Master Thesis

Exploring Decentralized Collaborative Filtering against Spam Mail

Simon P. Tobler

March 31th, 2008 - September 30th, 2008

Advisor: Dr. Spyros Voulgaris
          Prof. Gustavo Alonso
Information and Communication Systems Research Group
Preface

Acknowledgements

First of all I would like to thank my family for their support during my whole studies. They enabled me to be free of any obligations and allowed me to concentrate on my work during all these years.

I would like to thank Spyros Voulgaris for the idea of this thesis and the possibility to do research in the field of peer-to-peer applications. Additionally I thank him for his commitment and all the helpful meetings we had, which took place even when time was of great value. I thank him for proof-reading my thesis and I appreciated his friendly manners.

I would like to thank Keno Albrecht and David McLaughlin for their support in providing email logs from Spamato System and ETH email gateways, respectively. These logs were of fundamental importance to my thesis as they constituted the problem domain and founded the basis for analysis and simulation.

Andy Zbinden, Martin Kämpf and Thomas Buschor from Switch I would like to thank for their time and willingness to help me getting additional log data.

I thank my brother Philip Tobler for the free hardware supply in the first phase, where the experiments started to get very power consuming.

This leads me to the next acknowledgement, where I would like to thank the Information and Communication Systems Research Group for the access on their cluster computer IKR. Especially I would like to thank the system administrators Martin Wyser and Nico Schottelius for their immediate help on IKR related problems.

Finally I would like to thank my girl friend Sonja Schori for her support during my studies and Benjamin Steiner for proof-reading my thesis.
Abstract

Nowadays spam and virus mails get delivered daily to probably every user on the internet and almost everybody makes use of anti-spam software. Some of these anti-spam systems employ a collaborative component to include the users themselves into the decision process.

Normally a server-based architecture is used for the collaborative part. In this thesis we assess the feasibility of using a decentralized peer-to-peer component instead. Therefore we have a look at the problem of receiving spam mails from the users point of view, where we are interested in the relations between the users. We start by analyzing spam log data and construct a graph as follows: Each user receiving unsolicited bulk email constitutes a vertex. If two particular users receive the same spam mail, they are considered to be related to each other and the corresponding vertices have an edge in between.

In this graph we try to find some sort of structure, especially clustering. To achieve this goal we use different analysis methods which confirm each other and reveal different properties of the hidden structure, as they analyze the problem from different points of view.

Having all the tools in place, we additionally take a look at virus-infected emails and apply the same analysis steps to virus log data. In both cases we can detect some clustering but the results differ significantly.

After the analysis we evaluate collaborative filters. First we have a look at a server-based model to have a reference detection rate and afterwards we explore various ways of decentralized collaborative filtering. This way, we assess to what extent exploiting the clustering yields in better detection rates.
# Contents

1 Introduction 1
   1.1 Approach ......................................................... 2
   1.2 Related Work .................................................... 3

2 Analysis Methods 5
   2.1 Visualization of User Relations ................................. 5
      2.1.1 Graph ....................................................... 5
      2.1.2 Clustering Coefficient ..................................... 7
      2.1.3 Neato ......................................................... 8
   2.2 User-based Analysis ............................................. 8
   2.3 Spam-based Analysis ............................................. 9

3 Spam Analysis 11
   3.1 Spamato System .................................................. 11
   3.2 Spam Mail Data Source ......................................... 11
      3.2.1 Log Entries ................................................ 12
      3.2.2 Spam Statistics ............................................. 13
   3.3 Analysis ........................................................ 14
      3.3.1 Visualization of User Relations ............................. 15
      3.3.2 User-based Analysis ....................................... 18
      3.3.3 Spam-based Analysis ....................................... 21
   3.4 Analysis of Filtered Log Data ................................. 22
      3.4.1 Spam Data Preparation ..................................... 22
      3.4.2 Visualization of User Relations ............................ 27

4 Virus Analysis 30
   4.1 Clam Anti Virus .................................................. 30
   4.2 Virus Mail Data Source .......................................... 30
      4.2.1 Log Entries ................................................ 31
      4.2.2 Virus Statistics ............................................. 32
## Contents

4.3 Analysis .................................................. 33  
  4.3.1 Visualization of User Relations ....................... 33  
  4.3.2 User-based Analysis .................................. 35  
  4.3.3 Spam-based Analysis .................................. 38  

5 Collaborative Filtering Limits 39  
  5.1 Design .................................................... 39  
    5.1.1 Communication Paradigm ............................. 39  
    5.1.2 Parameters .......................................... 40  
  5.2 Simulation .............................................. 45  

6 Collaborative Filtering Simulation 47  
  6.1 Design .................................................... 48  
    6.1.1 Server-based Simulator .............................. 49  
    6.1.2 Optimal-view Simulator .............................. 49  
    6.1.3 Decentralized Simulator ............................. 49  
    6.1.4 Random-view Simulator .............................. 53  
    6.1.5 Parameters .......................................... 53  
  6.2 Simulation .............................................. 56  
    6.2.1 Spamato .............................................. 56  
    6.2.2 ClamAV .............................................. 57  

7 Conclusions 59  
  7.1 Clustering .............................................. 59  
  7.2 Collaborative Filtering ................................. 60  
  7.3 Decentralized Collaborative Filtering .................. 60  
    7.3.1 Practical Limitations ............................... 62  
  7.4 Approach for Collaboration .............................. 63  

Bibliography 64
Chapter 1

Introduction

Fighting spam has been an issue for a long time and a variety of solutions have already been proposed. Blacklists, Rule-based filters, Bayesian filters and Collaborative filters are widely used concepts. The goal is to filter out all emails sent by a small amount of professional spammers in a scalable and efficient way, which currently accounts to 90% of the email traffic.

Time has shown that fighting spam is actually a cat and mouse game. Professional spammers try to protect their business and react on each newly proposed filtering-technique by adapting their construction of spam mails. The only thing that keeps being stable is the perception of a spam mail by a human being. If this would not be the case, spammers would have missed their goal to advertise their products.

This causes most anti-spam systems to be combinations of the concepts listed above and to include the collaborative part to have some kind of human intelligence being part of the decision process. As for example Blacklists or Bayesian filters could operate locally on the clients machine, collaborative filters need to share information and are usually based on a central server that stores the decisions and opinions about received spam mails in a database, which can be queried again by every user of that system. But the problem of each centralized system is its scalability and availability. The problem of receiving spam mail affects almost every user on the internet. Therefore a spam filter system can potentionally be used by the biggest possible user base available. While centralized servers can be managed to work for regular operation on at least a subset of all existing users, a distributed denial of service (DDoS) attack is still possible to break down the system [1]. A server always means a central point of failure and this is the point where normally peer-to-peer systems are taken into account.
The idea of this thesis is to evaluate the possibility of replacing the centralized part of an arbitrary anti-spam system by a decentralized component. By having the collaborative part decentralized, an anti-spam system scales very well with increasing number of participants and gets increased robustness against targeted DDoS attacks. On the other side, several problems are introduced and need to be solved: How do the users know each other? What is about joining and leaving the system? What if there are different opinions about an email, e.g., an email gets classified as spam by a certain user or group of users but in the same time is classified as no spam by another group of users. Or how to limit network traffic so that the collaboration of all users does not overuse the networks capability.

In this thesis we will try to figure out whether it is possible to design such a decentralized system that addresses some of the problems from before and present and evaluate a possible solution.

In Section 1.1 we will roughly describe our approach how to explore decentralized methods against spam mail, and we have a look at related work in Section 1.2. In chapter 2 we will describe the methods used in chapter 3 and 4 to analyze spam mail and - additionally - virus mail logs. Based on our findings in the analysis phase, we will explore the limits of collaborative filtering in chapter 5 and evaluate our findings by means of a simulator in chapter 6. In chapter 7 we sum up what we have achieved and take some conclusions.

1.1 Approach

The goal is to have a dynamic unstructured network overlay connecting single peers. In order not to flood the internet with spam alerts, the participating users should get grouped so that several clusters appear. Users inside a cluster communicate with each other and share their knowledge about received emails. This way, the overall network traffic can be reduced significantly and collaboration takes place in a decentralized way.

Collaborative systems are always prone to manipulation if implemented without caution. An attacker may try to deceive the system or a user may be incautious and unaware. In this thesis we assume to have no such adverse users. Generally, these problems could be addressed by introducing some notion of trust for each neighbour. A trusted user should share a similar opinion and each user should keep a list of its trusted neighbours.
In this thesis we focus on the substantial part of our system design: Building clusters could be based on random selection of neighbours, they could be selected from the list of trusted users only or the system could exploit natural structures inferred through the reception of spam mails. Each kind of clustering would have its benefits and drawbacks. Random clustering would be simple to implement, but would probably not yield in a good success rate of spam mail detections. By only communicating to trusted users we would achieve a low false-positives rate, but the overall detection rate may be even lower as in the random case and building such clusters already needs more effort.

We suggest it would be best to exploit a natural clustering that is already present in the nature of spam mail delivery. If such a clustering is present, each user automatically belongs to a cluster and we just need to figure out to which cluster. The spam mail detection rate should then be higher as in the random case, as all neighbours receive similar spam mails.

1.2 Related Work

There are already other spam filters that are based on peer-to-peer systems. The main differences between our approach and the existing solutions lay in two design decisions:

1. Choice of the network overlay
2. The role of peers

We just give a short overview of the ideas from other studies. For further details and their explicit system design we refer to the corresponding papers and publications.

Approximate object location and spam filtering on peer-to-peer systems

Feng Zhou et al. [4] proposed a collaborative spam filter based on Tapestry [5]. Tapestry is a structured overlay location and routing infrastructure that provides location-independent routing of messages directly to the closest copy of an object using only point-to-point links and without centralized resources. In such a structured overlay, decentralized object location and routing across networks is mostly based on unique IDs. Zhou et al. focussed on an extension to these systems to publish an object using generic feature vectors instead of content-hashed globally unique identifiers, which enables their system to locate similar objects. Having generic feature vectors to identify an object, Zhou et
al. were able to develop a decentralized spam filter service using a structured network overlay. They used a two tier architecture where the user has to install a simple agent that itself connects to the second tier, the actual peer in the peer-to-peer network. These peers are typically dedicated computers; one per institution, department or company. The peers collect spam mail reports from their users and store these spam mails in the structured overlay using the generic feature vectors. Everytime a user receives a spam mail, the corresponding peer can deterministically locate the spam mail in the overlay network and if found, report it as spam to the user. This way, a central server is replaced by a peer-to-peer system.

