SchedMail: Sender-Assisted Message Delivery Scheduling to Reduce Time-Fragmentation

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Dezhong Wang

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Or

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

Although early efforts aimed at dealing with large amounts of emails focused on filtering out spam, there is growing interest in prioritizing non-spam emails, with the objective of reducing information overload and time fragmentation experienced by recipients. However, most existing approaches place the burden of classifying emails exclusively on the recipients' side, either directly or through recipients' email service mechanisms. This disregards the fact that senders typically know more about the nature of the contents of outgoing messages before the messages are read by recipients. This thesis presents mechanisms collectively called *SchedMail* which can be added to popular email clients, to shift a part of the user efforts and computational resources required for email prioritization to the senders' side. Particularly, senders declare the urgency of their messages, and recipients specify policies about when different types of messages should be delivered. Recipients also judge the accuracy of sender-side urgency, which becomes the basis for learned reputations of senders; these reputations are then used to interpret urgency declarations from the recipients' perspectives. In order to experimentally evaluate the proposed mechanisms, a proof-of-concept prototype was implemented based on a popular open source email client *K-9 Mail*. By comparing the amount of email interruptions experienced by recipients, with and without *SchedMail*, the thesis concludes that *SchedMail* can effectively reduce recipients' time fragmentation, without placing demands on email protocols or adding significant computational overhead.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Nadeem Jamali for the continuous support during my Master's study. Without his understanding and help, it would not have been possible for me to complete this thesis. While to many people, getting a degree is a big deal in their lives, to me knowing someone that can be my mentor and friend is bigger.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Ian McQuillan, Prof. Debajyoti Mondal, and Prof. Steven Prime, for their encouragement, insightful comments, and hard questions.

Last but not the least, I would like to thank my parents, Xiangen Wang and Ailian Chen, for supporting me spiritually throughout my life.

Contents

List of Tables

List of Figures

List of Equations

1 Introduction

According to a recent report [8] from *McKinsey Global Institute*, people in workplaces spend an average of 28 percent of their workweeks reading and answering emails. Another report from SaneBox [21] suggests that only 38 percent of the emails in the average inbox are relevant and important; the remaining 62 percent are not important and can be processed in bulk. Meanwhile, with the popularization of smart phones and tablets, an even larger proportion of emails are opened on the go, and the shifting of email communications from desktop to mobile is expected to grow [14]. While enjoying a more convenient and efficient platform of communication, people also have to put up with more frequent interruptions caused by the high volume of email notifications. This has led to a need for effective mechanisms for automated filtering and prioritization of messages to reduce the information overload and time fragmentation experienced by email users.

Spam filtering, as defined in [10], is "an automated technique to identify spam for the purpose of preventing its delivery". Most approaches to spam filtering in the existing literature can be classified into four categories: *content-based solutions*, *reputation-based solutions*, *network-based solutions*, and *economic solutions* [26]. In practice, email systems usually combine more than one approach to filter out spam emails. *Integrated solutions* therefore aim to provide flexible frameworks which enable most of the existing techniques to be applied in concert [26]. Some researchers also have examined existing approaches from a higher level, emphasizing the strategic goals (i.e., *what* to do) more than the specific methods (i.e. *how* to do) of handling spam. For example, Cormack et al. [10] summarized the filtering approaches in two main categories: *hand-crafted classifiers* and *machine learning methods*. Similar classifications can also be found in [4, 6].

Non-spam prioritization can be viewed as a generalization of filtering. Where the goal of filtering is to determine whether particular email messages are legitimate or not, the purpose of prioritization is to determine the importance of messages to a recipient at a finer grain. Consequently, some of the filtering solutions (especially network-based and economic solutions) can also be used for prioritization. There are mainly two different levels of prioritization discussed in literature, *tagging* (or "*foldering*") and *ranking*. The idea of *tagging* is to classify emails with similar attributes into groups, making it easier for users to handle messages of the same type, and very likely also the similar level of importance, all in the same place [12, 30, 16]. In contrast, email *ranking* is a finer way of prioritizing which requires each incoming email message to be assigned a numeric value according to its importance [35]. Most existing approaches to email ranking are based on social network analysis [7, 23, 40, 35, 46, 44], with the assumption that important messages are usually from the people who have been contacted frequently, which is not always true. Some other approaches are based on machine learning. For example, Yoo et al. presented a method to model

and predict personal email priorities with two machine learning algorithms: *ordinal regression* and *classifier cascades* [45].

There remain many open issues in email filtering and prioritization. One issue which is of particular interest to us is that most of the existing solutions tend to place the burden of filtering or prioritization entirely on the recipients' side [1], which is ironic considering that, before messages are read by recipients, senders typically know more about the nature of the contents of their outgoing messages. Although the importance of a message to a recipient is ultimately determined by what the recipient wants to happen, the recipient gets to know about the content of a message only after it has been read by them. Especially, senders who have some type of a relationship with a recipient may often have intimate knowledge of the recipients priorities, even if they do not know the recipients personal schedule (such as whether they are currently in a meeting). In such cases, short of the recipient making the filtering or prioritization decisions themselves, a sender's involvement could be the best input into the process in advance of message delivery.

This thesis presents a set of mechanisms that we call *SchedMail*, which can be added to traditional email clients, to support scheduling of email delivery, and consequently reduce fragmentation of email users' time. This is achieved by shifting a part of the responsibility of email filtering and prioritization from the recipients' side to the senders' side, without placing demands on email protocols or compromising privacy. We have developed specific mechanisms for senders to provide urgency values for their messages, and for recipients to automatically interpret sender-specified urgency values using sender reputations. In addition, there is a mechanism by which recipients can encourage cooperative senders to improve their prioritizing of messages. Finally, in order to experimentally evaluate the mechanisms, we implemented a prototype based on a popular email app *K-9 Mail*. A preliminary study involving a particular use case showed that SchedMail reduced email interruptions to the recipient by about one third.

The rest of the thesis is organized as follow: Chapter 2 presents an overview of some representative studies and products related to email filtering and prioritization; Chapter 3 introduces the detailed design of SchedMail; Chapter 4 presents the implementation of the prototype app as well as some configuration details; Chapter 5 evaluates SchedMail along three dimensions: *features*, *overhead*, and *effectiveness*; finally, Chapter 6 concludes the thesis and raises some questions for potential future work.

2 Related Work

In order to reduce information overload and time fragmentation experienced by email users, both academia and industry have been putting significant efforts into improving existing email systems and developing new email management tools. This chapter presents an overview of some representative works related to email filtering and prioritization, and then briefly discusses the open issues.

2.1 Research on Email

Researchers' early efforts aimed at dealing with large amounts of emails focused mainly on spam filtering, and then they were extended to non-spam prioritization as the amount of email communication traffic over the Internet expanded. During the past years, many efforts have contributed solutions for email filtering and prioritization by targeting different aspects of emails.

2.1.1 Email Filtering

Depending on the different angle of view, the approaches to email filtering in the existing literature can be categorized in various ways. For instance, based on the filtering methods applied, Kaushik et al. [26] classified the prevalent approaches into four categories (see Figure 2.1). *Content-based solutions* decide whether an email is legitimate or unsolicited by checking the content of the email to see if the content matches some predefined patterns [27, 41, 33]. *Reputation-based solutions* determine whether email senders are spammers based on the reputation of the senders, the senders' domains, or the senders' IP addresses, etc. [15, 37, 2, 42]; some approaches also rely on the reputation of spam reporters [47]. *Network-based solutions* are based on the idea that the emails sent by someone within the recipient's social network are highly likely to be non-spam messages [7, 46, 44]. *Economic solutions* try to address the question of who should pay the cost of dealing with spam and how, discouraging spammers by raising the cost of sending spam emails, in terms of human effort, computational cost, virtual currency, etc. [32, 22, 13]. In practice, email systems usually combine more than one of the approaches described above in order to filter out spam emails. *Integrated solutions* provide flexible frameworks which allow most of the existing techniques to be applied in concert [26].

Some researchers have examined existing approaches from a higher level, emphasizing *what* more than *how* things have been, or can be in handling spam. For example, Cormack et al. [10] summarized the filtering approaches into two main categories (see Figure 2.2): *hand-crafted classifiers*, which introduce a

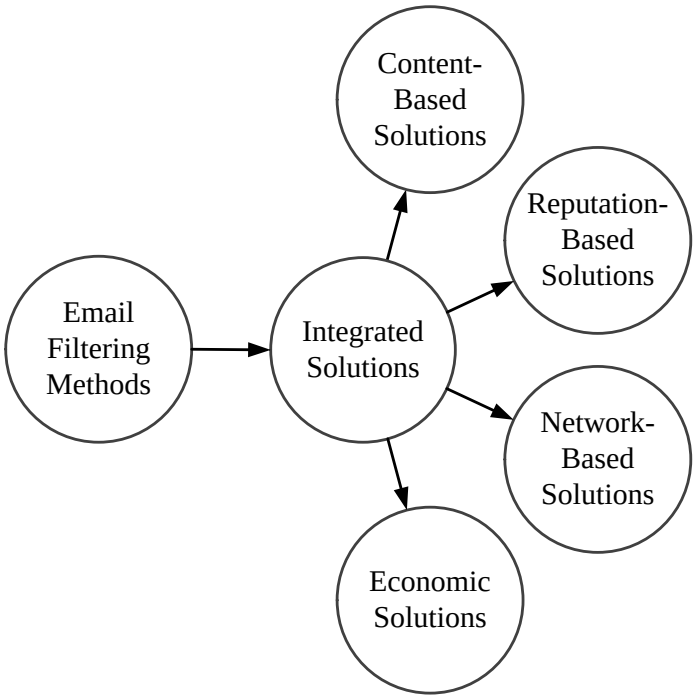

Figure 2.1: Email Filtering Methods

set of manually constructed rules for identifying spam emails; and *machine learning methods* (see also [4]), which try to replace much of the manual work by automatic machine learning. A certain strategic goal can usually be achieved with the support of different above-mentioned filtering methods (e.g., blacklists can be constructed in various ways, such as by email content analysis, sender reputation management, or social network analysis, etc.).

Another similar example of categorization can be found in [6], in which Caruana et al. also discussed two major types of filtering strategies: *machine learning* and *SMTP-based techniques*. While the former is similar to the *machine learning methods* in [10], the latter apply the same approaches as covered by the *hand-crafted classifiers* [10], but from a different angle. Where *hand-crafted classifiers* focus mainly on identifying the key concepts in various classification mechanisms, *SMTP-based techniques* place greater importance in how those mechanisms are applied to the different phases of SMTP sessions.

Although research on email filtering techniques has become relatively mature over the years, some challenges continue to exist: *text obfuscation* and *Bayesian poisoning* are two well-known types of attacks that can degrade the effectiveness of content-based solutions [3]; reputation-based solutions involve the extra burden of keeping the reputation databases like blacklists and whitelists up-to-date; network-based solutions can suffer from low effectiveness if the number of voters (i.e., the participants who run the algorithms of the solutions) in the network is not sufficient [7]; economic solutions as well as some SMTP-based techniques require changes to underlying network protocols which effectively makes them incompatible with the existing

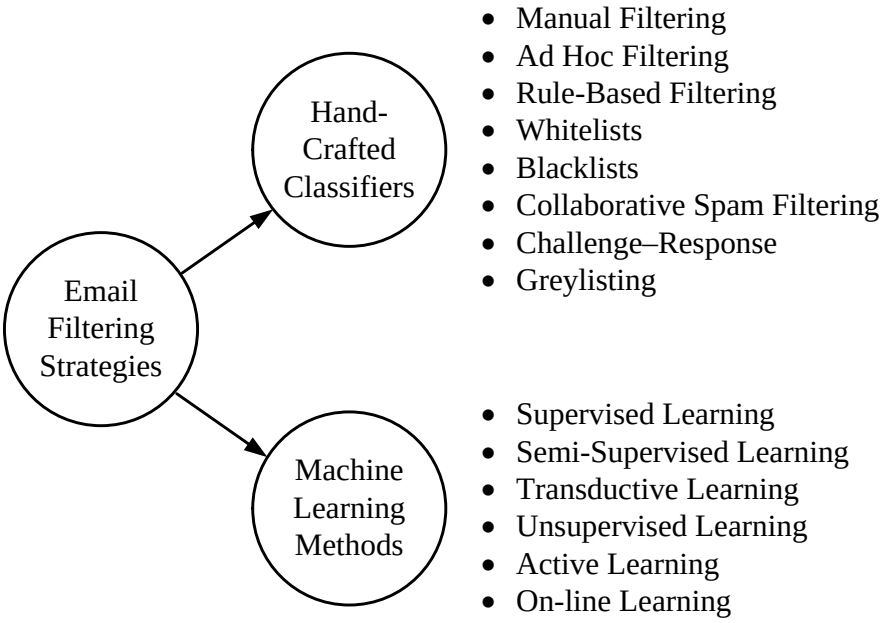

Figure 2.2: Email Filtering Strategies

network infrastructure. Because individual solutions cannot always guarantee satisfactory results, in practice, multiple solutions are usually combined together, or applied in parallel in order to complement each other.

2.1.2 Email Prioritization

Broadly speaking, email prioritization can be considered as an extended form of email filtering. While filtering gives coarse-grained answers about whether a given email message is legitimate or not, prioritization determines the finer-grained degree of importance (or urgency) of each message (spam messages would now have the lowest degree). In a narrow sense, the primary task of filtering is to distinguish spam and non-spam, whereas prioritization works mainly on identifying the importance of non-spam. This difference in points of focus implies that the technical details of filtering and prioritization can differ considerably; however, some of the filtering methods can also be used for prioritization. Like email filtering, prioritization rules can be created either *manually* or *automatically*. Typically, email systems with manual prioritization allow users to define a set of personalized rules, based on which the users' incoming messages can be organized according to their preferences. The techniques involved are quite similar to those for rule-based filtering. Since manual prioritization is rarely discussed separately in the literature, the approaches reviewed in this section are mostly related to automatic prioritization.

