these algorithms. User's may independently add new friends and connect to newly joined friends and family members.

We experiment on a snapshot of the Federated Social Network using the network formed on Identi.ca [9] as our basis. Identi.ca is a OSNS from StatusNet that gives each User the complete control on their activities and allow interoperability to other OStatus Subscribable accounts such as Blogger.com and Youtube.com. We use the Subscriber-Subscription data from Identi.ca as our dataset for experimentation. The remarkable property of this dataset is that Identi.ca Users subscribe to or are subscribed by other Users on the FSN, and these links are also included in the dataset. Hence we get a partial but relevant chunk of the FSN for our experimentation. The different parameters of this dataset are shown in Table 1. With more than 280,000 nodes we see that there are 6724 strongly connected components. This shows that the network is diverse and has various communities which do not overlap. We now give our experimentation strategies and the method to evaluate our results.

Table 1: Summary of dataset FSN Graph.

Parameter	Value
Number of nodes	285,198
Number of edges	1663690
Mean out-degree	11.7045
Strongly Connected Components	6724

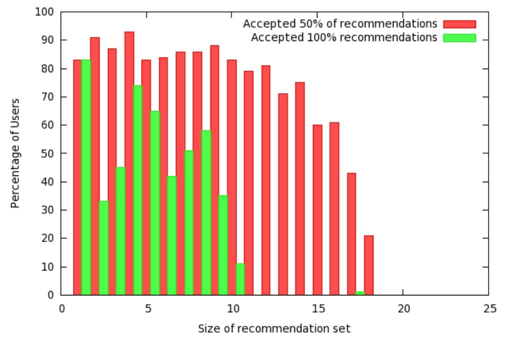

Figure 5: A plot showing number of people that accepted at least 50% of the recommendations given the size of recommendation set. It can be seen that the number of *positives* lies close to 4-5 connections being accepted. The plot also shows the percentage of Users that accepted 100% of the connections recommended to them.

5.1 Manual Feedback

A manual feedback is the best test of the system and can gauge the usefulness of our algorithm. The choice of friends a User would like to have is dependent on the User's interests. We used the Similarity function to find the similarity between the User and the prospective connections and hoped that the User will accept them. We were not able to test *Quick Connect* for these Users because each one owned their instance and did not have our system installed.

Table 2: ASPL for prospective recommendations

Community Details	Initial ASPL	Final ASPL
Developers of Identi.ca	1.807	1.602
Group of entrepreneurs	1.974	1.965
FOSS contributors	2.453	2.702
Family	1.333	1.000

Each User was provided with a set of prospective connections called the *recommendation set*. The User was asked to mark the connection as accepted on rejected. We call the accepted connections as *positives*. Figure 5. shows the percentage of positives against the size of the *recommendation set*. It was seen that the User's accepted not more than 4-5 prospective connections. For a *recommendation set* of size larger than 10, very few Users accepted all the connections. A User is recommended a set of size larger than 10 when the User is part of many communities (so that each community recommends at least one connection) or belongs to one large community (size of $\log n$). A trend was seen that Users marked at least one positive from each community and tended to discard a majority when most connections came from a single community. This means that the algorithm is able to find the best recommendable connection in each community to which the User belongs with the quality decreasing with increase in size of community.

5.2 How Good is a Recommendation?

For every community of the User which recommends a new connection we find out how much the cluster improved in its bonding. The average shortest path length of the cluster are good indications of how the individual communities to which the new User belongs has improved the bonding in the system. The average shortest path length in a cluster quantifies this value with a value of one meaning tighter interaction where everyone in the community knows each other (Clique) and a higher value meaning that the community is loosely interacting (everyone does not know each other). A value lesser than 1 means that some Users are not reachable from some other User in the network.

For a graph $G = (V, N)$, let $d(u, v)$ be the shortest path length between nodes u an v. The average shortest path is calculated as $\frac{\sum_{u,v \in N} d(u,v)}{|N||N-1|}$. For every community to which the User belongs we formed a graph containing the members of that community, the central User and the new recommendation. We find the average shortest path length (ASPL) for this graph.

We noticed that for communities which were already cliques (ASPL $= 1$), the new recommendation increased the value of ASPL. For many clusters the value of ASPL before adding the recommended User was 0.66 and after adding the new connection increased to 1.0 converting the community into a clique. This showed that a new connection was completing the networks hidden links.

Table 2. shows the initial and final ASPL for a User on the FSN. The communities shown is a rough guess of the kind of interaction the User has with them. For small communities like Developers and entrepreneurs on the internet, the ASPL value decreased, possibly filling up gaps in the networks. For a group such as FOSS contributors, the community seed recommended a User completely outside the network. Since very few Users in the community were connected to the new connection, the ASPL value increased.

The measure of ASPL can be used to rank the new connections in the network. A User which tries to converge the ASPL of the community towards 1 will be given higher ranking. The recommendations can be shown to the User in decreasing order of their rank.

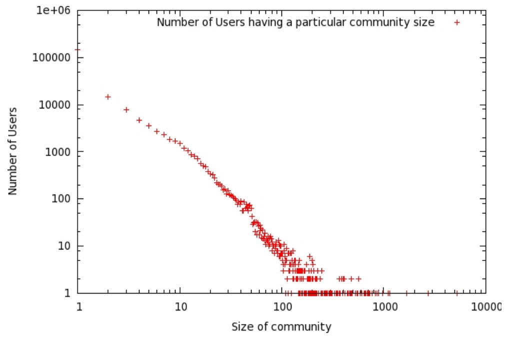

Figure 6: A log-log scale plot showing the different sizes of communities and number of Users belonging to communities of that size.