**Distinction** Our system differs from this solution in the usage of an unstructured network overlay and by trying to exploit the properties of a potentially existing topology. Zhou et al. used Tapestry to just replace the central storage by a distributed hash table. But in our approach, the peer-to-peer system is the collaborative spam filter. Another difference is the role of the user: In our solution every user participates as peer itself in the peer-to-peer system whereas on Zhou et al. the users are subordinated to a dedicated computer.

**P2P-Based Collaborative Spam Detection and Filtering**

Another solution was proposed by Ernesto Damiani et al. [6]. They focussed on a decentralized privacy-preserving approach to spam filtering. Damiani’s solution exploits robust digests to identify spam mails that are slight variations of each other and a structured peer-to-peer architecture between mail servers to collaboratively share knowledge about spam. The architecture consists of three tiers, where the clients directly connect to their mail servers, the actual peers. The third tier is made up of super-peers that collect and poll spam reports from the mail servers. The main idea of Damiani’s solution is to have the clients only known to the mail servers, which hides the client’s identity.

**Distinction** The main differences to our approach are the same as before: Unstructured versus structured overlay network and the role of the user in the system.

To our knowledge ours is the first attempt to build a truly decentralized collaborative spam filter system that makes use of an unstructured network overlay. The main advantages of an unstructured network overlay are its simplicity for joining and leaving the system and its robustness against node churn.
Chapter 2

Analysis Methods

In this chapter we explain the different analysis methods. As we are not sure to which extent clustering exists we apply several methods yielding different kind of results and visualizations of the clustering.

2.1 Visualization of User Relations

To get a feeling about the data we are analyzing and to identify interesting properties, we start off by visualizing the overlay network implied by the characteristics of the way spam mails are delivered. We always analyze a certain portion of our log data, e.g., all log entries recorded during a specific month, week, or day.

2.1.1 Graph

The goal is to have an undirected graph $G = (V,E)$, consisting of vertices $V$ and edges $E$. Each vertex corresponds to a unique user and two vertices have an edge in between, if the corresponding users received the same spam mail during the analyzed time period. Such users are related to each other and this relation is rated by the amount of how many spam mails were received in common by both of these users. This parameter is modeled as the weight $w$ of an edge. The degree $d$ of a vertex is the number of its neighbours, as $G$ is an undirected graph.
Due to the huge amount of delivered spam mails, the visualization gets normally very dense as there are many users sharing lots of spam mails. To reveal a hidden potential structure in the graph, it can be filtered as follows:

- $w_{\text{min}}$ Minimal edge weight
- $h$ Neglect honeypots

The minimal edge weight is some kind of measure for each graph, as edges not fulfilling the requirement are not drawn. We need to set this limit larger than one, because with a low weight we would not be able to see anything in the graph due to the big amount of edges. It is important to state that we do not filter vertices, but edges. Although if a vertex remains with a degree of zero, it is not drawn either. This measure is applied only in the visualization, as we are interested in a human readable graph. In other types of analysis and the simulations later on, we will not require a minimal edge weight in order to work on unbiased data.

The minimal edge weight is typically used if the amount of delivered spam mails is very high and lots of edges have a weight greater than some value. This way, the minimal edge weight removes weak edges only, and strong edges are left in the graph.

A honeypot is an artificially set up system which simulates a proper victim that is very prone to a specific threat - in this case spam mail delivery. Honeypots can be used to attract and detect adversaries because normal users have no knowledge about these honeypots. Actually, there are no real honeypots employed in our logs, but there are some users that behave exactly like a honeypot and have a very high degree $d$. So, if vertices have an extremely surpassing degree $d$, we simply refer to the corresponding users as "honeypots" - just for naming. In contrast to the minimal edge weight, filtering honeypots is directly filtering out vertices. It is obvious that such honeypots hide potential structures in a graph, as they are connected to a huge amount of other vertices including other honeypots. Tools like NEATO (introduced in Section 2.1.3) will draw graphs containing honeypots shaped as stars, having the honeypot-vertices being in the centre. In our analysis, we can specify a parameter $h$, which tells the application to ignore the $h$ most popular honeypots. This parameter is determined empirically and we increase it step-by-step starting at zero until there is a satisfying result or until the parameter does not make sense anymore, as it would specify all/too many users to be honeypots. Removing honeypots yields in lowering the connectivity of a graph and hence - if existing - helps clusters to appear. As these honeypots have per definition edges to virtually all other vertices they wont reduce the performance in the simulation later on if included again.
2.1.2 Clustering Coefficient

To substantiate our informal findings in the visualizations, we sometimes compute the clustering coefficient of the displayed graphs. This measure was first introduced in 1998 by Duncan J. Watts and Steven Strogatz [7] to determine whether a graph is a small-world network or not.

The clustering coefficient quantifies the connectivity of a graph $G = (V, E)$ where $V$ denotes the set of vertices and $E$ the set of edges between them. A distinction is made between a local clustering coefficient for a certain vertex of the graph and the global clustering coefficient of the whole graph.

The local clustering coefficient of a vertex $v \in V$ is denoted by the ratio of the number of edges effectively in the neighbourhood of $v$ and the number of maximal possible edges in the neighbourhood of $v$.

The neighbourhood $N_i$ for a vertex $v_i$ is defined as its immediately connected neighbours

$$N_i = \{v_j\}$$

where $e_{ij} \lor e_{ji} \in E$ and edge $e_{ij}$ connects vertex $i$ with vertex $j$.

The degree $d_i$ of a vertex is defined as the number of vertices in its neighbourhood $N_i$. In an undirected graph edges $e_{ij}$ and $e_{ji}$ are considered identical. Thus, if a vertex $v_i$ has a degree of $d_i$, totally

$$\frac{d_i(d_i - 1)}{2}$$

edges could exist among the vertices within the neighbourhood. Hence, the local clustering coefficient for undirected graphs is defined as

$$C_i = \frac{2|\{e_{jk}\}|}{d_i(d_i - 1)}$$

where $v_j, v_k \in N_i, e_{jk} \in E$. 

If the size of the neighbourhood is one, i.e., $|N_i| = 1$, then the clustering coefficient is defined to be 1.

The clustering coefficient for the whole system is given as the average of the local clustering coefficient for each vertex:

$$\bar{C} = \frac{1}{n} \sum_{i=1}^{n} C_i$$

2.1.3 Neato

The graphs were generated using NEATO, a third party software freely available through the GRAPHVIZ [8] software package. NEATO fits our needs best as it tries to find a low-energy configuration by constructing a virtual physical model and running an iterative solver. It follows an approach proposed by Kamada and Kawai [9], and places an ideal spring between every pair of vertices, such that its length is set to the shortest path distance between the vertices. The heavier the weight of an edge, the more NEATO will try to place the end points so that the length of the edge is set near to the predefined minimal length.

2.2 User-based Analysis

In the user-based analysis we focus on each single user and collect all received spam mails for each particular user. We build all unordered pairs of users $(A, B)$ where $A, B \in U$ and $U$ is the set of all distinct users, and compute for each pair the following four values:

- $N_0$ = Number of spam mails not reported by any of $A$ and $B$.
- $N_A$ = Number of spam mails received by $A$, but not $B$.
- $N_B$ = Number of spam mails received by $B$, but not $A$.
- $N_{AB}$ = Number of spam mails received by both $A$ and $B$. 
For $N_0, N_A, N_B, N_{AB}$ the following invariant has to hold:

$$N = N_0 + N_A + N_B + N_{AB}$$

where $N$ is the total number of spam mails delivered to anybody during the period in question.

For each result $N_0, N_A, N_B, N_{AB}$ we generate a plot, displaying the current value on the y-axis as percentage with respect to $N$, and listing all pairs of users on the x-axis. The number of unordered pairs, and hence the range of the x-axis, is defined as

$$|P| = |U| \cdot (|U| + 1) / 2$$

where $P = \{\{a, b\} | a, b \in U \text{ and } a \neq b\}$ is the set of all unordered pairs.

In the plots displaying the results for $N_A, N_B, N_{AB}$ the percentage is not computed with respect to $N$, as the result would always be very low. Instead the percentage is computed with respect to the spam mails effectively received by $A, B, \text{and } A + B$, respectively.

### 2.3 Spam-based Analysis

In the spam-based analysis we have a look at each single spam mail from the logs and compute the set of users that actually received the current spam mail. This leads to a so called receiver-set per spam mail, and these sets are compared to each other (all unordered pairs as in Section 2.2). In this comparison, we compute the intersection of the sets, yielding a similarity factor that expresses to what percentage the users of one set occur in the other set. This way we can see, whether many spam mails are delivered to the same set of users or not. The similarity factor is computed as follows:

$$f_{\text{sim}} = \frac{|S_A \cap S_B|}{\min |S_A|, |S_B|}$$

where $S_A, S_B$ are the receiver-sets of the corresponding spam mails $A$ and $B$. 
In the computation of $f_{\text{sim}}$, the receiver-set with smaller cardinality is always taken as base. This ensures, that an overlap of 100% can always be achieved.

We get the results in form of two plots:

**Similarity-Plot**
This plot summarizes, how many times a certain similarity factor has been achieved. For each different similarity factor listed on the x-axis, the number of pairs that achieved such a similarity factor is plotted on the y-axis.

**Size-Plot**
This plot shows the relation between the size of the participating receiver-sets and its achieved similarity factor. On the y-axis we have the mean size of all receiver-sets yielding a similarity factor listed on the x-axis.
Chapter 3

Spam Analysis

In this chapter we analyze real log data from a collaborative filter of Spamato System. Section 3.1 gives a rough overview on Spamato System and the spam mail data source gets elucidated in Section 3.2. Different analyses are performed in Section 3.3 and 3.4.