As shown in Figure 2.3, basically two different types of prioritization can be found in the literature: *tagging* (or "*foldering*") and *ranking*. The idea of tagging is to classify emails with similar attributes into the same groups (e.g., work, social, news), which makes it easier for users to handle messages of the same

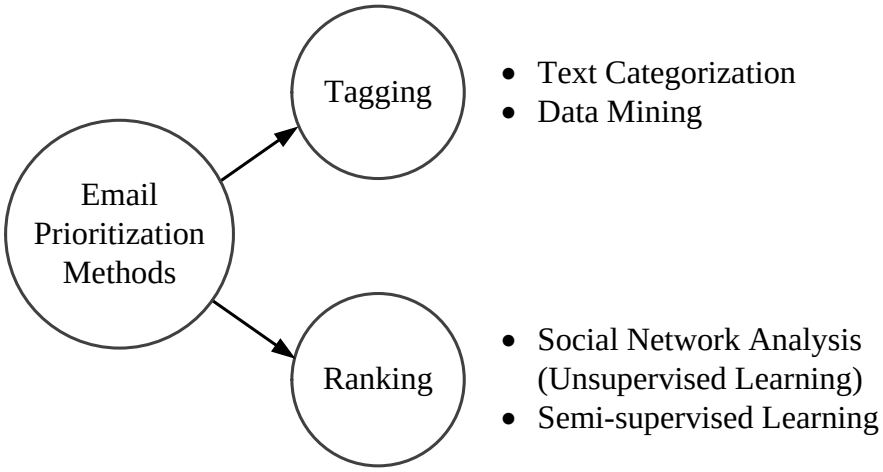

Figure 2.3: Email Prioritization Methods

type, and very likely also the similar level of importance, all in the same place. In early systems, email tagging was treated as a special case of text categorization [9]. However, since most of such approaches are individual-based, an inevitable concern is that an automatic text-based email system may perform well on some users but badly on others. For this reason, Koprinska et al. [29] say that the problem of email tagging is quite different from the standard text categorization problem. In order to decrease potential bias, Koren et al. [30] proposed to exploit the inboxes of the overall population of users (i.e., practically, a massive number of users), and find out some of the most popular tags which work for the general population. A shortcoming of this approach is that, as pointed out in [16], several thousands of commonly used tags can be extracted based on a large-scale corpus, so that the level of abstraction of the results may be not high enough for practical purposes; furthermore, a low level of abstraction also increases the probability of tag-overlapping, i.e., a certain email message can be bound with more than one tag (e.g., as both *Facebook* and *social network*). Grbovic et al. [16] hence suggested giving preference to fewer but consistent classes for a more user-friendly experience. They proposed six latent categories, one for human- and five for machine-generated messages, in view of the fact that the latter ones nowadays take up a large proportion of the commercial webmail traffic. Although experiments showed that the recommended categories covered nearly 90% of the messages in a massive email corpus, an obvious drawback of their approach is the lack of ability to classify human-generated messages. In contrast, Dredze et al.'s [12] idea of classifying email into activities focuses mainly on human-generated messages, and therefore, can be applied together with that of Grbovic et al.'s [16]. The approach is conceptually similar to automated tagging but in a more dynamical way: not only because the set of activities (tags) can grow over time, but also because the participants and their specific roles (instead of only the content of the messages) are also taken into account.

In comparison with tagging, email ranking is a finer way of prioritization which typically requires each

incoming email message to be assigned with a numeric value based on its importance [35]. Most approaches to email ranking in the existing literature are based on *social network analysis* (SNA). For example, Chirita et al. [7] proposed a so-called *MailRank* scheme that exploits the social communication network created via email interactions, in order to identify spam and then build up a ranking among the filtered non-spam. For each email address, MailRank maintains a public reputation score which is shared across the email system (*Basic MailRank*), as well as some personalized scores for all the connected MailRank users (*Personalized MailRank*). Experiments showed that MailRank performed well even in sparse networks, where only a small set of peers take part in the ranking of email addresses. The results generated by MailRank can be sometimes biased as it mainly analyzes only the flow of communication for clustering users and predicting importance. To increase accuracy, other researchers have tried to introduce more metrics during the SNA process. For instance, Johansen et al.'s approach [23] based on *communities of interest* is similar to *Basic MailRank*, but also takes the frequency of communication into account. Since the user clusters in [23] were induced from a community network rather than personalized networks, the approach works well only for a coarse level of prioritization (i.e., identifying incoming messages as either important or non-important). Yoo et al. [46, 44] tried to overcome this limitation by clustering users based on personalized networks instead (like *Personalized MailRank*), however using more metrics for measuring the social importance of users than *Personalized MailRank* did. Moreover, [46, 44] moved another step forward by propagating a set of pre-collected email data with user-annotated importance levels in the network, so that the importance of new messages can be predicted more accurately based on the communication history. More studies of prioritizing email messages by SNA can be found in [40, 35], and the basic principles behind them are similar: first, clustering users based on their historical social interactions, and identifying the social importance of the users based on the flow and/or frequency of the interactions; second, predicting the importance of messages according to the social importance of the senders, either in a public network or in the personalized networks of the recipients. Besides, various metrics can be added during the SNA process to refine the accuracy of results.

An inherent limitation of the existing tagging or ranking approaches to email prioritization is that messages from the same clusters (of either topics or users) are not always of the same importance, and the differences are sometimes nontrivial. Taking email tagging for example, a message tagged with "financial" can be important if it is a bank statement, but it can also be a sales promotion letter which is unimportant to most recipients. As mentioned above, trying to refine the tags does not help the situation because the email systems may end up with too many tags. The problems of SNA-based approaches to email ranking are similar, i.e., classifying emails solely by traffic patterns such as sender, quantity, frequency, etc., cannot capture the priorities of emails all the time [23]. For instance, a message from the manager of a company (i.e., an "important" person) can be an urgent meeting request, whereas it can also be a general holiday greeting. Furthermore, the structures of social communication networks are usually not sharable among different public email service providers due to privacy concerns, which makes it difficult to apply such approaches in public domains.

2.2 Emails in Industry

Email Filtering and prioritization are not only subjects of scientific research, but also wide fields of software development in industry. Blanzieri et al. presented a summary of both commercial and non-commercial software solutions for email filtering in [4], showing that in practice different filtering techniques are usually combined in order to achieve optimal results (as also discussed in Section 2.1.1). In comparison with filtering, email prioritization is still not a well-established industry field, partly because the importance of email messages is mostly user-dependent and thus is difficult to capture or predict. This section presents a brief overview of some market products, which provide email prioritization services, from the functionality perspective.

Gmail, as one of the most influential public email services, has been putting a lot of efforts into improving its webmail interface. The *Inbox Tabs* functionality was launched in 2013, aiming to classify users' incoming messages into categories such as *Promotions*, *Social*, *Updates*, etc. These categories make it easy for users to focus on messages that are important to them and read messages of the same type all in the same place [18]. Users can also choose different views for their inboxes, such as *Important First*, *Starred First*, *Priority Inbox*, etc. Especially, to predict which of the incoming messages are important, *Gmail* automatically takes into account a number of characteristics, including users' frequent contacts, users' topics of interest, recent use of stars, archive and delete, and so forth [19]. Similarly, ***Yahoo! Mail*** provides a so-called *Smart Views* function to its webmail clients, classifying machine-generated messages automatically into a fixed set of categories including *Social*, *Travel*, *Shopping*, *Finance*, etc. However, all human-generated messages are put into one category named *People*, which may not be able to satisfy the requirements of users who have intensive communications with real people. Although ***Outlook.com*** (previously *Hotmail*) webmail also provides an interface for users to sort their incoming messages into different categories, the purpose is mainly for archiving and searching instead of prioritization, as the classification can only be done manually. *Outlook.com* allows users to assign importance to outgoing messages as we do in SchedMail; nevertheless, the importance values are not used at the recipient-side for scheduling email receiving, and no means are provided for reconciling the conflicting interests of priorities between senders and recipients.

Other than server-side solutions, some software products attempt to prioritize incoming messages through client-side analysis (e.g., text categorization, communication-flow statistics, user-action logging). ***EmailTray*** analyzes user actions such as read, respond, delete, forward, etc., as well as interconnections between email senders, to rank incoming messages by importance as belonging in one of four different levels: *top priority*, *low priority*, *no priority*, and *spam*. *EmailTray* allows user-training by tuning the priorities of messages manually; however, since the priorities are sender-based rather than message-based, changing the priority of a message also changes the priorities of all the messages from the same sender. ***SaneBox*** prioritizes important messages and summarizes the rest mainly based on users' communication histories. It provides a

function named *Snooze Folders*, which allows users to temporarily archive selected messages, move them out of the inbox, and return them to the top when it's more convenient for the users to deal with them. A similar *Snooze* function is also built in *Gmail*. Although *Snooze* and SchedMail both work on scheduling incoming messages to be handled at an appropriate time in the future, based on the importance levels, the main difference is that SchedMail schedules the messages automatically on recipients' side while *Snooze* requires manual actions by recipients. ***Boomerang*** is a web app built on top of *Gmail* which allows a user to write a message and schedule it to be sent automatically at a selected time in the future. This potentially provides a sender-side mechanism for coping with email overload. For example, senders who do not want to disturb their recipients with non-urgent messages can schedule the messages to be sent later in the day. However, since this "sender-side scheduling" grants no control to the recipients, it works fundamentally differently from SchedMail.

Table 2.1: Software Products Which Provide Email Prioritization Related Features

Type	**Product Name**	**Tagging**	**Ranking**	**Snoozing**	**Sender-Side Scheduling**	**Training**
Server Side Solutions	Gmail	✓*	✓	✓	×	×
	Yahoo! Mail	✓	×	×	×	×
	Outlook.com	☒	×	×	×	×
Client Side Solutions	EmailTray	×	✓	×	×	✓
	SaneBox	×	✓	✓	×	✓
	Boomerang	×	×	✓	✓	×
	Mailstrom	✓	×	✓	×	×
	Newton	☒	×	✓	✓	×
	Superhuman	☒	✓	✓	✓	×

* ✓– supported; ×– unsupported; ☒ – manually supported.

There are also many other commercial products which offer limited supports of email prioritization, such as *Mailstrom*, *Newton*, *Superhuman*, just to name a few. Table 2.1 summarizes the prioritization-related features of above-mentioned software products.[1] The main principle behind these tools or services is quite similar: from the space angle, classifying messages into different categories and/or prioritizing them according to their importance; from the time angle, scheduling the delivery, receiving, and/or response of messages; moreover, user assistance sometimes is also expected (e.g., *Gmail*, *Yahoo! Mail*, and *Outlook.com* all allow users to report spam through a one-click action).

[1]The purpose of the table is for summarization rather than comparison, which means the products that support more features do not necessarily function better than the ones that support less.

2.3 Discussion

Email filtering and prioritization are active fields in both academic research and industry. However, work on filtering started earlier and is better-established than that on prioritization, partly because prioritization did not draw much attention from researchers until recently when information overload became a matter of concern. In addition, the importance of email messages is mostly user-dependent (e.g., messages that are important to some users may be unimportant to others), with the result that the research on prioritization is usually confronted with the difficulties of collecting private data for analysis and evaluation purposes [44, 45].

For similar reasons, prioritization-related functions implemented by the existing commercial products are still in preliminary stages. Prioritization is realized mostly by way of tagging or coarse-grained ranking (i.e., identifying email messages as either *important* or *unimportant*). Especially, there is still no satisfactory solution for classifying or ranking human-generated messages. However, to users who frequently receive a high-volume of emails from family members, friends, coworkers, etc., automatic fine-grained ranking is not only just useful but also essential. Although many email clients allow users to define rules manually for classifying incoming messages (i.e., rule-based solutions), the approach fails when the sources and/or content of the messages cannot be predicted in advance. Besides, the efforts involved in maintaining the rules can be overwhelming as the number and variety of messages increase.

Furthermore, a common deficiency of the existing approaches to email prioritization is that sender-side participation is barely involved, even though senders typically know more about the nature of the content of their outgoing messages. Some email clients such as *Outlook.com* do allow senders to assign importance to outgoing messages, however, there is no way for the recipient side to interpret the sender-side importance levels according to the recipients' preferences. This is important because recipients and senders often have different interests in determining the importance (or urgency) of a message: even when the sender is trying to be considerate, only the recipient knows their local schedule.

SchedMail focuses mainly on email prioritization, considering that prioritization is a less-explored area. It is not meant to replace the existing filtering or prioritization techniques, but to work as a complementary mechanism. SchedMail utilizes similar techniques to what have been reviewed in this chapter, i.e., prioritization, scheduling, user-assistance, etc., to address the time fragmentation issue. However, what makes SchedMail unique is that senders are significantly involved in the process of recipient-side email ranking. Scheduling of email reception thus is informed by a fine-grained ranking of messages, balancing recipients' interests with those of senders.

3 SchedMail

This chapter presents the detailed design of SchedMail. Section 3.1 describes the core mechanisms of SchedMail, i.e., how urgency values of messages are specified by the senders, interpreted by the recipients' email clients, and eventually used for scheduling the delivery of messages to the recipients. Section 3.2 presents some complementary mechanisms for managing the senders who are supposed to be amenable to the recipients' urgency interpretation.

3.1 Core Mechanisms

Figure 3.1 illustrates the core mechanisms involved in SchedMail. Message senders assign urgency values to their messages, which are recorded in the messages. All messages flow through the email servers to the recipients' side. Once on the recipients' side, each message goes through two steps. The first step interprets the *sender-specified urgency*[1] value by applying the sender's reputation to the value in order to determine corresponding *recipient-focused urgency*. The second step makes a delivery scheduling decision about the message based on a recipient-provided policy, which essentially maps various message urgency values to recipient-selected message delivery time.

3.1.1 User-Assisted Prioritization

Urgency (or importance, priority, which are interchangeable in this thesis) is not a new attribute for email messages introduced by SchedMail. Table 3.1 shows some commonly used header fields [39] that are related to message urgency. The values of these header fields are usually chosen by email senders from the user interfaces of their email clients. Considering that senders typically know more about the nature of the contents of messages, *sender-specified urgency* values can be very useful input parameters for the email ranking process on the recipients' side. However, in practice, the sender-specified urgency values have been used only in limited ways: most email clients display different indicators alongside messages according to their urgency levels; some others may go further, allowing users to sort messages by urgency levels. On the senders' side, for example, an outgoing message can be marked as *important* but there is no guarantee that it will appear prominently in the target mailboxes; on the recipients' side, there is no effective way to keep users from being interrupted frequently by notifications of message arrivals, while still helping them keep

[1] More specifically, it is the sender-side estimation of the urgency to the recipient.

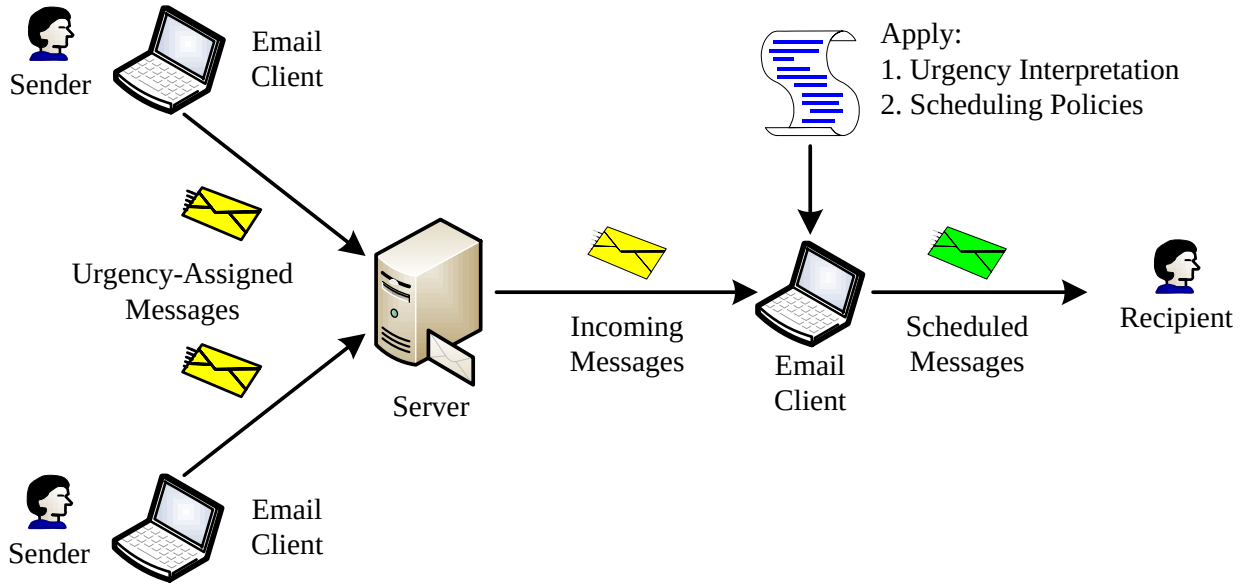

Figure 3.1: SchedMail Design

track of important messages. To make the situation worse, the senders may abuse urgency values (e.g., by assigning the highest urgency value all the time), while the recipients have no way to prevent or even report such improper behavior, such as giving feedback to the senders, reporting to their own email clients or some central authority.

