MailHistory: toward introspection and social translucence in email communication

Yiming Liu

December 8, 2007

> *"I would typically think it should be 24 hours. Even if they don't have the answer they should at least say, 'I got your email'..."* [12]

> *"I don't want the people to think that they can get an immediate [response]...[that] I'll drop anything for you..."* [12]

Over the last few decades, email has become an important communication and collaboration tool. As a global, networked messaging system, email allows individuals to exchange information and carry on conversations, without requiring participants to gather at a physical location (as in face-to-face communication), at a particular point in time (as in telephone conversations), and with relative alacrity (as opposed to postal mail).

Yet, email communication carries a fair degree of uncertainty, due to the nature of asynchronous messaging and the interaction design of the email system. Between the time that an email is sent and a response is received, there are few meaningful cues available to the sender about how the recipient is handling the message. "When will I get a response to this email?" is a common question for email senders [12]. Unless the participants have some other means of establishing context or a prior history of email interaction, answering this question is difficult. Deciding

how (and when) to follow-up on an apparently unanswered email may also pose a problem for the sender: whether to wait patiently (in hopes that the recipient did not overlook the message), or whether to call or send a follow-up message (and risk annoying the recipient, who may be busy or in the process of handling the message).

On the other hand, recipients must actively balance their information triage goals [10] with their desire to project a social image [10, 12]. In certain cases, users may wish to maintain an image of responsiveness (e.g. when interacting with their supervisors), or may wish to communicate a sense of unavailability (e.g. when particularly busy, when on vacation, or when promoted to a high-ranking position), or perhaps both at the same time depending on context. There are few means of introspection available to email users, to analyze their own patterns of interaction and see a reflection of their projected images. There are also few means of communicating these images to potential email senders, without first building up a history of email interactions.

In this paper, I describe MailHistory, a possible design of an email analytics tool/service that extracts, analyzes, and visualizes temporal patterns in the user's email interactions. Most of the extracted data and visualizations are presented to the user alone, while a subset is posted for public consumption. In presenting a socially translucent [6] view of the recipient's current and historical email management habits, the system may help senders to find the appropriate interpretation and response to email communication silence and recipients to conduct introspective analysis on their own communication behaviors.

1 Background

1.1 The Problem

Consider a typical email interaction: a sender S composes a message to some receiver R, asking R for some information or to perform some task. S sits down in front of an email client,

writes down a message, address it to R, and hits "Send".

Now, S must wait diligently for R's response. S usually has some upper bound on the expected response time. That is, S expects a response from R within some timeframe t_S, which may vary depending on the context of interaction, S's patience, etc. R also has some response time t_R, which may be based on the context of interaction, but also the current state of R. In the best-case scenario, R's response might arrive within the timeframe expected by S (that is, $t_S \geq t_R$), thus confirming all went well. Otherwise ($t_S < t_R$), barring an actual technological failure, there is only silence in the interim ($t_R - t_S$, where $t_S < t_R \leq \infty$). The meaning of this silence is left up to the interpretation of S.

There are a variety of possible interpretations of such silence.

- R overlooked the message.
- R is too busy to respond to the message.
- R is in the process of responding to the message, but is otherwise delayed.
- R does not care about the message.
- R does not care about S, and is ignoring communication from S.
- The message was caught by R's spam filter in a false-positive identification.

Misinterpretation of silence can have undesirable effects on both interpersonal relations and collaboration goals. A study of geographically dispersed teams [2] found that recipients' silence was often misinterpreted. Silence was often taken as consent, when the actual reason was disagreement or inattention on part of the recipient. Further, silence was also often attributed to intentional disregard or non-participation, when the true problem involved technical or informational failures. If persistent or lengthy enough, the risk of perceiving silence as a form of "cyberostracism" [16] may also cause further anxiety for the sender and friction between the parties.

Further, S's most reasonable course of action to communication silence differs by scenario. Simple oversight can usually be remedied by a follow-up message or a phone call to R, providing a gentle reminder. However, if R is busy, or is currently working on a reply, a follow-up will more

likely cause annoyance and social friction than achieve any useful purpose. It would be best for S to simply wait a bit longer. Unfortunately, there are few contextual cues (with the exception of prior knowledge of the habits of R) that would assist in this judgment, and especially so if S and R have had few prior contact. In other words, it is difficult for S to estimate t_R (and reconcile t_S with t_R, if possible), without knowing enough about R and his strategies in determining t_R.