3.1 Spamato System

Spamato is a spam filter framework [10]. It does not filter any emails on its own and is not able to distinguish between spam and no spam. Spamato simply provides an extendable system, which allows the bundling of different components that detect spam in cooperation. The Spamato Core filters emails received from Add-ons on the client side with the help of Filters and optional Plug-ins.

Spamato supports several independent spam filters as plug-ins for the system. A particular filter is the collaborative one named *Earlgrey* [2, 3]. Since we are interested in the users behaviour, we used log entries from this filter.

3.2 Spam Mail Data Source

The log files contain consecutive log entries from user-queries to the Spamato server. These queries are made when a user checks his mailbox at the mailserver and all emails are downloaded to the local computer. The installed Spamato Add-on computes automatically for each email a checksum based on the URLs contained in the email, and queries the Spamato server with these checksums. The Add-on receives a decision from
Chapter 3 Spam Analysis

<table>
<thead>
<tr>
<th>Log file</th>
<th>From</th>
<th>To</th>
<th>Time period</th>
<th>Size [KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>query15.log</td>
<td>12.09.07 10:19</td>
<td>28.09.07 18:02</td>
<td>~16 days</td>
<td>∼16 days 1'001'257</td>
</tr>
<tr>
<td>query16.log</td>
<td>28.09.07 18:03</td>
<td>24.10.07 09:20</td>
<td>~1 months</td>
<td>1'489'129</td>
</tr>
<tr>
<td>query17.log</td>
<td>24.10.07 09:21</td>
<td>14.12.07 20:01</td>
<td>~1.5 months</td>
<td>3'448'369</td>
</tr>
<tr>
<td>query18.log</td>
<td>14.12.07 22:38</td>
<td>02.01.08 18:39</td>
<td>~15 days</td>
<td>1'366'260</td>
</tr>
<tr>
<td>query19.log</td>
<td>02.01.08 23:12</td>
<td>17.03.08 19:57</td>
<td>~2.5 months</td>
<td>7'376'608</td>
</tr>
<tr>
<td>query20.log</td>
<td>18.03.08 00:05</td>
<td>18.03.08 00:49</td>
<td>~45 minutes</td>
<td>4'467</td>
</tr>
<tr>
<td>query21.log</td>
<td>18.03.08 00:50</td>
<td>09.04.08 23:16</td>
<td>~20 days</td>
<td>2'495'382</td>
</tr>
</tbody>
</table>

Table 3.1: Available log files from Spamato System.

the Earlgrey filter - spam, no spam, or unknown - and classifies the email adequately. These queries are logged on the Spamato server and constitute our log files.

In the following we will usually call these queries *reports*, as we think of them as a spam mail report to the Spamato server, i.e., a statement that a particular user received the current spam mail.

The log files have different sizes and cover different time periods. Table 3.1 gives an overview of a part of the material we got. In some few cases, there is data missing for the time between the log files. These missing parts are at most few hours and we try to avoid running detailed analyses over these gaps.

### 3.2.1 Log Entries

The Earlgrey filter is a URL-based filter. It extracts the unique domains contained in an email and checks whether the domains appear in the central Earlgrey database and if so, it considers the email as spam.

The Earlgrey logs consist of log records for each query a user sends to the server, that is, everytime a user receives an email that contains possible spam domains.
Chapter 3 Spam Analysis

A typical log record looks as follows:

```
[1179219072845] 8
UserID: c60a51b101399acb0feb96e1bb5bab80
URLs: [http://www.w3.org, http://sarmint.com]
All domains: [w3.org, sarmint.com]
Used domains: [sarmint.com]
Result: 69c77778a19d241154282bfb3bfe0586 [1b4...6b, 8d2...d8] 167 0 1
```

Line one shows the timestamp in square brackets and an internal log number. On the second line follows the MD5 hash\(^1\) of the user ID and the third line lists all URLs found in the email. The domains get extracted on line four and duplicates as well as all server-side whitelisted domains are removed, resulting in line five. On the last line we get the decision, which is one of the following:

- **Result: no domains**
  - If after removing duplicates and whitelisted domains nothing is left.

- **Result: no match**
  - If the domains seems to be legitimate.

- **Result: 69c77778a19d241154282bfb3bfe0586 [1b4...6b, 8d2...d8] 167 0 1**
  - A positive match, where the first string is the message ID\(^2\), the square brackets contain message IDs of other spam mails having the same hash of All domains\(^3\), and the three numbers represent the number of reports, revokes, and finally the Trooth [2] status code, i.e., one if the Trooth Trust System of Spamato was successfully employed.

### 3.2.2 Spam Statistics

To get a rough idea what data is recorded in these gigabytes of log files, we produced some basic spam statistics. Figure 3.1 shows the number of active users and the number of reported spam mails per day. The plot covers the time period of query15 to query21, i.e., September 12th 2007 to March 18th 2008.

---

\(^1\) Fictive in this log excerpt.

\(^2\) MD5 hash of the Used domains.

\(^3\) These message IDs can be different, because of changes in the server-side whitelist over time. So, equal All domains can get mapped to different hashes of the Used domains.
Chapter 3 Spam Analysis

Figure 3.1: Statistics of spam log data query15 to query21, listing for each day the number of users that were active and the number of spam mail reports sent to the Earlgrey server.

It is important to state that Figure 3.1a shows the amount of active users per day. It shows the amount of distinct users per day, which reported at least one spam mail on that day. This does not correspond to the number of registered users at that day. But generally the activity of the users is likely to grow with an increasing number of registered users.

There is a slight increase of activity to see in Figure 3.1a except during Christmas and New Year (around day 100). In Figure 3.1b we can see a clear increase of spam mail reports by advancing in time. These observations align with the statistics of well-known organizations dealing with spam [11, 12].

3.3 Analysis

In this section we analyze the logs in their original form. No changes have been made and all available data was analyzed. This includes potential false positives and/or other errors contained in the logs.
Chapter 3 Spam Analysis

Figure 3.2: Effect of different values for the minimal edge weight $w_{\text{min}}$ during a time period of one month from January 3rd to February 2nd 2008.

In the analysis, we concentrate on the following fields of a log entry:

- Timestamp
- UserID
- Result

All entries with a result not consisting of a message ID and entries without valid Trooth status (i.e., $\neq 1$) are ignored, because they are either no spam mails or invalid reports.

### 3.3.1 Visualization of User Relations

**Parameter Effects** Figure 3.2 shows the effect of the minimal edge weight $w_{\text{min}}$. If set to a higher value, more edges get neglected and some structure appears containing only strong edges in the sense of their weight. As there is an extreme amount of spam mails delivered during the period of a month, it’s comprehensible that we need to set the minimal edge weight very high to reveal a structure. The level of $w_{\text{min}}$ will be lower in further experiments, as we will analyze shorter time periods and therewith fewer spam mail reports.

The effect of the second parameter, the number of honeypots $h$, can be seen in Figure 3.3. As we have 9’540 active users during this month it should still be feasible to ignore
Figure 3.3: Effect of different values for parameter $h$ during a time period of one month from January 10th to February 9th 2008.

such an amount of honeypots and as honeypots are very lively users, the systems performance will not get lower once we take all users into account.

The finding of this paragraph is, that filtering by means of $w_{\text{min}}$ and $h$ is really needed if we want to discover any structure by visualizing the graph.

**Time period: One month** If we take a series of visualizations covering overlapping time periods, we get the result in Figure 3.4. We can see the development of the clusters and notice the increase in spam mail delivery, e.g., Figure 3.4a is slightly less dense than Figure 3.4c.

Overall, we really reveal some structure in Figures 3.2 to 3.4 by applying our analysis. The clustering we are looking for seems to be present, but it is not very concise and not very well-formed. But what if natural clustering is existing based on smaller time periods? In that case the clusters would overlap in our analysis, because up to now we had a look at a whole month per graph. If clustering is evolving on a weekly or even daily basis, all the clusters would superimpose themselves.
Chapter 3 Spam Analysis

Figure 3.4: Series of visualizations with parameters $w_{\text{min}} = 150$ and $h = 1000$ for a time period of one month each.

Figure 3.5: Series of visualizations with parameters $w_{\text{min}} = 90$ and $h = 300$ for a time period of one week each.
Chapter 3 Spam Analysis

Figure 3.6: Series of visualizations with parameters $w_{\text{min}} = 40$ and $h = 50$ for a time period of one day each.

**Time period: One week**  
On a weekly basis, the parameters $w_{\text{min}}$ and $h$ can be set to lower values. As before, $w_{\text{min}}$ and $h$ were determined empirically and we choose to present the values that provided the best results. We have some clustering too although it seems to be better in the monthly case. On average there were 7,448 users active during these overlapping weeks.

**Time period: One day**  
If we lower the time period to one day, we get the results shown in Figure 3.6. We picked three different days and analyzed them with $w_{\text{min}} = 40$ and $h = 50$. The results differ: While we can identify in Figures 3.6a and 3.6b two very nice clusters, Figure 3.6c shows no structure at all. Clustering is probably likely to evolve and each user downloads his emails in different frequencies, so that same spam mails may get reported at different points in time. Therefore we probably just caught a bad day to analyze. But nevertheless, we need to deal with such situations and be aware of such fluctuations.

### 3.3.2 User-based Analysis

Visualizations are convenient for getting a feeling of the situation quickly. But they can be misleading in both ways, positive and negative. In this section we analyze the data of one day, January 14th 2008, with the user-based analysis. We analyze unmodified and
unfiltered logs and try to detect the same clustering as with the visualization method.

On this particular day were 4'816 distinct users active and according to equation 2.1 we need to analyze 11'594'520 unordered pairs.

In Figure 3.7a we see the percentage of all spam mails, that were not reported by any of A and B. For almost all pairs of users this value is higher than 90% (but never 100%), so there are really lots of different spam mails and each user receives only a small portion of it. This is appreciated for clustering, as there is room to distinguish between clusters.

Figures 3.7b and 3.7c, show the same kind of plots with switched roles: They display the difference between the received spam mails of A and B. A large part of all pairs of users does not share any single spam mail (values of 100%). The corresponding users will not belong to the same cluster as they do not share any common spam mails. However, for $\frac{1}{6}$ of all pairs the percentage is between zero and 100. A percentage of zero means the two involved users belong to the same cluster as they reported exactly the same spam mails. Unfortunately there are only very few pairs yielding exactly 0%. Users sharing some common spam mails achieve a percentage between zero and 100. It is these users that make clusters overlapping as they do not belong to one cluster perfectly. The plot is not rising very fast, so there is a significant amount of users connecting the clusters.

