Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Studying Social-driven Mobility

Comparing Face-to-Face Meetings to Online Social Network Activity

M.Sc. Thesis

May 2010 – October 2010

Author: Georgios Nomikos

Advisors: Theus Hossmann, Dr. Franck Legendre and

Dr. Thrasyvoulos Spyropoulos

Professors: Bernhard Plattner, Maria Papadopouli

Abstract

The rapid increase of online social networks made a lot of researchers to turn their interest towards the analysis of online social applications focusing on reciprocal activity between users. Facebook, having the precedence in daily active usage, managed to reach 500 million active users until July of 2010, blowing its popularity, with the emergence of third-party application platform. Until now, many surveys have been conducted concerning the mobile behavior of users, in order to find out ways to maximize the efficiency of the networks. Especially, in opportunistic networks people make efforts to design heuristic rules to increment the performance of various topologies relying on mobility patterns.

In our project, we focused on collecting mobility data with a modern and attractive way and attempted to characterize them in different dimensions, such as duration, type of social links, location of contact etc. We implemented "Stumbl", a facebook application with which we could keep tracks of the daily users' meetings. We also took into account the flexibilities of Facebook platform in order to leverage facebook activity features of the participating users through our application. Therefore, after collecting this kind of data, we explored how the real-life meetings vary according to the types of affiliations between individuals and the places where they have been held. Interestingly, we noticed that users tend to have either high Facebook or meeting activity with their Facebook friends.

Table of Contents

Abstract 3
1. Introduction 9
2. Motivation 11
3. Related Work 12
4. Background 15
   4.1 Facebook Developer Platform 15
   4.2 Creating a facebook application 16
      4.2.1 How an IFrame canvas page works 16
      4.2.2 How an FBML canvas page works 17
5. Methodology 19
   5.1 Application Description 19
   5.2 The Backbone of Stumbl 23
6. Data Collection 25
7. Measurement Campaign 26
8. Preliminary Results 32
9. Discussion 36
10. Conclusions 37
11. Future Work 37
Appendix A 39
Appendix B 41
References 52

List of Abbreviations

CDN	Content Delivery Network
DAU	Daily Active Usage
FB	Facebook
OSN	Online Social Network
UGC	User Generated Content

List of Tables

TABLE 1: CATEGORIES OF USERS 27
TABLE 2: STRUCTURAL PROPERTIES OF MEETING (STUMBL) GRAPHS. 33
TABLE 3: DISTRIBUTION FITTING USING THE KOLMOGOROV SMIRNOV TEST 36

List of Figures

FIGURE 1: ONLINE SOCIAL WORLD [3] 10
FIGURE 2: IFRAME CANVAS PAGE RENDERING. 17
FIGURE 3: FBML PHILOSOPHY. 18
FIGURE 4: A LIST OF REQUESTED PERMISSIONS ACCORDING TO THE FACEBOOK PRIVACY POLICY. 20
FIGURE 5: FLOW CHART BASED ON THE FUNCTIONALITIES OF THE APPLICATION 22
FIGURE 6: PHP CLASS DIAGRAM 24
FIGURE 7: (UN)SUBSCRIBING ACTIVITY. 26
FIGURE 8: ENDURANCE OF STUMBL USERS 27
FIGURE 9: AVERAGE NUMBER OF MEETINGS PER USER THROUGH THE EXPERIMENTAL PERIOD. 28
FIGURE 10: (UNI)DIRECTIONAL FACEBOOK ACTIVITY OF STUMBL PAIRS (PARTICIPANT – STUMBL FRIEND). 29
FIGURE 11: AVERAGE ACTIVITY PER PAIR BASED ON THEIR AFFILIATIONS. 29
FIGURE 12: AVERAGE NUMBER OF MEETINGS PER USER ACCORDING TO THE PLACE OF MEETINGS. 30
FIGURE 13: PERCENTAGE OF USERS THAT SELECTED CERTAIN FREQUENCIES ABOUT THEIR MEETINGS. 30
FIGURE 14: PERCENTAGE OF USERS THAT SELECTED SPECIFIC TOTAL DURATION ABOUT THEIR MEETINGS. 31
FIGURE 15: MEETING GRAPH 32
FIGURE 16: MEETING GRAPH INCLUDING ONLY STUMBL USERS. 33
FIGURE 17: AVERAGE MEETING AGAINST FACEBOOK ACTIVITY PER PAIR PER PARTICIPATION DAY 34
FIGURE 18: CCDF (COMPLEMENTARY CUMULATIVE DISTRIBUTION) OF AVERAGE NUMBER OF MEETINGS PER PAIR PER PARTICIPATION DAY. 34
FIGURE 19: CCDF (COMPLEMENTARY CUMULATIVE DISTRIBUTION) OF AVERAGE FACEBOOK ACTIVITY PER PAIR PER PARTICIPATION DAY. 35

Acknowledgements

First of all, I would like to thank Professors Bernhard Plattner and Maria Papadopouli that offered me the opportunity to enrich my experience in ETH in Zurich.

I would also like to acknowledge the continuous and valuable support of my advisors, Theus Hossmann, Dr. Franck Legendre, and Dr. Thrasyvoulos Spyropoulos, during the whole period in which this research work has been carried out.

Finally, I would like to thank Merkourios Karaliopoulos for his patience and persistence to share with me his knowledge.

Στους γονείς μου, για την αγάπη τους και την βοήθεια τους....

Τερέζα μου σε ευχαριστώ...

1. Introduction

Social networking is a constantly rising phenomenon. Individuals can enter into relationships with people who share interests or a variety of activities. They interact, by exchanging information across economic, political, historical and other social topics. Quite often, people are organized into social communities according to a wide range of types of "*interdependency"* [5]. Such types of interdependency could be friendship, kinship, religious or working communities where people decide to cluster into.

This social behavior urged researchers to focus on social networking analysis. They noticed that the social environment quite often could be expressed as "*patterns or regularities in relationships among interacting entities*" [4]. For this reason, many scientists turned their considerable interest in leveraging social structures, studying not only the social ties between the units, but also trying to give a precise definition about the implications of those relationships. However, the analysis of social networks usually predicates self-reported data. Due to the restricted size of this kind of dataset, it is not so easy for the researchers to make sound conclusions and generalize.

Passing decades, the rapid evolution of the Cyworld in combination with the attraction of the social networks provoked the emergence of the online social networks (OSNs). Online social networks have become one of the most vital ways to disseminate bunches of information. They provide the possibility to the users to contact each other, to actively collaborate and share any type of content. According to Nielsen Online's latest report [7], the previous year both social networks and blogs have taken the fourth position among the most popular online categories, visited by the 67% of the global online population. These statistics imply that the social media have become a fundamental part of the global online experience with the personal emails leading in social activity.

Some of the most famous online social networks are Facebook, Twitter, MySpace, Orkut, Flickr, etc. These social sites represent communities of hundreds of active millions users (Figure 1), which encourage people to interact each other, taking the advantage to contact with other individuals.

The most popular category of the OSNs is the profile-based social networks. Particularly, the users that join such networks can create and adjust their personal profile, embedding any content or link. Some of the most famous profile-based social networks are Facebook (www.facebook.com), MySpace (www.myspace.com) and Bebo (www.bebo.com).

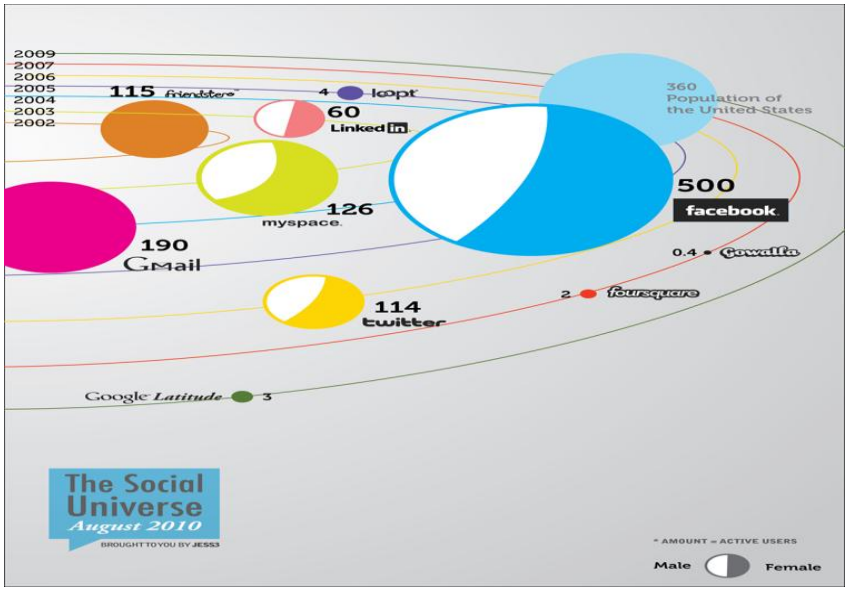

Figure 1: Online Social World [3]

In this project, we decided to work on Facebook the most popular online social network in the US. We tried to approach the relationship between the facebook and mobile activity of individuals by implementing a third party application, the 'Stumbl', in order to track this kind of data. Particularly, each facebook user could subscribe in our facebook application updating in a daily basis her Stumbl profile with feedback concerning real meetings with her facebook friends. More details of our methodology are discussed in the following sections.

The rest of this thesis is organized as follows; in Section 2 there is a detailed overview of our project describing the goals and the methodology that was used. In section 3, we present the four main methods used to track online social networks by describing some relative surveys concerning the mobile activity of individuals as well. Most notably, we underline the lack of related studies, which combine not only online social networking interaction but also mobility issues. Afterwards, we briefly describe facebook functionalities and introduce the Facebook Developer Platform (section 4). Section 5 describes our data collection methodology, and in section 6 we summarize the kind of data we managed to collect. In section 7 and 8, we show some preliminary results after a three-week experimental period. In Section 9, we discuss about the constraints that we dealt with through our survey and we have to resolve in the future in order to draw more representative and unbiased results. Finally, we conclude in Section 10, setting out some future optimizations in Section 11.

2. Motivation

The goal of this study was to invent a method with which we could easily collect fine-grained mobility data, determining additional features of their content, such as the frequency, duration and location.

Taking into consideration the rapid increase of OSNs, we wondered how we could also utilize this flurry of mutual interaction between the users. Since Facebook stands in the second position of the most popular online social services and was the pioneer in launching its Developer Platform [1] we decided to choose it as the basis of our study. Particularly, opening its doors to developers to create their own applications, it gave us the possibility to implement our idea offering a number of useful flexibilities.

Therefore, we implemented a Facebook application with which we would ask users to fill in on a regular basis when and for how long they have met a certain number of their facebook friends specifying the place of their meetings (fun, work, etc.) and the type of their affiliations (family, friend, colleague, etc.). With this dataset we attempted to study whether the place or the kind of relationships affect their social driven mobility.

Moreover, as the relation between virtual social interactions and physical meeting remains largely unexplored, we decided to combine the meeting feedback of the users with their reciprocal facebook behavior during their participation. In such a way, we examined whether there is any potential correlation between these two factors.

3. Related Work

Over the past few years, sociologists and anthropologists have realized that from the perspective of social network analysis can leverage social and behavioral structures among interacting users [4]. However, their typical tools did not permit to capture large-scale datasets since they were mainly relying on interviews. Nowadays, with the emergence of OSNs, researchers have managed to gather rich datasets through the online networking community. Therefore, this social revolution made two social based aspects be the point of interest. The first one is the social mobile behavior of individuals, and the second, is the social interaction between the users of OSNs.

From the online social interaction point of view there are four methods, which are widely used in order to draw a satisfying set of data. These methods are the implementation of third-party applications under web-based platforms, the clickstream, the crawling and the snowball sampling. Below, we refer to some representative studies, which have been based on these methods approaching the online social behavior from different perspectives.

