Labeled Thread Arcs

E-Mail Visualization

Purpose Statement, Usability Analysis, Engineering Overview, Aesthetic Review, and Creation Process

Jason Chen

8 May 09

Purpose

Motivation

After perusing in-class readings and searching online, I could not find a modern graphical representation of collections of e-mails. The main purpose of this project is to visualize someone's inbox while taking user-defined categories into account. It is assumed that the e-mail account being analyzed is well-maintained (e.g., no spam in the primary inbox) and that the flow of incoming messages is somewhat steady. (The number of received messages over time does not necessarily have to be large; it just shouldn't fluctuate much.)

Inspiration

- IBM's *Thread Arcs* component of *ReMail* (2003)
  http://www.research.ibm.com/remail/visualizations.html

Figure 1: Each arc denotes a link between a message and its parent in the thread tree. Different colors represent different people. © IBM

- Martin Wattenberg's *The Shape of Song* (2001)

http://transition.turbulence.org/Works/song/

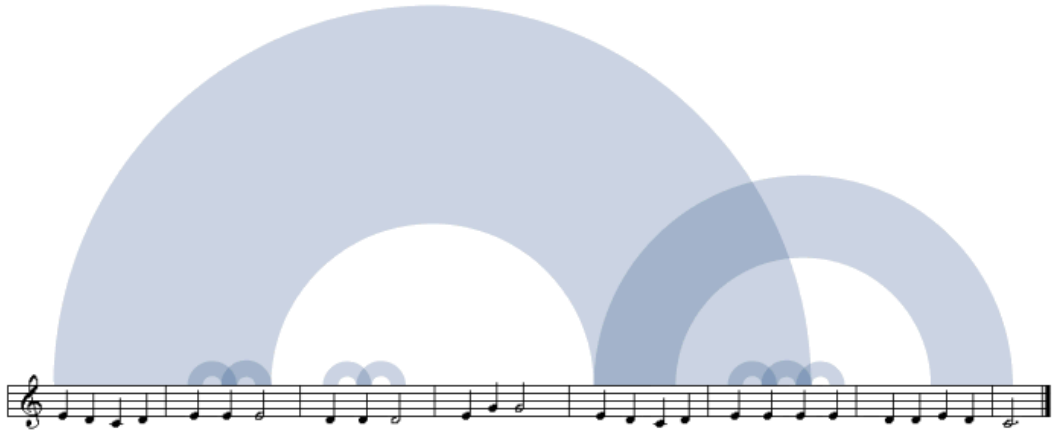

Figure 2: The picture was built from the first line of a very simple piece: *Mary Had a Little Lamb*. Each arch connects two identical passages. © Martin Wattenberg

Goal

Beyond simply crafting myself a tool to view my e-mailing habits on multiple scales, I hope to make this utility more widely available and/or popular than the sources of inspiration appear to be. The current implementation of *Labeled Thread Arcs* uses the Windows Presentation Foundation for the .NET Framework, so it can be made available as a web browser-based Silverlight application.

Usability

Legibility

Although there is not much to read, all default colors provide high contrast between foreground and background, especially in the case of text. Message nodes' colors are assigned based on their primary category's color which is assigned by the user outside of this application. Arcs and their "fillings" are colored opposite of the main canvas's background.

Understandability

Based on observations of other users, it's easy to understand that each semicircle represents a message in the inbox and that each arc shows association between the two linked messages. Intuition also guides them to the conclusion that long chains or trees of linked messages define a complete thread.

One of the major problems to tackle was sorting. I personally prefer it to be read most to least recent from left to right. For users who prefer to see the most recent messages on the right side, it is simple to switch between the two.

Efficiency

Visual

In terms of visual real estate, I feel that this visualization fills the proper amount of area and pads it with appropriate white space. Since the user decides whether to focus on individual threads/messages/threads or not, this will vary based on user interaction. Another factor is the number of categories a user has. In my case, the 7 labels I use result in a panel of reasonable height. A future improvement may be to enumerate the category in a scrollable list.

Technical

Messages are read from XML files generated from mailboxes retrieved through IMAP. Gmail keeps a copy of each message in every folder which represents a label applied to that message, so the downloaded data may contain many duplicates. Each (non-primary inbox) category a message belongs to produces a duplicate of that message in the XML file for that category.