Prioritization is essential to SchedMail, because incoming messages to recipients are scheduled based on their urgency levels. As mentioned above, the original sender-specified urgency values cannot be used directly on recipients' side for deciding delivery time, because senders and recipients have inherently independent interests in when messages should be read. SchedMail introduces the idea of *user-assisted prioritization*, which benefits from cooperation between senders and recipients, and attempts to balance their interests. As shown in Figure 3.2, user-assisted prioritization in SchedMail consists of three main steps: *assigning urgency*, *interpreting urgency*, and *assessing urgency*.

Assigning Urgency

To communicate the urgency assigned by a message's sender, either a new header field needs to be added, or one of the existing commonly used headers shown in Table 3.1 can be utilized. SchedMail adopts the *X-Priority* header field along with its options [20], mainly for two reasons. First, the *5-point rating scale* (also known as *Likert Scale* [31]) used by *X-Priority* is in line with the ranking scheme used by popular online markets such as Google's Play Store, Apple's App Store, Amazon.com etc. A coarser scale would not be able to sufficiently differentiate between messages to effectively inform a meaningful range of choices of message delivery times; and, a finer scale would create more options than may be meaningful to senders, and

Table 3.1: Urgency-Related Email Header Fields

Header Field	Options	Remarks
Importance [28]	High Normal Low	A hint from the originator to the recipients about how important a message is (not used to control transmission speed).
Priority [28]	Normal Urgent Non-urgent	The value of this field may influence transmission speed and delivery.
X-Priority [20]	1 (Highest) 2 (High) 3 (Normal) 4 (Low) 5 (Lowest)	This header field is non-standard but widely used.

may even lead to *decidophobia*.[2] Second, reusing an existing header field leads to greater compatibility when SchedMail (or more generally, SchedMail-supported email clients) is used to communicate with people using other email agents (servers or clients) which may use the same existing header field.

Each outgoing message composed by SchedMail is assigned with one of the five predefined urgency values. The urgency can be either explicitly selected by the sender through SchedMail's user interface, or implicitly assigned by SchedMail with a default value.

Interpreting Urgency

As previously discussed, message recipients often have different priorities, preferences and interests from senders. As a result, the urgency value specified by a sender for a message may not match what the recipient of the message would assign it. For this reason, a mechanism is required for interpreting *sender-specified urgency* values from the recipient's perspective. As shown in Figure 3.3, a *urgency interpretation function* is used to do this. The function is initialized to an identity function, and is subsequently learned based on recipient's assessment of sender-specified urgency values.

Let s be a sender, r be a recipient, p_s be a urgency value specified by the sender, and p_s^r be the corresponding recipient-focused urgency. The urgency interpretation function can then be denoted by $f_{urgency}$, which takes two arguments: p_s, the sender-assigned urgency, and e_s^r, which denotes the sender's reputation from the particular recipient's perspective. A general form of the interpretation function is as below:

$$p_s^r = f_{urgency}(p_s, e_s^r)$$

[2]Decidophobia, introduced by Kaufmann in his 1973 book *Without Guilt and Justice: From Decidophobia to Autonomy*, is the fear of making decisions [25].

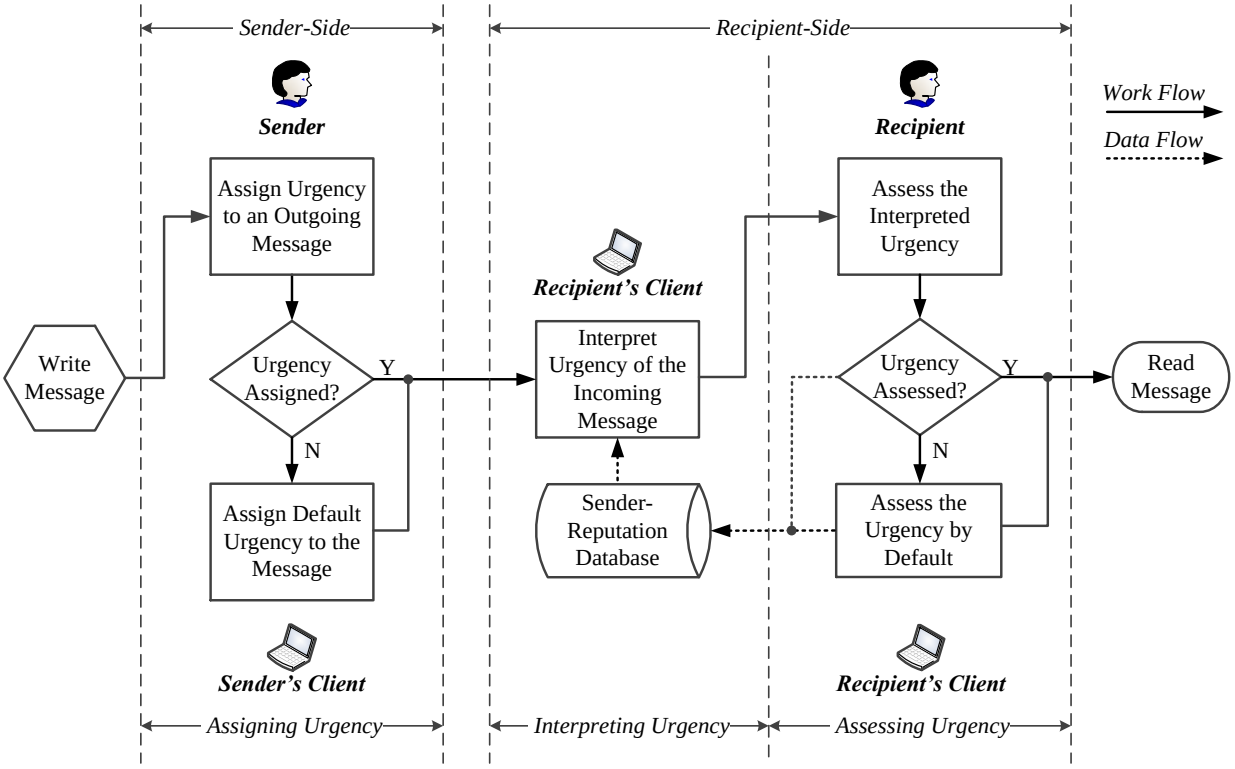

Figure 3.2: User-Assisted Prioritization

In the simplest scenario, e_s^r can be an offset value to the original sender-specified urgency. It may also take different forms depending on implementation, but should always be able to be converted to an offset value. $f_{urgency}$ therefore can be rewritten as below:

$$p_s^r = p_s + \delta(e_s^r) \tag{3.1}$$

where δ is a function for converting reputation scores to urgency offset values.[3] A detailed deduction of e_s^r and δ is presented in Section 3.1.2.

Assessing Urgency

As part of the user-assisted prioritization process, SchedMail allows recipients to assess sender-specified urgency values. Since urgency itself is a fuzzy variable (e.g., it is often hard to tell that the urgency level of a message is not the "lowest" but just "low"), recipients are not asked to assess the original urgency values directly. Instead, recipients have a feedback option as part of the message reading interface, allowing them to pick whether they would have liked to see the message *earlier* or *later*. The email client takes a recipient's

[3]Note that from the definition of *X-Priority*, lower values represent higher urgency levels. So, the sign before δ may need to be reversed accordingly in practice.

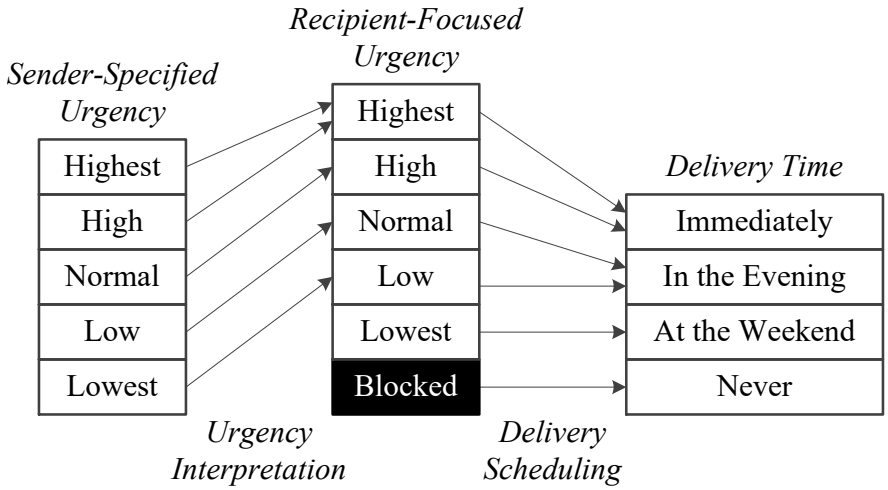

Figure 3.3: Urgency Interpretation and Delivery Scheduling

feedback input, if any, as a type of vote on the message's sender. For each incoming message, choosing *earlier* or *later* casts a *positive* or *negative* vote, respectively. If there is no assessment input received from the recipient (which possibly means the urgency of the message is *just right*, i.e., $p_s^r = p_s$), the email client counts a *neutral* vote automatically.

All these votes could be collected and stored in local databases which are maintained by the email clients. When a subsequent message is received, the recorded votes on the sender of the message are retrieved, and used for calculating the sender's reputation (i.e., e_s^r, see the previous subsection) in order to interpret the message's urgency. The interpreted urgency, again, can be assessed by the recipient. This iteration continues until the learning process reaches a relatively stable state. In practice, in order to reduce storage footprint and computational overhead, an optimized algorithm could store only the aggregates derived from the votes, and calculate e_s^r in an incremental way.

Let $\overset{+}{v}{}_s^r$, $\overset{-}{v}{}_s^r$, and $\tilde{v}{}_s^r$ denote *positive*, *negative*, and *neutral* votes from recipient r to sender s. $\overset{+}{V}{}_s^r$, $\overset{-}{V}{}_s^r$, and $\tilde{V}{}_s^r$ are collections of these votes, respectively. $V_s^r = \overset{+}{V}{}_s^r \cup \overset{-}{V}{}_s^r \cup \tilde{V}{}_s^r$ is the complete sequence of votes from r to s sorted in chronological order. Based on the discussion above, a general form of the function for calculating e_s^r is written as below:

$$e_s^r = f_{rep}(\overset{+}{V}{}_s^r, \overset{-}{V}{}_s^r, \tilde{V}{}_s^r) = f_{rep}(V_s^r) \tag{3.2}$$

In the function, f_{rep} depends on the voting time because typically recent votes predict senders' behaviors and reflect recipients' preferences more accurately, especially in considering that the senders may adjust their behaviors when the feedback-based incentive mechanism is applied (see Section 3.2). On the other hand, the content of V_s^r stays the same until a new vote is cast for s by r or r's email clients, regardless of how much time has elapsed since the last vote was received. Put another way, it is not actually the physical time of the

votes matters, but the relative time, i.e., the chronological orders of the votes in V_s^r.

Let $|V_s^r|$ denote the cardinality of V_s^r (i.e., the total number of votes from r to s), and similarly we have $|\overset{+}{V}{}_s^r|$, $|\overset{-}{V}{}_s^r|$, and $|\tilde{V}{}_s^r|$. Intuitively, the sender's reputation e_s^r should go up as $|\overset{+}{V}{}_s^r|$ and $|\tilde{V}{}_s^r|$ grow, and go down as $|\overset{-}{V}{}_s^r|$ grows. The recipient-focused urgency p_s^r thus is adjusted accordingly based on the definition of Function (3.1). This is meaningful as it addresses the conflicts of interests in when messages should be read between senders and recipients. To be specific, both senders' expectations and recipients' preferences are taken into account by the prioritization process. The senders hurt their own reputation if they keep abusing high urgency values, and consequently, the urgency of their messages are interpreted as low on the recipients' side. Moreover, user-assisted prioritization does not only help senders and recipients to focus on important messages, but also provides a means for those "considerate senders"who do not want to interrupt people with non-urgent messages.

3.1.2 Sender-Reputation Calculation

As discussed in Section 3.1.1, a sender s's reputation in the eyes of a recipient r, i.e., e_s^r, is calculated from r's assessments of the urgency of prior messages received from s. e_s^r is used to adjust the sender-specified urgency values to the corresponding recipient-focused urgency values. Although Function (3.2) illustrates a way of calculating e_s^r in general, the question about how raw reputation scores can be converted to urgency offset values in Function (3.1) remains to be answered.

Email networks are constructed under a typical client/server architecture. However, the user-assisted prioritization introduced by SchedMail is carried out in a peer-to-peer (P2P) manner. There are several existing solutions available for reputation management in P2P networks (such as [24, 17, 43]), but they cannot be adopted directly by SchedMail. For example, [17] utilizes a centralized *reputation computation agent* to prevent malicious peers from altering reputation scores for their own benefits, which is incompatible with the current email infrastructure. [24] requires collaboration among peers, which works well in content delivery networks, but raises privacy concerns for email networks. It is not only that senders' reputation scores cannot be exchanged anonymously, but also that the communication overhead for transferring the scores is not insignificant. Moreover, in typical P2P reputation problems, peers are ranked by their reputation scores; whereas in SchedMail, an email message is ranked based on the sender's reputation from the recipient's perspective, and the sender's indication of message urgency.

This section introduces two types of algorithms for calculating and normalizing senders' reputation scores, and then shows how the scores can be tuned for interpreting sender-specified urgency values.

Weighted Moving Average

In the field of data analysis, *Weighted Moving Average* (*WMA*) is a time series forecasting model that places more emphasis on recent changes in data. Each data point is multiplied by a weight, with the weighting

determined by the number of data points selected. The characteristics of the model provide us a convenient way of verifying what has been discussed earlier in Section 3.1.1, i.e., the latest votes usually reflect senders' behaviors and recipients' preferences more accurately than the older ones.

Let m denote the total number of votes from recipient r to sender s (i.e., $m = |V_s^r|$), and $V_s^r(i)$ be the i^{th} vote in V_s^r. $\{\, V_s^r(i) \mid 1 \leq N \leq m,\ m - N + 1 \leq i \leq m \,\}$, therefore, is a subsequence of V_s^r that contains the latest N votes. If the temporal factor is ignored, a function that maps the three types of votes to the corresponding reputation scores is given as below:

$$f_{score}(v_s^r) = \begin{cases} \overset{+}{c} & v_s^r \in \overset{+}{V}{}_s^r \\ -\overset{-}{c} & v_s^r \in \overset{-}{V}{}_s^r \\ \tilde{c} & v_s^r \in \tilde{V}{}_s^r \end{cases} \tag{3.3}$$

which says that sender s gains a reputation score $\overset{+}{c}$ from a *positive* vote cast by r, loses $\overset{-}{c}$ from a *negative* vote, and gains $\tilde{c}$ from a *neutral* vote. Let $w(i)$[4] be a monotonically non-decreasing function of i (i = 1, 2, ..., N), whose value is the weight of the reputation score contributed by $V_s^r(m - N + i)$. Based on the *WMA* model, a function for calculating the reputation score of s, denoted by $e_s^r(m)$, can be defined as below:

$$e_s^r(m) = \sum_{i=1}^{N} (\ w(i) * f_{score}(V_s^r(m - N + i))\ ) \tag{3.4}$$

where $e_s^r(m)$ is undefined when $0 < m < N$, and typically, we should have $\sum_{i=1}^{N} w(i) = 1$.