For R, on the other hand, email tends to become an exercise in *triage* [10]. As the number of incoming messages increases, the recipient must apply some personal information organization and retrieval strategies to handle these messages. These strategies involve applying priority to the incoming message queue, prioritizing some messages for response, leaving some for later, and simply ignoring others. The strategy being employed affects t_R for each message.

Social image plays a significant role in determining t_R for a message. The *responsiveness image* is a projected signal on part of an email recipient, designed to leave a particular impression about their email handling with senders [12]. R may actively try to respond to certain messages quickly, to leave impressions of efficiency and availability with those senders. In other cases, R may de-prioritize certain messages or certain senders, to cultivate a sense of unreachability or importance. In fact, R may have a single responsiveness image, or different responsiveness images with different senders.

The problem here is similar, as R has no reasonable means to communicate t_R to S, until the two have built up some history of interactions and some expectations (with which S may use to estimate t_R). Further, there does not exist a set of analytic tools that R may use to see the reflection of his responsiveness image(s). The history of interactions between R and each sender is obscured in most email clients' archives. The maintenance of the responsiveness image rests directly with R, with few introspective or reflective analytics tools to assist him.

1.2 Related work

Email in general has been a well-studied topic in computer-mediated communication. As email has become a multipurpose tool, prior work in this area [4] have examined the system in multiple theoretical and empirical contexts, including as *filing cabinets* (digital personal information repositories), *production lines* (collaborative workflows), and *genre of communication* (asynchronous messaging).

Of particular interest are studies in email handling and response strategies, which take on aspects of all three perspectives. In a qualitative study of email and email-filing practices, Whittaker and Sidner [15] noted that email has evolved beyond a simple asynchronous messaging interface, having become also a task management tool, event scheduling and planning mechanism, and information archival system. They also found that while users were highly positive about email, many were frustrated with the lack of timeliness of email responses and concerned about the impact on efficiency and productivity.

Ducheneaut and Bellotti [3, 1] conducted extensive studies of strategies in email management. Using the ”email as habitat” metaphor, they find that users tend to subsume many of their personal information tasks into their email clients. Further, users adopt varying and personalized approaches to dealing with increasing amount of email, such as triage, deliberate organizational schemes, usage of Inboxes and special folders as TODO lists, and other idiosyncratic methods for dealing with incoming email.

There also has been work in identifying *patterns* of email usage and behavior from a collaboration context and building email management tools using these patterns. Fisher and Dourish constructed an analysis and visualization toolkit called *Soylent*, which extracted and displayed social networks and temporal patterns from individual users' email archives, helping to uncover users' "social workscape" of collaborative interactions [7]. Likewise, Neustaedter et al.'s *SNARF* system computes social and temporal metrics and ranks messages by these metrics, placing the most socially important messages at the top of the inbox [9].

Viegas et al. offered *Themail*, a visualization system to characterize egocentric social relations, via temporal and contextual content drawn from email interaction histories [14]. In a similar vein, Viegas et al.'s *PostHistory* offered visualizations of social network and temporal patterns of interaction for each contact found in an email archive [13].

Tyler and Tang [12] performed a qualitative study on temporal rhythms in email usage, responsiveness, and implications for collaboration. Using interviews, they confirmed the intuition that users draw upon contextual cues to explain email responsiveness, from the very explicit (e.g. vacation/out-of-office auto-replies) to the implicit (e.g. prior personal or email interactions). Conversely, they found that some users consciously attempt to project a responsiveness image for various social contexts. Faster response times (even if only to signal receipt, rather than address the sender's main points) help to build an image of availability and efficiency, while deliberate delays in response times created an aura of importance and inaccessibility. In either case, when faced with a new email correspondent, people develop a degree of anxiety about whether a message would be read or replied to in reasonable time; this was true even if the two correspondents had prior personal contact, but not email contact.

Some researchers in this area express interest in designing email systems that incorporate responsiveness profiles, or some historical metrics about a particular user's email behavior [12, 8]. These profiles can serve as cues for reducing the opacity of the current send-and-wait email process. By creating a more socially translucent system [6], it is hoped that email interactions would entail less guesswork and anxiety on part of the correspondents. At the same time, responsiveness analytics would help users assess their own idiosyncratic email-management methods, showing their email behaviors and responsiveness images as others perceive them.

I propose one particular design of such a system in the following sections.