Due to the fact that OSNs tend to extremely restrict their privacy policy in order to protect the subscribed users from any malicious content, the techniques of crawling, snowball and clickstream desisted to be considered as effective. This happened because the majority of the users' profile or whole online social networks are not publicly accessible anymore without their permission. As a consequence, the emergence of Facebook platform gave the possibility to overcome the problem of gaining an over-controlled access in such social topologies. More precisely, this solution concerns the implementation of third-party applications in order to be integrated into the OSN as an indirect controlled approach. For instance, Nazir *et al*. [6] implemented their own Facebook applications, Fighters' Club (FC), Got Love (GL) and Hugged and managed to collect and analyze a huge amount of data. They compared the daily active usage of their applications with the one of the top 160 applications on Facebook for their experimental period, indicating that the DAU of the most popular applications follows a power-law distribution with an exponential cutoff. They observed that the degrees (number of bidirectional edges of a node which belongs to an activity graph) of user interaction on FC, GL and Hugged also follow a power-law distribution.

Likewise, in our project we decided to implement a facebook (third-party) application in order to gather data about the user interaction on Facebook. We avoided using the aforementioned techniques (crawling, clickstream, snowball sampling) in order to approach the content of the facebook profiles. Therefore, each time that a participant was updating her Stumbl profile, we were storing a snapshot of her facebook profile in the database through our application as will be deeper explained in section 5.

Concerning the rest of the methods, Benevenuto *et al*. [11] focused on detailed click-stream data on four popular social networks (Orkut, MySpace, Hi5 and LinkedIn) mostly concentrating on Orkut. They analyzed users' behavior in OSNs, determining the type, frequency and sequence of user activities. Constructing a click-stream model, they classified social interactions into groups, indicating that although "silent" user actions, such as browsing a friend's profile, are the most dominant behaviors, they were not so easy to identify from publicly available data (e.g. writing a scrap). Furthermore, Schneider *et al*. [12] studied the users' interactions through Facebook, LinkedIn, Hi5 and StudiVZ, four OSNs whose primary content are user maintained profiles, extracting actual user clickstreams. They analyzed OSN sessions and user action within the OSN, and concluded that users tend to utilize the same profile activities under long sessions with continuous interceptions. However, the fact that these surveys were conducted using the clickstream technique, constrained somehow their results. Particularly, a clickstream data set being from the perspective of the active user in combination with the highly dependency on the time range, make the correlation between events across time and users quite challenging. Since each user event (application-level) generates a large volume of clickstream data, this makes it difficult to capture certain user events, as it demands extremely large clickstreams. Finally, it is worth mentioning that the structure of OSNs affects this kind of data, incommoding the distinction of user activities.

In order to dig further into the user interactions in OSNs, collecting large-scale data, a number of crawl-based measurements studies have been performed. For instance, a recent study by Mislove *et al*. [13] focused on the graph theoretic properties of large OSNs. They crawled the social network graphs of Flickr, LiveJournal, Orkut and YouTube indicating that the majority of user accounts in the social graph are "*part of a single, large, weakly connected component*" (WCC). They showed that the indegree and outdegree of nodes in the above social networks have properties consistent with power-law networks. Due to the tendency of users to create symmetric links from other users who point to them, they noticed a high correlation between the outdegree and indegree of nodes. Another very interesting survey was held by Wilson *et al*. [9] concerning user interaction in Facebook. They compared the social and interaction graphs analyzing their properties, demonstrating that interaction graphs exhibit a significant absence of small-world clustering.

As far as the forth method, Ahn *et al*. [16] concentrated on three online social networking services, Cyworld, MySpace and Orkut using the snowball sampling. Snowball sampling is a useful tool to build networks. It randomly selects one node as seeder and performs a breadth-first search, until the number of selected nodes has reached the desired sampling ratio. It is like a rolling snowball, which increases as the sample builds up. Most notably, this technique is often used in hidden populations, which are difficult for researchers to have access.

To the contrary with all these studies, in our survey we chose not to go further into the user interaction via the OSNs or topology properties. Our goal was to capture an overview of the behavioral aspects of users on Facebook, and search any potential correlation with the social mobile activity.

Taking into account the mobile dynamics of individuals, relative surveys have been conducted attempting to show how the mobility issue interacts under distinct factors. For instance, Eagle *et al.* [17] demonstrated that as the time goes through the social mobile activity tends to be correlated with the contact activity. In other words, the probability to make new friends increases as the number of meetings increments as well. Another interesting study [15] focused on the proximity between the self-reported and observed mobility data of the participants using mobile phones in order to track this kind of data. They noticed that the recency of the meetings and the kind of the relationships are considered two dominant factors that can highly affect the correlation of the behavioral versus the observed data.

To the best of our knowledge, there is no similar survey, which combines the above two activity attributes. In general, some studies focus either on the analysis of the interaction between the users through the OSNs or on how the mobile behavior varies depending on the social environment. Hence, our ultimate goal was to gain a better understanding on how much the individuals meet up with their facebook friends under certain circumstances and whether there exists a plausible correlation between the facebook and mobile activity. Finally, we mainly demonstrated that the average facebook and meeting (mobile) activity for a random pair of individuals are log-normally distributed.

4. Background

Facebook is a social networking website that opened its doors to everyone (of ages 13 and older) in the 26th of September 2006. When subscribing in Facebook, users can set up their personal profiles having in their disposal a number of functionalities. For instance, each user has a "Wall" in her profile, a kind of message board, where she can post messages and any other personal or contact information, such as hobbies, college, birthday, hometown, phone numbers etc. In addition, they can dress their profiles uploading photos and videos with the possibility to organize them in different albums. An interesting feature is that Facebook users can "tag", or mark users in a photo. Every user can invite other users from Facebook in order to form "facebook friendships". Upon confirmation by the latter, facebook friends can share your profile information, having the possibility to exchange instant messages. Also, they are allowed to make comments or express their interest enabling a tag with the name "Like" to any content of their profile and facebook friends as well. It is noteworthy that Facebook notifies the users about their profile activity with regards to their Facebook friends. A quite useful characteristic of Facebook is the Mini-Feed extension. It is a detailed logging tool which presents to the users the 100 most recent of their actions into the Facebook. At each time, they can see at a glance the history of their activities, including any kind of interaction between common Facebook friends, and with third-party applications.

With all these possibilities, most of the facebook activities occur due to these friendship relationships. However, with the intrusion of the Developer Platform, Facebook managed to gain immense popularity especially due to non-friend interactions through social applications. In the following section we describe more about the Facebook Developer Platform. For the remainder of this study, we will refer to Facebook as "FB" sometimes.

4.1 Facebook Developer Platform

In May 2007, Facebook presented the Developer Platform [1], a key innovation in order to enhance user experience and increase Facebook's functionality. The deployment of this platform gave the possibility to the users to implement *third-party* applications protecting them from the intrusion of malicious content in the response data by the application servers. In general, Facebook platform is a web-based service with methods to fetch and distribute a variety of FB data. There is a list of tools available to the developers that can be used to add social context to their applications. For instance, there is the FB API, a detailed "handbook" with the facebook functions used for the interaction with the FB web site via HTTP requests. The application developers in order to retrieve profile-based data from the facebook servers, they can use "FQL" (Facebook Query Language, a SQL-based language)

queries. In addition, FBJS (FB JavaScript) is a solution for developers who desire to integrate in the FBML-based applications, JavaScript code. FBML is a FB language, which allows representing a number of objects (photos, profile names, etc.) with an easy way. Eventually, FB the last few months offered to the developers a number of social plugins. They are FB extensions, specifically designed so none of the FB user's data is shared with the sites on which they appear. With social plugins, users can see what their friends have liked, commented on or shared on sites across the web.

4.2 Creating a facebook application

When someone decides to create a platform application, facebook developers' guide offers a huge amount of information on how to set up the application, by providing information on what tools you need with the appropriate configuration. More precisely, this guide proposes how to configure the settings and integration points for a facebook platform application using your own host server.

A crucial choice concerning the configuration stage that the application engineers have to take into consideration when they create a facebook application, is whether they should select to use IFrames or FBML as the default for the application's canvas pages. The FBML or IFrames support determines a type of collaboration between the browser, the facebook server and the application server.

In the next session we shed more light on how IFrames and FBML work and what is the most effective decision relying on the needs of the application.

4.2.1 How an IFrame canvas page works

IFrame is considered to be the best solution when an application has already been developed. This is due to the fact that configuring the application in order to connect it to the facebook server is not time consuming. The developer has also the possibility to use JavaScript, HTML and the CSS. Especially, when the source code of the application basically consists of AJAX, it gains extra speed since the requests do not need to go through the Facebook proxy. Generally, IFrames offer a variety of manual configurations in regards with the FBML canvas pages in which most of the contents are automatically configured from Facebook. Debugging regular HTML and JavaScript code in IFrames is much easier than FBML, as Facebook started to come up with new tools extending the IFrames applications with more features.

Figure 2 shows how a traditional IFrame canvas page works. In the first step (1), when the user loads the application's canvas page on facebook (http://apps.facebook.com/_stumbl/), the

facebook server responses (2) with a Web page that contains the facebook chrome including the relative IFrame. It is worth mentioning that a new reconstructed URL is involved in the IFrame in order for the user to communicate with the application server (3). With this process the Facebook appends to the URL a number of parameters in order to give more information about which user is logged into the application, verifying that the request is coming from the genuine Facebook. Under some circumstances (4,5), the application needs to render social content, like names, profile names or make use of the facebook API. This process requires some API calls, which aggravate the whole process with another round trip of communication between facebook and the application server before sending back content to the user's browser.

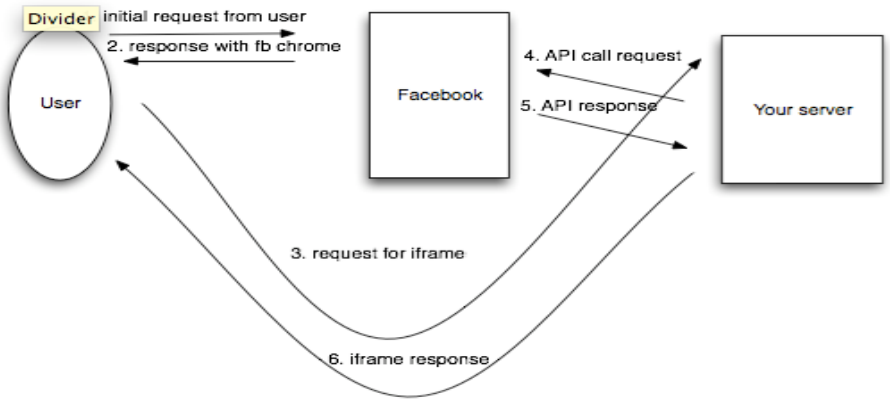

Figure 2: IFrame canvas page rendering.

Hence, up to the previous year, developers had to confront a tradeoff between an application with rich social experience and the latency to fetch all these data that an application needs from the facebook server. Fortunately, Facebook released a new feature, the XFBML, which optimized the platform applications. Now, with XFBML, developers can embed FBML tags like fb:name and fb:profile-pic directly into the HTML code that the application sends to the user's browser. In this way, when the application needs data, which do not require API calls, can directly connect with the facebook server without having to connect firstly to the application and then to the facebook server.

4.2.2 How an FBML canvas page works

In general, the FBML option lets the new facebook developers to start an application with a quick and easy way. They have access to a huge amount of facebook elements just using some FBML tags (<fb:name>) with the relative parameters. FBML is like HTML, but allows the inclusion of social

content inline in the FMBL markup. The application pages do not suffer from cosmetic issues, as the FBML takes over resizing automatically any content that exceeds the good-looking appearance. Furthermore, the FBML canvas pages comprise a judicious authorization mechanism, which facilitates the developer to verify that the request for FBML is coming from facebook on behalf of a particular user.

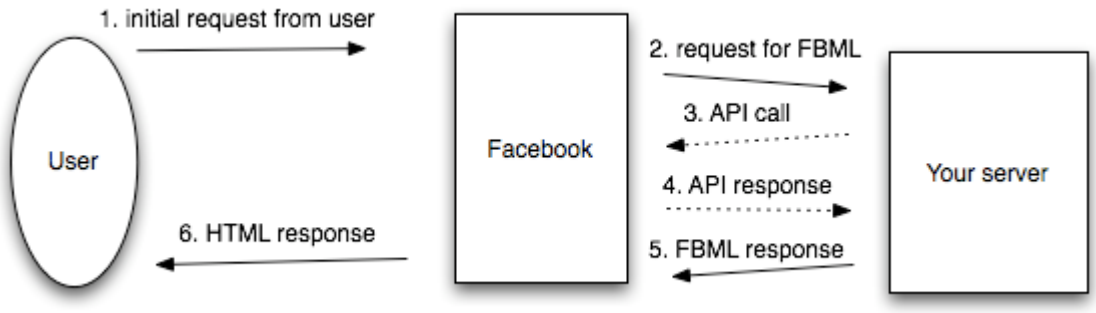

Figure 3: FBML philosophy.

