THE DEVELOPMENT OF A SOCIAL NETWORK ANALYSIS SOFTWARE TOOL

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Cansoy SOYDAN

July 2009
İZMİR

We approve the thesis of **Cansoy SOYDAN**

_____

Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Supervisor

_____

Prof. Dr. Şaban EREN
Committee Member

_____

Assist. Prof. Dr. Tuğkan TUĞLULAR
Committee Member

03 July 2009

_____

Prof. Dr. Sıtkı AYTAÇ
Head of the Computer Engineering
Engineering of Department

_____

Prof. Dr. Hasan BÖKE
Dean of the Graduate School of
Engineering and Science

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Assoc. Prof. Dr. Ahmet KOLTUKSUZ, for his guidance, patience, and encouragement. His valuable support, and confidence have been the driving force of this thesis work.

I also like to express my gratitude to Selma TEKİR, who had a real effort in helping me in the preparation of this thesis work. Her thoughtful advise often served to give me a sense of direction during my studies. I am deeply grateful for her support.

I would like to thank Türkiye Bilimsel ve Teknolojik Araştırma Kurumu for the scholarship throughout in my graduate study.

Finally, I should thank to my wife, Aslı SOYDAN, for her confidence in me, her support and her encouragement.

ABSTRACT

THE DEVELOPMENT OF A SOCIAL NETWORK ANALYSIS SOFTWARE TOOL

Nowadays the amount of spam is increasing in an uncontrollable way. This situation decreases the email trust of the Internet users and reduces the usability of the emails to the critical level. On the other hand, this makes a lot of money loss for the hosting companies. There are many anti-spam tools that are developed against spammers. Some of these tools are really succesful. However; since the spammers improve their techniques, spams gain immune to these tools. In this thesis, the problem of detecting spams in in-coming and out-going emails is adressed.. To achieve this goal, an email social network is constructed by using the traffic of emails between users. While constructing this network, only information that can be gained from the email structure is used. The results on the real data set show that the techniques applied have been effective and also point to new directions of research in this area.

ÖZET

BİR SOSYAL AĞ ANALİZİ YAZILIM ARACININ GELİŞTİRİLMESİ

Günümüzde spam sayısı kontrol edilemez derecede artmaktadır. Bu durum, internet kullanıcılarının e-postalara olan güvenini azaltmakta ve e-postaların kullanılabilirliğini kritik seviyeye düşürmektedir. Öte yandan servis sağlayıcıları için çok büyük miktarda finansal olarak zarar teşkil etmektedir. Bu durum çok fazla anti-spam araçlarının geliştirilmesine yol açmıştır. Bu araçların bazıları önemli oranda da başarılı olabilmektedir. Fakat spammerların tekniklerini her geçen gün geliştirmeleri bu araçlara karşı spammerlara önemli bir bağışıklık kazandırmaktadır. Bu çalışmada gelen e-postalarda spam bulma problemini ele aldık. Bu amaca ulaşmak için, kullanıcıların birbirleri arasındaki e-posta trafiğini kullanarak bir e-posta sosyal ağı oluşturuldu. Bu ağ oluşturulurken sadece e-posta yapısı içerisinden elde edilebilinen veriler kullanıldı. Gerçek veri seti üzerindeki çalışmalarımız sonucunda oluşan değerler uygulanan tekniğin etkili olduğunu ve bu konuyla ilgili yeni araştırmaların yapılabileceğini gösterdi.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Unsolicited commercial email, spam, is not a new problem causing complaints from many Internet users. Spam is increasingly being used to distribute virus, spyware, links to phishing web sites, etc. The problem of spam is not only an annoyance, but is also becoming a security threat. In this thesis a new point of view for the increasing threat is presented using social network structure.

A social network is a description of the social structure between actors, mostly individuals or organizations. It indicates the ways in which they are connected through various social familiarities ranging from casual acquaintance to close familiar bonds. Social networks and their analysis is becoming a rapidly emerging field of research. Massive quantities of data on large social networks are available from blogs, knowledgesharing sites, collaborative filtering systems, online gaming, social networking sites, newsgroups, chat rooms, e-businesses and so on (Staab, et al. 2005).

In this work a tool is implemented by using the properties of social networks The tool works on the basis of trust. An email social network is constructed by using the traffic of emails between users. By using the properties of the network it's planned to detect the people who the user should trust (whitelists) and should not trust (blacklists). While making this network there is no work for the user or the hosting companies. The goal is to give a meaning to the existing network. This network will be formed by only using information in the email headers. There is no need to the other information. The performance of the tool comes from this point.

This work aims to increase the success rate of creating whitelists and blacklists and also to decrease the number of people in gray lists. To achieve this goal, only information that can be gained from the email structure was used.

Chapter 2 provides background information about social network analysis and its metrics; mentions about some experiments about social networks and email social networks. Then spam and non-spam concepts are reviewed and related works are mentioned. In chapter 3 the structure of an email header is examined and how to read an email header is shown. In chapter 4 the application architecture of the tool is

analyzed. The algorithms and the results are summarized. Finally, chapter 5 gives the future work and chapter 6 gives and the conclusion of the thesis.

CHAPTER 2

BACKGROUND INFORMATION

2.1. Social Network Analysis

Social network analysis is the mapping and measuring of relationships and flows between people, groups, organizations, computers, web sites, and other information/knowledge processing entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. Social network analysis provides both a visual and a mathematical analysis of human relationships (Org Net 2009).

To understand networks and their participants, location of actors are evaluated in the network. Measuring the network location is finding the centrality of a node. These measures give us insight into the various roles and groupings in a network -- who are the connectors, mavens, leaders, bridges, isolates, where are the clusters and who is in them, who is in the core of the network, and who is on the periphery (Org Net 2009).

The network shows the distinction between the three most popular individual centrality measures: Degree Centrality, Betweenness Centrality, and Closeness Centrality (Org Net 2009).

2.1.1. Social Network Analysis Metrics

2.1.1.1. Degree Centrality

Degree centrality is simply the number of direct relationships that an entity has. An entity with high degree centrality (FMS Advanced System Group 2009):

- Is generally an active player in the network.
- Is often a connector or hub in the network.
- Is not necessarily the most connected entity in the network (an entity may have a large number of relationships, the majority of which point to low-level entities).

- May be in an advantaged position in the network.
- May have alternative avenues to satisfy organizational needs, and consequently may be less dependent on other individuals.
- Can often be identified as third parties or deal makers (FMS Advanced System Group 2009).

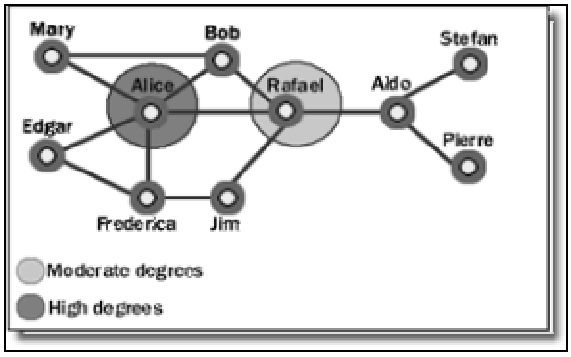

Figure 2.1. Degree Centrality Graph.
(Source: FMS Advanced System Group 2009)

2.1.1.2. Betweenness Centrality

Betweenness centrality identifies an entity's position within a network in terms of its ability to make connections to other pairs or groups in a network. An entity with a high betweenness centrality generally (FMS Advanced System Group 2009):

- Holds a favored or powerful position in the network.
- Represents a single point of failure—take the single betweenness spanner out of a network and you sever ties between cliques.
- Has a greater amount of influence over what happens in a network (FMS Advanced System Group 2009).

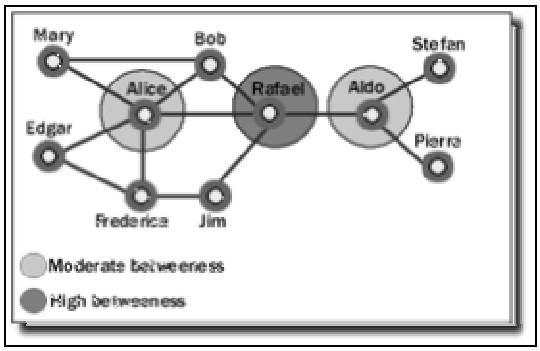

Figure 2.2. Betweenness Centrality Graph
(Source: FMS Advanced System Group 2009)

2.1.1.3. Closeness

Closeness centrality measures how quickly an entity can access more entities in a network. An entity with a high closeness centrality generally (FMS Advanced System Group, 2009):

- Has quick access to other entities in a network.
- Has a short path to other entities.
- Is close to other entities.
- Has high visibility as to what is happening in the network (FMS Advanced System Group 2009).

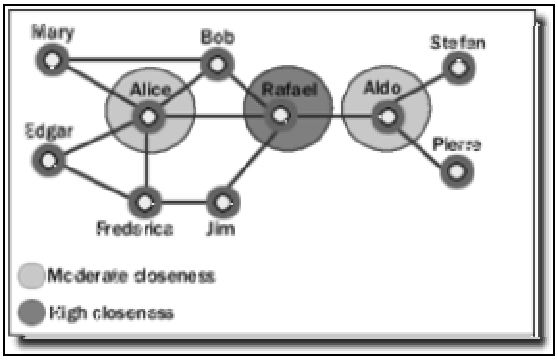

Figure 2.3. Closeness Graph
(Source: FMS Advanced System Group 2009)

2.1.2. Social Network Analysis Experiments

There have been many experiments about social networks and email social networks. In this part some of the most important experiments have been summarized. These works also have been used as a guide book for this project.