As $\overset{+}{c}$, $\overset{-}{c}$, and $\tilde{c}$ are all non-negative real numbers, from Function (3.3) and (3.4), the minimum value of e_s^r is $-\overset{-}{c}$ and the maximum value is $\max(\overset{+}{c}, \tilde{c})$. Normally, a *positive* vote should contribute more reputation score than a *neutral* vote, which means $\overset{+}{c}$ is no less than $\tilde{c}$ and the maximum value of e_s^r is therefore $\overset{+}{c}$. On the other hand, the *X-Priority* header field uses a 5-point rating scale to prioritize messages, which means the range of $\delta(e_s^r)$ in Function (3.1) can be limited to $[-4, 4]$ so that it is possible for any value of *X-Priority* to be adjusted to the lowest or highest level. By mapping $[-4, 4]$ linearly to the value range of e_s^r, i.e., $[-\overset{-}{c}, \overset{+}{c}]$, we can get one set of coefficients for Function (3.3), i.e.,

$$f_{score}(v_s^r) = \begin{cases} 4 & v_s^r \in \overset{+}{V}{}_s^r \\ -4 & v_s^r \in \overset{-}{V}{}_s^r \\ 0 & v_s^r \in \tilde{V}{}_s^r \end{cases} \tag{3.5}$$

In general *WMA* analysis, it is usually up to the data analyst's experience to come up with a weight function $w(i)$ which works for a specific application field. In other words, there is no formally correct way of

[4]Note the notation of $w(i)$ implies that it does not vary from different senders or recipients.

choosing a $w(i)$ that works the best for all cases. Here for the proof-of-concept purpose, a limiting yet widely-used form of $w(i)$ is adopted: $w(i) = \frac{i}{1+2+...+N}$. By substituting the weight function into Function (3.4), it gives:

$$e_s^r(m) = \sum_{i=1}^{N} \frac{i * f_{score}(V_s^r(m-N+i))}{1 + 2 + ... + N} \tag{3.6}$$

where the weight of each vote decreases in arithmetical progression. This specific version of *WMA* is also known as *Linearly Weighted Moving Average* (*LWMA*) [11].

The last remaining question for *WMA* is how to select a suitable value of N. Let Δe_s^r denote the variation of e_s^r along with each individual input vote $V_s^r(i)$, which can be calculated based on Function (3.6):

$$\Delta e_s^r = e_s^r(i) - e_s^r(i-1) = \frac{N * f_{score}(V_s^r(i)) - \sum_{j=0}^{N-1}(f_{score}(V_s^r(i-j-1)))}{1 + 2 + ... + N}$$

From Function (3.5) we know that Δe_s^r achieves the minimal value when $V_s^r(i)$ is a *negative* vote and $V_s^r(i-1)$, $V_s^r(i-2)$, ..., $V_s^r(i-N)$ are all *positive* votes; similarly, Δe_s^r achieves the maximal value when $V_s^r(i)$ is a *positive* vote and the rest are all *negative* votes. Therefore, the range of Δe_s^r is:

$$\begin{aligned} min(\Delta e_s^r) &= \frac{N * (-\bar{c}) - N * \overset{+}{c}}{N * (N+1) / 2} = \frac{-16}{N+1} \\ max(\Delta e_s^r) &= \frac{N * \overset{+}{c} - N * (-\bar{c})}{N * (N+1) / 2} = \frac{16}{N+1} \end{aligned} \tag{3.7}$$

Function (3.7) indicates that the larger the value of N, the less responsive is e_s^r to the latest votes. For example, if we want to control the variation of e_s^r so that it changes no more than one urgency level for every single input vote, we have to set the size of the moving window to be at least 15 (i.e., $N \geq 15$). Figure 3.4 shows some examples of the responsiveness of WMA to a consecutive series of positive votes, with different choices of N.

Simple Exponential Smoothing

In *WMA* we can give more weight to recent votes, but we are limited to the last N votes. *Simple Exponential Smoothing* (*SES*, a.k.a. *exponentially weighted moving average*) [5, 36], as a rule of thumb time series forecasting model, improves on *WMA* by taking all previous votes into account, while still favoring the most recent votes. *SES* offers the flexibility of either giving more weight to recent changes in data, or having a stronger smoothing effect by being less responsive to recent changes. The characteristics of the model provide us a convenient way of studying the trend of senders' changing their behaviors over time, which serves the purpose of determining whether recent votes should be assigned higher or lower weight.

Let $V_s^r(i)$ be the i^{th} vote cast by recipient r for sender s, and $e_s^r(i)$ be s's reputation from r's perspective right after $V_s^r(i)$ is received. Based on the *SES* model, a function for calculating the sender's reputation can

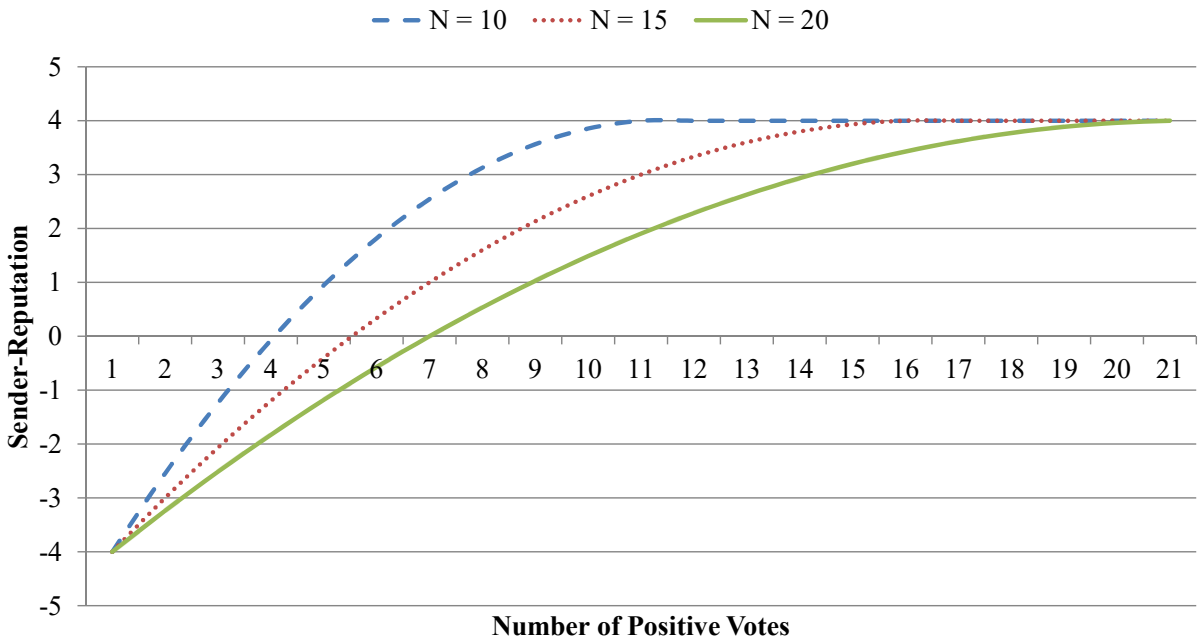

Figure 3.4: Responsiveness of WMA with Different Sizes of Moving Windows

be defined recursively as below:

$$e_s^r(i) = \begin{cases} f_{score}(V_s^r(1)) & i = 1 \\ \alpha * f_{score}(V_s^r(i)) + (1-\alpha) * e_s^r(i-1) & i > 1 \end{cases} \tag{3.8}$$

where α $(0 < \alpha < 1)$ is the *smoothing factor*, and f_{score} is defined in Function (3.3). The range of e_s^r calculated by Function (3.8) is $[-\bar{c}, \overset{+}{c}]$, which is the same as the reputation range derived with *WMA* model. For the same reason, the f_{score} defined in Function (3.5) can also be applied in *SES* model.

In Function (3.8), $e_s^r(1)$ plays an important role in computing all the subsequent reputation scores. Specially, the smaller the value of α, the more important is the selection of the initial reputation score [34, 36]. Although setting $e_s^r(1)$ to $f_{score}(V_s^r(1))$ is one method of initialization in theory, it would be too radical to apply the calculated reputation scores directly during the early stage (e.g., $V_s^r(1) = \overset{+}{v}_s^r$ means the second email from s to r is going to be raised to the highest urgency level regardless based on Function (3.5)).

This general drawback of *SES* in practice is usually addressed by allowing the process to evolve for a reasonable number of observations, and using the average of those observations as the initial forecast. In other words, we could bootstrap the initial reputation score by collecting a certain number (e.g., K) of initial votes from r to s and calculating the average score of them. Function (3.8) then can be refined as below:

$$e_s^r(i) = \begin{cases} \frac{\sum_{j=1}^{K}(f_{score}(V_s^r(j)))}{K} & i = K \\ \alpha * f_{score}(V_s^r(i)) + (1-\alpha) * e_s^r(i-1) & i > K \end{cases} \tag{3.9}$$

where the value of $e_s^r(i)$ is undefined when $0 < i < K$.

Another question about Function (3.8) and (3.9) is the choice of the *smoothing factor* α. Figure 3.5 shows some examples of the responsiveness of SES to a consecutive series of positive votes, with different values of α. In general, α can be any value between 0 and 1. Values of α close to one give greater weight to recently collected votes, while values of α closer to zero are less responsive to recent votes. Although there is no formally correct procedure for choosing α, one commonly adopted approach is to identify values of α that minimize a measure of forecast error like *Mean Absolute Deviation* (*MAD*) or *Mean Squared Error* (*MSE*) ([38, 36]). For example, in Function (3.9), the initial votes (i.e., $V_s^r(1)$, $V_s^r(2)$, ..., $V_s^r(K)$) could be utilized to find an optimized value of α which results in the smallest *MSE*.

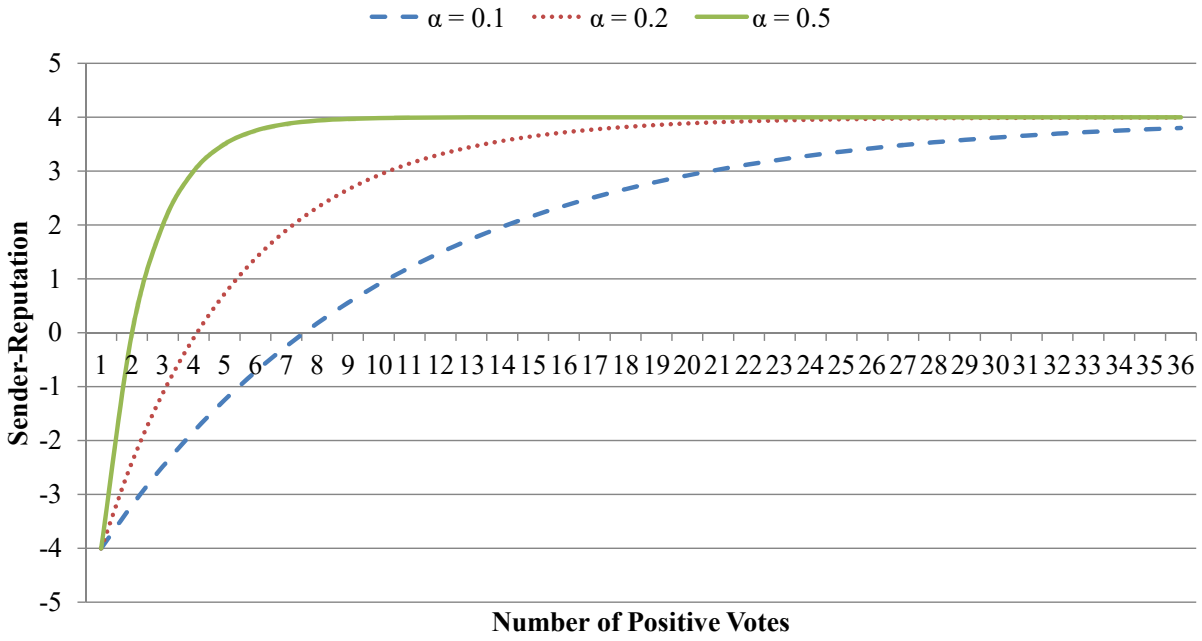

Figure 3.5: Responsiveness of SES with Different Smoothing Factors

Once we get all the required constants settled, including $\overset{+}{c}$, $\overset{-}{c}$, $\tilde{c}$, $e_s^r(K)$ and α, we can calculate the variation of e_s^r along with each individual input vote $V_s^r(i)$ $(i > K)$ based on Function (3.9):

$$\Delta e_s^r \;=\; e_s^r(i) \;-\; e_s^r(i-1) \;=\; \alpha \,*\, (f_{score}(V_s^r(i)) \;-\; e_s^r(i-1))$$

where if $V_s^r(i)$ is a positive vote (i.e., $\overset{+}{v}{}_s^r$), we should have $\Delta e_s^r = \alpha * (4 - e_s^r(i-1))$ based on Function (3.5). Since $e_s^r(i-1)) \subseteq [-\overset{-}{c},\, \overset{+}{c}] = [-4,\, 4]$, we get the range of Δe_s^r which is $[0,\, 8\alpha]$. Similarly, for a negative vote the range of Δe_s^r is $[-8\alpha,\, 0]$, while for a neutral vote the range is $[-4\alpha,\, 4\alpha]$. Therefore, for any type of input vote the overall range of Δe_s^r is:

$$\Delta e_s^r \;\subseteq\; [-8\alpha,\, 8\alpha] \tag{3.10}$$

Function (3.10) indicates the responsiveness of Function (3.8) and (3.9) to recent votes only depends on the choice of α. For example, if we want to limit the variation of e_s^r so that it can change at most one urgency level for every single input vote, we have to choose a value of α from $(0, 0.125]$ instead of $(0, 1)$. However, in practice, if we choose a higher α but still want the variation of e_s^r to below a certain value, e.g., Δ_{max}, we could put this restriction on Function (3.9) so that when $\Delta e_s^r > \Delta_{max}$, let $e_s^r(i) = e_s^r(i-1) + \Delta_{max}$.

Fine Tuning

Although the previous subsections showed how raw reputation scores calculated by Function (3.6) and (3.9) can be normalized to fit the value range of the urgency offsets, the outputs are still not ready to be used to adjust sender-specified urgency values directly. First, the reputation scores, which are floating-point numbers, must be rounded up or down to integers. Below we show two examples about how such conversion could be performed.

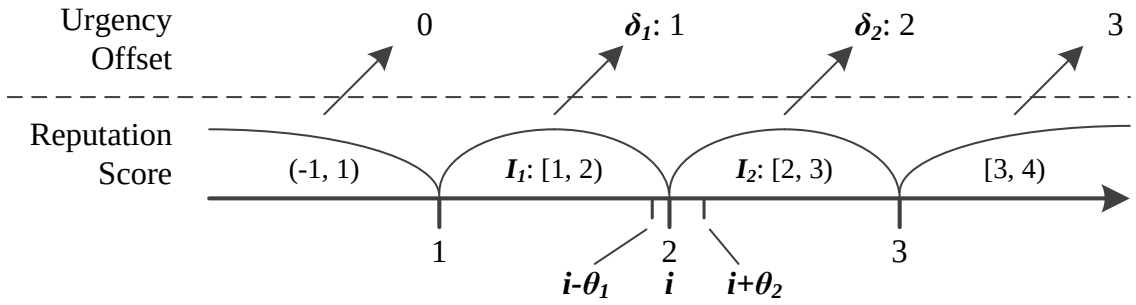

Figure 3.6: Rigid Reputation Score Mapping

Figure 3.6 illustrates a *rigid mapping* from reputation scores to urgency offsets. The value range of the reputation scores is divided into disjoint intervals. Each of the intervals is mapped to one and only one urgency offset value, and so is any reputation score within the interval. In a simulation we carried out, we learned that this type of rigid mapping could suffer from a problem which we call "urgency-bounce." As shown in the figure, i is the boundary of two adjacent intervals I_1 and I_2; $i - \theta_1 \in I_1$, $i + \theta_2 \in I_2$. Consider the following scenario:

a) The current reputation score of a sender s in the eyes of a recipient r, e_s^r, is $i - \theta_1$, and the corresponding urgency offset is δ_1;

b) r receives a new message from s, and then casts a *positive* vote for s; e_s^r becomes $i + \theta_2$, and the urgency offset becomes δ_2 ($\delta_2 = \delta_1 + 1$);

c) r receives another new message from s, but this time casts a *negative* vote; e_s^r changes back to $i - \theta_1$, and the urgency offset changes back, too.