In figure 3 we notice how an FBML canvas page works on the first page that a user loads from the application. FBML is considered quite faster for a number of reasons. First of all, FBML canvas pages are pretty straightforward like the IFrames. Figure 3 demonstrates that when the user requests (1) a canvas page (in our case http://apps.facebook.com/_stumbl/index.php), Facebook does not send back a response immediately; instead, it sends (2) an HTTP POST to a callback URL in the application server (http://people.ee.ethz.ch/~nomikosg/index.php, our server). Afterwards, Facebook expects from the application server to return (5) FBML, in order to convert it into HTML and finally, send (6) it back to the user's browser. Sometimes, the FBML canvas pages need to make intermediate API calls (3), (4) between facebook and the application server, adding some extra delay until the final response. However, when the application needs to show facebook data such as names or pictures in FBML, it can avoid making calls to the facebook API, using tags to reference the data directly. This results to one less round trip, skipping (3), (4) steps, as shown in plot 2. Moreover, there are facebook servers directly peered with some of the largest hosting companies that serve application pages. Hence, the round trip latency from facebook servers to the application servers can be much smaller than the latency from the user's browser to the application server in the case of the IFrame pages.

Although, FBML tends to provide a number of advantages over IFrame applications, Facebook tries to minimize this gap with new enhancements and extensions as well.

Since our application is not a game, it was not necessary to use an interface with special animations and graphics. Furthermore, 'Stumbl' is like a notebook with calendar concerning the meetings with the facebook friends. For this reason, we had to fetch facebook data in an easy and fast way. Thus, the best choice was to enable only the FBML tags integrating the functions of the facebook API.

5. Methodology

5.1 Application Description

The substratum of our survey was a facebook application, which was set up in order to collect a large-scale dataset. The name of the application was 'Stumbl' and the ultimate goal was to keep tracks on the mobility of the users and their facebook interaction. As a result, we constructed a third-party application, which was accessible by any facebook user. They could find it either by browsing in the application directory [8] or by directly getting in with the reception of a facebook (Stumbl) invitation. In the rest of this section, we will describe the features of our application, presenting a detailed guide about what is necessary to do when a user starts the application, and how she can proceed in the next steps.

To begin with, in figure 5 we present an overview about the whole process that a user had to follow in order to take part in our experiment. Particularly, when a user had decided to participate in our experiment, she had to join the application into the facebook, starting with the initialization phase. Hence, the first step was to give access to a list of requested permissions (Figure 4). According to the facebook platform every web facebook-connected application has to be authenticated and authorized from the facebook user. Therefore, Facebook, in order to protect the privacy of the users who have not explicitly authorized a certain application, they are obliged to give access or leave the application. By default, the acceptance of any other application only permits the access of basic profile information of users, such as their names, profile picture, the list of their facebook friends and any other information they have shared with.

As in this project we aimed at investigating the facebook activity between the participants and their friends, it was necessary to have more detailed information about their behavior into the facebook world. For this purpose, we needed to ask for additional, special extended permissions. For instance, permissions to access all the profile information of the application users and their facebook friends, such as their email, friend list, comments and likes about their uploaded videos, photos and albums that had been posted from any other facebook user (Figure 4). With these permissions we were allowed to draw only information with regards to the incoming FB activity. In other words, we

could store any entry that could be noticed in the facebook profile created either by the owner or a facebook friend. In order to have a more fine-grained level of FB reciprocal data, we tried to keep track of the outcoming interaction. With the term outcoming we refer to FB behavior of a user when she visits profiles of her friends. Due to the fact that it was not feasible to utilize the Mini-Feed tool, we were forced to request an additional permission "Access my data any time", as shown in figure 4 to collect this kind of data. According to the FB platform, we were keeping a session id for each participant, with which we could visit the profiles of her friends identifying any track that belonged to that participant. Each session id was valid only for 15 days if the user had never joined the application again in order to renew it.

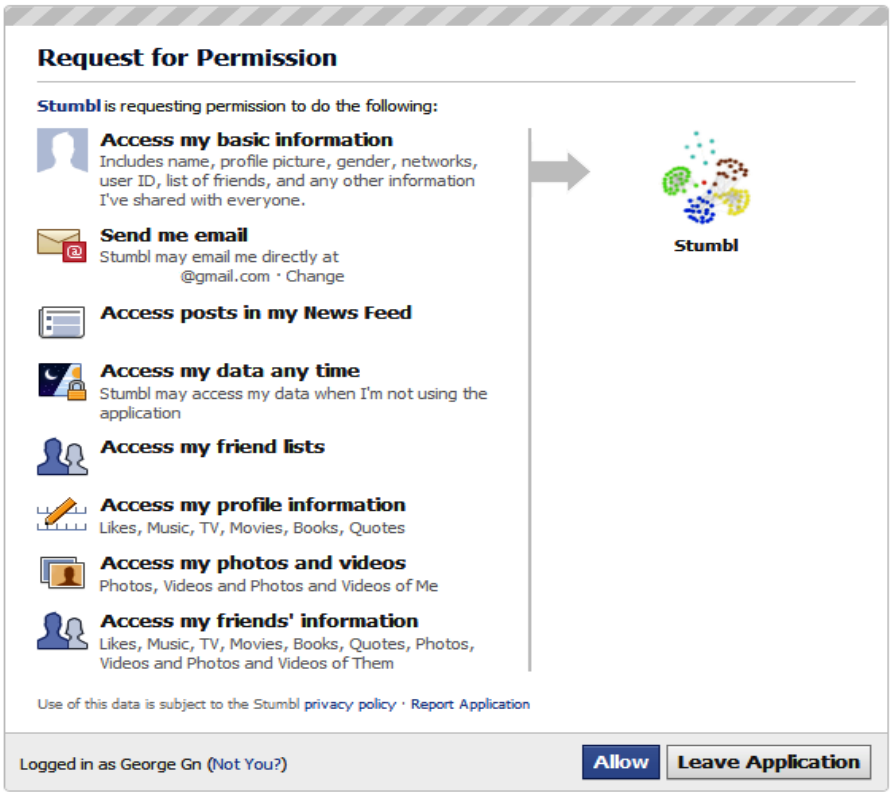

Figure 4: A list of requested permissions according to the facebook privacy policy.

Afterwards, since the application had obtained access permissions from the users, they were ready to participate in our experiment. Firstly, the participant had to choose up to twenty people from a list with all her facebook friends with whom she physically meets up at least once during the experimental period. After the fulfillment of her friend list, the user had to categorize them according to a list of affiliations (e.g. family, friend, colleague, acquaintance), in order to move forward to the next step. It's worth mentioning that the user had the possibility to choose more than one options (e.g. friend and colleague) in the categorization step.

Reaching the last and most important step (running phase, as it is shown in figure 5), the user had fulfilled her own application profile with a list of her Stumbl facebook friends. Through this phase, in a daily basis, she had to specify who of her Stumbl friends she had met with during the previous day, determining the frequency (e.g. 1 time or 2-3 times), the total duration (e.g. 0-10 or 30-60 minutes) and the place (work, home, sport, meal, fun) where the meetings were held. In the list of places, the user could select more than one. During this round, it was indispensable for all the users of the application, to visit their profile in a daily basis, giving feedback about their friends. Thus, the application offered the option of looking back to their profiles up to the first day of their participation, giving the possibility to make any desirable modification to their entries about the meetings.

In general, the application users could change their friend list, deleting or adding other facebook friends without exceeding the limit of 20 Stumbl friends. With this function in 'Stumbl', the users were allowed to adjust their friends according to their schedule of potential meetings in the near future. The application also proposed to users to add some of their facebook friends, which were out of the Stumbl friend list. In that case, the application took into account the friend list of each participant, checking whether there were participants that belonged to the friend list of others and were not included to their friend list.

One interesting aspect of 'Stumbl' was that the participants could closely monitor a bar graph, which included the top five friends with the highest meeting activity. The meeting activity was based on the highest total duration of the meetings through the whole period of participation. Except from their mobile behavior in the real world, they could additionally assess their facebook interaction. Since 'Stumbl' could keep tracks of their facebook actions (in the next session we are going to describe in detail the facebook data) the users were able to see an overview about how many comments and likes they had received from their facebook friends regarding their photos, albums, wall posts and videos throughout the experiment.

Due to the fact that this application was not a game or did not keep the user's interest so much, we had set up a kind of reminder, a program whose task was to send over emails to the users to update their profiles on a daily basis. At the beginning of the use of the application, we utilized the default e-mail that the user had submitted in the facebook profile. Afterwards, the application interface allowed the change of this e-mail address.

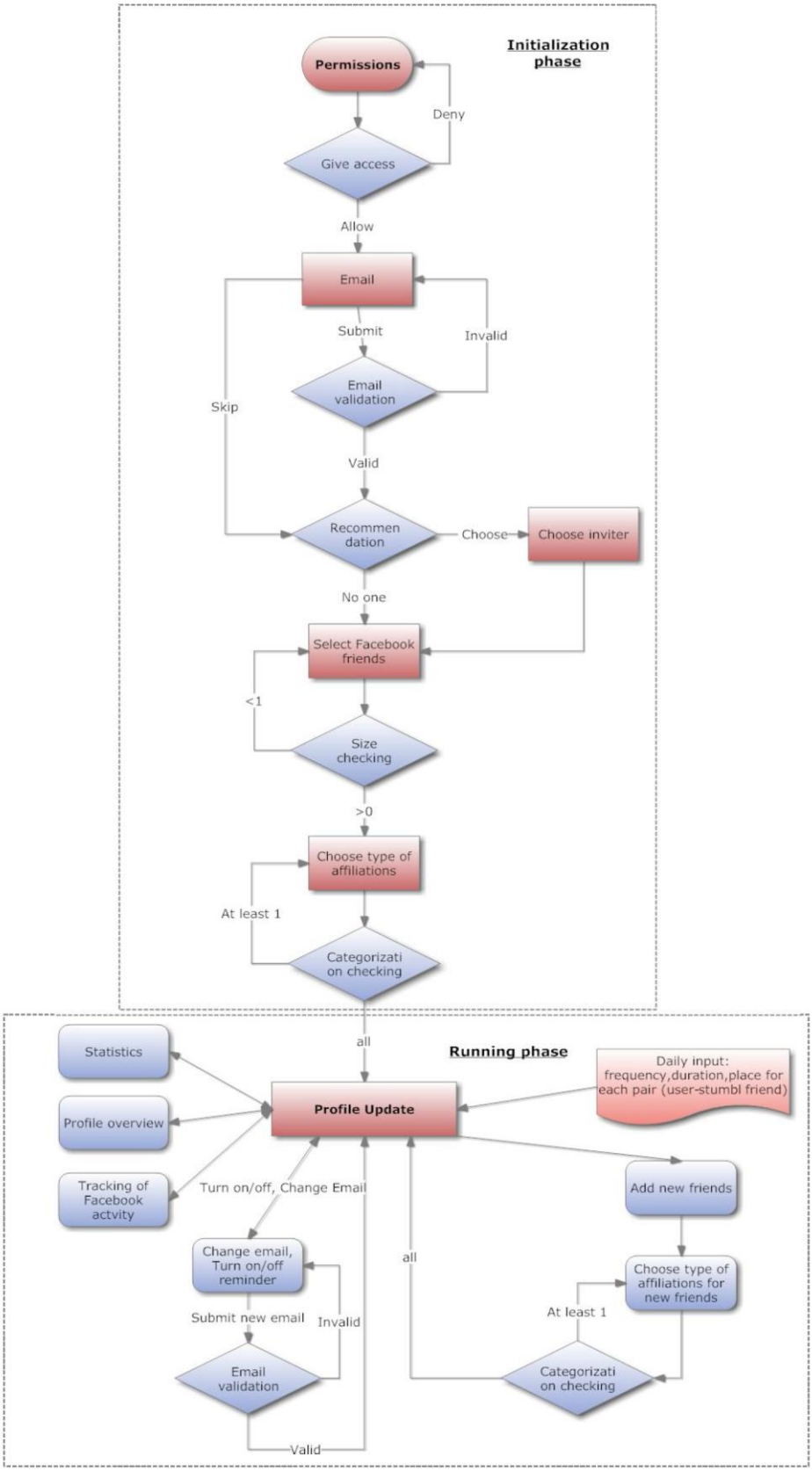

Figure 5: Flow chart based on the functionalities of the application.