All shapes are drawn as vectors so sufficient hardware is required for a smooth experience.

Patterns

There will typically be a few cases of noteworthy patterns (and combinations of them). They either are naturally highlighted or are displayed in ways that draw attention.

- Chains
  Threaded messages each have a depth relative to the first e-mail in the thread. As the thread gets deeper, arcs linking deeper messages become more opaque. This pattern is typical of conversations which progress "linearly".

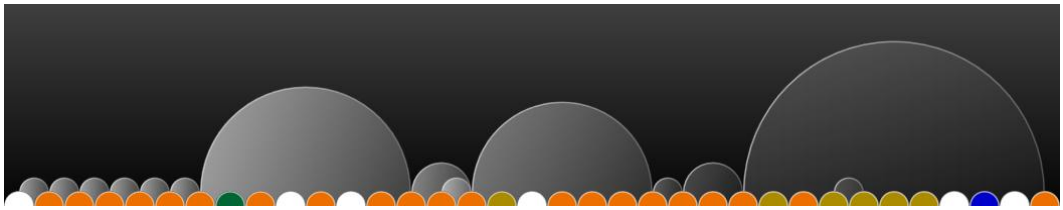

Figure 3: In a chain, arc opacity increases as depth does.

- Trees
  (Technically, any node is the root of a tree, but a tree in this context has at least two levels and the root node has more than one child.) Initial e-mails which spawn many direct responses are classified as parents in a tree. The many "branches" sprouting from the root node create a thicker line which brings attention to the bunch. This pattern is commonly seen on mailing lists when someone asks a question to which many people have answers.

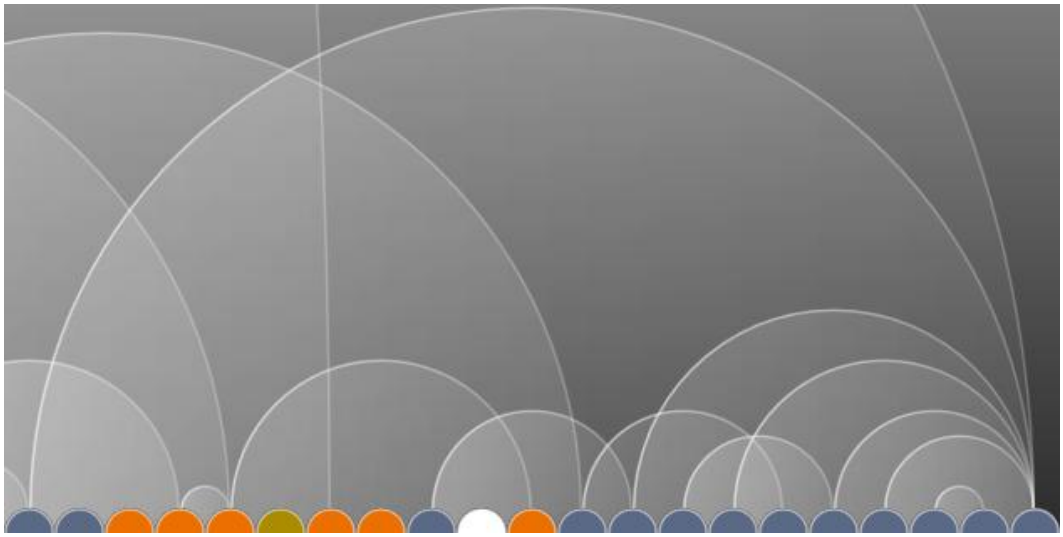

Figure 4: In a tree, a single message has received many responses.

- Umbrellas & Stacks
  Slow responses cause umbrellas to be cast over all of the other messages mailed during the delay. As a huge structure in a landscape of smaller arcs, it is naturally outstanding. From my personal data I couldn't find a clearly defined criterion for these types of e-mails, but it was usually an unrelated response in some informal communication.