2 Solution Design

MailHistory is envisioned to be a four-component, extensible analytics system for extracting and visualizing temporal trends in email. It allows senders to check the typical responsiveness behavior of recipients, and all users to examine their own empirical performance in context of the responsiveness image that they wish to convey.

2.1 Architecture

The system consists of two analytic components and two visualization components. Figure 1 presents the overall system architecture.

- MailAnalytics: a client-side analytics tool to process and extract temporal response patterns and other data from static email archives.
- MailDynamics: a client-side analytics tool that integrates with email client software to record ongoing email exchanges and temporal patterns.
- MailViews: a client-side visualization dashboard for MailHistory data
- MailHistory: a centralized visualization server for MailHistory data

2.2 MailAnalytics - static data analysis

The static data analysis component - tentatively called MailAnalytics - forms a primary piece of the MailHistory analytics architecture. Its purpose is to extract historical, temporal patterns of email correspondence for a given user, via a snapshot of his email archive. It presumes that the user has archived his mail consistently and comprehensively, and has such a historical archive of emails to give to the analytic component.[1]

[1] In an age of cheap disk space and massive email storage from online vendors (such as Google Gmail), I believe this to be a fair presumption.

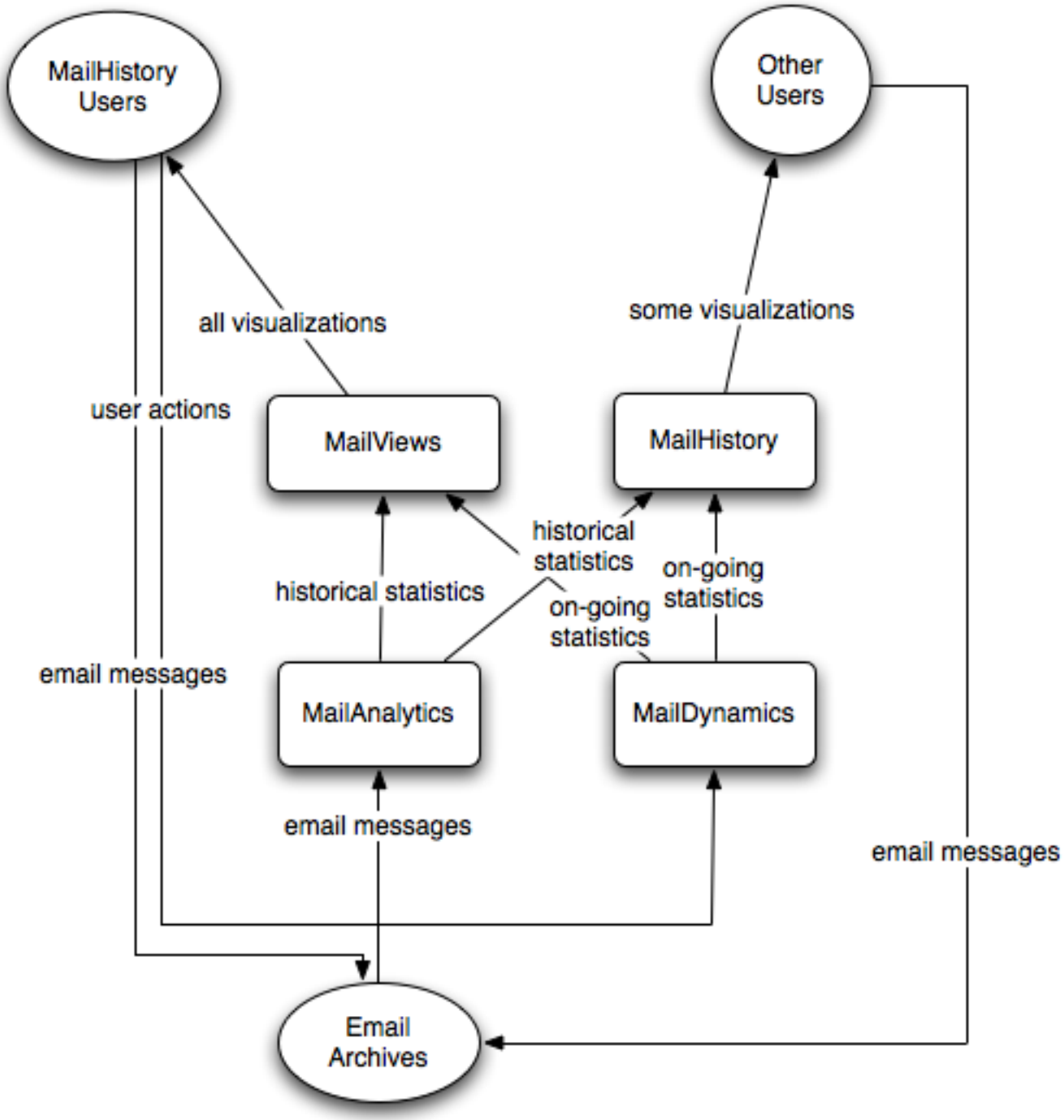

Figure 1: MailHistory consists of two client-side tools and two reporting interfaces.