2.1.2.1. Analysis Of Sigmod's CoAuthorship Graph

In the work of 'Analysis of SIGMOD's CoAuthorship Graph', it is aimed to investigate the co-authorship graph obtained from all papers published at SIGMOD between 1975 and 2002 so that to identity of the authors who, are closest" to all other authors at a given time. It is also showned that SIGMOD's co-authorship graph is yet another example of a small world which has received a lot of attention recently (Nascimento and Pound 2003).

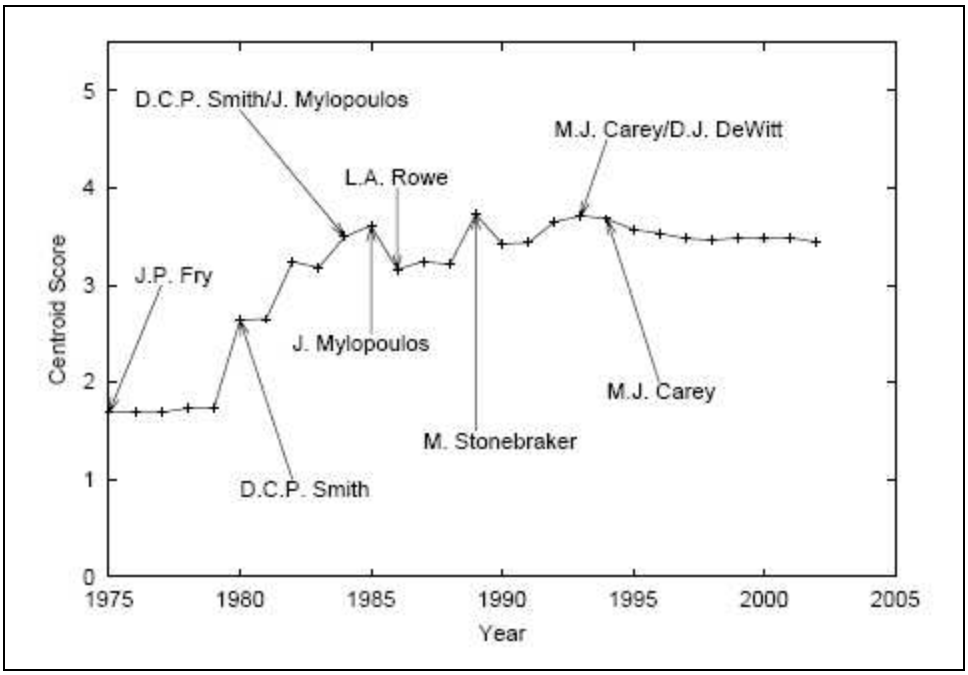

Figure 2.4. Minimum average path length over time
(Source: Nascimento and Pound 2003)

2.1.2.2. Small Worlds

Long a matter of folklore, the “small-world phenomenon” the principle that we are all linked by short chains of acquaintances was inaugurated as an area of experimental study in the social sciences through the pioneering work of Stanley Milgram in the 1960's. This work was among the first to make the phenomenon quantitative, allowing people to speak of the “six degrees of separation” between any two people in the United States. Since then, a number of network models have been proposed as frameworks in which to study the problem analytically. One of the most refined of these models was formulated in recent work of Watts and Strogatz ; their framework provided compelling evidence that the small-world phenomenon is pervasive in a range of networks arising in nature and technology, and a fundamental ingredient in the evolution of the World Wide Web (Kleinberg 2000).

Milgram's basic small-world experiment remains one of the most compelling ways to think about the problem. The goal of the experiment was to find short chains of acquaintances linking pairs of people in the United States who did not know one another. In a typical instance of the experiment, a source person in Nebraska would be given a letter to deliver to a target person in Massachusetts. The source would initially be told basic information about the target, including his address and occupation; the source would then be instructed to send the letter to someone she knew on a first-name basis in an effort to transmit the letter to the target as efficaciously as possible. Anyone subsequently receiving the letter would be given the same instructions, and the chain of communication would continue until the target was reached. Over many trials, the average number of intermediate steps in a successful chain was found to lie between five and six, a quantity that has since entered popular culture as the “six degrees of separation” principle (Kleinberg 2000).

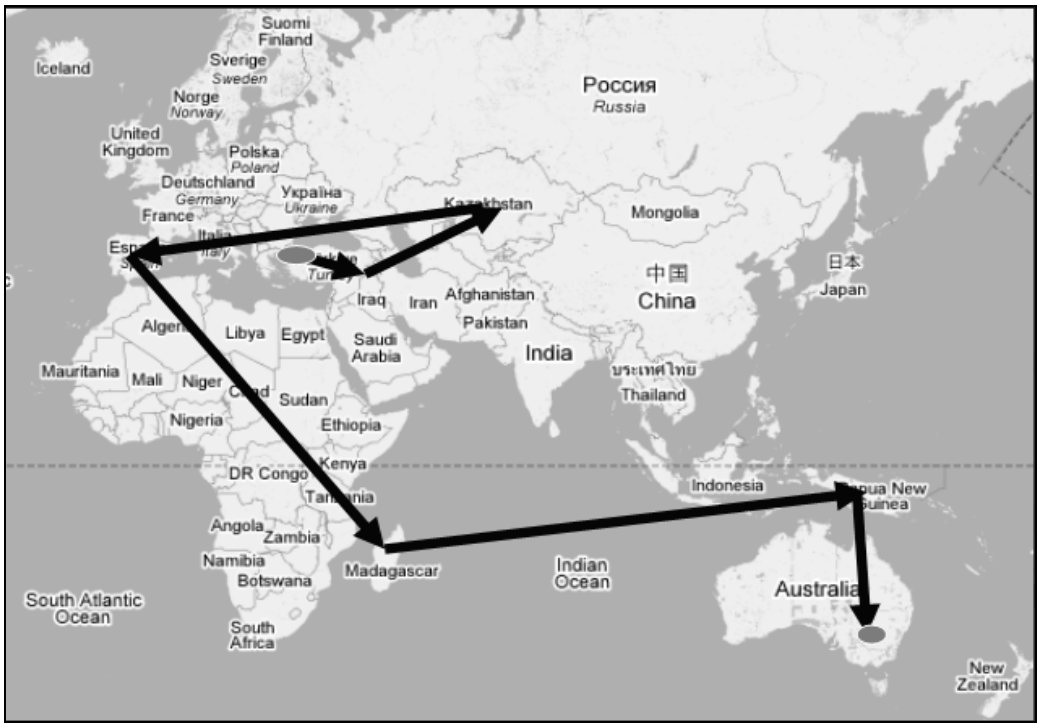

Figure 2.5. Six degrees of seperation

2.1.2.3. Semantic Analytics On Social Networks: Experiences In Addressing The Problem Of Conflict Of Interest Detection

The work of 'Semantic Analytics on Social Networks' describes a Semantic Web application that detects Conflict of Interest (COI) relationships among potential reviewers and authors of scientific papers. This application discovers various 'semantic associations' between the reviewers and authors in a populated ontology to determine a degree of Conflict of Interest. This ontology was created by integrating entities and relationships from two social networks, namely "knows," from a FOAF (Friend-of-a-Friend) social network and "co-author," from the underlying co-authorship network of the DBLP bibliography (Boanerges, et al. 2006).

Conflict of Interest (COI) is typically known as a situation that may bias a decision. It is important to detect COI for transparency in circumstances such as peer-review of scientific research papers or proposals (Boanerges, et al. 2006).

2.1.2.4. Mining Email Social Networks In OSS

In the work of ‘Mining Eail Social Networks in OSS’ data is mined from both the source code repository and the mailing list archives to examine the relationship between communication and development in an OSS project. A number of social network analysis measures and statistical techniques are used to analyze this data (Bird, et al. 2006).

- By using social network analysis the aim of this work is to determine
- level of email activity
- level of activity in the source code
- document change activity
- who plays significant social role (Bird, et al. 2006).

After mining and analyzing mailing list and source code repository data for the Project the observations from the network are

- The mailing list activity reflects a typical social network.
- Developers are the “key social brokers”.
- More active developers tend to be more important (Bird, et al. 2006).

2.1.2.5. Leveraging Social Networks To Fight Spam

In the work of ‘Leveraging Social Networks to Fight Spam’, a network is formed by using the header information of the emails. Using the some metrics of the network; whitelists, blacklists and gray lists are created. This study also shows the way to our study about how to construct an email social network (Boykin and Roychowdhury 2005).

2.2. Spam

In this part some answers are given to the questions like why spam is a problem what spam cost. Some background information is given about spam and spamming methods.

2.2.1. Definition Of Spam

The word spam means junk mails. The unsolicited emails that are received by any person in his / her mailbox are called spam. These junk mails are usually sent in bulk for advertising and marketing some products. Lots of space in your mailbox is occupied by these junk mails. Sometimes it eats up your valuable space so that the genuine mails are bounced back to the sender if the whole lot of space is occupied by the junk mails. Hence there comes a necessity to filter out those junk mails from your mailbox (Bean Software 2009).