As the communication between r and s continues, a) – c) may repeat a bunch of times. Consequently, the urgency offset, which is used for adjusting urgency values of the messages from s, bounces between δ_1 and δ_2 during the same communication period.

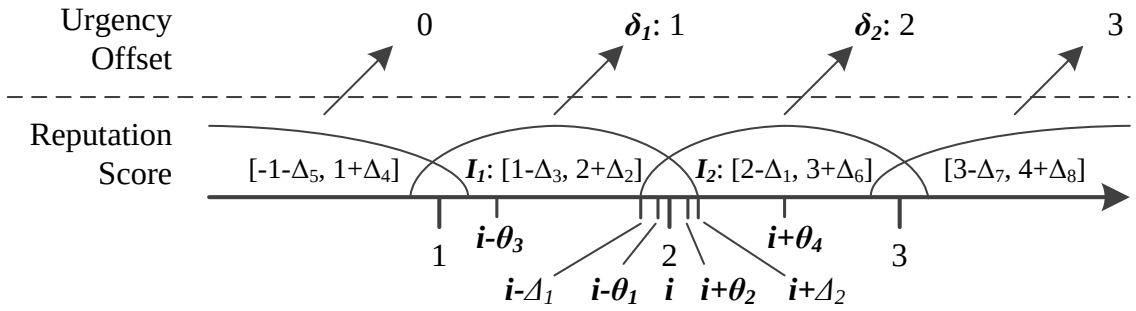

Figure 3.7: Flexible Reputation Score Mapping

In order to cope with the urgency-bounce issue, Figure 3.7 introduces *flexible mapping*. As in *rigid mapping*, the value range of the reputation scores is also divided into several intervals, and each of the intervals is mapped to one and only one urgency offset value. However, in *flexible mapping*, there is a non-empty intersection between every pair of adjacent intervals. As shown in the figure, I_1 and I_2 are two adjacent intervals; $I' = I_1 \cap I_2 = [\, i - \Delta_1,\, i + \Delta_2\,]$; $i - \theta_1 \in I'$, $i + \theta_2 \in I'$, $i - \theta_3 \in I_1 - I'$, $i + \theta_4 \in I_2 - I'$. Suppose the reputation score of a sender s in the eyes of a recipient r, e_s^r, is $i - \theta_3$, and the corresponding urgency offset is δ_1. As the communication between r and s goes on, e_s^r may fluctuate up and down. However, the urgency offset remains at δ_1 as long as e_s^r does not go beyond the range of I_1, in spite of the possibility that e_s^r may fall into I' at some point. Similarly, if the current e_s^r is $i + \theta_4$, the urgency offset remains at δ_2 unless e_s^r changes out of the range of I_2. To see how *flexible mapping* addresses the urgency-bounce problem exactly, let's go through the above sample scenario again:

a) The current e_s^r is $i - \theta_1$, and the urgency offset is δ_1;

b) r receives a new message from s, and then casts a *positive* vote for s; e_s^r becomes $i + \theta_2$, however, the urgency offset remains at δ_1 instead of changing to δ_2;

c) r receives another new message from s, but this time casts a *negative* vote; e_s^r changes back to $i - \theta_1$, and the urgency offset is still δ_1.

3.1.3 Urgency-Based Scheduling

The purpose of email prioritization in SchedMail is for supporting urgency-based scheduling of message delivery. As shown in Figure 3.3, the delivery scheduler takes the delivery policy specified by a recipient, and uses it to map the recipient-focused urgency value of a message to the time when the message should be delivered to the recipient. More specifically, the message is received by the email client, but kept hidden

from the recipient, until the appropriate delivery time, with the purpose of reducing the recipient's time fragmentation. The mapping function is generated automatically from the *user-specified policy*, which itself is created by the recipient through a policy specification interface.

Let P denote a collection of the five urgency levels defined by the *X-Priority* header field, and D denote a set of predefined delivery-time options, such as *immediately*, *in the evening*, *at the weekend*, etc. The *user-specified policy* is simply a mapping function from P to D which can be denoted as $f_{delivery} : P \rightarrow D$, and the mapping relations of $f_{delivery}$ is manually configured by the users from their email clients. Since the options in D are relative time rather than absolute (it is not feasible to use absolute time, either), in order to make delivery decision of a new message, the delivery scheduler should check not only the recipient-focused urgency (i.e., p_s^r, and $p_s^r \in P$) of the message, but also the sending time. Below gives a formal definition of the delivery scheduling mechanism first, and then makes a simple example to explain how it works.

Let T be the set of physical times. The delivery scheduling mechanism of SchedMail is a mapping function from $P \times T$ to D, denoted as $f_{schedule} : P \times T \rightarrow D$. A value from the domain of $f_{schedule}$, denoted as $(\, p_s^r,\, t\,)$, indicates the recipient-focused urgency as well as sending time of a message. For example, a recipient would like all *low*-urgency messages to be delivered *in the evening*. Supposing *evening* starts at 7:00 p.m. (which should also be configurable in practice), the time window for delivering *low*-urgency messages to the recipient, therefore, is from the same day as when the messages were sent, at 7:00 p.m., to any time later. If a *low*-urgency message is sent at 4:00 p.m. one day, but the recipient does not get a chance to check his emails until 8:00 a.m. the second day, the message should be delivered *immediately* instead of waiting until the *evening* of that day.

Although more different types of policies can be designed for convenience of the users, they should all work similarly to the above example. In practice, it would also be helpful to allow the users to configure multiple sets of policies, from which one could be selected to serve the current needs of the users the best. For example, a user can have two sets of policies named *default* and *vacation*. While *default* is configured for daily use, *vacation* postpones all the incoming messages to the end of vacation.

3.1.4 Non-Participant Interactions

It is not meaningful to use SchedMail to help recipients who do not use the mechanisms described above (other than by explicit verbal communication). However, if a sender does not use SchedMail, the mechanisms naturally convert into a coarser-grained sender-reputation based mechanism, which assigns the same urgency value to every message coming from the same sender. In other words, all messages coming from a sender without sender-side urgency values assigned, could be treated as having *normal* urgency, and then interpreted using the learned reputation of the sender.

Two short-cut mechanisms can also be developed to deal with non-participating senders directly, rather than through a learning process. *Blacklisting* simply maps the urgency value of every message from a sender

to *blocked*; *whitelisting* raises the urgency value of every message from a sender up to at least a certain specified floor.

3.2 Feedback-Based Incentive Mechanism

The previous section presented how senders' reputation can be learned at the recipients' side, and used for adjusting the urgency values of incoming messages for the recipients' own interests. However, there could be a potential drawback of this urgency adjusting mechanism, i.e., lowering senders' reputation arbitrarily could prevent the senders from sending really important messages in time. First, low reputation scores do not necessarily mean that the corresponding senders are inconsiderate: the senders may simply have a different understanding of the urgency scheme. Second, in some circumstances, senders, such as family members, friends, and colleagues, may be amenable to be influenced by recipient preferences. Considering that SchedMail relies on senders' judgement of the urgency of outgoing messages anyway, there is an opportunity to grant more control to those "trustworthy" senders, by allowing these senders to adjust their behaviors to accommodate the preferences of the recipients, and eliminating the need for the recipients to second-guess the urgency of the messages.

In order to achieve the goal, SchedMail classifies senders into three categories based on their trust-levels, i.e., *untrusted*, *semi-trusted*, and *trusted*. SchedMail manages the *untrusted* senders using the default urgency adjusting mechanism described in the previous section, while for the *trusted* senders it applies a so-called *feedback-based incentive mechanism*. This section presents the details of this mechanism first, and then briefly discusses how the two mechanisms can be combined to manage *semi-trusted* senders.

3.2.1 Managing Trusted Senders

As illustrated in Figure 3.8, for senders who are amenable to be influenced by recipient preferences, SchedMail supports an option at the recipient end, for recipients to indicate their assessments of the urgency values of incoming messages to the senders. This is intended to be an alternative to adjusting the senders' reputation, with the hope that the senders would accordingly adjust their urgency declarations in the future. In other words, rather than applying senders' reputation on recipients' side, SchedMail lets the senders decide whether or not to be subjected to the recipients' adjustment.

As a result, Function (3.1) can be slightly changed as below:

$$p_s^r \;=\; p'_s \tag{3.11a}$$

$$p'_s \;=\; p_s \;\mid\; p_s + \delta(e_s^r) \tag{3.11b}$$

The new variable p'_s indicates the sender-side urgency value, which can be used directly on the recipient's side based on Function (3.11a). This ensures the senders' ability of delivering important messages even when

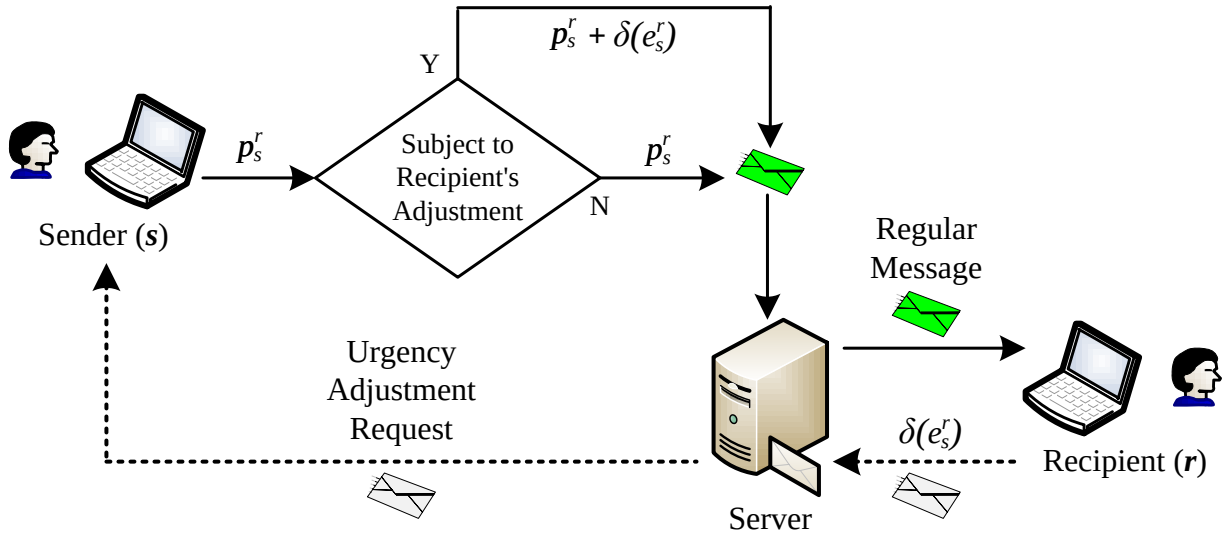

Figure 3.8: Feedback-Based Incentive Mechanism

their reputation scores are low. Although Function (3.11b) means the senders now have the privilege of choosing whether or not to respect the recipients' preferences, the decision about to whom and when the privilege is granted is totally up to the recipients. Especially, on the recipients' side each incoming message still flows through the urgency assessment process, so that the senders' behaviors can be learned. If certain senders are "suspected" to be uncooperative, the recipients could consider moving them into the *semi-trusted* category.

A sender's reputation e_s^r in (3.11b) may be calculated on either side of the conversations. If e_s^r is calculated on the recipients' side like in Function (3.1), the recipients only need to notify e_s^r to the senders when the value of $\delta(e_s^r)$ is actually updated. Besides, the recipients may also send a warning message to the senders prior to each urgency adjustment in order to improve user experience. In contrast, if e_s^r is calculated on the senders' side, the recipients have to pass each and every urgency assessment to the senders.[5] While the former scheme can reduce a greater amount of communication, the latter may allow the senders to adjust their behaviors in a finer level and a faster manner, depending on the specific choices of δ.

3.2.2 Managing Semi-Trusted Senders

In practice, it could be difficult for the recipients to decide whether certain senders should be *trusted*, or they probably do not want to bother making such decisions at all, or even the *trusted* senders can be sometimes uncooperative so that the incentive mechanism breaks. Therefore, other than *trusted* and *untrusted*, a more

[5]The senders and recipients mentioned here are not actually the email users but their email clients, i.e., SchedMail. The users should be kept out from this type of low-level communication, not only for ensuring the usability of SchedMail, but also for some privacy concerns (e.g., recipients would usually not want senders to know the details of urgency assessments).

flexible category *semi-trusted* is defined. The *semi-trusted* senders are managed the same way as the *trusted* senders at the beginning. However, as communication goes on, if the senders' reputation e_s^r exceeds a predefined scope ($E_{min},\ E_{max}$), they will be managed as *untrusted* instead (Note both E_{min} and E_{max} are global settings for a certain SchedMail client, and typically we should have $E_{min} < 0 < E_{max}$). Based on Function (3.1) and (3.11), the urgency interpretation function for the *semi-trusted* senders can be defined as below:

$$p_s^r = \begin{cases} p'_s & e_s^r \in (\, E_{min},\ E_{max}\,) \\ p'_s + \delta(e_s^r) & e_s^r \notin (\, E_{min},\ E_{max}\,) \end{cases} \tag{3.12}$$

As a typical example, all contacts in a recipient's address book can be categorized as *semi-trusted* senders unless otherwise specified. Those senders by default are able to assign arbitrary urgency values to the outgoing messages (when necessary), without worrying about the urgency values being reinterpreted on the recipient's side. However, if the senders abuse the privilege without respecting the recipient's urgency adjustment requests, their reputation would possibly go beyond the recipient's tolerable range, in which case the privilege is going to be revoked (i.e., the recipient's email client will start to reinterpret urgency values of the messages from the "punished" senders).

3.3 Discussion

It may appear that in SchedMail, non-cooperative senders would hurt their own reputations by abusing email urgency values, and consequently get punished in a way that the urgency values of their messages would be interpreted as low on the recipients' side. However, rather than imposing restrictions on non-cooperative senders, the main focus of SchedMail is to try to help people work cooperatively, i.e., enabling considerate senders to help recipients manage their email notifications, and giving control to recipients to decide when messages are delivered.

Although we use the term *reputation* in this discussion, what it measures is the gap between the sense of urgency on the two sides of email communication (or, *urgency gap* in short). The goal is not to identify unsolicited senders or to rank legitimate senders as in other email prioritization approaches. Accordingly, the purpose of collecting and analyzing a recipient's feedback is to learn if there is a consistency or pattern of the urgency gap. In practice, this kind of consistency or pattern might not always exist, in which case it is not meaningful for SchedMail to find it. As a typical example, a non-cooperative sender might assign random or high urgency values to their messages all the time regardless of the actual urgency of the messages. In such a case, SchedMail will struggle to learn a meaningful gap.