5.2 The Backbone of Stumbl

In the previous section, we described in detail the functionalities of the application and presented the expected Stumbl behavior of each participant. Through this session, we are going to dig further into the implementation of our application. Particularly, the majority of the source code consists of PHP and HTML code. Hence, we only analyze how the subparts of this core portion of code collaborate with each other. In our implementation we used JavaScript in combination with facebook tags in order to refine the Stumbl interface. In figure 6, there is an overview of how the PHP classes are connected and how they interact with the database. Below, we give some definitions concerning the operations of the PHP classes. More information, about the methods of each class and their operation can be found in Appendix B.

- **Application Class**

  The application class represents the core of our facebook application. It is a kind of state machine, which initializes the profile of a new Stumbl user or takes over the configuration of the interface according to the step that a user is allowed to go through. Moreover, it is responsible for logging the behavior of the Stumbl user.

- **Log Class**

  The log class describes the content of each loged behavioral entry, which is stored in the database. Particularly, it includes the facebook id of the Stumbl user, the timestamp and the kind of the event.

- **StumblUser Class**

  StumblUser is one of the most important classes in the application. This class represents a Stumbl user, which includes a number of useful functionalities. For instance, it offers information about the current step that a user is permitted to visit and her last time of her access in the application. In addition, it configures the Stumbl and facebook friend list, which are stored in the application's database and the reminder/email system as well. Thus, this class is also responsible for handling the update of the e-mail every time that a user submits a new mail address.

- **Message Class**

  The message class comprises the content of different messages that are going to be presented to the users throughout their participation in the application. Such messages could be considered for example posts about the average mobile activity of the users.

- **Friendship Class**

  A Friendship object represents a Stumbl user's friend. It includes the ID of the Stumbl user that belongs to and also offers methods in order to handle the type of relationship or potential meetings, which describe a kind of social interaction with that Stumbl friend.

- **Meeting Class**

  An instance of the meeting class depicts a meeting event between the Stumbl user and her friend. Through this class, it is possible to fetch and configure the content of the meeting, such as the frequency, duration and place of the event.

- **Relationship Class**

  Alike, the relationship class describes the relation between the Stumbl user and her Stumbl friend giving the flexibility to the relationship status to be in tune.

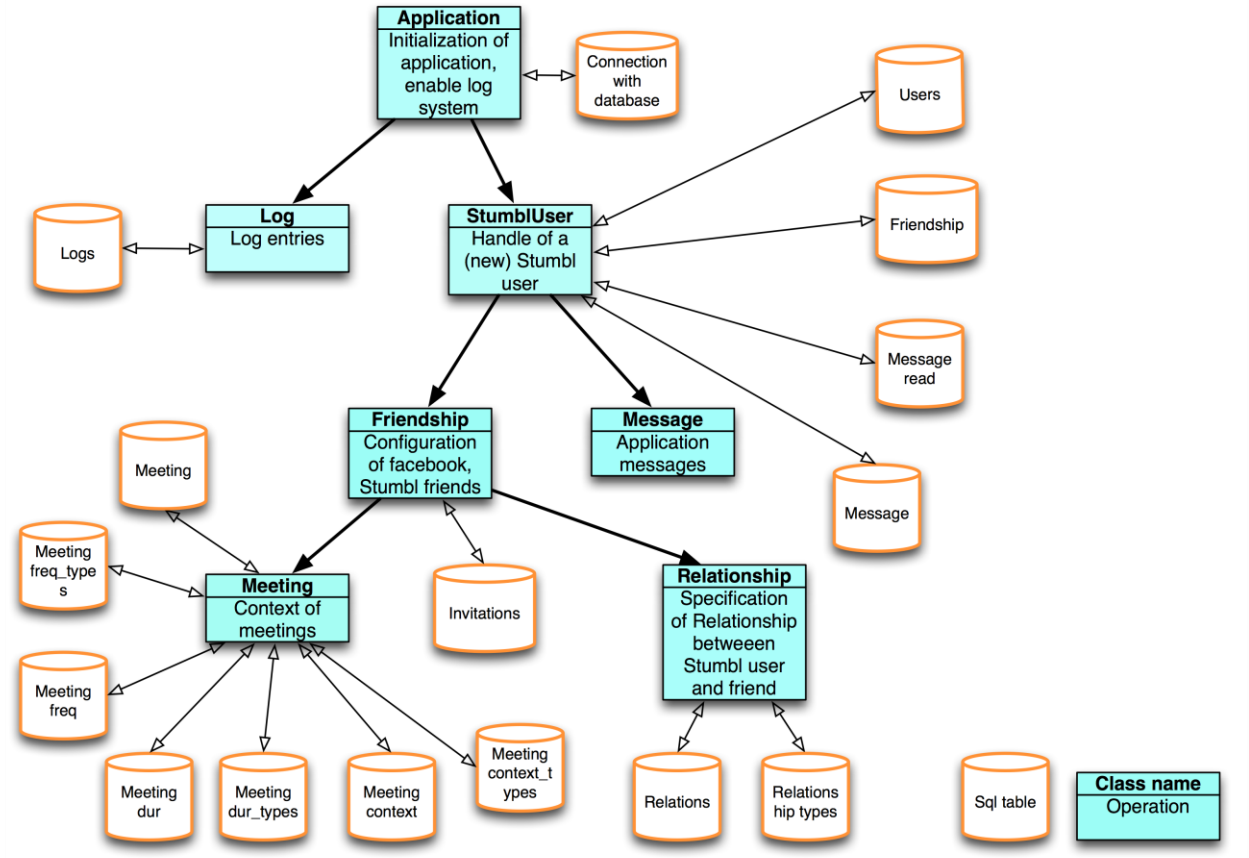

Figure 6: PHP class diagram

6. Data Collection

As it was mentioned in the previous section, the goal of this project was to collect as much data as we could on the mobile and facebook behavior of users. Therefore, we managed to keep tracks on each participant's facebook interaction and daily meetings. Particularly, for every facebook user that joined the application, at the first step, we stored the IDs of her facebook and Stumbl friends and the kind of affiliations (e.g. friend, colleague, family, acquaintance) that she had declared for each one. The IDs helped to regain the facebook identity of each user in order to present the Stumbl friend list through the application. Afterwards, every time that the user was updating her Stumbl profile, at the same time all the meeting entries concerning the frequency, the duration and the facebook IDs of the participants were being saved in the database. Since, we were interested in the mobile behavior of the users, we not only kept the place (e.g. sport, home, fun, meal, work) where a meeting had taken place but also stored a kind of networking data. For instance, we saved for each participant the IPs that they had, every time they visited their application profile.

As far as the facebook interaction is concerned, users had granted to the application a number of permissions. In order to specify any kind of reciprocal facebook activity between the users and their friends, we kept in the database information about their comments, likes and wall-posts. We were not interested in the real content of each facebook activity, so we only kept the kind of activity, such as the actor ID, the receiver ID and the exact timestamp of the interaction. The likes and comments concern the albums, photos, videos, as well as the photos and videos that the users had been tagged in. It is worth mentioning that with the term reciprocal (facebook activity) we refer to the incoming and outcoming activity. With the incoming, we pertain to any new entry in the user's facebook profile by her friends. On the other hand, the outcoming behavior concerns any visit of the users into the facebook profile of their friends, performing any kind of interaction in the context of Facebook, such as a wall post or making a comment to a public content, etc.

In the previous section, we described a reminder system for the users in order to update their profiles. For this purpose, it was necessary to store their e-mail addresses. It is worth mentioning that most of the sensitive data that we gathered and process were anonymized. We did not need actual content of any specific user, as they were not going to be used or appeared anywhere in the final results of our study.

For the needs of our project we set up an application server located in ETH. We configured it by taking into account the facebook policies and any other security constraints.

7. Measurement Campaign

After the main functionalities and the graphic interface of the application were fulfilled, we ran the experiment for a beta testing period of 1 week (19-25/7). We only distributed it to the members of the [1]CSG group receiving important feedback about some drawbacks and suggestions in order to proceed into the optimization phase. It is worth mentioning that some members of the group pointed out that our application was not discreet enough. As a consequence, we made some efforts to eliminate the number of the requested permissions (section 5.1) extending the application interface with optional properties. A month later, we published the application to the Facebook Directory [8] and we officially set off our project for three weeks (17/08-7/09).

Due to the fact that the application could be considered quite intrusive, we decided to do three raffles -one for each week- in order to attract more and more participants. Therefore, we implemented a point system in order to ascribe a condign reward to three of the subscribed participants. The below formula describes the way of the total point calculation:

$$Score = P\_d + (P\_d * S\_f\_u)$$

where *P_d* denotes the number of the participation days and *S_f_u* the number of Stumbl user's friends who also use the application. Actually, this form supports these consistent users that were interested in participating in the experiment for a satisfying period, prompting their facebook friends to join Stumbl as well.

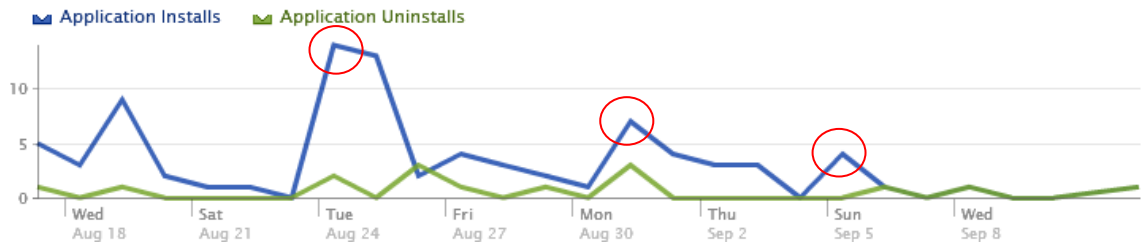

Figure 7: (Un)subscribing activity.

[1]Communication Systems Group (CSG), Information Technology and Electrical Engineering Department [D-ITET] at ETH in Zurich.

After the experimental period of three weeks, in figure 7 we have a first picture of the total application turnout. More explicitly, the blue line represents how many users subscribed in our application (Y axis) through the running period (X axis), while the green line indicates the number of Stumbl users, which decided to unsubscribe from Stumbl. We observe that there are three characteristic activity peeks (red circles), which demonstrate that in the previous day a raffle took place, urged quite more users to participate the day after. Even though the prize of each raffle was an iPad (Apple), the total number of the participants did not reach a satisfying amount of 'Stumbl' active users, as it is shown in table 1.

	# Users
Cautious users	85
Interested users	53
Active users	38

Table 1: Categories of users

Particularly, there were only 38 (active) users, which were updating their profile almost every day, and 53 (interested) users, which just started the application in order to have a look. Unfortunately, the majority of the facebook users (85 cautious users) refused to give access to the application.