Just by looking at the emails that you receive you may notice that some of the emails are not meant for you. Although they have been addressed to you, you may know that the sender is unknown to you and the subject of the email is something that you have not expected. Such emails in your inbox are spam. These spams are meant to promote some product or service. It is easy for a human to go through the mails that they have received and identify them as spam although it is a tough job for the software that are meant to filter spam (Bean Software 2009).

2.2.2. Considering Spam As A Problem

Spammers use tricks to force thousands of pieces of junk mail through someone else's mail servers, often looking as if they originated there. Thus, your Internet Service Provider (ISP) may appear to be the source of the spam when, in fact, they were a victim (Eagle-Ing. 2009).

Your ISP must pay for the data lines, the servers, and the disk drive arrays needed to carry all of its traffic, including spam. ISPs maintain an Abuse department that fields spam complaints and that tries to shut down any spammers using its resources. And ISPs pay for a technical support staff to field calls from customers, many of whom call to complain about spam (Eagle-Ing. 2009).

Some companies were reporting that over 50% of the mail received by their mail servers was spam. We routinely receive 30-50 pieces of junk mail each day. It has become a major problem for us. At one point the spam was causing the server to re-send messages repeatedly, thus leaving us with large numbers of duplicate messages (and spam) each day (Eagle-Ing. 2009).

Your ISP must pay for the extra staff and equipment needed to deal with spam. Those costs are passed on to their customers. So, not only do you have to spend time processing the junk mail you receive, you are paying for the privilege. Worse, if your ISP is used by a spammer they could be "black listed", meaning that other ISPs may refuse to accept mail from them (Eagle-Ing. 2009).

It costs spammers almost nothing to send millions of pieces of junk mail. It costs the ISPs and their customers quite a lot to deal with the spam they send. A number of free ISPs and email providers no longer offer those services or they have cut back on what they will provide for free, in large part due to the costs of spam. (Eagle-Ing. 2009).

2.2.3. Spamming Methods

2.2.3.1. Direct Spamming

Spammers may purchase upstream connectivity from “spam-friendly ISPs”, which turn a blind eye to the activity. Occasionally, spammers buy connectivity and send spam from ISPs that do not condone this activity and are forced to change ISPs. Ordinarily, changing from one ISP to another would require a spammer to renumber the IP addresses of their mail relays. To remain untraceable and avoid renumbering headaches, spammers sometimes obtain a pool of dispensable dialup IP addresses, send outgoing traffic from a high-bandwidth connection the IP address spoofed to appear as if it came from the dialup connection, and proxy the reverse traffic through the dialup connection back to the spamming hosts. (Ramachandran and Feamster 2006)

2.2.3.2. Open Relays and Proxies

Open relays are mail servers that allow unauthenticated Internet hosts to connect and relay email through them. Originally intended for user convenience (e.g., to let users send mail from a particular relay while they are traveling or otherwise in a different network), open relays have been exploited by spammers due to the anonymity and amplification offered by the extra level of indirection. (Ramachandran and Feamster 2006)

2.2.3.3. Botnets

Collections of machines acting under one centralized controller allows infected hosts to be used as a mail relay, and attempts to spread itself to other machines affected by the vulnerabilities over email. (Ramachandran and Feamster 2006)

CHAPTER 3

EMAIL SOCIAL NETWORK ANALYSIS

3.1. Email Header Analysis

In an email we can have so much information about an email like, who sent the email, who received the email, route of the email, sent date. This information is very helpful for many researches. To get this information it is necessary to know how to read and analyze an email header.

3.1.1. Definition Of Email Header

Every email comes with a header which is one part of an email structure. It has basic information such as from whom the email comes, to whom it is addressed, date/time it was sent and the subject of the email. The detailed technical information can be viewed in a full header (MyCert 2009).

Full header will have information such as the mail servers name that the email passed through on its way to the recipient, recipient and sender's IP address and even the name of the email program and its version used (MyCert 2009). In Figure 3.1 full header information of an email is showned.

```
Return-Path: ass@relay13.jaring.my
Received: from relay13.jaring.my (relay13.jaring.my
[192.228.128.124])
by ace.cdc.abu.com (8.7.1/8.7.1) with ESMTP id KAA18533
for <jacob@ace.cdc.abu.com>; Fri, 8 May 1998 10:01:01
+0800
Received: from hole.com (j19.kch18.jaring.my
[161.142.54.153])
by relay13.jaring.my (8.8.8/8.8.7) with SMTP id KAA21792
for <jacob@ace.cdc.abu.com>; Fri, 8 May 1998 10:05:21
+0800 (MYT)
Date: Fri, 8 May 1998 10:05:21 +0800 (MYT)
From: ass@pc.jaring.my
Message-Id: <199805080205.KAA21792@relay13.jaring.my>
To: jacob@ace.cdc.abu.com
Subject: happy holiday
Status: RO
X-Status:
```

Figure 3.1. Email full header information.
(Source: MyCert 2009)

3.1.3. Reading Of An Email Header

The header contains the "name" and "address" of the sender, recipient and anyone who is being copied, the "date" and "time" the mail is sent and the "subject" of the mail. The header exists mainly for the computer to route mail to you. The "received:" item indicates the mailers. It shows what mailers the mail is routed through before it goes to the recipient. Usually, over the internet, the mail will go through several mailers before it finally reaches the recipient. This information will help in tracing the source IP address of the sender (MyCert 2009).

The Return-Path line mean the address in which the reply for this mail will be sent to.The received lines were the routing information which told where the mail went and the time it arrived to the respective mailer. In order to follow the flow, they had to be read backwards. So, the particular mail originated from hole.com and mailed to relay13.jaring.my. Further, it went to ace.cdc.abu.com which was the recipient's Internet host. So, if your mail bounced, this part in the header showed how far the mail went and which machine rejected it.The message-Id line was intended mainly for tracing mail routing and uniquely identified each mail. The 'From' line showed who sent the mail and his/her email. This 'From' information can be easily faked/forged. The 'To' line listed the email address/es of the recipients of the mail. There might be also a

Cc line which listed all the people who received copies of this mail. This address could also be a hidden list of emails; thus your email may not appear in here even though you received the mail. The subject line gave some idea of what the mail is about. The Date line lists the date and time this mail was originally sent. It was sent on the sender's local time zone (MyCert 2009).

CHAPTER 4

APPLICATION ARCHITECTURE

4.1. Architecture Of The Application

In this work, it is aimed to determine spam components and non-spam components of the email social network which is created by the information of the email headers. To make this decision clustering coefficient values are calculated of these components.

A social network's most distinctive property is the tendency to cluster. Using this property a graph is created by using 'To', 'Cc' and 'Bcc' header information. This graph shows the connections of the people who have sent us emails and who we have sent emails. After that, the graph is divided into the components and then clustering coefficient values are calculated of these components.