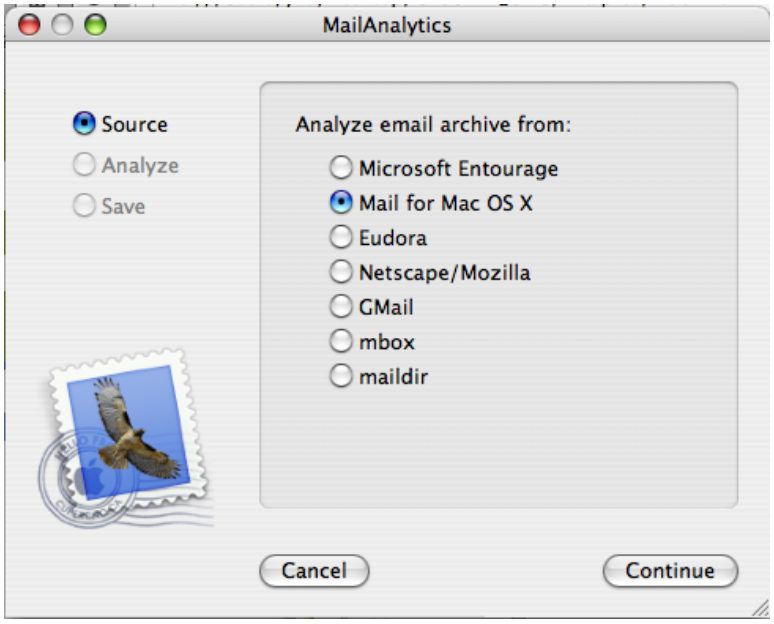

Figure 2: MailAnalytics tool using a Mac OS X assistant interface, with many archive parsing options. Clicking "Continue" will bring up the next section, dealing with the specific analytic tools to run.

2.2.1 Source

By default, MailAnalytics accepts `mbox` [2]files, which can be reliably created from most email clients. Given login credentials and the address to an IMAP server, MailAnalytics can also attempt to download each message residing on that server, using the IMAP server as an archive. Figure 2 shows the MailAnalytics tool, as it appears on Mac OS X and using the system's native "Assistant" GUI metaphor.

2.2.2 Analysis

Once the source has been established, MailAnalytics proceeds to make two analytic passes on an email archive. First, the parser is run to obtain metadata for each message, and then MailAnalyticPlugins are run to obtain the actual statistics desired.

[2]`mbox` is a set of plaintext file formats that represent a corpus of emails. They contain their original Internet message headers and can be parsed reliably.

On the first pass, the MailAnalytics mailbox parser obtains each email message from the archive. The parser extracts temporal and message metadata from the message headers, and stores them in a temporary SQLite database.[3].

On the second pass, each registered analytic plug-in is called with a handle to the SQLite database. Each plug-in extracts whatever patterns or statistics that are of interest, and write them to an XML output file. MailAnalytics itself provides some basic plug-ins to fulfill its design goals:

- ResponseTime: This plug-in identifies message-response pairs (via the use of `In-Reply-To` and `References` headers). It computes the response time as the difference between the Sent time of the original message and the Sent time of the response message. It also records the sender and sending time for each message-response pair, so response times can be aggregated by sender and timeframe.
- TrafficVolume: This plug-in records the size of each message and the count of messages received by hour (and thus other timeframes, such as by day, by week, by month, etc. can be inferred from this data.) It also associates each message thus processed with its original sender, so that the data can be aggregated by sender if necessary.
- CorresponderCount: This plug-in extracts a (sender, timestamp) pair for each message received. A visualization widget can then simply pick an aggregation size (per day, per week, etc.) and select a beginning/end time to show the number of people corresponded with in a given timeframe.
- ContentWords: For each inbound and outbound message, message text is tokenized, and a frequency table of word-tokens is built. This table is associated with sender and timestamp for sorting or aggregation purposes.
- EmotionScore: For each message, an emotion score is assigned. In this first version, the score is generated from comparing tokens to a pre-evaluated set of positive or negative words. Rudimentary negation detection is provided, by detecting negation words such as “not”, “n’t”, etc.