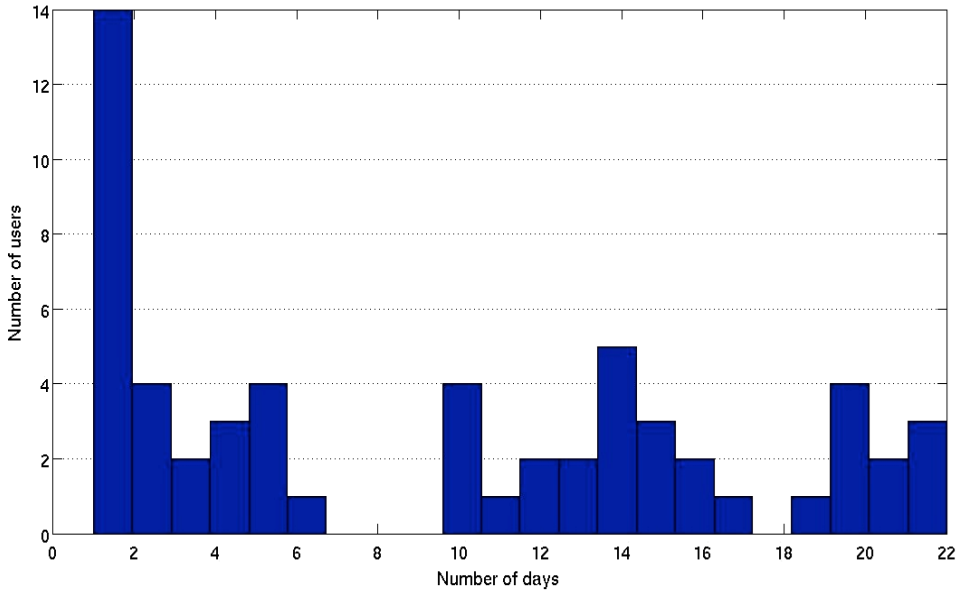

Figure 8: Endurance of Stumbl users

Concerning the Stumbl active users, we noticed different participating behaviors through the experiment. In figure 8 we can observe that there are three regimes of users. Each bin represents how many users participated for a certain number of days. Hence, we can distinguish the first group (1-6 days) where the "tryout" users belong to. Those are users that gave entries only for the first week. Afterwards, we have the regular users (10-17 days) which participated for about two weeks. Finally, we have the "power" users as it is shown in the right corner of the figure 8. They were the most active users updating their profile in a daily basis.

Even though we did not manage to attract a satisfying number of active users, the next plot (figure 9) demonstrates that the participants tended to not have quite many meetings as the week went through. Apart from certain variations in the meeting activity, one can observe that a diurnal pattern takes place as the average number of meetings per user (Y axis) reduces at the end of each week. However, we would expect the reverse behavior that is to see the highest meeting activity during the weekends. This might have happened because the majority of users had meetings with their Stumbl friends at work (figure 12) without keeping in touch at the weekends.

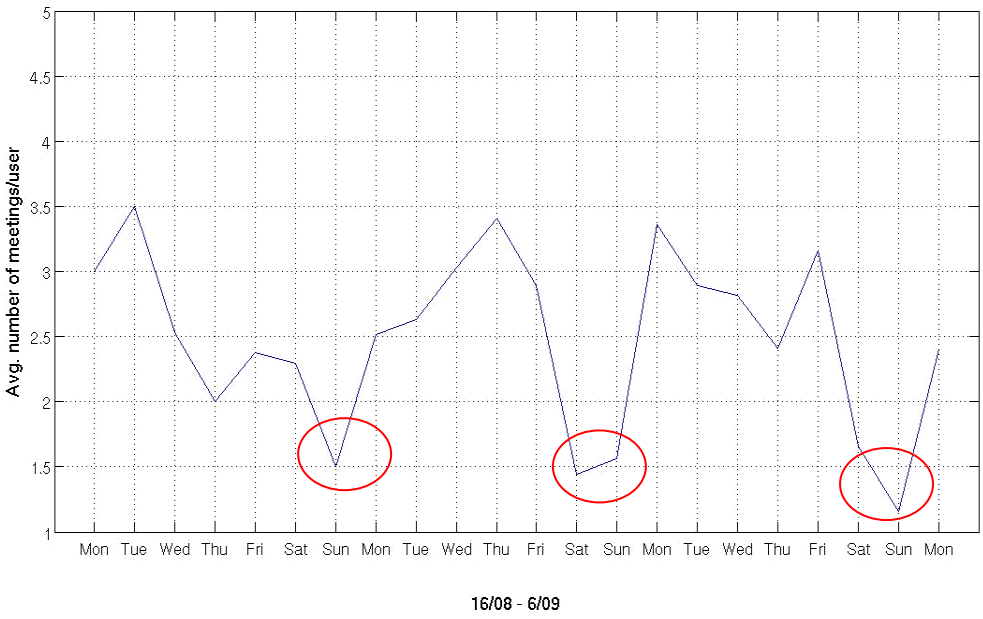

Figure 9: Average number of meetings per user through the experimental period.

In general, during the period of our experiment most of the users did not interact with their Stumbl friends. We analyzed the pairs (participant – Stumbl friend) according to whether they presented facebook activity or not. In figure 10 we notice that the highest percentage (67.4%) of pairs did not

have any actions (blue box), while only 8.4% of pairs visited the facebook profile of each other exchanging any kind of facebook content (brown box).

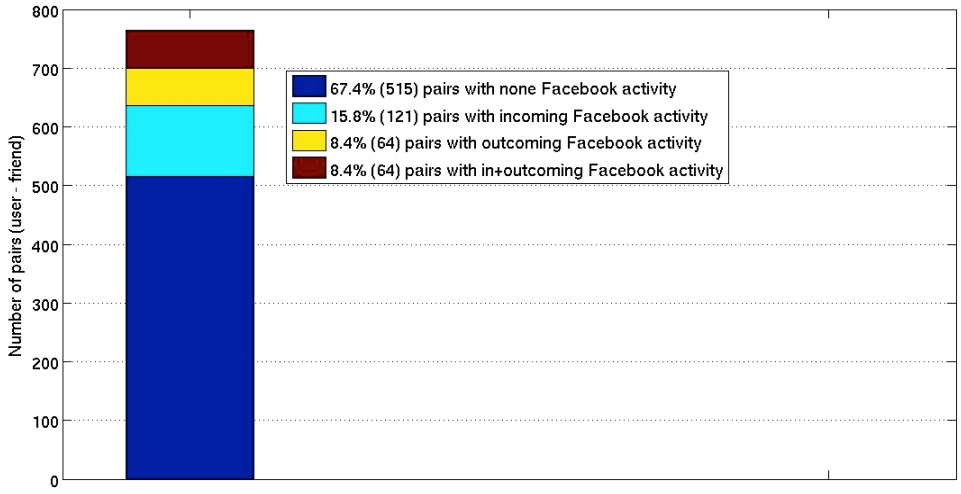

Figure 10: (Uni)directional facebook activity of Stumbl pairs (participant – Stumbl friend).

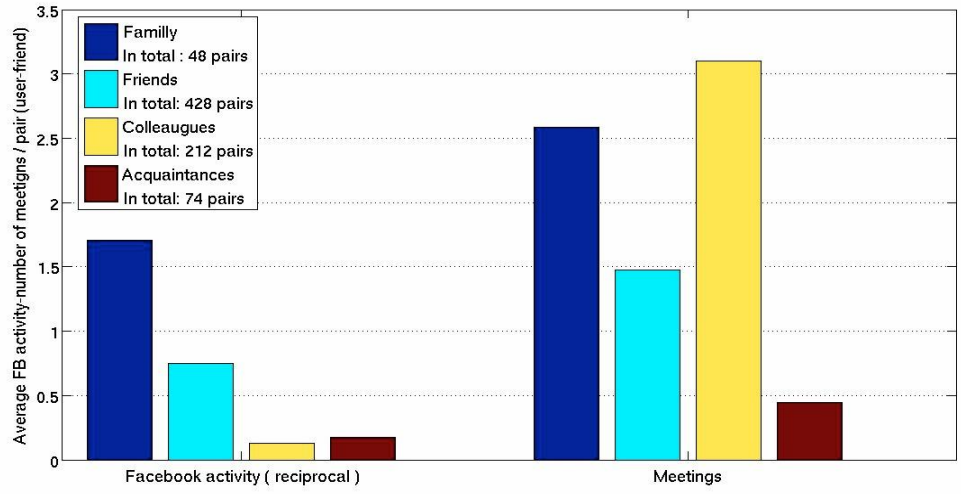

Figure 11: Average activity per pair based on their affiliations.

After having obtained an overview on how the mobile and facebook activity spans between the participants and their facebook friends, we decided to focus on how the type of affiliations affects either their online or their social attitude. In figure 11, there are two groups of bars that describe the facebook activity (reciprocal) and the meetings respectively. The Y-axis represents the average facebook activity per pair for the left group, and the average number of meetings per pair for the right one. They are both categorized in pairs based on family, friends, colleagues and acquaintances. We

notice that the family pairs are quite dominant including the colleagues in the case of meetings. This happens because the majority of the Stumbl users were also colleagues from the CSG group.

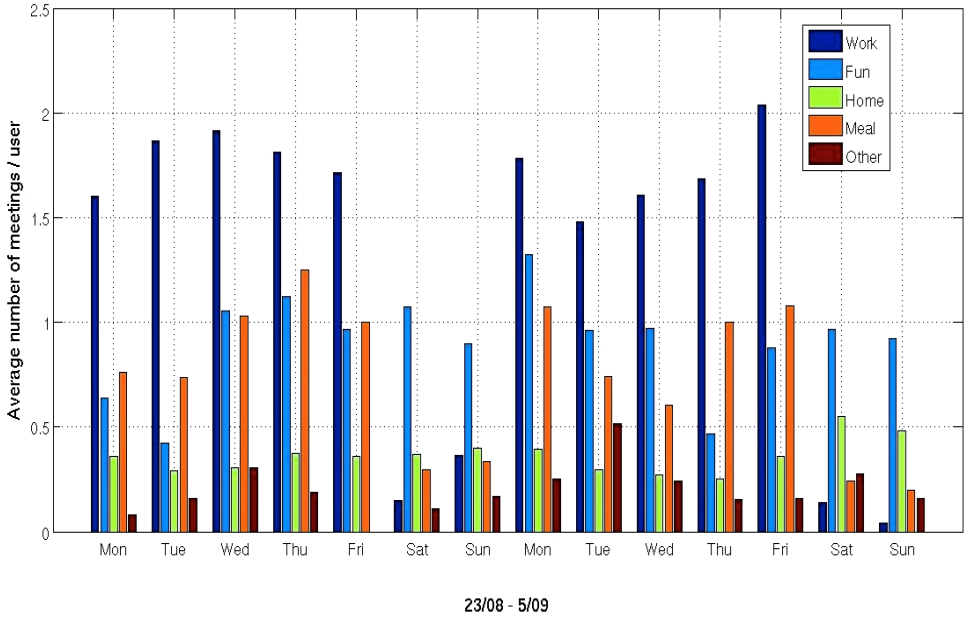

Figure 12: Average number of meetings per user according to the place of meetings.

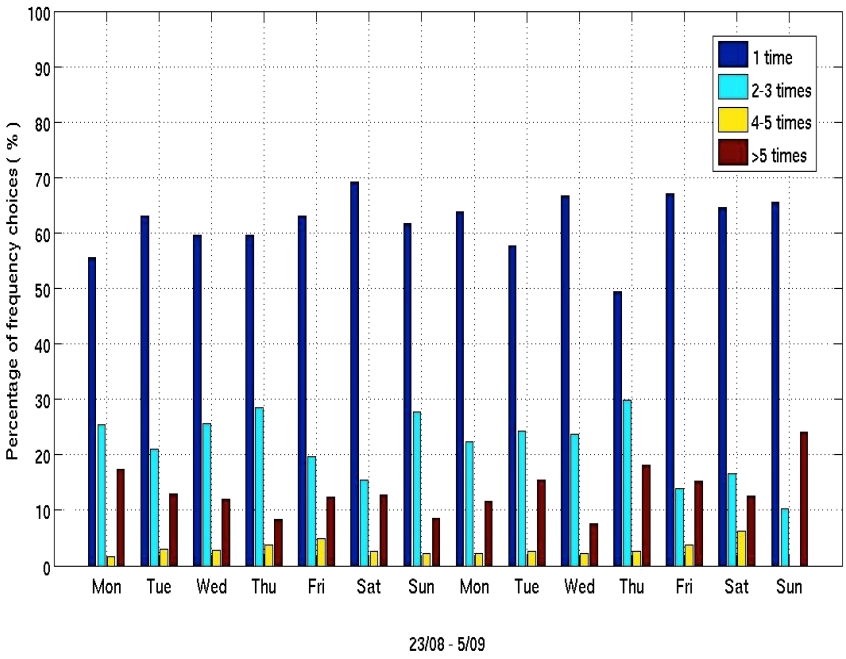

Figure 13: Percentage of users that selected certain frequencies about their meetings.