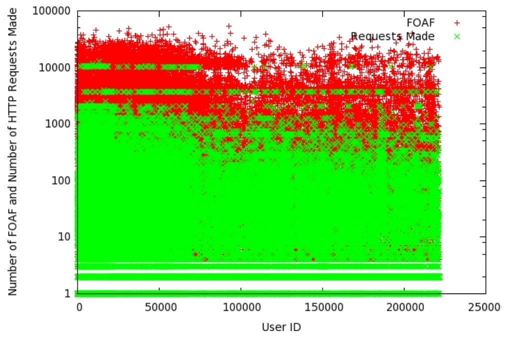

Figure 7: A log-linear scale plot showing the number of FOAF and number of actually HTTP requests made to find new friends.

5.3 Clustering Advantage

The motivation for clustering a User's FSN into communities was to find recommended seeds and to reduce the number of HTTP requests made to the Social Graph API or to WebFinger(if it existed). Figure 6. shows the size of communities to which a particular User belongs. We see that more than 50% of User's belong to a single community. Such User's are either new and have very less subscriptions or tend to interact with a closed group of User's. This reenforces our idea of using recommender seeds from a community to find newer connections for our User. User's from same community will interact closely and will not overlap significantly with other communities. For example, Users from a family may choose to interact only with their family members. A User from that community may be part of another community , say Presley Fan Club. The community finding exercise can also give the User an indication of why the new connection is being recommended.

Figure 7 shows the number of HTTP requests sent to find new friends for each User on the FSN. The number of such requests is reduced drastically. If all the recommendations are accepted by a User, the size of each community of the

FSN centered around the User increases. Hence, the recommendations count will keep increasing and will expose the User to newer prospective friends in every iteration. The $\log n$ upper bound for choosing the size of recommendation set gives rise to slow growth of FSN of a User. The feedback collected from the User's of FSN showed a trend of having lesser *positives* when the recommendation set was very large.

6. CONCLUSIONS

The federated nature of the social network enpowers Friend Finding algorithm to be executed on individual Nodes of the FSN. The network can evolve independently and expand at each node as required by the User. We showed a simple mechanism to find new friends on this network and show how we can depend on the Friends Of a Friend information alone to extract this information. Clustering of a FSN centered around a User into communities allows expansion of a Users FSN in all the domains he is interested. Community based categorization of recommendations is a valuable addition to our algorithm and can be used in various applications where the community feature can be exploited.

The algorithm makes many heuristic assumptions and exact evaluation of them can be only made by analyzing the behavioral pattern of the User for whom the recommendation is made. This is a first attempt to experiment a friend finding algorithm on federated networks. The actual impact of this algorithm can be felt only when the network evolves considerably using this system.

7. ACKNOWLEDGMENTS

We would like to thank Evan Promodrou and his team of StatusNet developers for constant support and interaction during the Google Summer of Code 2010. We would also like to thank Dr. Sanjeev Saxena for his guidance. A part of this work was a result of discussions with him during the work carried out under his supervision for the Masters Program at IIT Kanpur. The authors thank the anonymous referees for their valuable comments that helped improving this paper.

8. REFERENCES

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[2] S. G. API. http://code.google.com/apis/socialgraph/.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143, New York, NY, USA, 2006. ACM.

[4] D. Boyd and N. Ellison. Social network sites: definition, history, and scholarship. *Engineering Management Review, IEEE*, 38(3):16 –31, 2010.

[5] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1):1–38, 2009.

[6] P. Contacts. http://portablecontacts.net/draft-spec.html.

[7] A. Culotta, R. Bekkerman, and A. Mccallum. Extracting social networks and contact information from email and the web. In *In Proceedings of CEAS-1*, 2004.

[8] A. Galloway. Protocol, or, how control exists after decentralization. *Rethinking Marxism: A Journal of Economics, Culture & Society*, 13(3):81–88, 2001.

[9] Identi.ca. http://identi.ca.

[10] S. Inc. http://www.status.net/.

[11] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[12] K. J. Liben-Nowell, D. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[13] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *In Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMCâĂŹ07)*, 2007.

[14] M. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38:321–330, 2004.

[15] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, Feb 2004.

[16] OAuth. http://oauth.net/.

[17] OMB. http://en.wikipedia.org/wiki/openmicroblogging.

[18] A. Passant, T. Hastrup, U. Bojars, and J. Breslin. Microblogging: A semantic web and distributed approach. In *4th Workshop on Scripting for the Semantic Web (SFSW2008)*.

[19] S. Plugins. http://status.net/open-source/add-ons/plugins.

[20] F. Project. http://www.foaf-project.org/.

[21] O. Project. http://www.ostatus.org/.

[22] S. Protocol. http://www.salmon-protocol.org/.

[23] PubSubHubBub. code.google.com/p/pubsubhubbub/.

[24] S. Queues. http://status.net/wiki/queues.

[25] RDF. http://www.w3.org/rdf/.

[26] M. Rowe. Interlinking distributed social graphs. In *Proceedings of Linked Data on the Web Workshop, World Wide Web Conference 2009*, April, Spring 2009.

[27] O. Spec. http://www.ostatus.org/specification.

[28] F. Specification. http://xmlns.com/foaf/spec/.

[29] X. Specification. http://gmpg.org/xfn/1.1.

[30] A. Streams. http://activitystrea.ms/.

[31] vCard. http://www.imc.org/pdi/vcard-21.doc.

[32] WebFinger. http://code.google.com/p/webfinger/.

[33] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 981–990, New York, NY, USA, 2010. ACM.

[34] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. pages 506 –515, apr. 2008.