In Figure 3.7d we have the percentage of spam mails received by both nodes A and B. For most pairs this is 0% but there are some pairs (the same as in Figures 3.7b and 3.7c not having 100%) with a certain percentage of shared spam mails. The affected users probably belong to clusters as they share some spam mails and at the same time do not share every spam mail (overlapping clusters).
Chapter 3 Spam Analysis

Figure 3.7: User-based analysis of January 14th 2008

(a) $N_0$: Spam mails not reported by any of $A$ and $B$
(b) $N_A$: Spam mails reported by $A$ but not $B$
(c) $N_B$: Spam mails reported by $B$ but not $A$
(d) $N_{AB}$: Spam mails reported by $A$ and $B$
3.3.3 Spam-based Analysis

The spam-based analysis looks at the clustering from the spam mails point of view. We present the results for the same day as in the previous section, January 14th 2008.

In Figure 3.8a there are lots of pairs of receiver-sets not overlapping at all (similarity factor is zero) and fewer but still a reasonable amount of pairs that have a similarity factor of one, which means they are completely the same. In between the amount of pairs having a certain similarity factor is decreasing with increasing similarity factor. As we are looking for clustering this is a very good property. Many spam mails do not reach the same users, many others get indeed delivered to the same users and the rest makes sure the clusters are overlapping.

Information about the size of receiver-sets can be seen in the Size-Plot in Figure 3.8b. As we always took the smaller receiver-set as base for the computation of the similarity factor, it is true that the points belonging to the compared receiver-set are always above the points belonging to the receiver-set taken as base. While the receiver-sets taken as base keep being small with increasing similarity factor, the mean size of the compared receiver-sets increases. This is obvious as a bigger set is more likely to achieve a high intersection with a small set. Towards similarity degree one the mean size of both receiver-sets drops instantly. This is explained as follows: A spam mail is very unlikely to be delivered to a big set of users that is exactly the same set of users of another spam...
mail. But it is comprehensible that a small receiver-set can consists of exactly the same few users as another small set. This indicates again, that clusters will be overlapping, as there are no big isolated receiver-sets. Towards similarity zero the plot of the base drops too: There are many small distinct receiver-sets.

Summary

We have found some clustering using different analysis techniques. However, the clusters are not very distinct and not very well-formed. There are lots of users related to almost any other user and some parts of the graphs are not clustered at all. These results are useful, but not very ideal. The question remains: Can we do better?

3.4 Analysis of Filtered Log Data

3.4.1 Spam Data Preparation

The Spamato system has been designed to make use of several spam filters in order to achieve a high filter accuracy. There exists a filter-process that invokes spam filters, and a decision maker to combine the results. If we have a look at only one filter the result can be misleading. Therefore we prepare the log data and filter out certain entries.

Abnormal log entries can be caused by the following reasons:

- Nontypical user behaviour such as
  - Joining and leaving the system within a short period; e.g., just testing Spamato System for a short time.
  - Sporadic participation; e.g., using Spamato very infrequently and/or nontypical behaviour patterns.

- Mapping of different spam mails onto the same signature\(^4\) due to partial evaluation of spam mail content only; Nothing but the URL’s are taken into account to identify a spam mail.

- Delayed email delivery.

\(^4\)Cryptographic hash functions, such as MD5, itself are not completely collision free. But we think this source of errors is negligible in this case.
Chapter 3 Spam Analysis

Figure 3.9: Spam reports of a user who runs Spamato only for a few days (3.9a), and a user who checks his/her mails infrequently (3.9b).

Behaviour patterns

In Figures 3.9, 3.10, 3.11, and 3.12 we can see the different behaviours of users and spam mails in the period from September 12th 2007 to March 18th 2008 (query15 to query21). Each plot reveals the behaviour of one particular user or spam mail, and shows the percentage of reports/deliveries per day, respectively.

Spam report patterns We illustrate the user behaviour by presenting a typical example for each activity pattern. Figure 3.9a shows a typical activity pattern for users, that participated in Spamato during a short time period only. They probably installed the Add-on and gave Spamato just a short try. Others participate for a long period of time, but check their emails only very infrequently which is illustrated in Figure 3.9b.

The most interesting users for our analysis are the regular ones. As Figure 3.10a and 3.10b illustrate, there are some highly active users that check their email accounts very frequently and therefore report the spam mails in a timely manner.
Figure 3.10: Spam reports of specific users who are using Spamato for a long period and check their mails regularly.

Figure 3.11: Delivery patterns of specific spam mails getting reported over and over again during periods of several weeks.
Chapter 3 Spam Analysis

Figure 3.12: Delivery patterns of specific spam mails getting reported during short periods of time only.

Spam delivery patterns  Figure 3.11a shows a particular spam mail, that gets repeatedly delivered to the users very often and Figure 3.11b shows another spam mail that gets repeatedly delivered, but less frequently. This is mostly due to users downloading their emails only very sporadically. Therefore, in order to filter out such artifacts, we only consider users that report something every day of the period in question.

Figures 3.12a and 3.12b show the typical spam mail delivery pattern, where content of spam mails change frequently and therefore the same signatures occur during short time intervals only.

Spam Data Filter

We think by analyzing adequate users and spam mails only we can perhaps detect better clustering, as such log data is very consistent. It is important to first filter the users, before invalid spam mails get disregarded.

Users  The criteria for being kept in the logs or filtered out is the number of days a user was active during a given period. We define a user to be active on a single day, when he has reported at least one spam mail during the day, no matter at which time.
Chapter 3 Spam Analysis

As we look after the activity of a user it does not matter whether the spam mail reported by the user was valid, invalid, or a false positive log entry. The fact that the user was active is proven in any case. Therefore we can first filter the users and then the spam mails without introducing any errors.

For a given time period, we simply defined a user to be a regular one and therefore being kept, if he was active on at least 90% of the time period. Broken days were included as a whole, e.g., a user needed to be active on 27 days only during a monthly time period of 31 days.

If after spam mail filtering a user is left without any spam mail he is not further considered as he would not have relations to other users anyway. This explains the fact that filtering logs of one single day still yields fewer users although the threshold for a user to be active would be zero.

Spam mails For each spam mail we stored the time period stretched by its first and last reception. We defined the lifetime of a spam to be 20 days at most, so if a spam mail was received later than 20 days after its first appearance, it was filtered out. The time period in question is always extended for ±20 days, to know whether a spam mail at the very beginning or end should be kept or not. A problem arises with spam mails first received towards the end of our data source, where we do not have any more log entries. We do not know whether these spam mails will be received 20 days later again or not. Therefore we analyzed log data until 20 days before the end of the covered time period only.

Such a filtering is quite aggressive and the analysis should not be used as a basis to design a system. But the goal is to get a very concise data source and to find out how the analysis would behave on such a filtered data source.

Results The filtering process leads to the results summarized in table 3.2. We immediately see, that a huge amount of log entries are filtered out. It is comprehensible that fewer users and spam mails get filtered out with decreasing size of the time period, as they need to meet lower requirements.

There are only few users active on 90% of all days of one month. Already significantly more users are active during one week and on the daily time periods. The daily time
Chapter 3 Spam Analysis

### Table 3.2: User- and spam-filtering results

<table>
<thead>
<tr>
<th></th>
<th>period</th>
<th>users before</th>
<th>after</th>
<th>decr.</th>
<th>spam mails before</th>
<th>after</th>
<th>decr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>03.01. - 04.02.08</td>
<td>9'547</td>
<td>1'521</td>
<td>84.1%</td>
<td>206'188</td>
<td>44'813</td>
<td>78.3%</td>
</tr>
<tr>
<td>months</td>
<td>10.01. - 11.02.08</td>
<td>9'499</td>
<td>1'543</td>
<td>83.8%</td>
<td>212'633</td>
<td>48'066</td>
<td>77.4%</td>
</tr>
<tr>
<td></td>
<td>17.01. - 18.02.08</td>
<td>9'533</td>
<td>1'559</td>
<td>83.6%</td>
<td>218'681</td>
<td>50'308</td>
<td>77.0%</td>
</tr>
<tr>
<td>weeks</td>
<td>10. - 17.01.08</td>
<td>7'434</td>
<td>2'040</td>
<td>72.6%</td>
<td>74'643</td>
<td>13'128</td>
<td>82.4%</td>
</tr>
<tr>
<td></td>
<td>11. - 18.01.08</td>
<td>7'461</td>
<td>2'069</td>
<td>72.3%</td>
<td>79'383</td>
<td>13'764</td>
<td>82.7%</td>
</tr>
<tr>
<td></td>
<td>12. - 19.01.08</td>
<td>7'449</td>
<td>2'110</td>
<td>71.7%</td>
<td>79'139</td>
<td>13'911</td>
<td>82.4%</td>
</tr>
<tr>
<td>days</td>
<td>09.01.08</td>
<td>4'903</td>
<td>2'170</td>
<td>55.7%</td>
<td>20'143</td>
<td>11'091</td>
<td>44.9%</td>
</tr>
<tr>
<td></td>
<td>14.01.08</td>
<td>4'816</td>
<td>2'175</td>
<td>54.8%</td>
<td>22'637</td>
<td>12'870</td>
<td>43.1%</td>
</tr>
<tr>
<td></td>
<td>06.02.08</td>
<td>5'043</td>
<td>2'550</td>
<td>49.4%</td>
<td>29'222</td>
<td>12'738</td>
<td>56.4%</td>
</tr>
</tbody>
</table>

Concerning the spam mails, we can see that a big part of it gets filtered out. Spam mails may be reported a long time after their delivery to the user’s mailbox, because the user checks his mailbox only very infrequently. Therefore many spam mails get filtered out and only the ones reported within the defined time period are kept.

### 3.4.2 Visualization of User Relations

We choose the same time periods as in Section 3.3 to visualize. The values for the parameters $w_{\text{min}}$ and $h$ are generally lower as we have filtered out most data and therefore fewer vertices and edges. Because of the lighter edges, parameter $w_{\text{min}}$ behaves rawly and the visualizations are very sensitive to its value. Therefore we fixed this parameter for each series of visualization and let $h$ vary only.