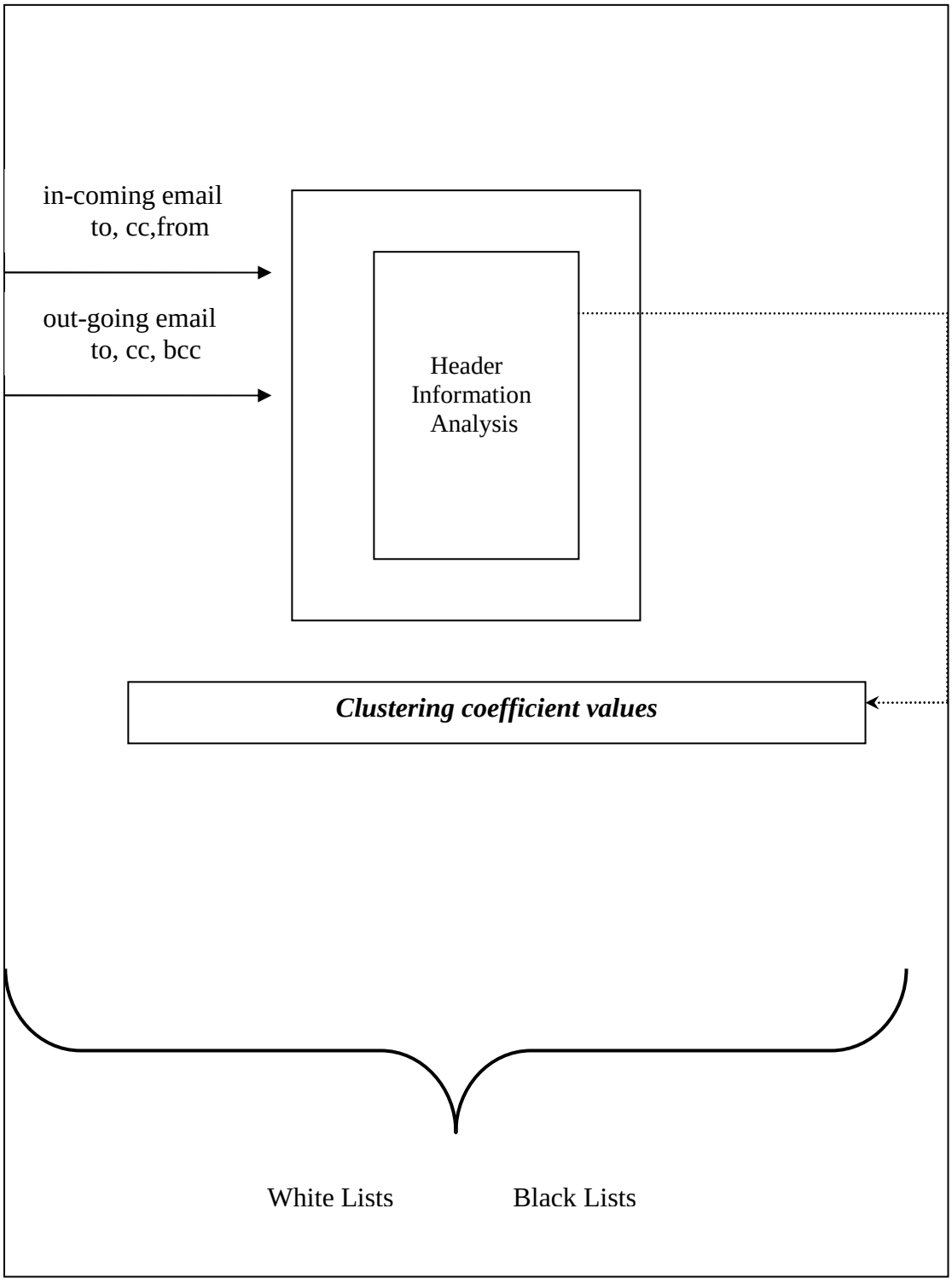

Figure 4.1. Application Architecture.

In Figure 4.1 the application architecture is shown. The application only needs email datasets which consists of ‘from’, ‘to’, ‘cc’ and ‘bcc’ header information. Then application analyzes these datasets and creates the whitelists and blacklists using the

clustering coefficient values of the emails. Here is the algorithms of the analysis part of the application.

4.2. Algorithm Of The Application

4.2.1. Main Algorithm

1- Loop for each email in inbox
    1.1- Put 'To' and 'Cc' information into 'received list'
    1.2- Remove owner from 'received list'
    1.3- Create nodes for 'from' header and the email adresses in 'received list'
    1.4- Make connections between from node and the nodes in 'received list'
2- Loop for each email in outbox
    2.1- Put 'To', 'Cc', 'Bcc' information into 'sent list'
    2.2- Create nodes for the email adresses in 'sent list'
    2.3- Make connections between the nodes in 'sent list'
3- Loop for each node in graph
    3.1- Calculate the clustering coefficient value of each node
3.2- Add this node and this node's connected nodes in a connected component(recursive) (In Section 4.2.3)
4- Loop for each connected component
    4.1- Calculate the average clustering coefficient value of the component
5- Draw the graph
6- Display the data

Figure 4.2. The Main Algorithm

In Figure 4.2 the main algorithm is shown. In the first part of the algorithm nodes are created from each inbox email's 'from', 'to' and 'cc' header. Then the owner node is removed since we are interested in friends' social network. We do not need the owner of the email. After that, connections are created between from node and the other cc and to nodes. In Figure 4.3 how to remove owner is shown for the first part of the algorithm.

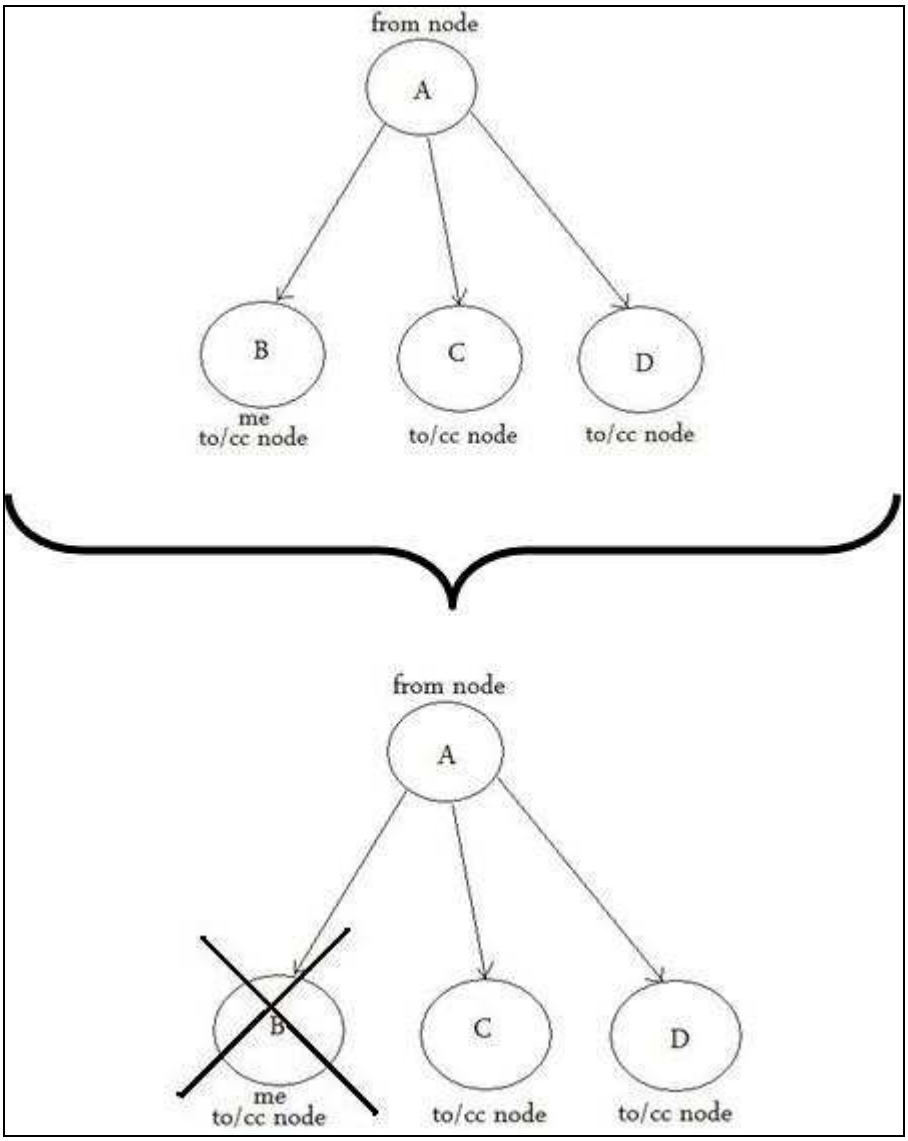

Figure 4.3. Connections Between 'From' Nodes and To Nodes, 'CC' Nodes.

In the second part of the algorithm nodes are created from each sent email's 'to', 'cc', and 'bcc' header. Then connections are created between the nodes. They are connected since as mentioned in 'friend of a friend' project (Foaf Project 2009) if we send an email to some people we can establish relations between the people in that email. In Figure 4.4 how to remove owner and how to make connections between nodes are shown for the second part of the algorithm.

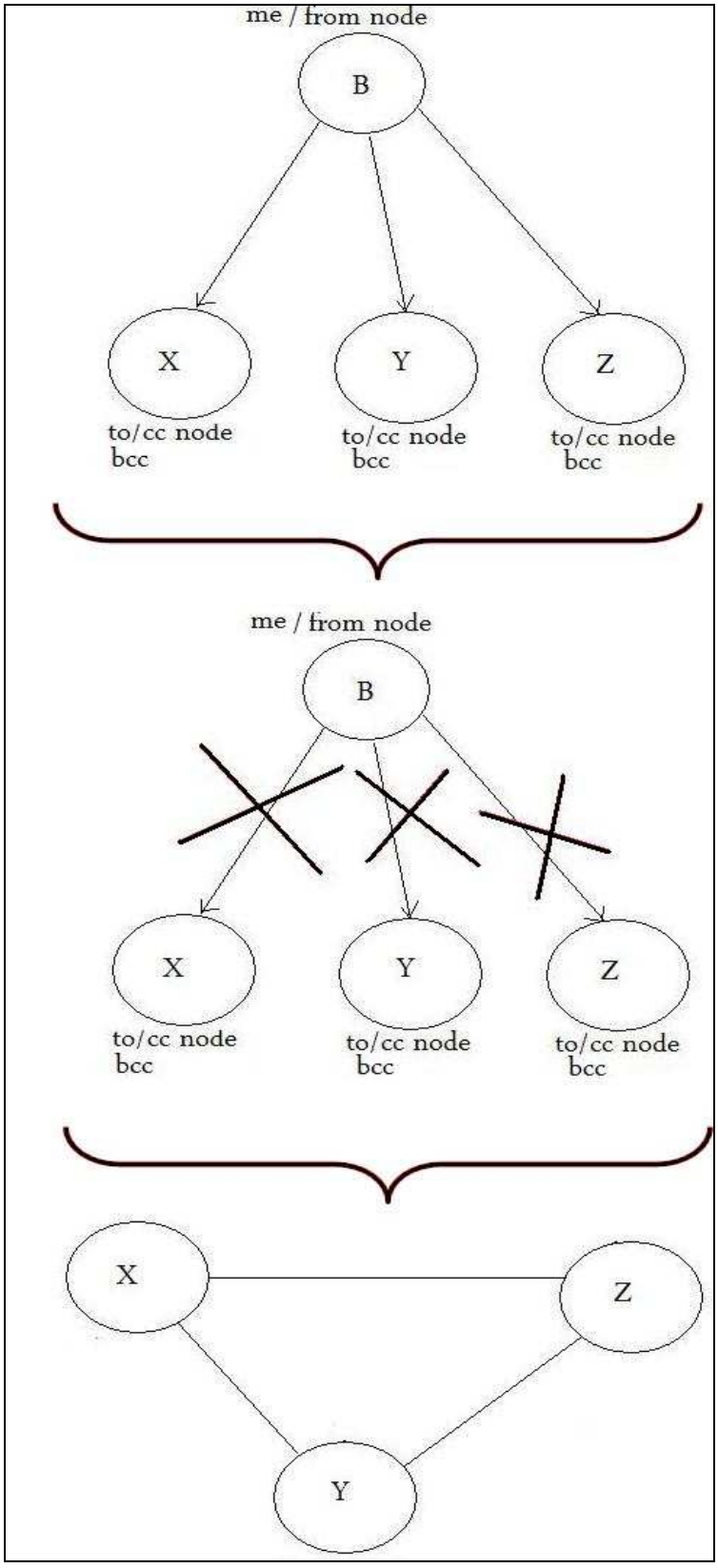

Figure 4.4. Connections Between ‘To’ Nodes, ‘CC’ Nodes and ‘BCC’ Nodes.