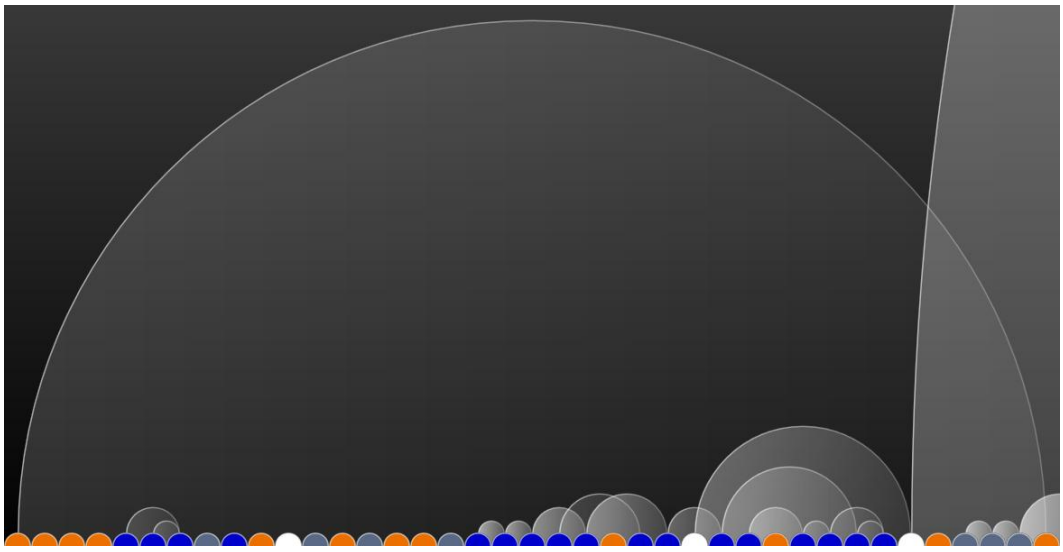

Figure 5: An umbrella covers many other messages. This is actually a fairly small umbrella in my data.

Exploration

Although the visualization is not intended to be a full-fledged e-mail reader, hovering over a message node provides a preview of the message in addition to other essential data: subject, author, date and time sent, and a list of labels (since the color only represents the first of the labels).

Developing the application into a complete e-mail suite definitely remains a possibility.

Engineering

Criteria & Constraints

- Reads e-mail and category data from XML
- Displays e-mail data
  - Indicates category
  - Shows threading
  - Organized by time
- Is intuitive to use
- Provides enjoyable discovery experience

Restrictions

The original scope of the project covered only Gmail. However, without any official API from Google, IMAP became the only viable method of retrieving messages and labels. With this change also came the expansion of mail service coverage. Theoretically, any e-mail provider which allows IMAP access will be able to be read into this application. Service-specific tweaks may be needed for each individual one. Currently this is only known to work well with Gmail.

As previously mentioned, Gmail keeps an extra copy of every message for each label it has. Although text doesn't take up much space, one possible solution is to remove the message contents of all multi-labeled e-mails in the XML files of the extra copies. If mail/label management capabilities are added in the future, however, that erased data must be carefully re-inserted or copied from the correct source.

Possible Features

- Indicate messages sent by logged in account holder
- Highlight single threads
- Trace thread arcs (particularly for extremely long umbrellas)
- Support e-mail services besides Gmail
- Data fetched directly through IMAP
  - Store data in eml and/or mbox format
  - Allow login to different supported services
- Drag-and-drop message management
  - Trash
  - Sent
  - Labels
- Act as webmail service

Aesthetics

Interactivity

A panel containing a listing of check boxes and labels (for toggling the highlight animation) is generated based on the XML files each named for the mailbox folder (label) they contain. Once a label is checked, the appropriate nodes are animated. This is currently the slow part of the program and is a strong candidate for re-implementation.

Hovering over a message node provides information (subject, author, date and time sent, a list of labels if necessary, and a message preview) in a system tooltip. This may change to a custom pop-up panel type.

Zooming is possible via the standard "`Ctrl+-`/`Ctrl++`" key combinations. Buttons and a slider will be added in a planned toolbar at the bottom.

Color Usage

The default color scheme is a black background with white strokes on message nodes and white arcs. The nodes themselves are colored based on the label of the message they represent. These colors are defined by the user in Gmail. The panel with check boxes is colored by the label colors. Text on these labels is either black or white depending on the label color.