**Time period: One month** Figure 3.13 shows the result for a monthly series. There appears some clustering as expected but the difference to the unfiltered case is not very significant. We can detect some clusters, but there are still many randomly looking vertices/edges.
Figure 3.13: Series of visualizations with parameter $w_{\text{min}} = 10$ and varying $h$ for a time period of one month each.

Figure 3.14: Series of visualizations with parameter $w_{\text{min}} = 3$ and varying $h$ for a time period of one week each.
Chapter 3 Spam Analysis

Figure 3.15: Series of visualizations with parameter $w_{\text{min}} = 1$ and varying $h$ for a time period of one day each.

**Time period: One week**  The results in the weekly series of Figure 3.14 do not show any new results either. There is some clustering, but it did not get better or very concise. The clusters seem to be a little bit more isolated to each other, but on the other side there are also more unclustered vertices distributed all over the graph.

**Time period: One day**  There are some small clusters in Figure 3.15 which look very nice. But as observed in the weekly series, there is much noise in the background that seems not to belong to any cluster.

As log data filtering did not reveal breaking results and great differences to the unfiltered case, we omit the user- and spam-based analysis as the plots do not provide any new insights.
Chapter 4

Virus Analysis

Another type of spam mails are the ones containing viruses, aiming at breaking into computers and getting access to critical information, such as credit card numbers. We obtained logs about virus mails from our mailserver at ETH and we stray from the subject of filtering classical spam mails to filtering virus-infected emails. The goal to find clustering and fight unsolicited bulk email in a collaborative way is still the same.

4.1 Clam Anti Virus

Clam AntiVirus [13] is an open source (GPL) anti-virus toolkit for UNIX, designed especially for email scanning on mail gateways. This is in contrast to Spamato System, which is a spam filter framework and uses Add-ons installed on each clients computer.

4.2 Virus Mail Data Source

We have much less log data from ClamAV than from Spamato System. The differences are enormous. Whereas we have multiple months of logs from Spamato System, we have

<table>
<thead>
<tr>
<th></th>
<th>distinct users</th>
<th>distinct emails</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spamato System</td>
<td>~12'000</td>
<td>~400'000</td>
</tr>
<tr>
<td>ClamAV</td>
<td>~15'500</td>
<td>~5'500</td>
</tr>
</tbody>
</table>

Table 4.1: Number of users and emails during approximately two months.
few weeks of ClamAV logs only. But the logs also differ with respect to their actual content. During a time period of approximately two months, there are roughly the same amount of distinct users receiving spam and virus mails, but the amount of distinct spam mails excels the amount of virus mails by a factor of $\sim 70$. Table 4.1 lists the specific numbers for query19 and 45 days of ClamAV logs.

4.2.1 Log Entries

We received daily extract files of ClamAV logs from the mail gateway of ETH Zurich. Each file contains information about all emails identified to carry a virus. The data collecting process started at July 18th 2008 and continued until August 31th 2008. Therefore we do not have quantities of logs, but very fresh and almost real-time data of high quality. The format of each log file looks as follows:

```
Column #1 is the date
Column #2 is the time
Column #3 is a MD5 hash of the recipient address
Column #4 is the name of the ClamAV signature that caused the rejection
```

These log files contain data of high quality. The crucial differences and/or improvements over Spamato’s log files are the following:

- Fixed and big set of accounts
- Logs recorded at the moment a virus mail arrives
- Includes all emails that contain a virus

---

1Fictive in this log excerpt.
Chapter 4 Virus Analysis

Figure 4.1: Statistics of virus log data from July 18th 2008 to August 31th 2008, listing for each day the number of users that were receiving virus mails and the number of detected virus mails.

The tapped mail gateway handles around 15’500 user accounts and an account at ETH is far more fixed for a certain time period than in Spamato System, as the email services at ETH are not for free and not available to everybody. In Spamato, the queries about spam mails are sent to the server and logged at the time they are downloaded on the client computer. Therefore big delays are introduced and the picture of spam mail arrival is blurred. ClamAV runs on the mail gateway itself and each email arriving at the mailbox on the server is immediately scanned and automatically reported. This may be an advantage for detecting a structure in the graph of virus mail receptions. However, such a mail reception/checking pattern at an end-user level could only be achieved by having all users using push-email, for example. Apart from false positives and not detected viruses ClamAV reports every virus mail. This is in contrast to Spamato’s filter Earlgrey, where we get spam mails including URL’s only. Most spam mails do - but by far not all. In Spamato, some spam mails may be missing, which could distort our findings.

4.2.2 Virus Statistics

Figure 4.1a does not show the amount of registered users for each day but the number of users which have received at least one virus mail on the corresponding day. This
means, that the bars can be constituted by different users and to get the total number of distinct users, we would have to sum up all the different users from all the bars.

In contrast to Section 3.2.2 where we could see a trend in the activity of the users, there is no such evolution to see in Figure 4.1a. We would probably need much more data to plot in order to see trends instead of some fluctuations only.

In Figure 4.1b all virus detections of ClamAV are displayed per day. This includes duplicates and shows the number of virus mails found during the corresponding day. This way we can see the amount of virus mails sent to our mailserver.

4.3 Analysis

In this section we analyze the ClamAV logs in the same way as in Section 3.3.

4.3.1 Visualization of User Relations

As we have daily log-extract files we will combine 30 consecutive days to have one month of data available, seven consecutive extract files to get a data file of one week and we will pick a few single days to perform our analysis on it.
Time period: One month If we have a look at a time period of one month and analyze it using several different values for parameter $h$, we get the result shown in Figure 4.2. By increasing parameter $h$ to filter out more honeypots we reveal no structure in the graph. Varying the minimal required edge weight $w_{\text{min}}$ does not help either. We just have one fairly well connected cluster with some outliers yielding in total clustering coefficients decreasing from $\sim 0.85$ in Figure 4.2a to $\sim 0.55$ in Figure 4.2c.

Time period: One week Figure 4.3 shows some graphs obtained by analyzing data of one week. As in the monthly case, there is still no real structure to see although there are some artifacts. The clustering coefficients in these settings lay for each graph around $\sim 0.75$. During this one week we still had around 5’900 active users.

Time period: One day Figure 4.4a looks already promising, as we can see several "bumps". Trying to isolate these bumps by increasing the minimal edge weight as in Figure 4.4b fails, because this removes lightweight edges and therefore only retains highly popular nodes. But increasing parameter $h$ to filter out the $h$ most popular users yields

---

2In this chapter an active user means that an email was delivered to the users mailbox on the mailserver. It does not mean that the user itself was active and did something.

3This is true for the ClamAV logs, as there are very few virus mails delivered compared to spam mails and hence edges rarely get very heavy.
Figure 4.4: Series of visualizations with varying parameters $w_{\text{min}}$ and $h$ for a time period of one day, July 21th 2008.

in spectacular graphs, which need to be considered carefully as we are removing about $\frac{1}{4}$ of all users (there are approximately 2'200 distinct users active during one day). The clustering coefficients are quite high and reach at max $\sim 0.96$ in Figure 4.4c.

In the next Figures, 4.5a, 4.5b, and 4.5c, we made heavy use of parameter $h$ to isolate the clusters. Compared to Figure 4.4c or even 4.4a we notice, that we virtually do not loose any clusters by increasing $h$, but remove all interconnecting nodes and edges which rendered the graph unreadable. The clustering coefficients for all three sub-figures in Figure 4.5 are as expected high and range from $\sim 0.93$ to $\sim 0.96$.

### 4.3.2 User-based Analysis

In the previous section we have seen, that there seems to be no clustering on monthly and weekly periods, but there is some structure on a daily basis. We try to detect these interesting structures with the user-based analysis method and present the result of examining one single day, i.e., July 25th 2008, visualized in Figure 4.5b. The underlying data was not modified or filtered in any way. So all the log data available was taken into account for the analysis.

In Figure 4.6a we see, that for each pair of users more than 70% of all virus mails were not received. In other words, only a very small percentage of the overall virus mails was delivered to any of the two users. Note that 100% is never achieved as both users A and
Figure 4.5: Series of visualizations with parameters $w_{\text{min}} = 1$ and $h = 650$ for a time period of one day each.

$B$ need to have received at least one virus mail in order to appear in the logs. So there is much room for clustering - if these mail deliveries are disjoint.

This requirement is analyzed in Figures 4.6b and 4.6c. Essentially the plots show the same, but with switched roles. For most pairs 100% of all virus mails received by one user were not received by the second user. So, these users should belong to different clusters. But there is a small amount of pairs, for which every received virus mail was also received by the other user (i.e., percentage is zero). This means they received the same emails and belong together. Whether they build a single cluster or several clusters cannot be said. The rising edge of the plot is very steep and shows pairs of users with overlapping email reception and therefore connecting the different clusters.

Finally, Figure 4.6d tells us what percentage of all virus mails received by user $A$, $B$, and both $A$ and $B$ were received by both $A$ and $B$. As in Figure 4.6a most pairs do not share any virus mails, but some do. Some few pairs even received the same virus mails for 100%. Note that the amount of pairs with $N_{AB} = 1$ needs not to be (and indeed is not) the same amount of pairs with $N_A = 0$ or $N_B = 0$ from Figures 4.6b and 4.6c, as in 4.6b and 4.6c the second user may additionally have received other emails and hence does not necessarily have received exactly the same virus mails. But the amount of pairs with non-zero percentage is equals the amount of pairs with non-100 percentage from Figures 4.6b and 4.6c.
Chapter 4 Virus Analysis

Figure 4.6: User-based analysis of July 25th 2008

(a) $N_0$: Virus mails not reported by any of A and B

(b) $N_A$: Virus mails reported by A but not B

(c) $N_B$: Virus mails reported by B but not A

(d) $N_{AB}$: Virus mails reported by A and B
Chapter 4 Virus Analysis

4.3.3 Spam-based Analysis

We further investigate July 25th 2008 using the spam-based analysis. The underlying data was not modified or filtered afore, so we are again analyzing all the available data.