In a more comprehensive system, not only the senders' but also the recipients' behaviors can be taken into consideration during the above-mentioned learning process. For instance, for a given incoming message, if no direct feedback is received from the recipient, instead of always recording a *neutral* vote by default,

the email client may also be able to record a different vote depending on how the message is handled. For example, a *positive* vote could be recorded if the recipient replies to a message instantly, whereas a *negative* vote could be recorded if a message is deleted immediately after being opened.

The two algorithms presented in Section 3.1.2 for sender-reputation calculation are a proof of concept of the approach. Iterative improvements would likely be needed before either can be deployed in a real email system. Particularly, for example, both algorithms involve a variable (i.e., the size of the moving window N of *WMA*, and the smoothing factor α of *SES*) which can be tuned to adjust the responsiveness of the algorithms to recent recipient feedback. In practice, we might not want to pick the same N or α for all senders, or even the same sender at all times. A reasonable value of N or α could be selected based on the frequency of communication in the history. Roughly speaking, the output of the algorithms should reflect older feedback more if the communication is sporadic, whereas it should take more of the latest feedback into consideration if the communication is intensive. Although it is technically hard to identify the boundary between *sporadic* and *intensive*, the bottom line is trying to learn the pattern of the urgency gap during a certain period of time as quickly as possible, if such a pattern exists.

While talking about improving existing algorithms for calculating senders' reputation, or designing new algorithms from scratch, one guideline we need to keep in mind is that there always should be a balance between complexity and accuracy of such algorithms. The algorithms should be effective, but not so resource-intensive or attention-demanding for users that it would contradict the design principle of SchedMail, i.e., to shift a part of the burden of email filtering and prioritization from recipients to senders.

There is a potential problem with urgency-based scheduling mechanism presented in Section 3.1.3. Suppose a message m_1 with *normal* urgency is received at time t_1 and scheduled for delivering *in the evening* at t_e; and soon afterwards, another message m_2 with *high* urgency is received and scheduled for delivering *immediately* at t_2 ($t_1 < t_2 < t_e$). If both m_1 and m_2 are from the same sender, and the topics of the two messages are closely related, the recipient would possibly be confused by a broken message context while m_2 is being read, but m_1 is still waiting for delivery.

The problem can be solved implicitly if the sender could simply assign the same urgency level to the two messages. Although the chances are relatively low for multiple messages to be related in this way while the sender is giving them different urgency levels, in order to always present complete message contexts to a recipient, a possible solution could be for the delivery scheduler has to apply an additional rule to guarantee that messages from the same senders are delivered in the original time order: once a certain message is delivered, all earlier pending messages from the same sender, if any, must also be delivered at the same time, regardless of the previously scheduled delivery time. A more sophisticated solution could be by analyzing more email parameters (e.g., titles, contents) to determine which earlier messages are relevant. This type of analysis is beyond the scope of this work.

4 Implementation

In order to demonstrate how the core mechanisms of SchedMail can be applied to existing email clients, a prototype app is implemented based on an open source project *K-9 Mail*.[1] The app is developed using Android Studio, and tested on Android 8.0 platform. For ease of description, the enhanced version of *K-9 Mail* is referred to as *SchedMail App* (or simply *SchedMail*).

As a modern email client, *K-9 Mail* is designed with various useful features like IMAP push email, multi-folder sync, flagging, PGP (*Pretty Good Privacy*) encryption, etc. *SchedMail* extends it by adding two key features: *user-assisted prioritization* and *urgency-based scheduling*. As *prioritization* is the prerequisite for *scheduling*, the full mechanisms of SchedMail require close cooperations between senders and recipients in order to achieve optimal outcomes. However, for individual users, these two features can be enabled and used independently.

4.1 User-Assisted Prioritization

By means of *user-assisted prioritization*, users send messages with appropriate urgency values in the hope that the messages can be scheduled fairly and in a timely manner for the recipients.

4.1.1 Configurations

As shown in Figure 4.1, *Enable Prioritization* is the overall switch for the prioritization functionality. Once it is turned on, SchedMail inserts an *X-Priority* header field (see Table 3.1) to each of the outgoing messages. The assigned value of the field is interpreted on the recipient's side and then used to schedule the delivery of the message.

Allow Urgency Adjustment by Default determines whether urgency values of outgoing messages should be adjusted according to the recipients' preferences by default. This option can be overridden per message in the email composing page as shown in Figure 4.2.

4.1.2 Assigning Urgency

While composing a message (Figure 4.2), the sender can manually assign an urgency value to the message, or leave the urgency as *normal* by default. If *Subject to Recipient's Adjustment* is selected, SchedMail auto-

[1] K-9 Mail: `https://github.com/k9mail/k-9`

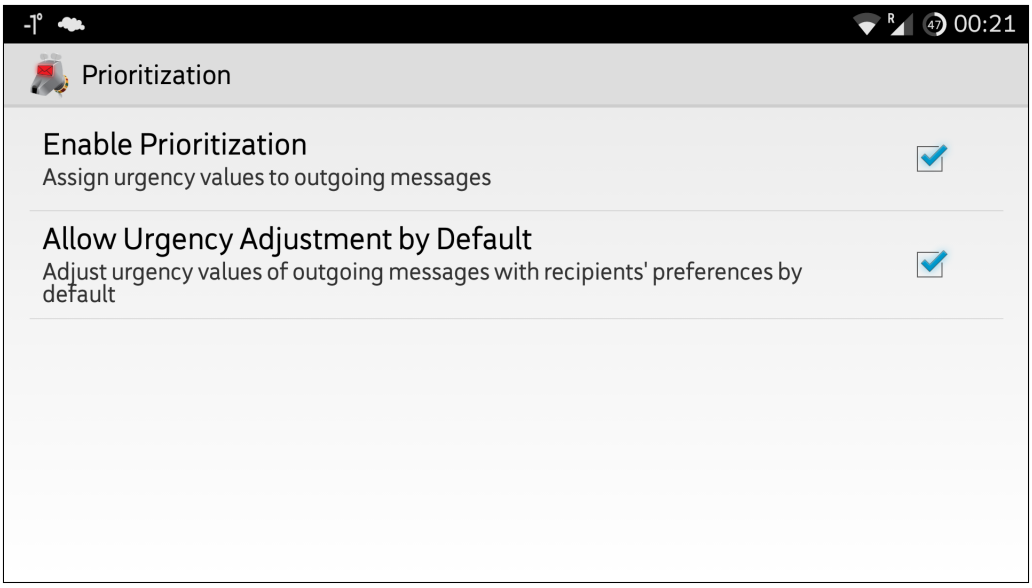

Figure 4.1: Prioritization Configurations

matically adjusts the urgency value based on the sender's reputation before sending the message, otherwise the original urgency value is used.

SchedMail draws its value from those considerate senders who are willing to help recipients manage their email notifications, which implies an application scenario that demands a high level of sender-side involvement. With senders who do not use SchedMail, or do not put enough effort into assigning urgency values to their outgoing messages, the overall effectiveness of SchedMail in terms of reducing time fragmentation may be degraded on the recipient's side.

4.2 Urgency-Based Scheduling

With the help of *urgency-based scheduling*, recipients attempt to schedule delivery of messages according to their urgency.

4.2.1 Configurations

As shown in Figure 4.3, *Enable Scheduling* is the overall switch for the scheduling functionality. When it is enabled, an incoming message is scheduled based on its urgency level calculated using the sender-assigned urgency value and the sender's reputation according to the recipient, as well as the delivery policies configured by the recipient. In the prototype, there are four delivery options predefined for each urgency level, i.e, *immediately*, *in the evening*, *at the weekend*, and *never*, however, these options are easily configurable.

Enable Urgency Assessment allows a recipient to judge the urgency values assigned by senders to the received messages in the message reading interface (see next subsection). When it is disabled, the recipient

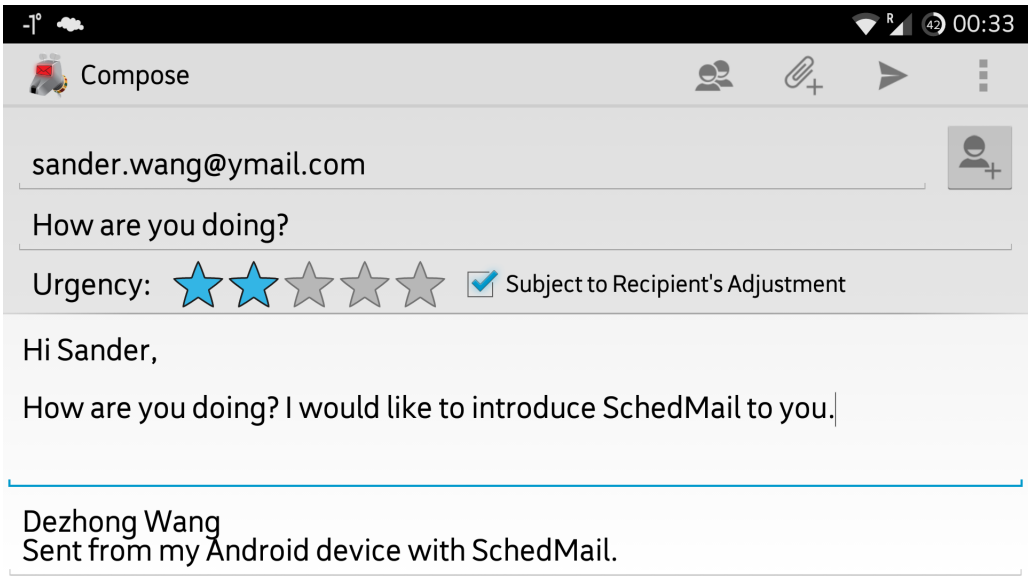

Figure 4.2: Assign The Urgency of a Message

does not judge urgency values assigned by senders, and SchedMail stops updating senders' reputation scores. This also means that if urgency assessment is disabled right after an email account is set up, there is no attempt to judge a sender's reputation, and the sender-specified urgency values are used directly for scheduling incoming messages.

Enable Urgency Adjustment Requests controls whether or not recipients' email clients should request senders to adjust the urgency values of their future messages, which usually happens when the senders' reputation scores are approaching certain thresholds where they would need to be second-guessed at the recipient end. If this option is enabled, SchedMail sends adjustment requests to the corresponding senders when appropriate, in the hope that the senders would make an effort to accommodate recipients' preferences. The requests are sent as regular emails but with a custom header field indicating the value of the urgency offset. Since the format of the requests is known to the email clients on both sides, the email clients can easily identify the requests, hide them from the users, and delete them automatically after processing, without requiring any user interaction. This does not only save the users' time and efforts, but also protects the recipients' privacy so that they would not have to expose their preferences of incoming messages.

4.2.2 Assessing Urgency

On the recipient's side, as shown in Figure 4.4), the user can judge the urgency of an incoming message by clicking the $P\downarrow$ or $P\uparrow$ button, to cast a *negative* or *positive* vote, respectively. If the user thinks that the message has been delivered at about the right time, no action is required and the email client records a *neutral* vote automatically at the back-end.

SchedMail chooses to implement an *up-or-down* voting interface instead of some other possible alternatives

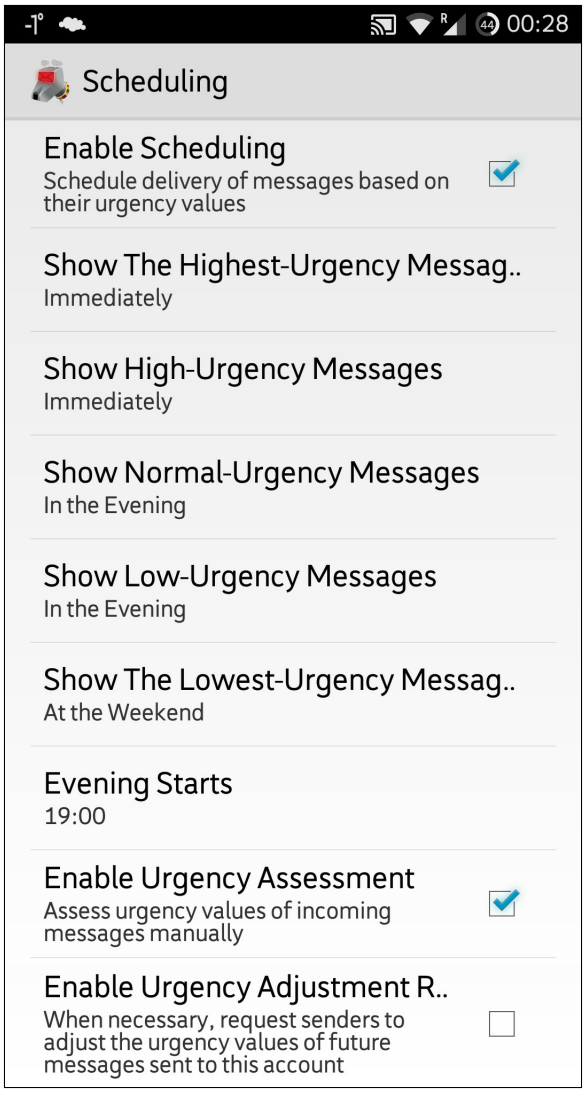

Figure 4.3: Scheduling Configurations

(e.g., a drop-down list for recipients to pick arbitrary urgency values according to their perspective) mainly because of two considerations. First, a minimalist user interface that requires only one user-click is more likely to improve the level of user engagement. Although a more involved user interface can possibly gather more accurate feedback, it may easily become disused because of its high demand of user effort. Second, the recipient's email client decides when to display an incoming message, based on not only the urgency value of the message but also the delivery policy corresponding to that urgency value. The vote that the email client tries to collect, therefore, is actually feedback to a composition of both the urgency value and the delivery policy. Otherwise, for example, if the email client were to directly ask the recipient their perspective about the urgency values, the recipient would have to think about what the urgency values mean to them before taking actions, which would significantly increase the amount of attention required.

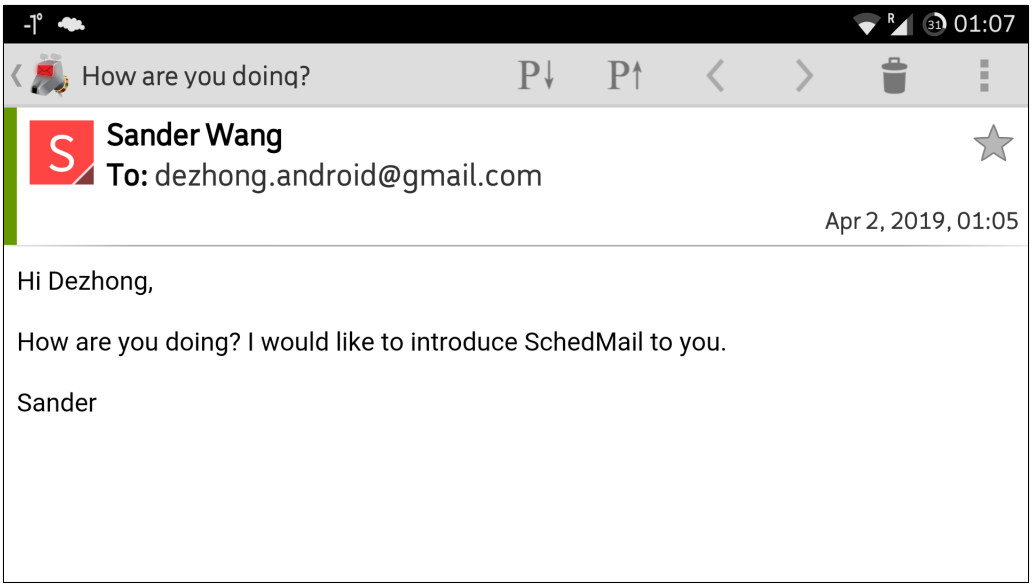

Figure 4.4: Assess The Urgency of a Message

4.2.3 Message Delivery

The inbox page of the email client is designed so that it only displays messages which satisfy the user-specified delivery policies. If the users update the delivery policies at any time, all pending messages are reevaluated so that new messages may show up in the inbox page immediately. In order to avoid messages being stuck at a remote location while awaiting delivery (in the case of a network outage), messages are always downloaded to local storage whenever available but kept hidden from the recipients until the appropriate delivery time.