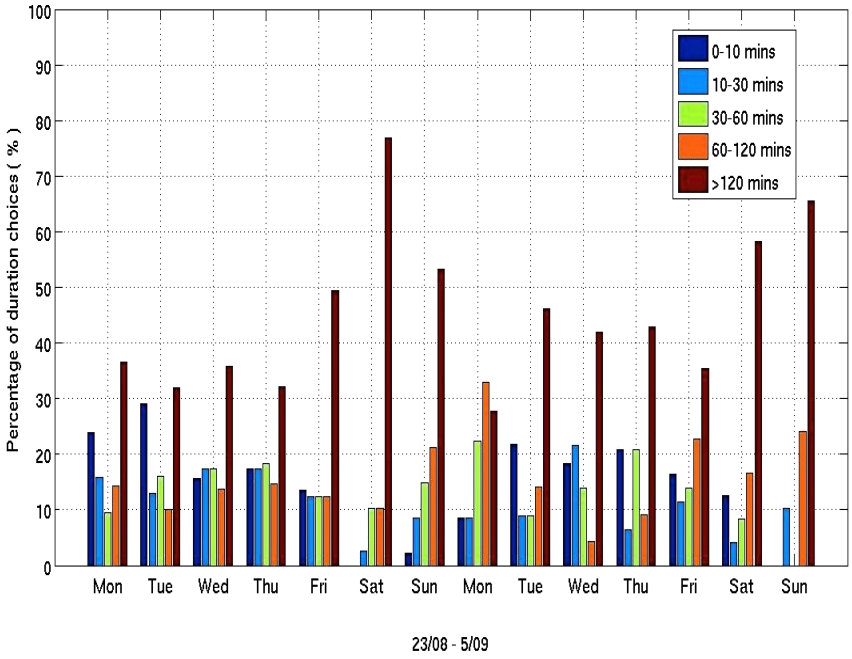

Figure 14: Percentage of users that selected specific total duration about their meetings.

In figures 12, 13, 14 we show the discrepancies of meeting activity from another point of view. To begin with, in figure 12 we calculated the average number of meetings per user (Y axis) according to the place where they were held. The following two plots (figure 13, 14) present the percentage of users (Y axis) that selected a specific choice for each day concerning the frequency and duration of their meetings respectively. We have to mention here that we selected only the last two weeks of the experiment in order to reach a satisfying number of participants deriving more accurate results in the analysis phase. Taking into account only figure 12 we can see that in the weekdays the 'work' dominates as place, while at the weekends 'fun' places prevail, as we would expect. On the other hand, in figures 13, 14, we notice that only one choice of the total frequency (1 time) and duration (>120 mins) of most of the meetings seem to keep the first position over time. This means that only the factor place possibly affects the mobile activity of the users. In addition, since the majority of the users have mainly chosen "1 time" and ">120 mins" for frequency and total duration respectively, we assume that the individuals prefer to have a few long meetings in their daily life, regardless of the place.

8. Preliminary Results

In the previous section we showed how the mobile and facebook data behave when different factors take place, such as the frequency, the total duration, the place and the kind of the relationships between the participants and their Stumbl-facebook friends. In order to gain a better understanding on the mobility of the users in regard with their Stumbl friends, we constructed a meeting graph (figure 15) the properties of which are presented in table 2. Note that each edge represents the occurrence of a meeting between two individuals (nodes), while the size of nodes is related with the number of their neighbors.

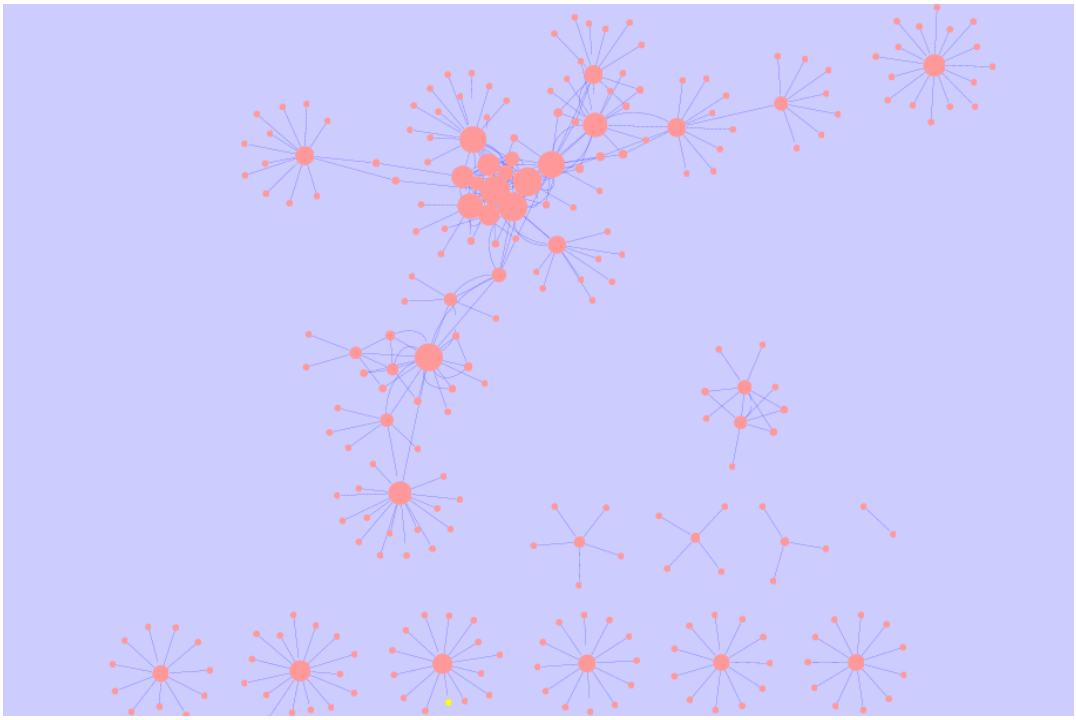

Figure 15: Meeting graph

We calculated the number of communities and specified their modularity, which is quite close to 1. We conclude that the graph consists of dense communities, which are sparsely connected. We also estimated the average clustering coefficient, which is extremely low as it is shown in table 2. We suggest that this clustering coefficient is artificially low because in the meeting graph rather than most of the nodes represent the Stumbl friends of the participants, we did not manage to attract all these individuals to take part in our experiment in order to give their meeting entries. Hence, we would expect to see in each community in the graph many of their "triangles" to be closed. Since we did not accomplish to get almost any "friends of friends" of the Stumbl users to participate, these links do not show up.

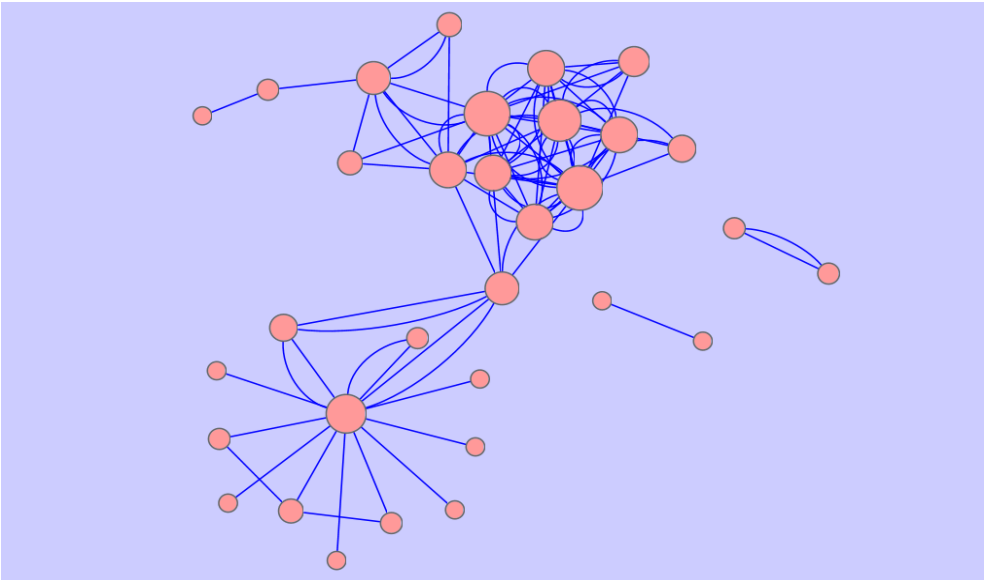

Figure 16: Meeting graph including only Stumbl users.

In order to derive more accurate results we decided to construct the meeting graph including only the Stumbl users (figure 16). According to table 2, in this case the clustering coefficient is quite higher. Since we utilized the mobility data for each user, the clustering coefficient becomes more representative in relation to the meeting graph (figure 15), without having ego networks as in the first case. Moreover, we calculated the average shortest path of the graph in order to see whether there is a small world network. As a consequence, we calculated the average clustering coefficient and shortest path of a same size Erdos-Renyi random graph and we showed that it almost has the same avg. shortest path but a quite lower clustering coefficient (table 2). This proves that the meeting Stumbl graph presents *small-world* properties [14].

	# Nodes	**# Edges**	**Avg. Clustering Coefficient**	**Avg. Shortest Path**	**# Communities**	**Modularity**
Meeting graph	260	364	0.09	2.34	19	0.79
Meeting Stumbl graph	32	95	0.45	2.71	5	0.4
Random Stumblgraph	32	95	0.13	2.19	-	-

Table 2: Structural properties of meeting (Stumbl) graphs.

In order to dig further into the meeting activity of the users with their facebook friends, we show the relationship of the number of meetings against the facebook activity. In figure 17, we measured for each pair (Stumbl user-Stumbl friend) the average number of meetings (Y axis) and facebook actions (X axis) per participation day of the user. According to the results, the users either have high meeting or facebook activity. This comes out from the fact that we do not notice pairs that have both high average mobile and facebook behavior.

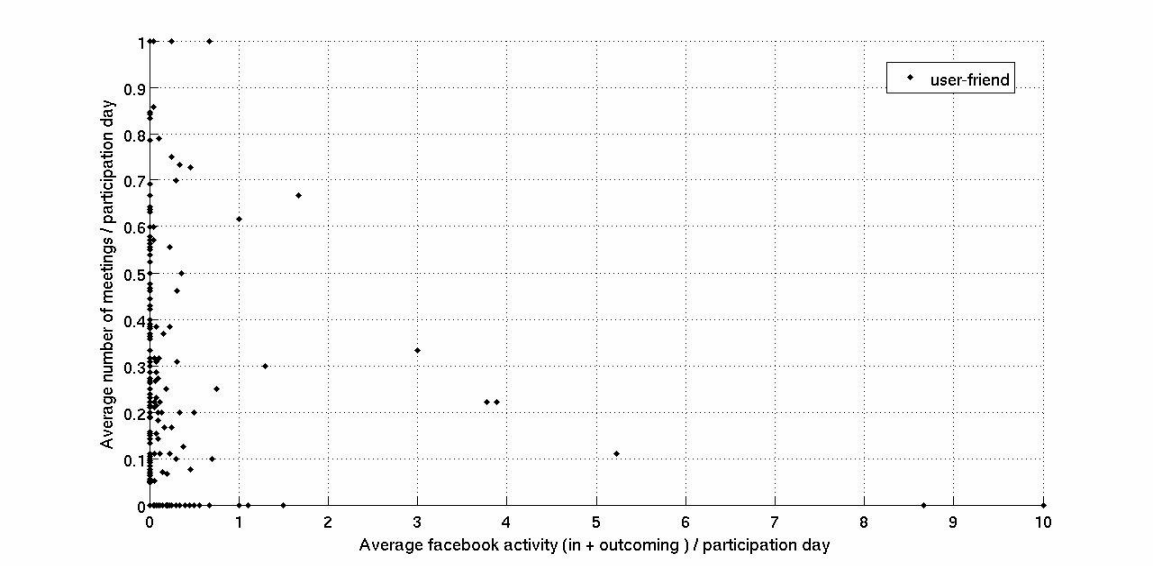

Figure 17: Average meeting against facebook activity per pair per participation day.