Figure 4.7a looks almost the same as Figure 3.8a from the analysis of the Spamato logs. The only thing that attracts our attention is the generally small amount of pairs caused by the small amount of virus mails delivered. If we have a look at the Size-Plot in Figure 4.7b, we observe the same shapes as in Figure 3.8b, but the ratio between the sizes of the compared receiver-set and the base receiver-set is higher. In other words, we achieve in Figure 4.7b high similarity factors by having big receiver-sets too whereas in the Spamato case the sizes of the compared receiver-sets had big differences. This indicates that there are clusters of notable size, as big groups of users received the same virus mails.

The trend of having better clustering inside the ClamAV logs in contrast to the Spamato logs was confirmed for a third time.
Chapter 5

Collaborative Filtering Limits

In this chapter we simulate an ideal collaborative mail filter to see whether the sparse clustering from the previous chapters can be exploited to increase the detection rate of unsolicited bulk email. Furthermore we will use these simulations to determine the optimal value of some parameters.

5.1 Design

The ideal simulator assumes global knowledge and has access to an oracle. It is a Java application computing the ideal neighbourhood for each user (called its view) and then simulating spam mail reception and detection based on the logs. The computation of the ideal view for each user is based on one week of training data, and the resulting view is then evaluated on one week of test data. Only users active on each day during this time period of two weeks are considered, but all spam mails are taken into account. This way, we avoid the problem of having users in our simulation which are not active at all in the test set or only active for a few times.

5.1.1 Communication Paradigm

Basically there are two possible approaches:

1. Push communication
2. Pull communication
Push communication

In a push-based communication, each user notifies all his neighbours from its view about the reception of a spam mail without any prior request. The advantage is, that a user is already informed about a spam mail upon reception of it and therefore does not lose any time by asking his neighbours. On the other side, there are many needless messages sent as not everybody of the current view will receive the reported spam mail.

Pull communication

A pull-based communication pattern relies on requests. Each user remains idle until he gets a spam mail or a request to lookup the local spam history. If a spam mail arrives and cannot be resolved locally, the user sends requests consecutively to its neighbours until he receives a positive answer or he has asked every user from its view. Therefore notifications are only sent on demand which saves some network bandwidth. It can be optimized by sorting the users from the view according to their rating with respect to the current user or by aggregating several spam checksum into one request/notification. But in contrast to the push paradigm, a user needs first to exchange some messages before he can decide whether a certain email is spam or not.

We decided to use the pull communication pattern in our simulation, as there will be fewer/smaller messages sent in the optimal case.

5.1.2 Parameters

There are some basic parameters which need to be determined:

$V$ This parameter determines the size of the neighbourhood of each user. Each user will have knowledge about $V$ neighbours only and communicate with these users only. Users theoretically belonging to the same cluster should likely get the same neighbours as other users from the cluster.

$K$ The parameter $K$ determines how many hours we are allowed to look into the past for getting spam mails being part of the ratings of our potential neighbours. We need to restrict this, as the users need to exchange their profile in order to compute the rating between themselves.
Chapter 5 Collaborative Filtering Limits

Figure 5.1: Effect of different view-sizes on the detection rate of spam mails (5.1a) and the requests per spam mail (5.1b). We set $K = \infty$ and vary $V \in \{10, 20, 30, 50, 100\}$.

We start by isolating feasible values for the parameters $V$ and $K$ using Spamato logs. Subsequently we will pick the most appropriate values with respect to the achieved detection rate versus workload and compare the results to the behaviour of a completely random graph - as this would be the case if there is no clustering at all.

In the evaluation of the detection rate the spam mails that were first seen by a certain user were neglected, as these spam mails cannot be resolved in any case (at least one user needs to receive the spam mail as very first). This improves the readability of the plots, as the very best result we could achieve is always 100%.

Spamato

For Spamato we used one week of logs starting at January 10th 2008 as training data, and the following week as test set.

First we run some experiments for different values of $V$ whereas $K$ is kept fixed at the maximum value.

In Figure 5.1a we can see the effect of different view-sizes on the detection rate of spam mails. As mentioned before, in the ideal case the detection rate would be always 100% in this plot. On the x-axis all users are listed and the y-axis shows the percentage of spam
mails for each user, that were resolved successfully by asking the neighbours from the view. With increasing view size the detection rate increases, as there are more neighbours to ask and therefore the probability to know somebody who has already received the current spam mail is higher. The clustering coefficients of the system rise from \( \sim 0.32 \) with \( V = 10 \) up to \( \sim 0.55 \) with \( V = 100 \).

If we would increase the view-size up to the size of the whole network, this would be identical to a client/server model with regard to the detection rate. However, we are interested in the case when everybody knows a small subset of users only. The improvement of the detection rate is continuously increasing, but already with a view-size of 30 we have more than half of the users detecting over 80% of their spam mails successfully and 90% of all users still having a hit ratio of 50%.

In the second plot, Figure 5.1b, we can observe the impact of bigger views on the amount of request messages sent. In this plot the "first seen" spam mails were also included because for each such email a request is sent to every user from the view. The y-axis shows the average number of requests needed per spam mail to resolve it. In each case there are some few users that always need to ask everybody from their view. These are probably some very vulnerable users receiving most spam mails as very first. In the plot we can see that the bigger the view-size, the faster the amount of request messages increases. This is the drawback from having a very good detection rate - the workload increases too.

From these experiments we can say that a view-size of about 30 would be feasible. The detection rate is still quite good and the amount of messages moderate.

Now that we have an idea about the view-size, we need to check different values for parameter \( K \), as the profile of each user would grow unbounded if \( K \) is not set. We set \( V = 30 \) and let the simulator run for different values of \( K \).

If we use spam mails received during the last hour only for rating the users and building the views, we get a smaller detection rate as we do not have the best neighbours selected. The problem is, that for many users the amount of spam mails received during the last hour is equals zero. And with a profile not containing any spam mails, we cannot compute the rating between the users and end up with a randomly filled view. Figure 5.2 visualizes this fact. There is a big gap from \( K = 1 \) to the next run with \( K = 6 \) and still a considerable improvement for \( K = 12 \). The clustering coefficients of the system increase from \( \sim 0.28 \) with \( K = 1 \) to \( \sim 0.38 \) with \( K = 6 \) and keep stable around 0.4 for the rest of the experiments. Further increasing of \( K \), does not yield in big improvements and with an average of 37 spam mails received during the last \( K = 12 \) hours we are
already at the upper bound for the profile size. In the last $K = 24$ hours there are in fact 59 spam mails received in average.

We decide to stick to $K = 12$ as our optimal value for getting relevant spam mails.

**ClamAV**

We let the same experiments as before run on ClamAV logs, to detect potential differences in the choice of the parameters. Because of the superior properties of the ClamAV logs over Spamato logs, it was sufficient to require the users to be active on 70% of the training and test set only.

In Figure 5.3 we can see the simulation based on different values for parameter $V$. The plots look almost exactly the same as in Figure 5.1. They increase in the same rate compared to each other with increasing $V$ and the only difference is, that ClamAV overall performs better (although a threshold of 70% only). But the different values for $V$ let the plots develop in the same way as in the simulation with Spamato logs. Analogue the clustering coefficients increase from $\sim 0.39$ with $V = 10$ up to $\sim 0.63$ with $V = 100$.

We decide to choose $V = 30$ as our view size as well as before.

To determine parameter $K$, we let the simulation run again on different values for $K$. Apart from having overall better performance, the plots in Figure 5.4 look similar to the
Figure 5.3: Effect of different view-sizes on the detection rate of virus mails (5.3a) and the requests per virus mail (5.3b). We set $K = \infty$ and vary $V \in \{10, 20, 30, 50, 100\}$.

Figure 5.4: Effect of different values for parameter $K$ on the detection rate of virus mails. We set $V = 30$ and vary $K \in \{1, 6, 12, 24, 48\}$.
Chapter 5 Collaborative Filtering Limits

Figure 5.5: Detection rate and percentage of successful requests of a simulation exploiting clustering and another simulation using a random view only.

<table>
<thead>
<tr>
<th></th>
<th>Random</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spamato System</td>
<td>$\sim 0.03$</td>
<td>$\sim 0.38$</td>
</tr>
<tr>
<td>ClamAV</td>
<td>$\sim 0.06$</td>
<td>$\sim 0.42$</td>
</tr>
</tbody>
</table>

Table 5.1: Clustering coefficients for the ideal simulation.

ones of Spamato in Figure 5.2. But as fewer virus mails are delivered than spam mails, we can easily go for $K = 24$ and profit of its better performance, as in the average case 20 virus mails only were delivered in the last 24 hours which is still less than in Spamato with $K = 12$. A short look at the clustering coefficients does not provide new insights. As usual they are increasing from $\sim 0.19$ with $K = 1$ up to $\sim 0.46$ with $K = 48$.

5.2 Simulation

If we put our findings together and let a simulation run, we get the results shown in Figure 5.5. Each figure shows two plots: The detection rate exploiting clustering and its percentage of successful requests (e.g., how many request out of all ended in a hit) and analogue the detection rate of a completely random graph and its rate of successful requests.
Chapter 5  Collaborative Filtering Limits

The difference between the clustering-plot and the random-plot relies completely on the underlying clustering we were looking for in chapter 3 and 4. The random graph shows the result we would achieve in a decentralized way if there would not be any clustering at all. There is a reasonable improvement by exploiting the clustering which also shows up in the higher clustering coefficients summarized in table 5.1.

The percentage of successful requests is quite low. In Figure 5.5a it is below 20% for most users even if clustering is exploited. This is a high price for the detection rate we get. The random case is even worse. At least the ratio looks little better in Figure 5.5b.

If we compare the performance using Spamato logs versus ClamAV logs in Figure 5.5, we observe that ClamAV logs perform better. This aligns with our findings from chapter 3 and 4 where we found better clustering in the ClamAV case.
Chapter 6

Collaborative Filtering Simulation

In the previous chapter we had a look at an ideal simulation. This allowed us to get a feeling for the parameters and find reasonable values for them. But the simulation was unrealistic in many ways:

Global knowledge
Each user was able to ask an oracle for its best neighbours found during the training set.