Given this design, there may be some redundant work involved in the data analysis process. However, the intent of this design is to be extensible - that is, each analysis plug-in can be run alone, and other analysis plug-ins can be written and inserted into the architecture. See Figure 3.

[3]SQLite is chosen as it can be wholly embedded into the main MailAnalytics program, rather than requiring a separate server like MySQL

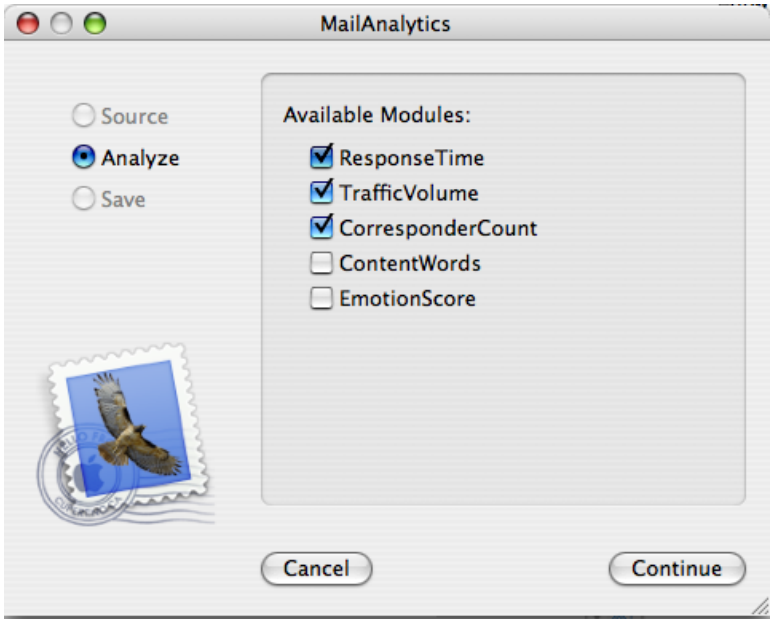

Figure 3: MailAnalytics tool with available modules.

2.2.3 Save

Once processing is complete, MailAnalytics will prompt the user to save the results of the analysis, for use with the client-side visualizer MailViews.

2.2.4 Interaction

In a typical interaction, a user installs the MailHistory package. At the end of the installation process, MailAnalytics is launched. He is prompted to select a data source. He selects an *mbox* as the email archive and hits Continue.

He then receives a listing of checkboxes, representing the analytic plug-ins about to run. Having no objections to the built-in plug-ins, he leaves all of them checked and hits Continue.

The application then presents a progress bar indicating the state of the analysis. Since the data is being read from a static file on the local drive, the processing completes relatively quickly. Ten minutes later, processing is complete, and the user is able to hit Continue again.

The application prompts the user to save the results (as a SQLite database) to a default directory.

Hitting Save will accept this choice.

2.3 MailDynamics - dynamic data capture

The dynamic monitoring component - MailDynamics - forms the second piece of the MailHistory architecture. Its function is to reside within a mail client and record statistics about each incoming and outgoing message. MailDynamics has no visible user interface, except for a preference panel indicating the specific monitoring plug-ins (analogous to the MailAnalytics analysis plug-ins) installed.

The primary purpose of MailDynamics is to ensure that a user's responsiveness profile is continually freshened with the latest behavior data. It is also a potential fallback, in case that the user has maintained no archive of his email history. Via MailDynamics, it is at least possible to build up a profile of usage data, and continue to do so without forcing the user to change his archival strategy.

MailDynamics collects data in a real-time, message-by-message fashion. Results are appended to an existing SQLite database (if available from a MailAnalytics run) or a new database file (if not).

Since there are many mail clients, each of which differ in extensibility and flexibility, MailHistory cannot possibly support all of them. The use of MailDynamics is thus optional. Periodic snapshot updates via MailAnalytics is sufficient for MailHistory usage.

In a typical scenario, MailDynamics monitors the user as he performs his normal email tasks. When he receives a message, MailDynamics silently processes the message in the background and invokes each of its plug-ins to handle the message. When the user responds to a message, this action also triggers another round of MailDynamics processing. Each plug-in will again be asked to handle the message, and whatever metadata it is interested in can be recorded. One plug-in, for example, might find the timestamp for the original message and record the response-time on this correspondence.