In the third part, clustering coefficient value of each node is calculated and nodes are divided into components.

Lastly average clustering coefficient values of the components are calculated, the graph is drawn and the clustering coefficient values of the components are displayed.

In this algorithm the important parts are calculating clustering coefficient values and dividing the network to the components.

4.2.2. Calculating Clustering Coefficient Value of A Node

```
1- edgesInNbd equals to zero
2- i equals to zero
3- While i is less than number of the neighbours of the node
3.1- k equals to zero
3.2- While k is less than i
                    3.2.1- If the i th neighbour and k th neighbour has a connection
                    3.2.2- Add one to edgesInNbd
4- return edgesInNbd * 2.0 / (node's connections' Count * (node's connections' Count - 1))
```

Figure 4.5. Calculating Clustering Coefficient Algorithm.
(Source: Pemmaraju 2008)

In Figure 4.5 Calculating Clustering Coefficient Algortihm is shown. The clustering coefficient value of a node is calculated by dividing the number of connections that node's neighbours have, by possible connection number that node's neighbours can have. In Figure 4.6 an example of relations of the nodes are showned.

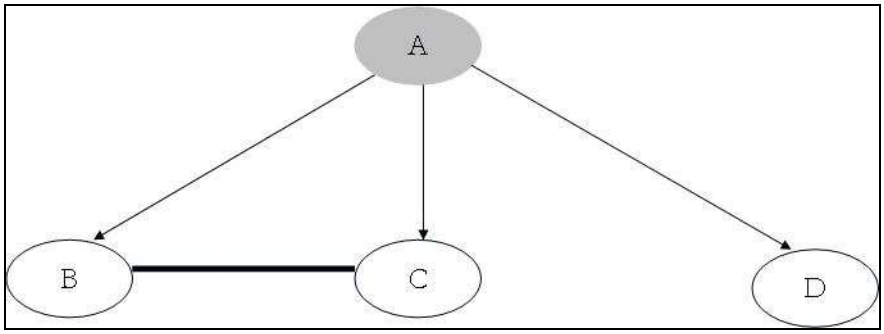

Figure 4.6. Connections Between Nodes.

In Figure 4.6 to calculate the clustering coefficient value of node A first of all the connections that node A's neighbours have should be counted. When the network is examined it is seen that there is one connection between node B and node C. So the first number is 1. The possible connection number that node A's neighbours can have is 3 (from the formula n*(n-1) / 2).When the first number is divided by the second the clustering coefficient value of node A is calculated. The result is 0.333.

4.2.3. Dividing The Network To The Components

```
1- Loop for each node
1.1-If this node does not exist in any connected component
1.1.1-Create a new connected component(List)
1.1.2-Call AddNodeToConnectedComponent(this node, createdConnectedComponent)
1.2-End
2-End
AddNodeToConnectedComponent(node, ConnectedComponent)
1-If this node does not exist in this ConnectedComponent
1.1-Add this node to this ConnectedComponent
2-Else return
3-Loop for each neighbour node of this node
3.1-Call AddNodeToConnectedComponent(this neighbour node, this ConnectedComponent)
4-End
```

Figure 4.7. Dividing The Network To The Components.

In Figure 4.7 dividing the network to the components is shown. In the first part of the algorithm it is controlled that if a node is in any connected component. If it is not, a component is created for this node and the neighbour nodes.

In the second part, component gets bigger with the neigbour nodes until it reaches the first node. A recursive algorithm is run in this part.

4.3. Application Design

Unified Modeling Language(UML) is a language or notation intended for analysing, describing and documenting all aspects of a software intensive system (Sintef 2009).

The purpose of a use case diagram to identify and partition system functionality. They separate the system into actors and use cases. Actors represent roles that can are played by users of the system. Use cases describe the behavior of the system when one of these actors sends one particular stimulus (Object Mentor 2009).

The purpose of a class diagram is to depict the classes within a model. In an object oriented application, classes have attributes, operations and relationships with other classes. The UML class diagram can depict all these things quite easily (Object Mentor 2009).

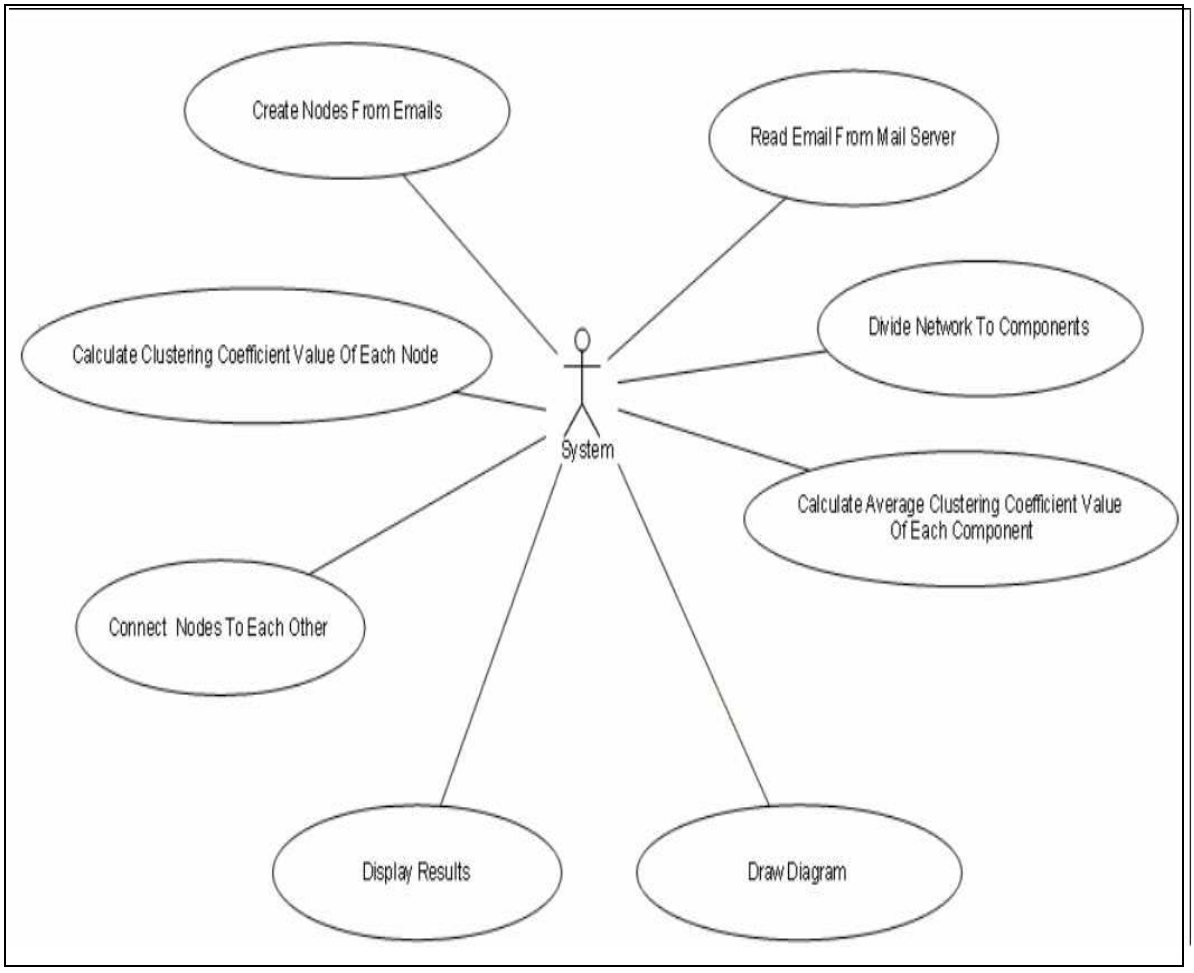

Figure 4.8. UML-Use Case Diagram

In Figure 4.8 UML-Use Case diagram of the application is shown. The application firstly get emails from mail server. According to the algorithm, for emails, nodes are created and connected to each other. The clustering coefficient values of each nodes are calculated. Then the constructed network is divided to the connected components. Average clustering coefficient values of the connected components are

calculated and lastly diagram and the results are displayed.