Users
Not every user was taken into account, because they were required to be active for a certain amount of days. But in reality there are users which are not active frequently. Whether they are not active because the user itself is not checking his emails or whether the user just does not receive a lot of spam/virus mails does not matter.

Messages
All messages were sent without any delay or loss. This includes requests as well as notifications.

View
Due to the simulators design of using training data and applying the simulation on test data, the view was not evolving. In reality, the view needs to evolve as the network is underlying changes and peers are fluctuating (node churn).
To get realistic results, we adapted our simulations as follows:

**No global knowledge**
There is no global knowledge anymore. Each user knows only few neighbours (random neighbours in the beginning) and develops his neighbourhood in time.

**All users**
All users are taken into account. Not a single one was left out.

**Message-Events**
The simulation is event-driven. Each message (email, request, or notification) was sent in terms of an event, having a delay or even loss (except emails).

**Evolving view**
The view evolves as the underlying network overlay is evolving.

One artificial issue still exists: We are given a limited set of log data. There are always users appearing first towards the very end of our log data, and hence there is no more data to let them develop their neighbourhoods and achieve good results.

Due to the fact that ClamAV logs are gathered at the mailserver and we are building a client-side anti-spam system, they need to be seen as containing a set of users where everybody has push-email in place and everybody has turned his receiving-device always on. Looking forward to the future it should be more or less feasible to assume such conditions.

### 6.1 Design

We would like to compare our realistic simulation with upper and lower limits. Therefore we designed several different simulations, where we assume there is nobody cheating and everybody has the same opinion about a certain email\(^1\).

\(^1\)We just simplify the decision process to one trustworthy statement of a user.
6.1.1 Server-based Simulator

In order to build simulators for upper and lower bounds, we need to disobey some weak requirements of a realistic simulation:

- Global knowledge
- No message delays or loss

In the server-based model we simulate the existing solution of having one central server known to everybody, maintaining a database of all reported spam mails. With a server-based model we get the best possible detection rate that can be achieved using a collaborative filter only, and it will be the general upper bound for collaborative filters of any principle.

If we neglect the first reception of each spam mail, we get a detection rate of 100% for every user. This is truly the case and comprehensible, as each request can be answered by the server.

6.1.2 Optimal-view Simulator

The optimal-view simulation behaves exactly the same as a decentralized system. Each user has its local view and has to send messages to its neighbours to ask for information about a spam mail. However, to have it being ideal the simulation does not meet the weak requirements like in the Server-based Simulator. It lets each user evolve his view by adding the best possible neighbours in each step using global knowledge. Requests and notifications are sent without delay or loss.

6.1.3 Decentralized Simulator

The decentralized simulator respects all the requirements for a realistic simulation and is run on PeerSim [14], a peer-to-peer simulator which makes it possible to evaluate protocols in a realistic way.
Chapter 6 Collaborative Filtering Simulation

Figure 6.1: Communication in the three-layered design. Each layer communicates with the respective layer of other nodes.

Our decentralized simulation consists of 3 layers and the application on each client computer works as follows:

1. A **Peer Sampling Service** offers in each cycle a set of fresh, random nodes participating in the P2P system to the next layer.
2. A **Structure Creating Service** provides in each cycle a set of nodes, having particular properties with respect to the **Selection Function**\(^2\), to the next layer.
3. The **Application** receives emails, checks its own database for possible matches and resolves the classification of unknown emails by asking its neighbours or if unsuccessful the user himself. All messages are sent and received in terms of an event.

The overall design is sketched in Figure 6.1 and each layer is explained in the following in more detail.

---

\(^2\)The **Selection Function** defines how to compare the profiles of two nodes and returns the best matching neighbours.
Chapter 6 Collaborative Filtering Simulation

Peer Sampling Service

The first layer is an implementation of CYCLON [15]. CYCLON is an epidemic-based protocol for the self-organization of a network that resembles a random graph and is used to build an unstructured network overlay. The implementation is cycle- and event-driven which means, that each node in PeerSim starts gossiping in each cycle with one neighbour but the messages exchanged are events that have a certain delay or even loss (see Section 6.1.3). The messages contain a subset of the sender’s view to swap neighbours.

Structure Creating Service

As next layer VICINITY [15] is implemented. VICINITY is an epidemic-based protocol too, but it controls which neighbours to keep in order to build a network whose links reflect a certain proximity. The implementation is also cycle- and event-driven and each node in PeerSim gossips and exchanges a subset of nodes with its neighbours. Unlike in CYCLON, nodes do not swap neighbours when gossiping but each node decides independently which neighbours to keep to optimize its view. The Selection Function of our VICINITY implementation is designed such that the existing clustering found in chapters 3 and 4 gets exploited, i.e., all neighbours get rated with respect to the current profile and the ones with the highest rating will be kept in the local view only.

Application

The last layer is the application layer which is completely event-driven. It models the collaborative email filter and relies on the neighbourhood provided by VICINITY. Each email is received by means of an event and checked against the local database. If no match is found, the application sends a message to each neighbour from its sorted neighbourhood until it receives a notification or it reaches the end of the neighbourhood. In the latter, the user itself is prompted for a decision. As the application layer makes up the design of our own collaborative filter we give a short pseudocode in listing 6.1 to sketch the behaviour of the application layer in more detail.
Listing 6.1: Application layer

```java
Event in {Emails, Request, Notification}

Upon each incoming Event:

case Event is list of Emails {
   /* new emails arrived at inbox */
   for all emails in Event {
      check whether email is already classified
   }
   if there are unclassified emails {
      select best neighbour from view
      send Request containing all unclassified emails to neighbour
      if no neighbour available {
         get user decision
      }
   }
   update profile on vicinity layer
}

case Event is a Request {
   /* another user asks for classification of emails */
   for all emails in Request {
      check whether email is already classified
   }
   send Notification containing all classified emails back
}

case Event is a Notification {
   /* received a response from a neighbour */
   for all emails in Notification {
      remove email from unclassified
   }
   if still unclassified emails left {
      select next neighbour from view
      send Request containing all unclassified emails to neighbour
      if no neighbour available {
         get user decision
      }
   }
}
```
Message delay and loss

Message delays and loss are modeled using a King Latency dataset [16]. King’s estimates are based on direct online measurements (using DNS infrastructure) which fits our needs as best.

6.1.4 Random-view Simulator

As last simulation we would like to know what detection rate can be achieved having no clustering at all and therefore filling the whole view with random neighbours. Users are selected randomly from the set of all users and all messages are sent again without delay or loss. Such an underlying random graph models the lower bound for our simulation as a random graph has the lowest possible degree of structure. Our enhanced collaborative filter should not perform worse than a simple random collaborative filter.

6.1.5 Parameters

In the realistic simulation each user gets initialized with a view filled by \( V \) random other users. As users are joining and leaving the system, views need to evolve and the clusters change by time. This introduces an additional and last core parameter: \( U \), the amount of hours between the updates of the neighbourhood. In the ideal simulation we had no parameter \( U \), as we trained the views on training data and tested its performance on the test data.

To determine parameter \( U \) we let the Optimal-view Simulator run on all available users and plot the results including the first-seen spam/virus mails.

Spamato

Figure 6.2 shows the results for optimal-view simulations using different values of parameter \( U \). Already in chapter 5 the results obtained from Spamato logs were not very performing. But we did not expect that they develop that worse in a realistic environment. Nevertheless. We leave the analysis and conclusions to a later section and focus on determining parameter \( U \).
Figure 6.2: Effect of different values for parameter $U$ on the detection rate of spam mails. We set $V = 30$, $K = 12$, and vary $U \in \{1, 6, 12\}$.

If $U > K$ holds, then spam mails received during a time period of $U - K$ hours get ignored because only spam mails from the last $K$ hours are taken into account for building the view. As we would wait too long for updating our view, some spam mails get received but are never used for clustering. Therefore we increased $U$ up to 12 hours only.

The difference between $U = 6$ and $U = 12$ hours is not very significant. But if we lower it to $U = 1$ we get a bigger gap between the plots and higher efficiency.

**Clam AV**

If we let the optimal-view simulation run on ClamAV logs, we get the result of Figure 6.3a. We are glad to get roughly as good results as in chapter 5 where we had simulations based on a completely ideal environment. Because these results are much more promising as the results based on Spamato logs, we have a closer look at the parameter $U$. Parameter $U$ differs from $V$ and $K$ in the sense of its range. $U$ has not only influence on the Application layer where mainly $V$ and $K$ were influential, but also on the Peer Sampling and Structure Creating Service. Therefore we additionally let our decentralized simulation run on different values for $U$ (Figure 6.3b).

Short time intervals to update the view, $U = 1$ or $U = 6$, clearly lead to the best detection rate. To some extent this relies on the concept of the underlying layers, as the Peer Sampling Service needs to get in a steady-state in order for the Structure Creating
Figure 6.3: Effect of different values for parameter $U$ on the detection rate of virus mails using the optimal-view simulation (6.3a) and decentralized simulation (6.3b). We set $V = 30$, $K = 24$, and vary $U \in \{1, 6, 12, 24\}$.

Service to work properly which itself needs some cycles to get asymptotically close to the ideal view. Dead neighbours or newly joined users get detected faster with frequent updates, so higher cycle frequency is very helpful.

But there is a big difference between Figure 6.3a and 6.3b: The Optimal-view Simulator has a big performance increase from $U = 6$ to $U = 1$ whereas the same change in $U$ lets the Decentralized Simulator only perform slightly better. The Optimal-view Simulator has global knowledge and can therefore ask an oracle at any point in time to provide its best $V$ neighbours out of all 15'449 users. The Decentralized Simulator has just a local view and is able to get other peers known only by means of gossiping in every step. This is a huge advantage of the Optimal-view Simulator and lets it perform much better even with a fast update rate.
6.2 Simulation

In this section we present the results obtained using the optimal values for parameters $V$, $K$, and $U$. In the experiments again all available users were taken into account - no matter whether they appeared at the beginning of the log or at the end of the log for the first time nor whether they are highly active or not. In each figure four plots are drawn:

Server-based Simulation
as upper bound of the detection-rate for any system with any design.

Optimal-view Simulation
as upper bound for a decentralized system.