4.3 Discussion

The overall development effort involved in adding SchedMail mechanisms to *K-9 Mail* was less than one man-month. In the final working version that we used for experiment, five Java classes with approximately 1800 *source lines of code* (*SLOC*) were added. We think that the workload for adding SchedMail mechanisms to other existing email clients should be comparable.

In the prototype, the *feedback-based incentive mechanism* (see Section 3.2) is implemented in a way that both senders' and recipients' privacies are protected. For the recipient in the feedback loop, the only user-specific data passed to the sender's email client is the recipient's urgency offset for that sender. However, the urgency offset value sent to the sender is contextless, because the urgency of a message is only meaningful if it is mapped to a specific delivery policy at the recipient end, and so is the urgency offset. For the sender in a feedback loop, the decision on whether or not to apply the urgency offset requested by the recipient, is totally up to the sender and transparent to the recipient.

Although the recipient's privacy about the scheduling of email delivery is protected in the feedback loop, their judgement of the sender-assigned urgency of the message is not fully hidden from the sender. It is still technically possible for the sender to extract the urgency offset value from the local database of their email client. An easy option could be letting the recipient decide to which senders do they want the feedback-based incentive mechanism to be applied, i.e., who should be treated as "*trusted*" senders (see Section 3.2.1).

There are two levels of policies involved in SchedMail: *system policies*, such as the algorithms designed for sender-reputation calculation, the rules defined for urgency-based scheduling of message delivery, etc.; and *user policies*, such as senders' decisions on whether or not to be subject to recipients' urgency adjustment requests, recipients' preferences of mapping different urgency levels to the corresponding delivery options, etc. While the performance of a specific implementation of SchedMail could vary because of different *system policies* plugged in and *user policies* configured, the main focus as well as the main contribution of SchedMail is coming up with the infrastructure where systems can customize their own policies for various application scenarios. For example, when deployed in an enterprise email system, SchedMail can adopt reputation management algorithms which are optimized specifically for users within private networks, whereas in a public email system the choices of such algorithms should mostly consider a general population.

For these reasons, the prototype we presented in this chapter serves only a proof-of-concept purpose, i.e., to demonstrate a way of adding SchedMail mechanisms to an existing email client. In order to deploy the mechanisms in an email system for wide use, the above-mentioned *plug-in* policies must be designed carefully and refined iteratively.

5 Evaluation

The main contribution of this thesis is enabling use of explicit sender input in the decision-making of when email messages are delivered. Considering urgency and schedule as two separation of concerns, the goal is to have sender-recipient collaboration to decide the urgency of a message, and allow a recipient to have a policy to translate urgency to time of delivery. Our primary way of evaluating this work is to confirm that it is realizable, and that it can be implemented without placing significant additional demand on the users' attention, or on the underlying email infrastructure. Section 5.1 addresses the first question by comparing the features of our proof-of-concept prototype with comparable existing systems. Section 5.2 evaluates the user-attention and computational resource cost of SchedMail by looking closely at the attention-demand of every task to be carried out by the users, and computational cost of the functions required for supporting SchedMail.

In addition, we also carried out a preliminary informal study to get some insights into possible effectiveness of SchedMail in reducing users' time fragmentation, as explained in Section 5.3. The effectiveness of this approach is fundamentally limited by the stability of a gap between the perception of urgency of messages on the two sides of email communication.

5.1 Features

We do a qualitative evaluation by comparing SchedMail with three of the most relevant approaches to email prioritization: *content-based solutions* (mainly for tagging), *network-based solutions* and *economic solutions* (mainly for ranking). Table 5.1 shows a summary of the comparison.

a) *Sender-Side Contribution.* SchedMail involves sender-side contribution for email prioritization. Although *economic solutions* are able to do similar things in principle, they require changing of email protocols. Other approaches like *challenge-response* [10] require sender effort to dissuade spammers, but are only suitable for spam filtering.

b) *Urgency-Based Scheduling.* SchedMail schedules the delivery of messages based on recipients preferences. Message delivery scheduling in existing systems can only be done by senders manually, which gives recipients no control over the delivery time.

c) *Fine-Grained Interaction.* SchedMail involves a feedback loop between senders and recipients, trying to balance both senders and recipients interests when there is a mismatch of urgency assessments. The

Table 5.1: Comparison with Prioritization Approaches

	Feature	SchedMail	Content-Based Solutions	Network-Based Solutions	Economic Solutions
a)	Sender-Side Contribution	✓*	×	×	✓
b)	Urgency-Based Scheduling	✓	×	☒	☒
c)	Fine-Grained Interaction	✓	×	×	×
d)	Privacy Protection	✓	×	×	✓
e)	Scalability	✓	×	×	×
f)	Coexistence	✓	✓	×	×

* ✓– supported; ×– unsupported; ☒ – It is possible but has not been done before.

other solutions can do nothing more than just deleting unwanted messages at the recipient end.

d) *Privacy Protection.* The only user-specific data generated by SchedMail is the reputation value learned at the recipient end. The data is only stored in the recipient's email client. When the optional mechanisms for sending feedback to the sender are enabled, the only extra data shared between the two sides of the conversation is the recipient's urgency offset value for the corresponding sender, which is hidden from both users by their email clients. Even though it is technically possible for the sender to extract the offset value from their local database, which means that, to some extent, the recipient's preferences of incoming messages could be exposed, the decision about with whom and when the preferences are shared is entirely up to the recipient. Moreover, the recipient's privacy about the scheduling of email delivery is always protected, considering that the urgency-based scheduling polices are still only stored on the recipient's side. Other approaches like *content-based solutions* need to scan the contents of messages, and *network-based solutions* need to analyze social relationship structures.

e) *Scalability.* SchedMail operates in a peer-to-peer manner without needing special support from email servers, while most of the other approaches put the majority of work on recipients servers.

f) *Coexistence.* SchedMail can coexist with other email clients. Email scheduling at the recipient end works even when senders do not use SchedMail. However, in this case, the effectiveness of SchedMail would become limited to sender-level decision making because messages from the non-participating senders are all treated as having *normal* urgency.

SchedMail utilizes similar techniques for addressing the time fragmentation issue experienced by recipients to those which have been presented in many existing approaches, e.g., user-assistance, prioritization, scheduling. However, what makes SchedMail unique is that senders are significantly involved in the process of recipient-side email ranking, and therefore, there is qualitatively richer interaction between senders and

recipients.

5.2 Overhead

Because a part of the goal of SchedMail is to tackle the email overload problem, it is important to keep the extra efforts expected from the users, and the resources required, to the minimal.

5.2.1 User Effort

In order to make SchedMail operate effectively, there is only a constant amount of effort required of the sender, i.e., the effort of clicking once to pick one of a selection of urgency values, for each message sent.

The situation is slightly more complex on the recipients' side, but ultimately it requires less work. There would be a constant amount of effort, which also involves one click to indicate "up" or "down" disagreement with a sender-assigned urgency value. In addition, depending on the choices offered to a recipient in setting the delivery scheduling policy, there would be an occasional effort involved in fine-tuning the policy.

5.2.2 Computational Cost

There is negligible additional processing cost at the sender end. At the recipient end, there is per-message-received processing cost for interpreting the sender-specified urgency value, applying the scheduling policy, and revising the sender's reputation. The first two are simple mathematical function calls; the last involves pulling the senders record from a local database, applying the revision function – which could be a simple mathematical function – and storing the record back. Because of the relative infrequency of email arrivals, this does not represent a significant cost to any modern computing device.

The only additional data transferred for the core mechanisms is the urgency value assigned by the sender, which are sometimes already sent by existing popular email clients anyway. There is additional storage required to maintain information about each sender's reputation. However, for the simplest reputation management system like *SES*, only one numeric value is stored per user, which is insignificant.

5.3 Effectiveness

The specific type of time fragmentation we address in this thesis is caused by interruptions from notifications of email arrivals. As presented in Chapter 3, SchedMail tackles the issue by grouping non-urgent messages and scheduling them to a later time slot for bulk processing. Intuitively, the extent of fragmentation should be positively related to the number of interruptions. The question of how much SchedMail can help in reducing the recipients' time fragmentation is therefore equivalent to: to what extent can SchedMail lower the frequency of interruptions. In order to answer the question, we need to come up with a formal way of measuring the level of interruptions first.

For a recipient r, let m_i be the i^{th} message delivered to r at time t_i. M_j^k ($1 \leq j \leq k$) is the sequence of messages between j and k, i.e., (m_j, m_{j+1}, ..., m_k). ϵ denotes a constant which is a relatively small period of time. A function for calculating the *number of interruptions* introduced by M_j^k (denoted as I_j^k) is defined recursively as follows:

$$I_j^k = \begin{cases} 1 & k = j \\ I_j^{k-1} & k > j,\ t_k - t_{k-1} \leq \epsilon \\ I_j^{k-1} + 1 & k > j,\ t_k - t_{k-1} > \epsilon \end{cases} \tag{5.1}$$

The core implication of Function (5.1) is that, if the time interval between two consecutive messages m_i and m_{i+1} is below a preselected threshold ϵ, the arrival of m_{i+1} is not considered a new interruption to the recipient. In practice, a proper value of ϵ can be set to the average amount of time for processing a single message (e.g., ten minutes in our experiment).

From the definition of Function (5.1), a function for calculating the *frequency of interruptions* – number of interruptions per message (denoted as F_j^k) can be derived as follows:

$$F_j^k = \frac{I_j^k}{|M_j^k|} = \frac{I_j^k}{k - j + 1} \tag{5.2}$$

Now suppose ${M'}_j^k$ contains the same sequence of messages as M_j^k, but is scheduled with different delivery times by SchedMail. ${I'}_j^k$ and ${F'}_j^k$ are the corresponding *number* and *frequency* of interruptions calculated using Function (5.1) and (5.2), respectively. The effectiveness of SchedMail, in terms of reducing the frequency of interruptions (or time fragmentation), could then be measured using the following:

$$\phi \ = \ 1 - \frac{{F'}_j^k}{F_j^k} \ = \ 1 - \frac{\frac{{I'}_j^k}{k-j+1}}{\frac{I_j^k}{k-j+1}} \ = \ 1 - \frac{{I'}_j^k}{I_j^k} \tag{5.3}$$

5.3.1 Experiment

A preliminary case study was carried out to get further insights into this. Experimental data was collected from the email communications between the author and seven voluntary users during a three-month period. The volunteers who usually would communicate with instant messengers, for the study of the thesis, were asked to use emails instead. In order to take the *non-participant interactions* (see Section 3.1.4) into consideration, the volunteers include not only five SchedMail users (identified as $S_1 \sim S_5$), but also two regular email client users (identified as R_1, R_2). The SchedMail users were asked to do their best to estimate and assign the urgency values of their outgoing messages. Recipient-side urgency assessment and urgency-based scheduling of message delivery, however, were only carried out on the author's side (the volunteers might also have tried the features but no data was recorded).

In the specific SchedMail prototype that was used for running the experiment, senders' reputation scores were calculated utilizing the *SES* model (see Function (3.9)): the *smoothing factor* α, which controls the responsiveness of the model to recent votes, was set to 0.2 (see Figure 3.5); and the number of messages used for bootstrapping the initial reputation scores, K, was set to 5. The message delivery policies were configured as Table 5.2. Since all participants were trusted users, the policy "*never*" which was introduced for blocking emails at certain urgency levels was never applied in the experiment.

Table 5.2: Email Delivery Policies

Recipient-Focused Urgency	**Delivery Policy**	**Delivery Time**
Highest	*Immediately*	Right after the messages are received
High		
Normal	*In the Evening*	Between 19:00 $\sim$ 23:59
Low		
Lowest	*At the Weekend*	Between Saturday 09:00 $\sim$ Sunday 23:59

The daily distribution of the emails delivered to the author's email box during the experiment is shown in Figure 5.1. The horizontal axis denotes the actual dates when the messages were sent, which were not always the same as the dates when the messages were notified to the recipient due to the delivery policies applied. The vertical axis denotes the total number of messages received from the volunteers on a daily basis. For each message, the following attributes were recorded:

a) The sender and recipient of the message;

b) The time when the message was sent;

c) The urgency value of the message assigned by the sender or the sender's email client;

d) The time when the message was received by the recipient's email client;

e) The time when the message was notified to the recipient (not necessarily the same as the time when the message was read);

f) The urgency assessment vote cast by the recipient, if any;

g) The immediate reputation score of the sender.

5.3.2 Analysis

As illustrated in Table 5.3, the experimental data has been analyzed in three steps. First, based on messages from each individual sender, we calculated the following values that were relevant to the effectiveness of SchedMail: the number and frequency of interruptions (denoted as I and F, respectively) that could have

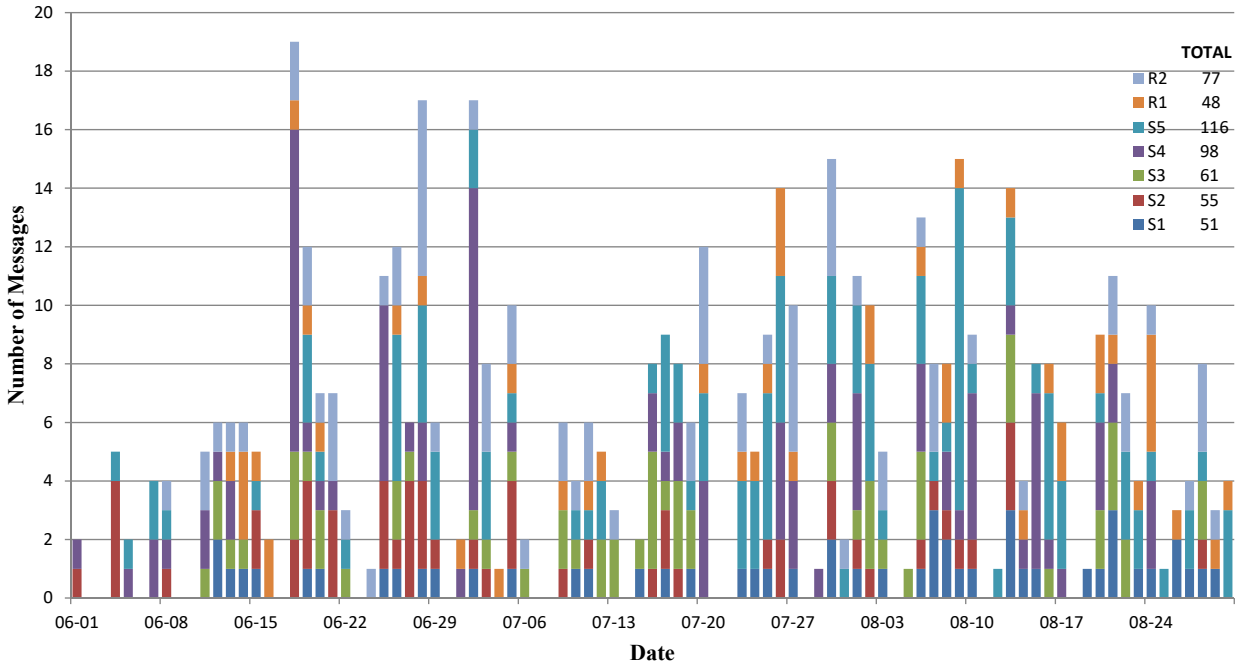

Figure 5.1: Daily Email Distribution

been experienced by the recipient if SchedMail mechanisms had not been utilized; the actual number and frequency of interruptions (denoted as I' and F', respectively) experienced by the recipient during the three-month period, with SchedMail mechanisms applied; and the corresponding effectiveness ratio of SchedMail according to Function (5.3). Second, we examined the same values as which had been calculated for individual senders, but based on messages from two different groups of senders, respectively: the senders who used SchedMail during the experiment (i.e., $S_1 \sim S_5$), and the rest who did not (i.e., R_1 and R_2). Last, we calculated the same group of effectiveness-relevant values based on all sampled messages.