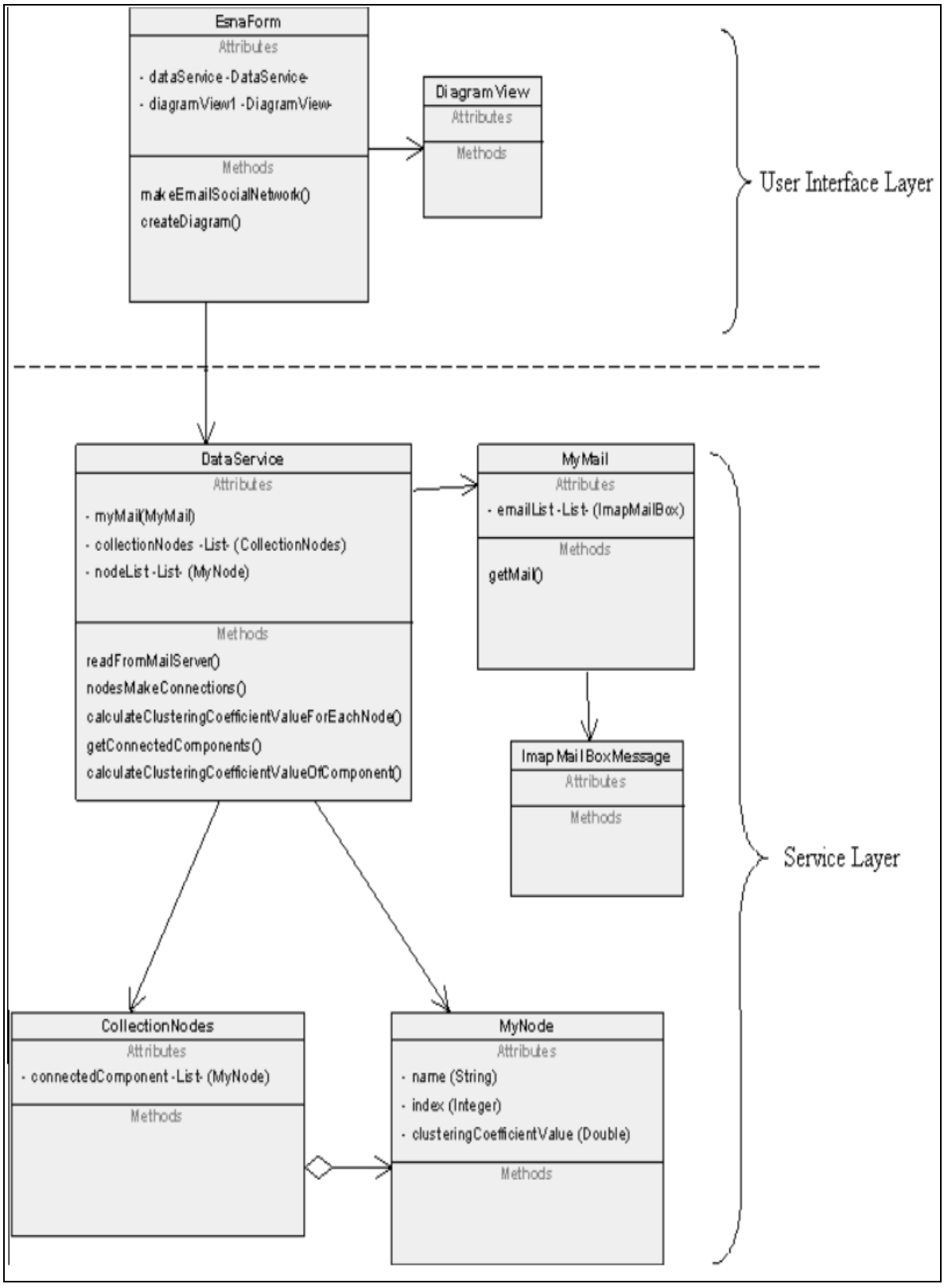

Figure 4.9. UML Class Diagram.

In Figure 4.9 Uml Class Diagram of the application is shown. The application has two layers. User Interface Layer is designed for the presentation of the constructed network and the average clustering coefficient values of the connected components in the email social network. EsnaForm class uses DiagramView (Mind Fusion 2009) class to draw the network and DataService class to display the data. Service Layer is designed for performing business logic. DataService class uses MyMail class to get emails from mail server, uses MyNode class for creating nodes and uses CollectionNodes class for creating connected components. MyMail class uses ImapMailBox (Source Forge 2009) class to connect to mail server and read emails form the server DataService class also makes connection between nodes, calculates clustering coefficient values of the nodes and divides the network to connected components.

4.4. Test Application

A simple software application was developed by using these algorithms. The spam and non-spam data gained from Oscar Boykin - University of Florida- (Boykin and Roychowdhury 2005) was used. The results are depicted in Table 4.3 and Table 4.4

The test email data consist of two data sets. Datasets have five-week periods email traffic of different people. In each dataset there are two sub datasets which have spam and non-spam emails. Each email only has ‘To’ ‘Cc’ and ‘From’ header information. The number of spam and non-spam emails for datasets are shown in Table 4.1 and Table 4.2.

Table 4.1. Number Of Spam and Non-Spam Emails For Dataset 1

Number of Non-Spam Emails	149
Number of Spam Emails	6349

Table 4.2. Number Of Spam and Non-Spam Emails For Dataset 2

Number of Non-Spam Emails	319
Number of Spam Emails	11654

4.4.1. Analyzing the Results

After applying the desktop test program on these data it was seen that spam components' clustering coefficient values are zero. The clustering coefficient valuses of the components are shown in Table 4.3. and Table 4.4. .

Table 4.3. Clustering Coefficient Values Of Spam and Non-Spam Emails' Social Network For Dataset 1.

Non-Spam Average	0,216769734801489
Biggest Component(65) Average	0,198419202472683
Spam Average	0
Biggest Component(4769) Average	0

Table 4.4 Clustering Coefficient Values Of Spam and Non-Spam Emails' Social Network For Dataset 2.

Non-Spam Average	0,20690710789395
Biggest Component(267) Average	0,229189411820991
Spam Average	0
Biggest Component(5413) Average	0

Average clustering coefficient values of non spam datasets are approximately 0.2. On the other hand average clustering coefficient values of spam datasets are 0. When we look at the results we can easily see that clustering coefficient values are really important factor to distinguish spam emails and non-spam emails.

4.5. Web Application Tool

A web software, which creates and analyzes email social network of a person was developed and deployed. (Cansoy Soydan 2009).

In this web implementation after the user enters his/her email user id, password and imap server name the software draws his/her email social network, divides the network to the components and calculates the clustering coefficient values of each component. User can extend his/her email social network by entering his/her other email account information. So that the software can draw and analyze all email friends of a person. In this application koolwired component (Source Forge 2009) was used for imap protocol and mindfusion component (Mind Fusion 2009) was used for drawing the social Networks.

4.5.1. Sample Application

In the first step user enters his/her email user id, password and imap server name and presses Create/Add button. In Figure 4.10 the user interface of the apllication is shown.

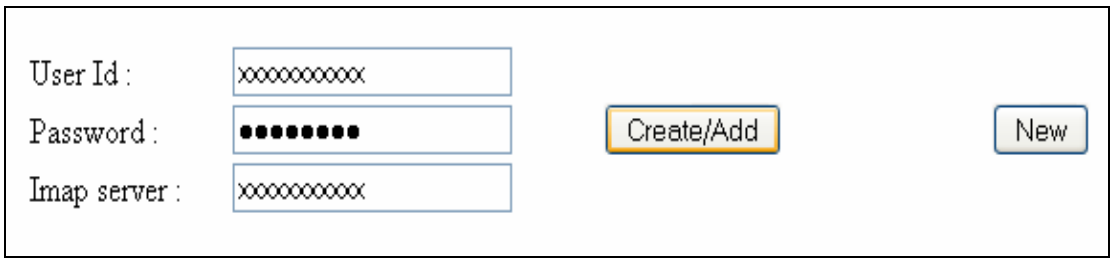

Figure 4.10. Entering The User Email Information.

Application draws email social network, divides network into clusters and calculates clustering coefficient values of each cluster.