The COM APIs provided by Microsoft Outlook and the extension architecture for Mozilla Thunderbird are sufficiently rich to implement MailHistory. It may be possible to support Apple Mail and Microsoft Entourage using AppleScript interfaces on OS X, but this has not been investigated.

2.4 MailViews - mail-client integrated visualization

The output of the analytic infrastructure is visualized via MailViews, a personal visualization dashboard consisting of various widgets. The dashboard loads the result files from the analytic tools and makes these available as data feeds.

Each visualization widget takes a data feed it wishes to use, performs some visualization routines, and draws the output to its widget space. Start and end times can be changed to reveal more or less of the timeline. Users may develop, install, or new widgets as they see fit - as long as data is available, the widget will be able to run its visualization. See Figure 4.

One widget, for example, is the mean-response-time widget, which represents a simple line graph of historical response times. Its timeline can be changed by specifying a new date range, and its interval buckets can be changed by clicking on Day, Week, or Month. Other widgets might include a data volume visualizer, the top corresponders list, etc.

MailViews communicates regularly with the public MailHistory server, (described in the following section). First, it periodically supplies the MailHistory server with data. Privacy controls on the back of each widget indicate how much of the visualization is available to public interface visitors. Widgets can be public, in which case full data available in the feed is shared with the public server. They can be friends-only, in which only a whitelist of users are allowed to access results. Finally, they can be set to private, in which case the data will appear in the personal MailViews dashboard but nowhere else.

Second, using MailViews, the user may search for other correspondents whose email patterns he is interested in seeing. The same widgets that can show his personal responsiveness profile may

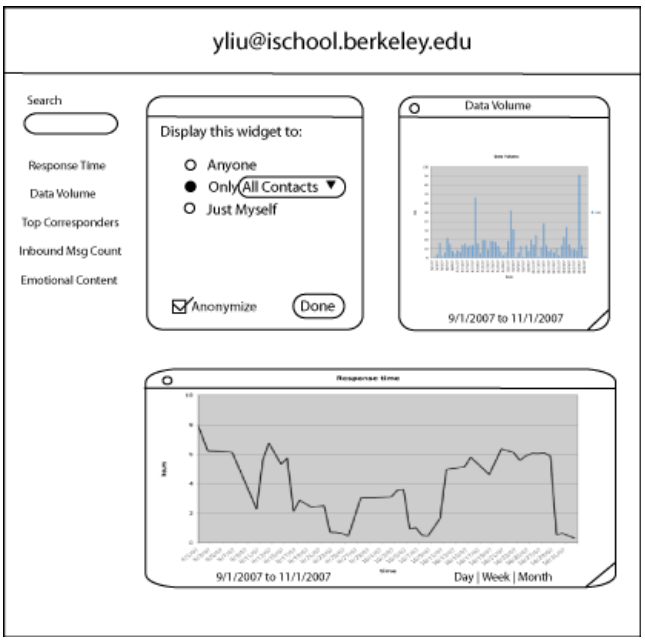

Figure 4: Sketch of the main visualization dashboard. Each visualization widget takes a data source and draws a visualization to its widget space. Start and end times can be changed to reveal more or less of the timeline. Privacy settings on the back of each widget controls if the visualization is available to public interface visitors.

also read MailHistory data feeds, and run visualizations for these other users. Behind the scenes, MailViews simply contacts the MailHistory server and asks for data feeds related to a given user. Based on privacy settings, some amount of data is supplied, and visualizations are drawn by each widget.

2.5 MailHistory - public visualization

MailHistory is a public server that stores email analytic data for sharing. MailViews will periodically update this server with the data that the user has specified to be shared. MailHistory then publishes the data feeds as XML.