Decentralized Simulation
as the proposed collaborative filter system.

Random Simulation
as lower bound.

We do not remove the first occurrence of every email from our statistics anymore, as we now have a lower and upper bound (in contrast to the ideal simulation of chapter 5).

6.2.1 Spamato

Surprising results emerge in Figure 6.4: A collaborative system based on a server-based architecture does not yield an extremely good detection rate opposed to our expectations. This is due to the fact, that there is a huge amount of spam mails delivered and they often differ from each other. Actually, most spam mails differ because Spammers often use so called "snowflakes" or other slight variations to trick conventional spam filters. Using robust digests helps to reduce this problem, but it does not completely solve it. As we are operating on unfiltered and unmodified log data we get each single spam mail reported.

The other simulations behave likewise: Using random neighbours performs extremely bad. The detection rate is below 40% for almost all users, and unfortunately exploiting clustering does not improve the detection rate dramatically either. Whether we use the Optimal-view Simulator or the decentralized one seems not to matter very much. Using
Figure 6.4: Comparison of spam mail detection rates using different methods of collaborative filtering. The parameters were set to $V = 30$, $K = 12$ and $U = 1$.

our Decentralized Simulator we get very close to the optimal performance of a collaborative mail filter (optimal-view).

This drastic decrease in performance relies on the fact, that we are considering every and each user. There are lots of users not suited to participate in a peer-to-peer system, as they are not online frequently or even for short time periods only and never again. Such users are not able to build ideal neighbourhoods and therefore achieve a very poor performance only.

6.2.2 ClamAV

If we let the same experiments run on ClamAV logs, we get the results in Figure 6.5. We immediately notice that the Server-based Simulation performs very well and achieves a hit ratio of 1 for almost every user. This is a big contrast to the Spamato logs, where we had a poor performance for the Server-based Simulation. This difference is caused by the different properties of our log source: Whereas we have an extrem high amount of different spam mails in Spamato and compared to it only few distinct users, we have a fractional amount virus mails only but more distinct users in the ClamAV logs (see table 4.1 in chapter 4). Therefore each virus mail is likely to be received by most users and everybody except the first user gets a hit on querying the central database.
Figure 6.5: Comparison of virus mail detection rates using different methods of collaborative filtering. The parameters were set to $V = 30$, $K = 24$ and $U \in \{1, 6\}$.

According to the generally better performance, randomly selecting neighbours yields in a notable detection rate and exploiting clustering enables the system to increase the detection rate significantly once more. In both Figures, 6.5a and 6.5b, we end up having a detection rate of over 50% for $3/4$ of all users with our realistic decentralized simulation.

As mentioned in Section 6.1.5 $U = 1$ lets the Optimal-view Simulator perform clearly better than the decentralized version. However, for $U = 6$ we get very close to the Optimal-view Simulation by using a realistic decentralized one.
Chapter 7

Conclusions

In this thesis we have explored and assessed the possibility of a decentralized collaborative spam and virus mail filter. Although it turned out that our innovative system-design for collaborative filtering is not sufficiently efficient, our findings are still very helpful to understand the problems of collaborative filtering and give some insights into the delivery of unsolicited bulk email.

7.1 Clustering

The analysis has shown that there is some little clustering hidden in the graph connecting users who received the same unsolicited bulk emails. The existence of this clustering has been confirmed using several different analysis methods.

The visualizations gave some initial insight into the topic. After the appropriate filtering had been applied, some nicely clustered graphs could be observed, particularly in daily ClamAV logs. Although the user-based analysis confirmed our visual findings in general, the clustering turned out to be very poor and not well-formed. There are many users interconnecting all the clusters, as they do not belong to a single cluster exclusively. For Spamato logs the results were even worse. Although some clusters could be identified on daily, weekly, and monthly periods, there still remained a large number of unclustered users in each case.

During the analysis of Spamato logs we found out that there are certain usage patterns that make detection of clustering impossible. For example, many users do not
check their emails frequently, while others participate in the system for short time periods only. Regarding the emails, there were lots of spam mails getting reported over and over again during periods of several weeks. This is due to the fact that some users do not check their emails on a daily basis, but possibly also due to weak signatures created by Spamato’s *Earlgrey* filter, which takes only URLs into account for computing the signature and therefore may map different spam mails to the same signature. However, filtering those users and spam mails did not yield in a big improvement of the observed clustering. After all, a system needs to deal with such issues anyway.

### 7.2 Collaborative Filtering

The general approach to collaborative filtering of unsolicited bulk email is to use a server-based architecture. Indeed, a server-based solution constitutes an upper bound for the detection rate of unsolicited bulk email, as the human decisions about received emails are collected at one single point and can be accessed by every other user of the system. However, the detection rate varies significantly depending on the type of unsolicited bulk email, namely spam or virus mails. Whereas the detection rate for virus mails is very good (100% for more than 9/10 of all users), the detection rate for spam mails is just acceptable (more than 80% hits for only 3/4 of all users).

These differences rely on the fact that *distinct* spam mails outnumber virus mails by far. Therefore many users are the first to receive unsolicited bulk email and do not get a positive match from the central server. When discovering these findings, we did not consider degradation of the detection rate due to possible concurrency in reporting and querying particular email signatures. That is, if a number of users receive a spam mail simultaneously. Hence, the detection rate may be lower in a real system.

We conclude that even a server-based collaborative filter should not be run alone but combined with several different filters, including a collaborative one, to achieve high and reliable detection rates.

### 7.3 Decentralized Collaborative Filtering

Using our findings from the analysis to decentralize the collaborative filter did not yield in great success. The clustering found during the analysis can indeed be exploited by a
Chapter 7 Conclusions

A peer-to-peer based system where each peer has local knowledge only, but the achieved detection rate is no more than moderate.

In a decentralized collaborative filter each user knows only few other users, i.e., its neighbourhood, and can communicate with these users only. There are two different paradigms for communication: Push or pull. In a push-based collaboration, each user would send spam/virus notifications to all its neighbours without prior request. Therefore a user may have knowledge of an unsolicited bulk email before its reception and hence a decision can be made immediately. On the other side, many useless notifications are sent as the neighbours may not receive the actual email. Another approach is to use a pull-based collaboration where each user sends out request messages to its neighbours upon reception of an unsolicited bulk email, hence notifications are returned on demand only. If a neighbour has already seen the current spam/virus mail, he will reply by a notification message and the requesting user stops contacting further neighbours. Therefore every notification is useful. The drawbacks of pull communication is that a user first needs to contact one or more of its neighbours before he can reach a decision and a pull-based communication additionally needs request messages. We decided to use pull-based collaboration, as the total amount of messages was lower than in the push case.

The main problem in decentralized collaborative filtering is how to determine one’s neighbourhood. We explored two boundary cases. In the first case each user had a neighbourhood consisting of $V$ random users, which led to the worst case scenario as random users are very unlikely to be related to the current user. This way, a system is modelled which has no clustering at all or where clustering is just not taken into account. In the second case each user was provided with the best possible neighbours out of all existing users, thus exploiting clustering as good as possible. This led to the best case scenario of a decentralized collaborative filter, as every user has the optimal neighbourhood.

Compared to the server-based collaboration, decentralized collaboration with random neighbours achieved an extremely low detection rate only. Using optimal neighbours the detection rate improved significantly compared to the random case but is just moderate in comparison to the good performance of server-based collaboration. These findings were no good prospects with regard to real decentralized collaboration where the neighbourhood may be improved by means of exchanging neighbours only.

Surprisingly, we observed the positive fact that we can get very close to an optimal filter by using local knowledge and exchanging information within the individual neigh-
bourhood only. This is an important result as it shows that the reason for a decentralized
filter to fail lies in the problem domain, i.e., in the environment where we wanted to
apply decentralization.

For ClamAV logs the achieved detection rates are far better than for Spamato logs.
However, the findings above generally apply to both types of unsolicited bulk email,
namely spam and virus mails.

**7.3.1 Practical Limitations**

Apart from having low detection rates in decentralized collaborative filtering there are
additional practical limitations that render decentralization impractical.

First, to get from an extremely weak detection rate to at least a moderate one (from
Spamato to ClamAV case), we would need users who are always online and report their
spam/virus receptions in a timely manner. Additionally the users should participate in
the system for a long time to let their neighbourhoods evolve to an acceptable degree of
homogeneity.

Second, for users to be able to communicate directly to each other, there would be
need for techniques capable of penetrating firewalls and NAT devices. Nowadays many
users are at the back of a NAT router or a firewall, possibly even subject to security
policies of a company. This poses a serious problem, as communication is essential and
the communication partners often change.

Third, we assumed to not have any malicous or unaware users participating in the
system. Furthermore we assumed every user to have the same opinion about a certain
email. Hence, our system takes for granted that users trust each other. We have not
explored the effect of users maliciously blaming legitimate emails as being spam, or char-
acterizing spam mail as legitimate. This is a non-trivial issue, that would significantly
complicate the system.

Finally, if all users were always online, if they were able to directly communicate with
each other, and if they could trust each other, the problem of having too much traffic
would still exist. Even in the best case, $\frac{3}{4}$ of all users achieve $20\%$ of successful requests
only. In other words, $80\%$ of all sent request messages are useless.
## 7.4 Approach for Collaboration

The differences between centralized and decentralized collaborative filtering are summarized in the following table:

<table>
<thead>
<tr>
<th>Server-based</th>
<th>Decentralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ High detection rate</td>
<td>+ No single point of failure</td>
</tr>
<tr>
<td>+ Simple</td>
<td>+ High scalability</td>
</tr>
<tr>
<td>− Single point of failure</td>
<td>− Moderate detection rate</td>
</tr>
<tr>
<td>− Low scalability</td>
<td>− Complexity</td>
</tr>
<tr>
<td></td>
<td>− Network traffic</td>
</tr>
</tbody>
</table>

Having these pros and cons of each concept we can assess whether the employment of a decentralized email filter is worth its costs. Our conclusion is, that the advantages of the decentralized system are insufficient for constructing a successful and promising email filter. Although it would be a nice approach and there are some very desirable properties, the costs are too high: The resource consumption is to high in respect to the achieved benefit. We thus get a far better ratio using conventional concepts and techniques.
Bibliography


64