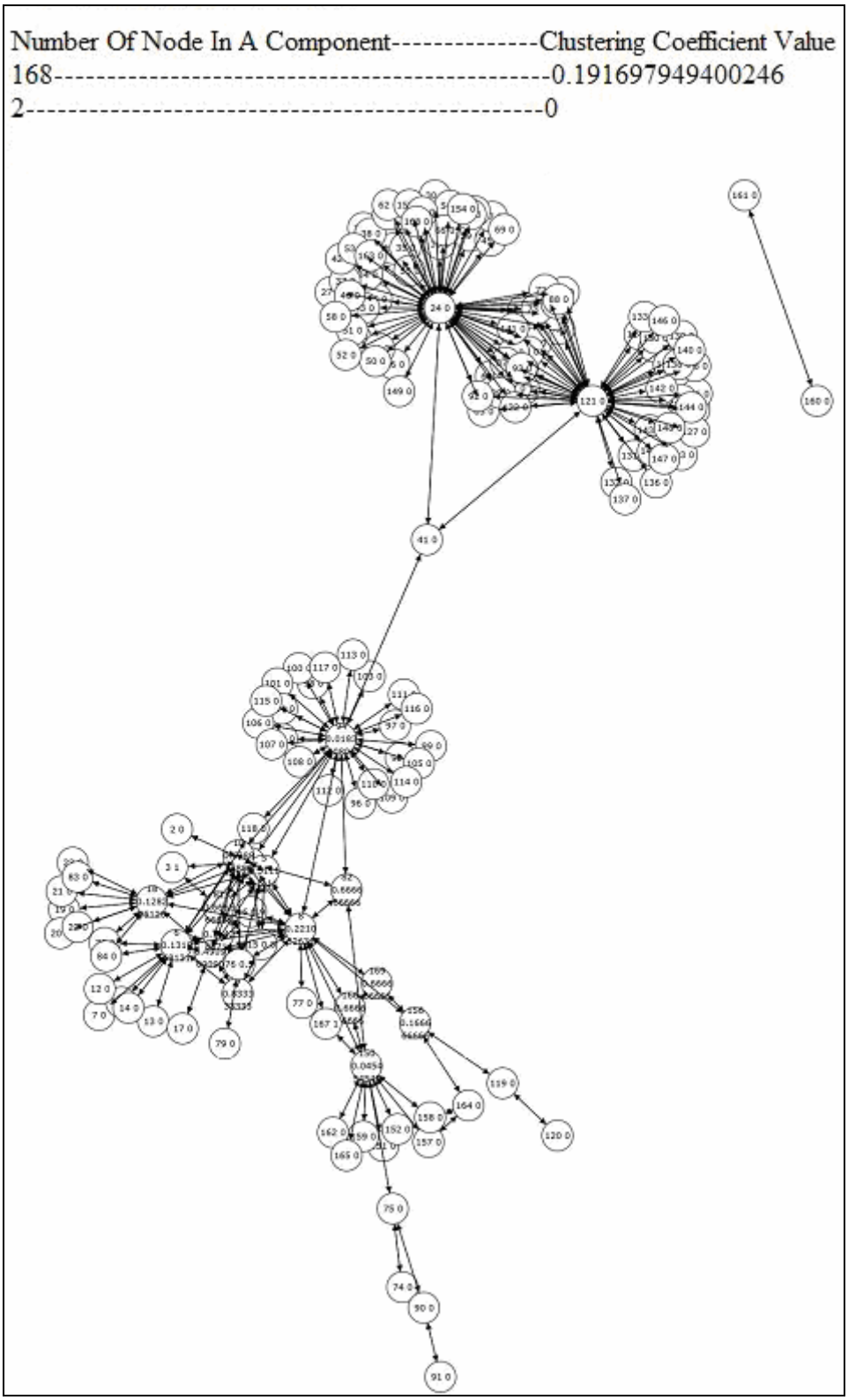

Figure 4.11. User Non-Spam Email Social Network.

In Figure 4.11 a social network of a person is shown. In this network there are two clusters. First cluster has 168 nodes and its clustering coefficient value is 0,19. This value means that in this network we see a friends social network of this person. So that we can easily say that in this social network there are no spam emails. The second network is too small to make a decision.

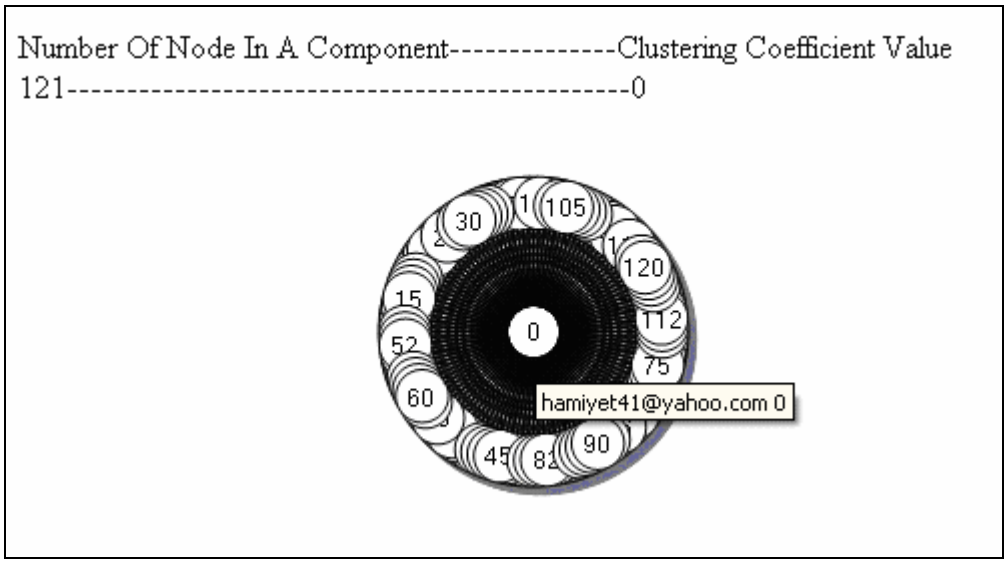

Figure 4.12. User Spam Email Social Network.

In Figure 4.12 a spam social network is depicted. This email social network's clustering coefficient value is equal to zero. A spammer sends an email to 120 people and these people has no relationship at all.

User can extend the graph with other his/her email addresses' social graph with entering the new email addresses. Alternatively he/she can enter new button to start the steps from the beginning.

CHAPTER 5

FUTURE WORK

To improve the application the other header information of the emails can be used. Here are the some examples of how they can be used.

The ‘received’ header information is used to trace the path that the email uses. By using the inconsistencies of this header, whitelists and blacklists can be formed. There is an example of this situation in Figure 5.1

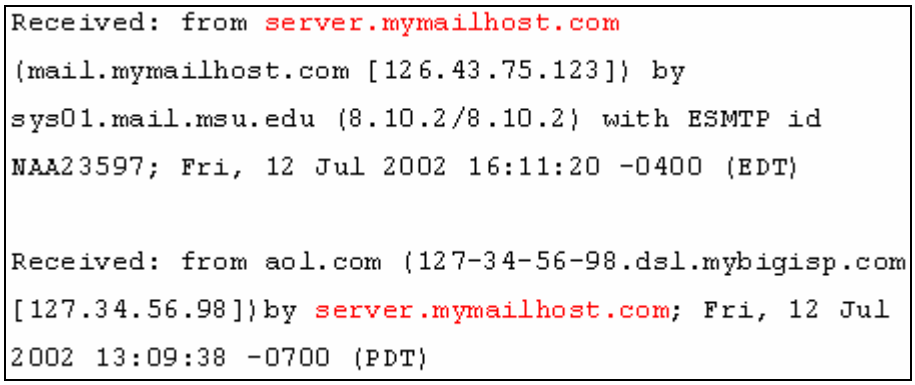

Figure 5.1. Path Inconsistency In Email Received Header.
(Source: Al-Zarouni 2004)

The header is analyzed from below to the top. The above email has used the path in the first ‘received’ header, from ‘aol.com’ to ‘server.mymailhost’. And then the email has followed the path from ‘server.mymailhost’ to ‘sys01.mail.msu.edu host’. Looking at the two headers, it’s apparent that ‘from’ and ‘by’ fields are consistent. When there is an inconsistency, the sender should go to the blacklist (Al-Zarouni 2004).

The most important information that is gained from ‘received’ header is the ip address of the sender host. Tools like ‘WHOIS’ can be utilized to determine if the domain name and the ip address are consistent (Al-Zarouni 2004).

If there is an inconsistency between the host name in the below ‘received’ header and the email address of the sender we can easily say that this person is trying to send spam (Al-Zarouni 2004).

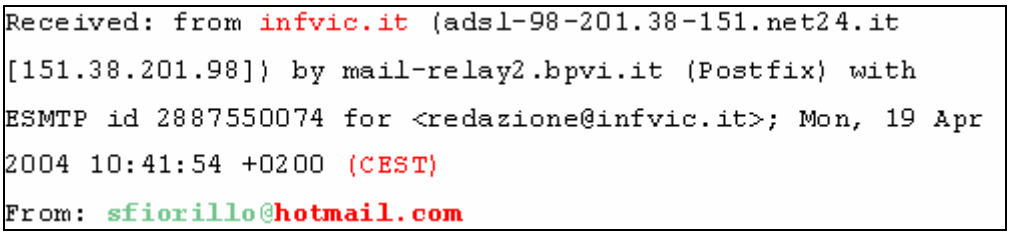

Figure 5.2. Location Inconsistency In Email Received Header.
(Source: Al-Zarouni 2004)

In Figure 5.2 if we look at the 'received' header above; the email is sent from a host located in Europe. However, 'hotmail.com' host is not located in Europe. There is also another inconsistency that the email is sent using 'SMTP' instead of using 'HTTP' or 'HTTPS' (Al-Zarouni 2004).

The other email header information that can be used is 'Message-ID 'header. The 'Message-ID' header gives the time information and other helpful information. By using 'Message-ID' header, inconsistent situations can be determined and blacklists can be updated (Al-Zarouni 2004).

CHAPTER 6

CONCLUSION

The objective of this work is to distinguish spam and non-spam emails using a tool which will use the properties of the email social networks. The tool will only use the information that can be gathered from emails. If some improvements are made, the tool can be used stand alone against spammers. On the other hand, this tool can be used with other anti-spam tools. This tool can automatically generate a training set for content-based filters and with content-based filters this tool will provide important success against spammers.