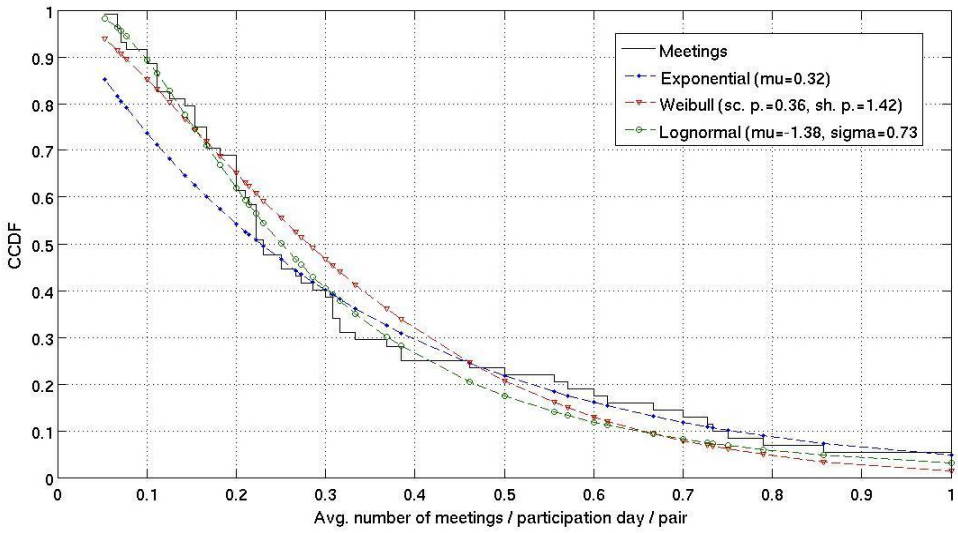

Figure 18: CCDF (Complementary Cumulative distribution) of average number of meetings per pair per participation day.

Afterwards, we plotted the CCDF (complementary cumulative distribution function) concerning the average mobility and facebook activity (figures 18, 19) in order to examine the distribution that they possibly follow. We used the maximum likelihood estimator to compute the fitting parameters for three potential distributions (Exponential, Weibull, and Lognormal). As a goodness-of-fit test we tried the Kolmogorov Smirnov, the output values of which are shown below in table 3. More precisely, in each row of the table we have the results that we drew from the Kolmogorov Smirnov test for the relative distribution, while the couple values refer to the CCDF of the mobility and facebook factor respectively. It seems that the CCDF distributions (figures 18, 19) are best approximated by a log-normal distribution with means μ = -0.38, -1.6 and standard deviations σ = 0.73, 1.14 as we notice in the plots. In addition, in the table 3 we can see that the null hypothesis in the case of the log-normal distribution is not rejected having the greatest p-value. The null hypothesis describes that the results were drawn from the same distribution (log-normal). Taking into account the values of these metrics we conclude that the average number of meetings or facebook actions for a random pair during a week is log-normally distributed.

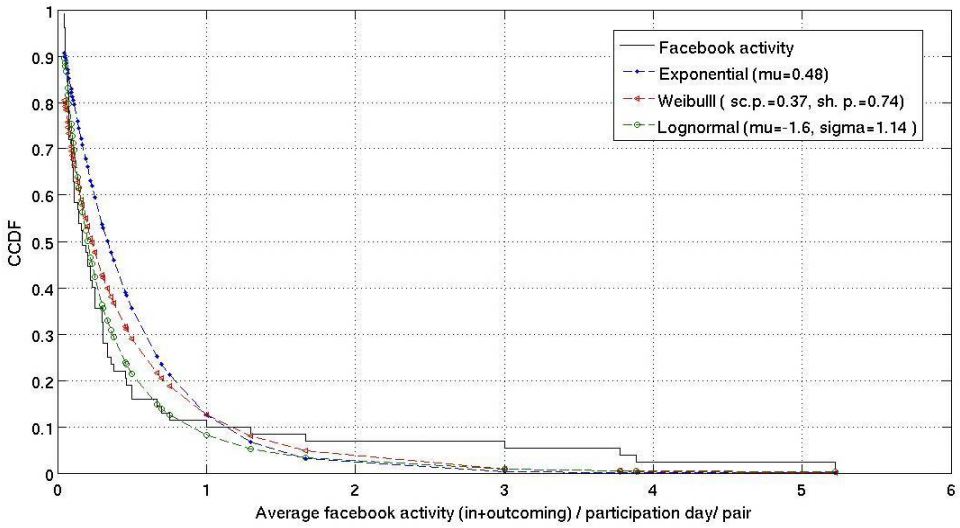

Figure 19: CCDF (Complementary Cumulative distribution) of average facebook activity per pair per participation day.

	H: null hypothesis	P: p-value	K: k-test
Exponential	1 Reject | 1 Reject	0.0035 | 0.0002	0.3 | 0.36
Weibull	0 | 0	0.4 | 0.08	0.15 | 0.21
Log-normal	0 | 0	0.54 | 0.4	0.13 | 0.15

Table 3: Distribution fitting using the Kolmogorov Smirnov test.

9. Discussion

Taking into consideration our results in combination with the size of our dataset, we assume that a kind of bias takes place. In addition, we observed that the majority of the participants were from the CSG group of ETH. This means that a lot of Stumbl users were also colleagues and facebook friends with each other. Therefore, a number of meetings were held, independently from their facebook interaction. This behavior could diversify the final results about the interaction between facebook and mobile activity. Particularly, we examine how facebook friends interact with each other, and how this reflects to their social life. In the case of colleagues, part of their mobility is obviously regulated from their job. Hence, we assume that a number of their meetings with their friends (colleagues) could not be taken into consideration as facebook side effects.

From the beginning of that project we had the consciousness that we are going to approach a quite intrusive aspect of the personal life of individuals. Therefore, we tried to inform the audience in detail about our identity and the goals of our project. Due to the fact that, we had implemented our application under the Facebook Platform, we had the advantage that a participant had to be logged in Facebook before using Stumbl. With this policy Stumbl users were protected by facebook security techniques. Nevertheless we noticed that the facebook users were quite hesitant to take part in our application due to privacy concerns. In order to ensure the integrity and confidentiality of Stumbl, we decided to make a raffle with three Apple iPads since it is quite rare to find applications in Facebook to give away such expensive devices and of course made the participation more attractive. Eventually, we showed that this bonus was not satisfying enough to attract a bulk of new Stumbl users. This behavior demonstrates that facebook users and especially users that utilize Facebook as a kind of career portfolio hardly trust third-party applications to have access in their profiles even if they come from a valid source.

As a consequence, the lack of a rich data set does not permit to assume that our results are considered as fully representative. Nevertheless, we managed to acquire a contact trace covering various nodes around ETH and Zurich.

10. Conclusions

In this project, we tried to quantify the discrepancies of the meetings between facebook friends, analyzing different components of them, such as the frequency, the duration and the place of meetings. After implementing a facebook application through which we collected data, we observed that the mobile and facebook activity for a random pair during a week follow a lognormal distribution. Even though we did not manage to fetch meeting data from all of the Stumbl friends, we showed that the meeting graph including only Stumbl users is actually a small world network. Afterwards, we attempted to approach the mobile behavior in relation to the average facebook action per pair. Our preliminary results demonstrated the absence of a linear correlation between these two attributes, implying to be negatively correlated. In order to verify this assumption, we need further analysis in order to derive more accurate results. Nevertheless, we outlined the research challenges in analyzing online social networking information in regards to real mobility data.

11. Future Work

Due to the fact that we did not manage to collect a large data set of the meetings of users with their facebook friends, and also to the fact that the users did not present high facebook activity, it was not feasible to conclude whether there exists a kind of correlation between these two factors. We assume that the small experimental period and the intrusiveness of Stumbl application were the two main factors that possibly affected the number of participants. Particularly, as we discussed in previous section, we ran our project at the end of August, a period that the majority of facebook users were away for vacations. In addition, facebook users seemed to avoid taking part in Stumbl, because they were quite cautious about privacy matters that follow data collection. As a consequence, we decided to let our application publicly accessible in the future, making at first, some crucial changes based on the feedback that we received from the Stumbl users. For instance, we have to reconfigure the initial list of permissions that a user has to grant in order to get in. Furthermore, it would be necessary to take into account some security concerns and speed optimizations since a number of participants complained that the application was not responding. We suppose that this happened under certain circumstances, when facebook workload was quite high. At last, we have to promote the application in a more efficient way; since, we will not offer iPads in order to attract new users, it is

appropriate to make the application more seductive to counteract this lack of motivation. With all these refinements we aspire to shed more light into the relation between Facebook and mobility.

Appendix A

MySql Tables

Tables of Mobility	Description
Users	Information about the Stumbl participants.
Invitations	Entries about Stumbl newcomers, the participation of whom came out from Stumbl Inviters (already Stumbl users)
Friendship	Stumbl-facebook friends of the users.
Relations	Types of affiliations between users – Stumbl friends.
Relationship_types	Available types of relationships that a user can select.
Meeting	Entries about who meets with whom.
Meeting_context	Entries about the place of meetings of each user.
Meeting_context_types	Available places of meetings that a user can select.
Meeting_dur	Entries about the total duration of the meetings.
Meeting_dur_types	Available choices of total duration about the meetings that a user can select.
Meeting_freq	Entries about the frequency of the meetings.
Meeting_freq_types	Available types of frequencies about the meetings that a user can select.
Message	Context of messages that a user can notice through the application.
Message_read	Messages that a user has already read.
Logs	Tracks concerning the behavior of the users within the application.

Tables of Facebook	
Photos_likes	**Incoming facebook activity**
Videos_likes	
Albums_likes	
Tag_videos_likes	
Tag_photos_likes	
Comments_photos	
Comments_videos	
Comments_albums	
Comments_tag_videos	
Comments_tag_photos	
Comments_posts	
Posts	
Photos_likes_others	**Outcoming facebook activity**
Videos_likes_others	
Albums_likes_others	
Tag_videos_likes_others	
Tag_photos_likes_others	
Comments_photos_others	
Comments_videos_others	
Comments_albums_others	
Comments_tag_videos_others	
Comments_tag_photos_others	
Comments_posts_others	
Posts_others	

Appendix B

Documentation of Methods

Application Class

- display ()

  Implements the state machine of the application. Takes over the interface configuration of the application based on the step that a Stumbl user is permitted to visit.

  - Parameters: none

  - Sql table: none

- render_0 ()

  Renders step 0 or advances to step 1. Configures the email address of the Stumbl user and checks if the user has facebook friends who also use Stumbl.

  - Parameters: none

  - Sql table: user

- render_1 ()

  Renders step 1 or advances to step 2. Presents the recommendation form in order the new Stumbl user to specify whether she joined the application due to a Stumbl invitation, submitting the relative inviter (facebook friend) too.

  - Parameters: none

  - Sql table: none

- render_2 ()

  Renders step 2 or advances to step 3. Presents all the facebook friends of the user and gives the possibility to select up to 20 of them as Stumbl friends. If the user does not select any of them, the function does not permit to proceed in the next step.

  - Parameters: none

  - Sql table: none

- render_3 ()

  Renders step 3. Presents all the Stumbl friends that the user has selected through the previous step 2, in order to specify for each one the relative relationship status.

  - Parameters: none

  - Sql table: none

- render_4()

  Presents the relative application interface for the step 4.

  - Parameters: none

  - Sql table: none

Log Class

- add_entry ()

  Adds a new log event for a specific Stumbl user in the database.

  - Parameters: $type: Kind of log event. (Defined in config.php file)
    $message: Description of log entry.
  - Sql table: logs

StumblUser Class

- initialize_user ()

  Checks if it is the first time that the user joins the application.

  - Parameters: none
  - Sql table: none

- get_uid ()

  Returns the facebook user id of the Stumbl user.

  - Parameters: none
  - Sql table: none

- get_stats_teaser ()

  Takes over the calculation of random statistics concerning the mobile and facebook activity of the participants.

  - Parameters: none
  - Sql table: none

- get_score ()

  Measures the current points of the Stumbl user based on the number of accepted invitations that he has sent, and the duration of her participation period.

  - Parameters: none
  - Sql table: none

- get_score_preview ()

  Presents the current score of the Stumbl user.

  - Parameters: none
  - Sql table: none

- get_email ()

  Returns the current email address of the Stumbl user.

  - Parameters: none
  - Sql table: none

- add_user ()

Adds a new Stumbl user in the database.

- Parameters: none
- Sql table: users

- get_reminder ()

  Checks if the reminder system is enabled or disabled.