In this particular case study, SchedMail achieved a promising overall effectiveness ratio at **34.0**% with all sampled messages. However, the different effectiveness ratios listed in Table 5.3 are for reference only, rather than for comparison purpose. The effectiveness of SchedMail can be impacted by various factors, e.g., the spread and significance of incoming messages, users' understanding of the urgency schemes, and the configurations of delivery policies. Most critically, it is impacted by the existence or absence of a stable gap between the perception of urgency on the two sides of email communication. Therefore, it is not always meaningful to compare the values of different effectiveness ratios. The goal of SchedMail is to enable a greater balance between lowering time fragmentation and seeing messages at the right time. Otherwise, for example, a recipient could just set an aggressive delivery policy to postpone all messages to the *weekend*, achieving high effectiveness in reducing time fragmentation, but at the expense of missing urgent messages, which is typically impractical.

Table 5.3: Effectiveness of SchedMail

Users	**Emails**	I	F	I'	F'	**Effectiveness**
R_1	48	47	97.9%	44	91.7%	6.4%
R_2	77	72	93.5%	44	57.1%	38.9%
S_1	51	49	96.1%	44	86.3%	10.2%
S_2	55	50	90.9%	40	72.7%	20.0%
S_3	61	61	100%	43	70.5%	29.5%
S_4	98	84	85.7%	56	57.1%	33.3%
S_5	116	110	94.8%	79	68.1%	28.2%
$R_1 \sim R_2$	125	117	93.6%	85	68.0%	27.4%
$S_1 \sim S_5$	381	335	87.9%	235	61.7%	29.9%
All	506	438	86.6%	289	57.1%	34.0%

5.4 Discussion

There are two levels of focus during the evaluation of SchedMail: *implementability*, i.e., whether the strategic mechanisms (e.g., user-assisted prioritization, urgency-based scheduling) can be implemented without placing demands on email protocols, or adding significant computational overhead; and *effectiveness*, i.e., to what extent the plugged-in policies (see Section 4.3) can help users reduce their time fragmentation. The mechanism issues are the more important concern to us than the policy issues, because the mechanisms construct the foundation of SchedMail, while the polices are substitutable for specific implementations.

6 Conclusion and Future Work

This thesis presented mechanisms we have developed to shift a part of the burden of email prioritization from recipients to senders, with the objective of enabling recipients to decide which messages should be delivered to them and when. Although there is already a large body of work on email ranking, what makes our solution unique is that senders are significantly involved in the process of recipient-side ranking of messages. Scheduling of email delivery thus is informed by a sender-assisted fine-grained ranking of messages, balancing recipients' interest to reduce interruptions with seeing messages at the right time.

A prototype, *SchedMail*, has been implemented by installing the core mechanisms required for this functionality into a modern mobile app, *K-9 Mail*. SchedMail offers the ability to schedule delivery of messages based on learned reputations of sender and sender-assigned urgency values for messages. The prototype implements the mechanisms without requiring any change in email protocols, and without a significant computational overhead.

The approach is evaluated primarily in terms of the novelty of features available in SchedMail, and the user attention and computational resources required to support SchedMail. We showed that SchedMail involves a set of original features which do not exist in other relevant approaches: user-assisted prioritization, urgency-based scheduling, as well as fine-grained interactions between senders and recipients. SchedMail supports these features without sacrificing user privacy, placing demands on email protocols, or adding significant computational overhead. Additionally, an informal preliminary study was carried out to obtain insights into the effectiveness of SchedMail in reducing time fragmentation. To this end, effectiveness was carefully defined, and analysis showed that SchedMail reduced about one third of the recipient's time fragmentation in a particular use case.

There remain a number of opportunities for improving SchedMail. For example, we did not address the problem of synchronizing users' reputation databases across the number of devices that a user may consume emails on. This could be addressed implicitly if the mechanisms are implemented in a webmail client, in which case the reputation data is stored on the server side. Otherwise, if no direct support is provided by the email service providers, some third-party cloud storage service could still be utilized for storing and synchronizing the data.

There is a lack of adequate verification for the algorithms (i.e., *WMA* and *SES*) we presented for computing senders' reputation. The algorithms have been widely used in business analysis but not much for predicting short-term peer-to-peer reputations. Although the purpose of introducing the algorithms into SchedMail is just a proof of concept, different algorithms would probably generate different reputation schemes and

consequently impact the results of evaluation.

With the development of machine learning technologies, a lot more sophisticated algorithms which are suitable for modeling the reputations for our needs could be designed. Nevertheless, one thing we need to keep in mind is that the key objective of SchedMail is not to develop a perfect reputation management model. While adopting an existing model, we also need to create a balance between its complexity and accuracy. For example, if an accurate model consumes too many resources for reasoning about its inputs, it would contradict the goal of SchedMail which is to shift a part of the resource demand for email prioritization to the senders' side.

Due to privacy concerns, it has been a common difficulty for researchers to gather email related data for their research use. Therefore it is inherently difficult for relevant studies to be thoroughly evaluated. For the same reason, the data we collected during the study covers only a very small portion of common use cases, which could somewhat bias the conclusions we drew about the effectiveness of SchedMail in reducing time fragmentation. However, for SchedMail specifically there is still room for improvement, since during the process of evaluation it only requires some email metadata (e.g., delivery time, urgency offset values) instead of message contents.

Finally, the approach we presented in this thesis is easily generalizable to SMS, instant messaging, and various types of application-triggered notifications on personal devices. We are looking into possible challenges in extending SchedMail to these communication mechanisms.

References

[1] Eric Allman. The Economics of Spam. *Queue - Distributed Development*, 1(9):78–80, December 2003.

[2] Dmitri Alperovitch, Paul Judge, and Sven Krasser. Taxonomy of Email Reputation Systems. In *27th International Conference on Distributed Computing Systems Workshops*, page 27, Toronto, Canada, June 2007.

[3] Alexy Bhowmick and Shyamanta Hazarika. E-Mail Spam Filtering: A Review of Techniques and Trends. *Advances in Electronics, Communication and Computing*, pages 583–590, January 2018.

[4] Enrico Blanzieri and Anton Bryl. A Survey of Learning-Based Techniques of Email Spam Filtering. *Artificial Intelligence Review*, 29(1):63–92, March 2008.

[5] Robert Goodell Brown. *Smoothing Forecasting and Prediction of Discrete Time Series*. Englewood Cliffs, NJ: Prentice-Hall, 1963.

[6] Godwin Caruana and Maozhen Li. A Survey of Emerging Approaches to Spam Filtering. *ACM Computing Surveys*, 44(2), February 2012.

[7] Paul Alexandru Chirita, Jorg Diederich, and Wolfgang Nejdl. MailRank: Using Ranking For Spam Detection. In *Proceedings of The 14th ACM International Conference on Information and Knowledge Management*, pages 373–380, Bremen, Germany, November 2005.

[8] Michael Chui, James Manyika, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Hugo Sarrazin, Geoffrey Sands, and Magdalena Westergren. The Social Economy: Unblocking Value and Productivity Through Social Technologies. *McKinsey Global Institute*, July 2012.

[9] William W. Cohen. Learning Rules that Classify E-Mail. In *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning and Information Access*, pages 18–25, March 1996.

[10] Gordon V. Cormack and David R. Cheriton. Email Spam Filtering: A Systematic Review. *Foundations and Trends in Information Retrieval*, 1(9):335–455, April 2007.

[11] John Devcic. Weighted Moving Averages: The Basics. Online: `https://www.investopedia.com/articles/technical/060401.asp`.

[12] Mark Dredze, Tessa Lau, and Nicholas Kushmerick. Automatically Classifying Emails into Activities. In *Proceedings of The 11th International Conference on Intelligent User Interfaces*, pages 70–77, Sydney, Australia, January 2006.

[13] B. Curtis Eaton, Ian A. MacDonald, and Laura Meriluoto. Filtering and Email Pricing As Solutions To Spam. *Canadian Journal of Economics*, 46(3):881–899, August 2013.

[14] Lauren Fisher. Email Marketing Benchmarks: Key Data, Trends and Metrics. *eMarketer, Inc.*, February 2013.

[15] Jennifer Golbeck and James Hendler. Reputation Network Analysis for Email Filtering. In *Proceedings of The First Conference on Email and Anti-Spam*, Mountain View, California, USA, July 2004.

[16] Mihajlo Grbovic, Guy Halawi, Zohar Karnin, and Yoelle Maarek. How Many Folders Do You Really Need?: Classifying Email into a Handful of Categories. In *Proceedings of The 23rd ACM International Conference on Information and Knowledge Management*, pages 869–878, Shanghai, China, November 2014.

[17] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A Reputation System for Peer-to-Peer Networks. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 144–152, California, USA, June 2003.

[18] Google Inc. Add or Remove Inbox Categories & Tabs in Gmail. Online: `https://support.google.com/mail/answer/3094499?hl=en&ref_topic=3394594`.

[19] Google Inc. Importance Markers in Gmail. Online: `https://support.google.com/mail/answer/186543?hl=en&ref_topic=3394594`.

[20] Qualcomm Inc. Eudora® Email 7.1 User Guide for Windows. Online: `https://fossies.org/windows/mail/eudora/Eudora_71_User_Manual.pdf`, September 2006.

[21] SaneBox Inc. Email Overload in The Enterprise. Online: `http://d1faw2u3edxi8l.cloudfront.net/pdfs/Email%20Overload%20White%20Paper.pdf`.

[22] Nadeem Jamali and Hongxing Geng. A Mailbox Ownership Based Mechanism for Curbing Spam. *Computer Communications*, 31(15):3586–3593, September 2008.

[23] Lisa Johansen, Michael Rowell, Kevin Butler, and Patrick McDaniel. Email Communities of Interest. In *Proceedings of The 4th Conference on Email and Anti-Spam*, Mountain View, California, USA, August 2007.

[24] Sepandar D. Kamvar, Mario T. Schlosser, and Hector GarciaMolina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International Conference on World Wide Web*, pages 640–651, Budapest, Hungary, May 2003.

[25] Walter Arnold. Kaufmann. *Without Guilt and Justice: From Decidophobia to Autonomy*. David McKay, April 1973.

[26] Saket Kaushik, William Winsborough, Duminda Wijesekera, and Paul Ammann. Email Feedback: A Policy-Based Approach To Overcoming False Positives. In *Proceedings of The 2005 ACM Workshop on Formal Methods in Security Engineering*, pages 73–82, Fairfax, Virginia, USA, November 2005.

[27] Ahmed Khorsi. An Overview of Content-Based Spam Filtering Techniques. *Informatica*, 31(3):269–277, October 2007.

[28] Graham Klyne and Jacob Palme. RFC 4021: Registration of Mail and MIME Header Fields. Online: `https://tools.ietf.org/html/rfc4021`, March 2005.

[29] Irena Koprinska, Josiah Poon, James Clark, and Jason Chan. Learning to Classify E-mail. *Information Sciences*, 177(10):2167–2187, May 2007.

[30] Yehuda Koren, Edo Liberty, Yoelle Maarek, and Roman Sandler. Automatically Tagging Email by Leveraging Other Users' Folders. In *Proceedings of The 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 913–921, San Diego, California, USA, August 2011.

[31] Rensis Likert. A Technique for the Measurement of Attitudes. *Archives of Psychology*, 22(140):5–55, June 1932.

[32] Thede Loder, Marshall Van Alstyne, and Rick Wash. An Economic Solution to the Spam Problem. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 40–50, New York, USA, May 2004.

[33] Wanli Ma, Dat Tran, and Dharmendra Sharma. A Novel Spam Email Detection System Based on Negative Selection. In *Fourth International Conference on Computer Sciences and Convergence Information Technology*, pages 987–992, Seoul, South Korea, November 2009.

[34] Steven Nahmias. *Production and Operations Analysis*. McGraw-Hill/Irwin, March 2008.

[35] Ronald Nussbaum, Abdol-Hossein Esfahanian, and Pang-Ning Tan. History-Based Email Prioritization. In *International Conference on Advances in Social Network Analysis and Mining*, pages 364–365, Athens, Greece, July 2009.

[36] National Institute of Standards and Technology. NIST/SEMATECH e-Handbook of Statistical Methods, 6.4.3.1. Single Exponential Smoothing. Online: `http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc431.htm`, October 2013.

[37] Vipul Ved Prakash and Adam O'Donnell. Fighting Spam with Reputation Systems. *ACM Queue - Social Computing*, 3(9):36–41, November 2005.

[38] Handanhal V. Ravinder. Determining The Optimal Values Of Exponential Smoothing Constants Does Solver Really Work? *American Journal of Business Education*, 9(1):347–360, May 2013.

[39] Pete Resnick. RFC 5322: Internet Message Format. Online: `https://tools.ietf.org/html/rfc5322`, October 2008.

[40] Ismael Rivera, Myriam Mencke, Juan Miguel Gomez, Giner Alor-Hernandez, and Angel Garcia-Crespo. Collaborative OpenSocial Network Dataset Based Email Ranking and Filtering. In *Third International Conference on Systems*, pages 241–246, Cancun, Mexico, April 2008.

[41] Chun Wei, Alan Sprague, Gary Warner, and Anthony Skjellum. Mining Spam Email to Identify Common Origins For Forensic Application. In *Proceedings of The 2008 ACM Symposium on Applied Computing*, pages 1433–1437, Fortaleza, Ceara, Brazil, March 2008.

[42] Mengjun Xie and Haining Wang. A Collaboration-based Autonomous Reputation System for Email Services. In *30th IEEE International Conference on Computer Communications*, pages 1–9, San Diego, California, USA, March 2010.

[43] Li Xiong and Ling Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, July 2004.

[44] Yiming Yang, Shinjae Yoo, and Frank Lin. Personalized Email Prioritization Based on Content and Social Network Analysis. *IEEE Intelligent Systems*, 25(4):12–18, August 2010.

[45] Shinjae Yoo, Yiming Yang, and Jaime Carbonell. Modeling Personalized Email Prioritization: Classification-based and Regression-based Approaches. In *Proceedings of The 20th ACM International Conference on Information and Knowledge Management*, pages 729–738, Glasgow, Scotland, UK, October 2011.

[46] Shinjae Yoo, Yiming Yang, Frank Lin, and Il-Chul Moon. Mining Social Networks For Personalized Email Prioritization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 967–976, Paris, France, July 2009.

[47] Elena Zheleva, Aleksander Kolcz, and Lise Getoor. Trusting Spam Reporters: A Reporter-Based Reputation System for Email Filtering. *ACM Transactions on Information Systems*, 27(1):3:1–3:27, December 2008.