This enables some crucial interactions. First, as described in the previous section, a user may take the MailViews dashboard and run visualizations for other users. Second, user-authorized third-party application can read a MailHistory data feed and presents some alternative visualization

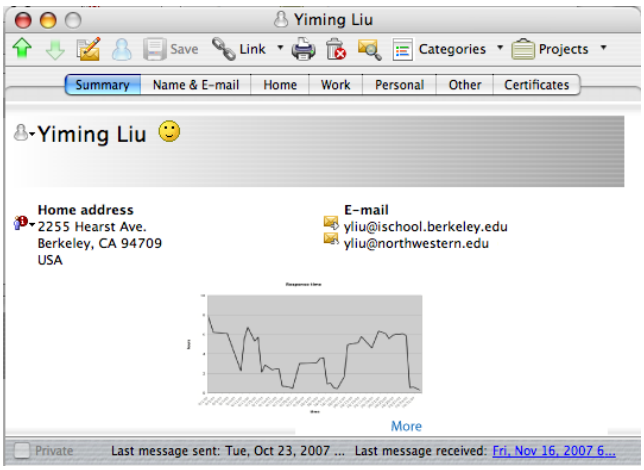

Figure 5: Mockup of the in-context visualization. In the Address Book of the mail clients, each entry can be annotated with current temporal pattern information for that user. The widget displayed can be swapped out for other widgets as appropriate. Clicking on More invokes the main visualization dashboard, targeted on this individual.

- perhaps as a badge for one's homepage or a Flash widget for one's blog.

Email senders would query the MailHistory server for visualizations and reports of their intended recipients' correspondence history. The resolution of MailHistory information should be performed in a similar fashion to how LDAP lookups are performed from within these clients now. [4] Whenever an Address Book entry is viewed, for example, the client would poll the MailHistory server for visualizations. If these visualizations are available, they will be displayed in the Address Book card screen.

3 Design rationales

The MailHistory system is intended to present a set of responsiveness and workload metrics that help users evaluate themselves in context of their desired responsiveness image. Further, MailHistory is designed to introduce social translucence into previously opaque email processes.

MailHistory creates a *visible* environment of interaction, by revealing previously private and

[4]The Lightweight Directory Access Protocol is an email directory service supported by most email clients.

dificult-to-extract metrics for public inquiry. It creates *awareness* through the fact that all MailHistory users are aware of others' scrutiny, at least at a high level, of their general email habits. They themselves can also apply an investigative lens into their own email behavior, by viewing these analytic results. Thus, this also instills *accountability*. A sender has some means to determine the typical patterns of correspondence for each user, and can set a reasonable time for follow-up or a switch to another, more synchronous communication channel for urgent concerns. The recipient, on the other hand, is able to analyze his own patterns of behavior and assess how well his empirical performance matches up to his desired responsiveness image.

This design is informed by the canonical principles for the creation of socially translucent systems and visualizations [5]. For example, MailHistory portrays actions, not interpretations. The user may interpret meaning into the temporal visualizations, using it as a larger pool of contextual information - which would not have otherwise been available. It also supports micro/macro readings [11] in its time-based activity graphs, allowing users with some prior context to interpret meaning into the pattern.

On the other hand, MailHistory does allow customization of visualizations. This is justified by the intuition that email, while a collaboration tool, is not necessarily a public, *group* collaboration tool as represented by Babble. While MailHistory is intended to inject some measure of translucence into the utter opacity of email and responsiveness, this development must be coupled with consideration of privacy and security implications inherent in email communication. Revealing top corresponders, for example, is unnecessary for the public visualization, but may be very useful for the user's private view. On the other hand, the user may *choose* to reveal this information, perhaps as a means of signaling that these are his "top friends".

3.1 Use as research prototype

MailHistory may be turned into a research prototype to collect temporal rhythms and responsiveness profile information. Anonymized aggregation of this data may grant some insight into

how users use these new cues in comparison with the old, opaque way of email communication.

Some specific questions of interest:

- Is there less anxiety when previous strangers attempt collaboration over email? One might imagine designing an experiment with a control group of traditional mail users and a group of MailHistory-enhanced users. The concept of "anxiety" may be difficult to operationalize, however.

- Would a translucent mail system reduce friction in team collaboration contexts? The composition of the team is a factor here. I would surmise that a team whose members know each other very well would not see much difference from a MailHistory system. A newly formed team, on the other hand, might see some improvement in communication. Perhaps a qualitative study is appropriate here.

- How useful is MailHistory as a purely introspective tool? SNARF, Themail, and other visualizations have been reported as helpful for introspection into email behavior. Do temporal visualizations function the same way?

- How useful is MailHistory as a social translucence tool? Would users change their email behavior, knowing that others can see some of their habits? If so, in what ways?

- Do people selectively reveal more of themselves to friends than the public? What kind of data would they reveal only to friends, but not to the public?

4 Remaining challenges

In an age of spam, any service that would to reveal information about individuals behind email addresses is likely to face abuse. This must somehow be accounted for in the system design.