  - Parameters: none
  - Sql table: none

- load_data ()

  Loads the Stumbl profile of the user.

  - Parameters: none
  - Sql table: users

- update_email ()

  Change the current email address of the Stumbl user in the database.

  - Parameters: $new_email: Mail address.
  - Sql table: users

- update_reminder ()

  Turns on or off the reminder system in the profile of the Stumbl user.

  - Parameters: $new_reminder: Takes zero or one to disable or enable the reminder system respectively.
  - Sql table: users

- update_session ()

  Updates the session_id of the Stumbl user in the database. According to the accepted permissions, the session id gives the possibility for offline access in the facebook profiles of the Stumbl users.

  - Parameters: $new_session: Session_id
  - Sql table: users

- update_last_access ()

  Updates in the database the timestamp of the last access of the Stumbl user in the application.

  - Parameters: none
  - Sql table: users

- update_last_step ()

  Updates the step that a Stumbl user is allowed to visit within the application in the database.

  - Parameters: $new_last_step: Takes 0 to 4 to describe the next step.
  - Sql table: users

- load_messages ()

Collects all the available and unread informational messages in order to be previewed to the Stumbl user.

- Parameters: none
- Sql table: message, message read

- remove_message ()

  Removes particular message from the message list of the Stumbl user.

  - Parameters: $msg_id: Unique id of an application message.
  - Sql table: none

- get_messages ()

  Returns the message list of the Stumbl user.

  - Parameters: none
  - Sql table: none

- get_friend ()

  Returns the friendship object of a Stumbl friend of the user.

  - Parameters: $f_id: Unique facebook id.
  - Sql table: none

- get_friend_rid ()

  Returns the friend id of a Stumbl friend.

  - Parameters: $r_id: Unique relation id of a pair (Stumbl user – facebook friend).
  - Sql table: none

- initialize_friends ()

  Checks if the Stumbl user has already a Stumbl-friend list in the database.

  - Parameters: none
  - Sql table: friendship

- update_friends ()

  Synchronizes the current facebook friend list with the stored facebook friend list in the database.

  - Parameters: none
  - Sql table: none

- update_stumbl_friends ()

  Synchronizes the current Stumbl friend list with the new stored facebook friend list in the database.

  - Parameters: $friend_string: A comma separated string of f_ids (facebook ids).
  - Sql table: none

- get_all_friends()

  Returns all the facebook friends.

- Parameters: none
- Sql table: none

- get_sorted_all_friends ()

  Returns all the facebook friends sorted by last name.

  - Parameters: none
  - Sql table: none

- get_sorted_nonselected_friends ()

  Returns a sorted list of new facebook friends that are not included in the stored facebook friend list in the database.

  - Parameters: none
  - Sql table: none

- get_uncategorized_friends ()

  Returns an array of selected Stumbl friends, the relationship type of which has not specified yet.

  - Parameters: none
  - Sql table: none

- get_stumbl_friends ()

  Returns an array of the Stumbl friends of the user.

  - Parameters: none
  - Sql table: none

- get_sorted_stumbl_friends ()

  Returns a sorted array of the Stumbl friends based on the last name.

  - Parameters: none
  - Sql table: none

- get_nr_fbfriends ()

  Returns the size of the facebook friend list of the Stumbl user.

  - Parameters: none
  - Sql table: none

- add_friends_db ()

  Adds a new facebook friend of the Stumbl user in the database.

  - Parameters: $new_friend: facebook id.
  - Sql table: friendship

- delete_friend_db ()

  Disables a specific Stumbl friend from the Stumbl friend list of the user, deleting any other entry (relationship, meetings) from the database.

  - Parameters: $old_friend: facebook id.

- Sql table: none

- load_friends ()

  Loads the facebook friends of the Stumbl user.

  - Parameters: none

  - Sql table: friendship

Friendship Class

- get_u_id ()

  Returns the facebook user id of the Stumbl user.

  - Parameters: none

  - Sql table: none

- get_in_list ()

  Returns 1, if the facebook friend is Stumbl friend too, otherwise 0.

  - Parameters: none

  - Sql table: none

- get_f_id ()

  Returns the facebook id of the facebook (Stumbl) friend.

  - Parameters: none

  - Sql table: none

- get_r_id ()

  Returns the r_id, which represents a unique pair of a Stumbl user – facebook friend.

  - Parameters: none

  - Sql table: none

- get_meeting ()

  Returns the object about the specified meeting between the Stumbl user and the Stumbl friend.

  - Parameters: none

  - Sql table: none

- add_invitation ()

  Each pair of a Stumbl user – facebook friend can be determined in the database by a unique identifier the r_id. When a Stumbl user sends an invitation to a facebook friend in order to join the application, and he responds, then the method stores this r_id in the database when the facebook friend decides to start the application.

  - Parameters: none

  - Sql table: invitations

- update_in_list()

Updates the in_list field in the database when a (new) Stumbl friend is added or deleted. It also updates her relationship and meeting status.

- Parameters: $new_in_list: Takes one or zero to assign or cancel a facebook friend as Stumbl friend respectively.
- Sql table: friendship

- delete_db()

  Deletes from the database a specific pair of a Stumbl user – facebook friend.

  - Parameters: none
  - Sql table: friendship

- load_relationships()

  Loads the list of the specified relations that concern the Stumbl friend.

  - Parameters: none
  - Sql table: relations

- get_relationship_labels()

  Returns an array with all the labels of the relationship with the Stumbl friend.

  - Parameters: none
  - Sql table: none

- has_relationship ()

  Returns the number of the specified types of relations with the Stumbl friend.

  - Parameters: none
  - Sql table: none

- has_relationship_type ()

  Returns true if a certain type of relationship is specified with the Stumbl friend, otherwise false.

  - Parameters: $type_id: Relation type id.
  - Sql table: none

- get_relationship_type ()

  Returns a relationship object based on a certain type of relation with a Stumbl friend. If there is not any type, it returns null.

  - Parameters: $type_id: Relation type id.
  - Sql table: none

- add_relationship ()

  Adds a new type of relationship with a Stumbl friend.

  - Parameters: $type_id: Relation type id.
  - Sql table: none

- del_relationship ()

Deletes from the database a relationship with a Stumbl friend based on the relation type.

- Parameters: $type_id: Relation type id.
- Sql table: none

- destroy_relationship ()

  Destroy a relationship object of a Stumbl friend of a Stumbl user.

  - Parameters: none
  - Sql table: none

- load_meeting ()

  Returns a meeting object with a Stumbl friend based on the current date or a requested date.

  - Parameters: $m_date: Timestamp, Y-m-d, Y (a full numeric representation of a year, 4 digits) -m (numeric representation of a month, with leading zeros) - d (day of the month, 2 digits with leading zeros).
  - Sql table: none

- destroy_meeting ()

  Deletes a meeting object of a Stumbl friend of the Stumbl user.

  - Parameters: none
  - Sql table: none

Meeting Class

- has_data ()

  Returns 0 if neither meeting frequency nor duration have been specified for a meeting with a Stumbl friend. Returns 1 if only the meeting frequency has been specified otherwise it returns 2 if both meeting frequency and duration have been specified.

  - Parameters: none
  - Sql table: none

- get_m_id ()

  Checks if there is a meeting entry with a Stumbl friend in the database.

  - Parameters: $r_id: Unique relation id of a pair (Stumbl user – facebook friend).
  - Sql table: meeting

- add_meeting_db ()

  Adds a meeting event with a Stumbl friend in the database.

  - Parameters: $r_id: Unique relation id of a pair (Stumbl user – facebook friend).
  - Sql table: meeting

- reset_meeting ()

  Deletes the frequency and duration about a meeting with a Stumbl friend from the database.

- Parameters: none
- Sql table: meeting_freq, meeting_dur

- load_frequency ()

  Loads the frequency of a meeting with a Stumbl friend from the database.

  - Parameters: none
  - Sql table: meeting_freq

- delete_frequency ()

  Deletes the frequency of a meeting with a Stumbl friend from the database.

  - Parameters: none
  - Sql table: meeting_freq

- add_frequency ()

  Specifies the frequency value of a meeting with a Stumbl friend, adding a meeting frequency entry in the database as well.

  - Parameters: $m_f_id: Meeting frequency type id.
  - Sql table: meeting_freq

- get_frequency_label()

  Returns the label of the specified meeting frequency with a Stumbl friend.

  - Parameters: none
  - Sql table: meeting_freq_types

- get_m_f_id ()

  Returns the specified type of the frequency of the meeting with a Stumbl friend.

  - Parameters: none
  - Sql table: none

- load_duration ()

  Loads the duration of a meeting with a Stumbl friend from the database.

  - Parameters: none
  - Sql table: meeting_dur

- delete_duration ()

  Deletes the duration of a meeting with a Stumbl friend from the database.

  - Parameters: none
  - Sql table: meeting_dur

- add_duration ()

  Specifies the duration of a meeting with a Stumbl friend, adding a meeting duration entry in the database as well.

  - Parameters: $m_d_id: Meeting duration type id.
  - Sql table: meeting_dur

- get_duration_label ()

  Returns the label of a specified meeting duration of the meeting with a Stumbl friend.

  - Parameters: none

  - Sql table: meeting_dur_types

- get_m_d_id ()

  Returns the specified type of the meeting duration with a Stumbl friend.

  - Parameters: none

  - Sql table: none

- get_frequency_types ()

  Returns all the active types about the meeting frequency.

  - Parameters: none

  - Sql table: meeting_freq_types

- get_duration_types ()

  Returns all the active types about the meeting duration.

  - Parameters: none

  - Sql table: meeting_dur_types

- get_context_types ()

  Returns all the active types about the context of the meetings.

  - Parameters: none

  - Sql table: meeting_context_types

Relationship Class

- get_type_id ()

  Returns the specified relationship type for a Stumbl friend.

  - Parameters: none

  - Sql table: none

- get_label ()

  Returns the label of the specified relation type of the relationhip with a Stumbl friend.

  - Parameters: none

  - Sql table: relationship_types

- get_relationship_types ()

  Returns all the active types, which determine a relationship.

  - Parameters: none

  - Sql table: relationship_types

- add_relationship_db ()

Adds a new relationship entry for a Stumbl friend in the database.

- Parameters: none
- Sql table: relations

- del_relationship_db ()

  Deletes a relationship entry for a Stumbl friend from the database.

  - Parameters: none
  - Sql table: relations

References

[1] Facebook developer platform. http://developers.facebook.com/

[2] http://wiki.developers.facebook.com/index.php/
Choosing between an FBML or IFrame Application

[3] http://jess3.com/geosocial-universe/

[4] Wasserman S., Faust K. *(1994) Social Network Analysis: Methods and Applications* (Cambridge Univ Press, New York).

[5] Wikipedia, Social Network, http://en.wikipedia.org/wiki/Social_network.

[6] Nazir, A., Raza, S., and Chuah, C.-N. Unveiling Facebook: A Measurement Study of Social Network Based Applications. In *Proc. ACM Internet Measurement Conference* (2008).

[7] Nielsen Online Report. Social networks & blogs now 4th most popular online activity, 2009.

[8] Facebook Application directory, http://www.facebook.com/apps/directory.php

[9] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *ACM EuroSys*, 2009.

[10] Facebook. http://en.wikipedia.org/wiki/Facebook

[11] Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. Characterizing user behavior in online social networks. In *Proc. ACM IMC* (2009).

[12] Schneider, F., Feldmann, A., Krishnamurthy, B., and Willinger, W. Understanding online social network usage from a network perspective. In*Proc. of ACM Internet Measurement Conference* (2009).

[13] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks.In *ACM/USENIX IMC*, 2007.

[14] M.E.J. Newman. The Structure and Function of Complex Networks. In *SIAM Review*, 2003

[15] A. Mtibaa, A. Chaintreau, J. LeBrun, E. Oliver, A.-K. Pietilainen, and C. Diot. Are you moved by your social networks application? In *WOSN'08: Proceedings of ACM SIGCOMM Workshop on Online Social Network*, August 2008.

[16] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of Topological Characteristics of Huge Online Social Networking Services. In Proceedings of the 16th international conference on World Wide Web (WWW'07), Banff, Canada, May 2007.

[17] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274– 15278, 2009.