For the email systems security, in terms of application cost, the email social networks are good solution. The application should work in host level instead of client level. In this way there would be less application cost and the application would work more efficiently. The proposed application can use information from blacklists services in the internet if it is needed.

As spammers improve their attacking technics, the tools like implemented in this work will be developed with using different perpectives. If the spam problem is solved, the email social networks that ESNA (Cansoy Soydan 2009) –the tool developed in this work- constructed can be used for different purposes in many areas.

BIBLIOGRAPHY

Al-Zarouni, M., 2004. Tracing E-mail Headers, *Australian Computer, Network & Information Forensics Conference.*

Bean Software, 2009. Introduction to Spam and Anti-spam , http://www.beansoftware.com/Tutorials-Articles-Guides/Anti-Spam-Introduction.aspx (accessed June 5, 2009).

Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Gertz, M., 2006. Mining Email Social Networks in Postgres, *Proceedings of the 3rd International Workshop on Mining Software Repositories, New York.*

Boanerges, A. M., Nagarajan M., Ramakrishnan, C., Ding, L., Kolari, P., Sheth, A., Arpinar, B., Joshi, A. and Finin, T., 2006. Semantic Analytics on Social Networks: Experiences in Addressing the Problem of Conflict of Interest Detection, *Proceedings of the 15th International WWW conference (WWW'06).* 407 - 416 (2006)

Boykin, P. O. and Roychowdhury, V. P., 2005. Leveraging Social Networks To Fight Spam, *IEEE Computer.* 38(4): 61- 68

Cansoy Soydan, 2009. ESNA(Email Social Network Analyzer) , http://www.esna.cansoysoydan.com / (accessed June 3, 2009).

Eagle - Ing, 2009. Spam Definition, http://www.eagle-ing.net/ClickPicks/Internet/Spam-01.shtml (accessed January 15, 2009).

FMS Advanced System Group, 2009. Social Network Analysis (SNA), http://www.fmsasg.com/SocialNetworkAnalysis/ (accessed June 9, 2009).

Foaf Project, 2009. The Friend of a Friend (FOAF) project, http://www.foaf-project.org/ (accessed July 2, 2009).

Kleinberg, J., 2000. The Small-world Phenomenon: An Algorithmic Perspective, *Proceedings of the 32nd ACM Symp. on Theory of Computing* 163 - 170 *(STOC).*

Mind Fusion, 2009. NetDiagram(ASP.NET control) , http://mindfusion.eu/netdiagram.html (accessed June 9, 2009).

MyCert, 2009. E-mail Header Analysis http://www.mycert.org.my/en/resources/email/email_header/main/detail/509/index.html (accessed January 5, 2009).

Nascimento, M. A., Sander, J. and Pound, J., 2003. Analysis of SIGMOD's Coauthorship Graph, *SIGMOD Record* 32(3) (2003).

Object Mentor, 2009. UML Tutorial: Part 1 -- Class Diagrams. , http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf (accessed June 12, 2009).

Object Mentor, 2009. UML Use Case Diagrams , http://www.objectmentor.com/resources/articles/usecases.pdf (accessed June 12, 2009).

Org Net, 2009. Social Network Analysis, A Brief Introduction , http://www.orgnet.com/sna.html (accessed May 8, 2009).

Pemmaraju , Sriram V. 2008. Algorithm Of Calculating Clustering Coefficient Value Of A Node. http://www.cs.uiowa.edu/~sriram/21/fall08/code/myListGraph.java (accessed July 3, 2009).

Ramachandran, A. and Feamster, N., 2006. Understanding the network-levelbehavior Of Spammers, Proceedings *of the ACM SIGCOMM. ACM Press New York.* 36(4): 291 – 302

Sintef, 2009. Tutorial on UML, http://www.sintef.no/time/ELB40/ELB/UML/UML.pdf (accessed June 1, 2009).

Source Forge, 2009. Koolwired.Imap(IMAP client library) , http://sourceforge.net/projects/imapnet/ (accessed June 3, 2009).

Staab, S., Domingos, P., Mika, P., Golbeck, J., Ding, L., Finin, T., Joshi A., Nowak, A. and Vallacher, R.R., 2005. *Social Networks Applied,* IEEE Intelligent Systems. 20(1): 80–93

APPENDIX A

SOURCE CODES OF SOME METHODS

1. Read email from mail server:

```
public void readFromMailServer(SessionData sessionData)
{
            int k=0;
            // Loop for each email in inbox and outbox
            while (k < sessionData.bundles.Count)
            {
                MailMessageCollection bundlee =
                 (MailMessageCollection)sessionData.bundles[k];
                int i;
                for (i = 0; i <= bundlee.Count - 1; i++)
                {
                    MailMessage bundle = bundlee[i];


                  if(!sessionData.owners.Contains(bundle.From.Emai
                  l))
                    {

                        Email eposta = new Email();
                        int j;
                        string name;
                        string addr;

                        //Put  'To'  and  'Cc'  information  into
                        'received list'
                        for (j = 0; j <= bundle.To.Count - 1;j++)
                        {
                            name = bundle.To[j].DisplayName;
                            addr = bundle.To[j].Email;
                            eposta.addToHeader(addr);
                        }

                        for (j = 0; j <= bundle.Cc.Count - 1; j++)
                        {
                            name = bundle.Cc[j].DisplayName;
                            addr = bundle.Cc[j].Email;
                            eposta.addCcHeader(addr);
                        }

                        eposta.setFromHeader(bundle.From.Email);

                        //Remove owner from 'received list'
                        ownerExist(eposta, sessionData.owners);
                        RemoveFromAdressFromToCcList(eposta);
```

```
                        //Make connections between from node and
                        the nodes in received list'
                        if (HasSentOther(eposta))
                        {
                            CreateNodes(eposta);
                        }
                    }
                else
                if(sessionData.owners.Contains(bundle.From.Email
                ))
                    {
                        //Put 'To', 'Cc', 'Bcc' information into
                        'sent list'
                        Email eposta = new Email();
                        int j;
                        string name;
                        string addr;
                        for (j = 0; j <= bundle.To.Count - 1; j++)
                        {
                            name = bundle.To[j].DisplayName;
                            addr = bundle.To[j].Email;

                        if (!sessionData.owners.Contains(addr))
                            {
                                eposta.addToHeader(addr);
                            }
                        }
                        for (j = 0; j <= bundle.Cc.Count - 1; j++)
                        {
                            name = bundle.Cc[j].DisplayName;
                            addr = bundle.Cc[j].Email;
                            if(!sessionData.owners.Contains(addr))
                            {
                                eposta.addToHeader(addr);
                            }
                        }
                        for (j = 0; j <= bundle.Bcc.Count - 1;j++)
                        {
                            name = bundle.Bcc[j].DisplayName;
                            addr = bundle.Bcc[j].Email;
                            if(!sessionData.owners.Contains(addr))
                            {
                                eposta.addToHeader(addr);
                            }
                        }
                        // Create nodes for the email adresses in
                        'sent list'
                        // Make connections between the nodes in
                        'sent list'
                        if (eposta.getToHeader().Count > 1)
                        {
                            CreateNodesForOutbox(eposta);
                        }
                    }
                }
                k++;
            }
        }
```

2. Calculate the Clustering Coefficient Value of Each Node:

```
    public void calculateClusteringCoefficientValue()
        {
            //Calculate the clustering coefficient value of each
            node
            int i = 0;
            while (i < CollectionNodes.nodes.Count)
            {

                MyNode node = CollectionNodes.nodes[i];

            node.setClusteringCoefficientValue(CalculateClustering
            CoefficientValueForEachNode();
                i++;

            }
```

```
private double CalculateClusteringCoefficientValueForEachNode
               (MyNode node)
        {
            // Calculating Clustering Coefficient Value of a Node
            int edgesInNbd = 0;

            for (int j = 0; j < node.connections.Count; j++)
                for (int k = 0; k < j; k++)
                if (nodeHasConnections(node.connections[j],
                    node.connections[k]))
                        edgesInNbd++;

            if (node.connections.Count <= 1)
            {
                return 0;
            }
            else
            {
                return edgesInNbd * 2.0 /
                 (node.connections.Count * (node.connections.Count
                 - 1));
            }
        }
```

3. Creating connected components:

```
public void getConnectedComponents()
   {
        //Add this node and this node's connected nodes in a
        connected component(recursive)
       int i = 0;
       while (i < CollectionNodes.nodes.Count)
       {
           ArrayList conectedComponents = new ArrayList();
           MyNode node = CollectionNodes.nodes[i];

           if (!bigConnectedComponentsHasThisNode(node))
           {
             CollectionNodes.connectedComponents.Add(conectedCom
             ponents);
             searchConnectedComponents(node,
             conectedComponents);
           }
           i++;
       }
   }
```

```
private void searchConnectedComponents(MyNode node, ArrayList
conectedComponents)
       {
           // Dividing the network to the components
           if (!listHasThisNode(node, conectedComponents))
           {
               conectedComponents.Add(node);
           }
           else return;
           int j = 0;
           while (j < node.connections.Count)
           {
                 searchConnectedComponents(node.connections[j],
                 conectedComponents);
               j++;
           }

       }
```