MailHistory makes some assumptions about how email is likely to be stored and used by a user. However, email technology has become increasingly diverse. Many users may entirely depend on

the web-based version of Gmail, for example. In many instances, it may not be trivial for either analytic component of MailHistory to collect sufficient data. In the case of the exclusive web Gmail user, running MailDynamics would tend to be difficult, unless a special extension is made to extend the Gmail interface. Furthermore, while Gmail allows the downloading of its data (so that MailAnalytics might process them), the average user is unlikely to be able to do so[5]. A deployable version should integrate with the POP or IMAP interfaces for common services such as Gmail, and provide automated means to extract email corpora for non-technical users.

Privacy implications are always present in a system that purports to handle personal emails. The system accounts for this by handling all data analysis client-side, and only uploading the data that the user specified. It also has per-widget privacy controls, so that no unwanted surprises should be published. However, users may still not want to share any of their analytic data with the public service, thus removing one of the two main uses of MailHistory.[6] Some incentive structure may be required to encourage participation.

5 Project status

The analytic component prototype is complete, though the schema representing these results has yet to be finalized. The other components have not been built. Thus, I am submitting only screenshots, design sketches, and textual descriptions in lieu of a full prototype.

6 Conclusions

In this paper, I have presented MailHistory, an email analytics service that provides a translucent view into email response times and other behaviors. I surveyed current work in analysis of

[5]Such a procedure involves activating POP access, setting up a client to download the entirety of the Gmail archives for that account, and possibly exporting it to a compatible mbox format

[6]In fact, this might even turn into a Prisoner's Dilemma game, where one party would prefer that the other user shares his information, while he himself does not.

temporal patterns in email, and proposed an architecture for extracting, analyzing, and publishing these patterns. There are some interesting questions in this space that a MailHistory prototype may help answer, and this work is a first step toward that direction.

References

[1] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., and Grinter, R. E. Quality versus quantity: E-mail-centric task management and its relation with overload. *Human-Computer Interaction*, 20(1-2):89–138, 2005.

[2] Cramton, C. D. The mutual knowledge problem and its consequences for dispersed collaboration. *Organization Science*, 12(3):346–371, may 2001.

[3] Ducheneaut, N. and Bellotti, V. E-mail as habitat: an exploration of embedded personal information management. *interactions*, 8(5):30–38, 2001.

[4] Ducheneaut, N. and Watts, L. A. In search of coherence: A review of e-mail research. *Human-Computer Interaction*, 20:11–48, 2005.

[5] Erickson, T. Designing visualizations of social activity: six claims. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 846–847, New York, NY, USA, 2003. ACM Press.

[6] Erickson, T. and Kellogg, W. A. Social translucence: an approach to designing systems that support social processes. *ACM Trans. Comput.-Hum. Interact.*, 7(1):59–83, 2000.

[7] Fisher, D. and Dourish, P. Social and temporal structures in everyday collaboration. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 551–558, New York, NY, USA, 2004. ACM Press.

[8] Kalman, Y. M. and Rafaeli, S. Email chronemics: Unobtrusive profiling of response times. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4*, page 108.2, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Neustaedter, C., Brush, A., Smith, M., and Fisher, D. The Social Network and Relationship Finder: Social Sorting for Email Triage. 2005.

[10] Neustaedter, C., Brush, B. J. B., and Smith, M. A. Beyond "from" and "received": exploring the dynamics of email triage. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1977–1980, New York, NY, USA, 2005. ACM Press.

[11] Tufte, E. *Envisioning information*. Graphics Press Cheshire, Conn.(PO Box 430, Cheshire 06410), 1990.

[12] Tyler, J. R. and Tang, J. C. When can i expect an email response? a study of rhythms in email usage. In *ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, pages 239–258, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[13] Viégas, F. B., Boyd, D., Nguyen, D. H., Potter, J., and Donath, J. Digital artifacts for remembering and storytelling: Posthistory and social network fragments. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4*, page 40109.1, Washington, DC, USA, 2004. IEEE Computer Society.

[14] Viégas, F. B., Golder, S., and Donath, J. Visualizing email content: portraying relationships from conversational histories. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 979–988, New York, NY, USA, 2006. ACM Press.

[15] Whittaker, S. and Sidner, C. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, New York, NY, USA, 1996. ACM Press.

[16] Williams, K. D. Cyberostracism: effects of being ignored over the internet. *Journal of personality and social psychology*, 79(5):748–62, 2000.