Tooltips that pop up currently use the standard form controls of the host system.

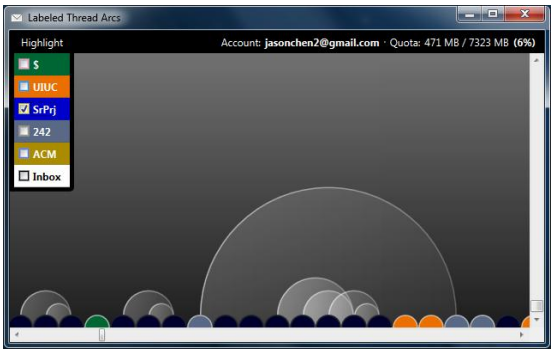

Figure 6: The list at the top left corner has a check box for each label to toggle whether e-mails tagged with that label should be highlighted by a fade loop animation. (In the screenshot, "SrPrj" messages are faded away and will fade back in shortly.)

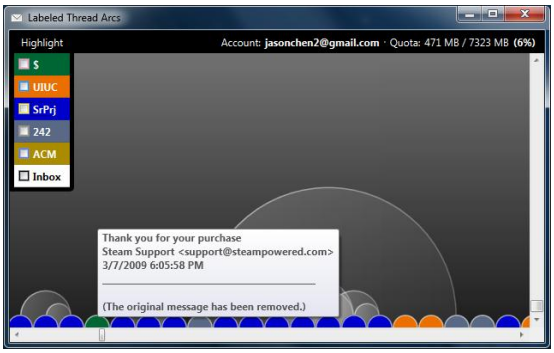

Figure 7: Hovering over a message pops up a tooltip containing pertinent data. (In the screenshot, the mouse is over the green node.)

Creation

Concept & Sketch

The original mock-up of the visualization used green arcs to indicate messages sent by the user. The nodes had thinner strokes so it would have been difficult to see white on white, as seen in the sketch itself!

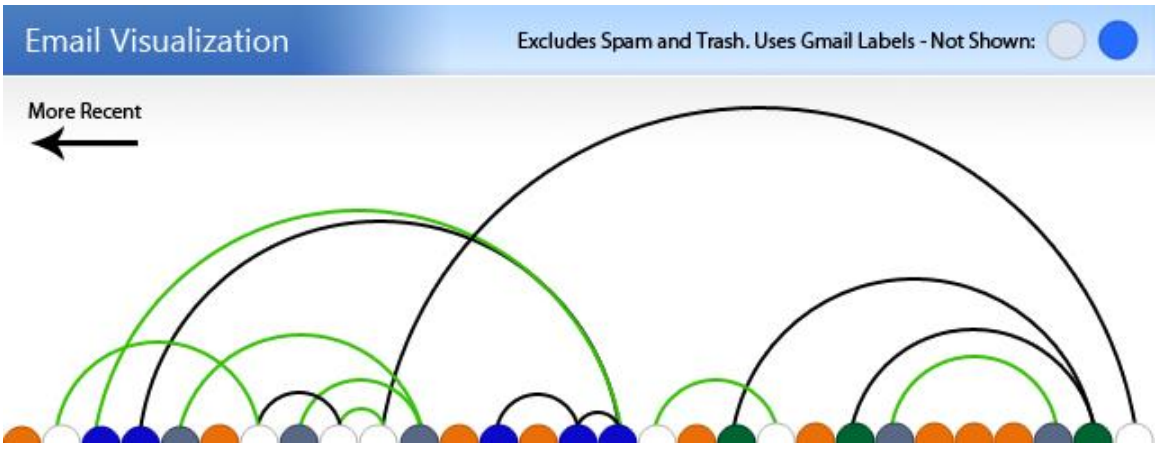

Figure 8: This image shows the basic concepts behind the visualization.

Prototype

The first implementation of this tool clearly shows the intended direction at the time it was made. Perhaps too much energy was spent on the customization and "potential" features of the project that not enough consideration was given to the actual visualization core of the application. The biggest flaw of the visualization was the possibility of the outer ring, which indicated sender colors (initially picked at random but re-assignable), getting distracting.

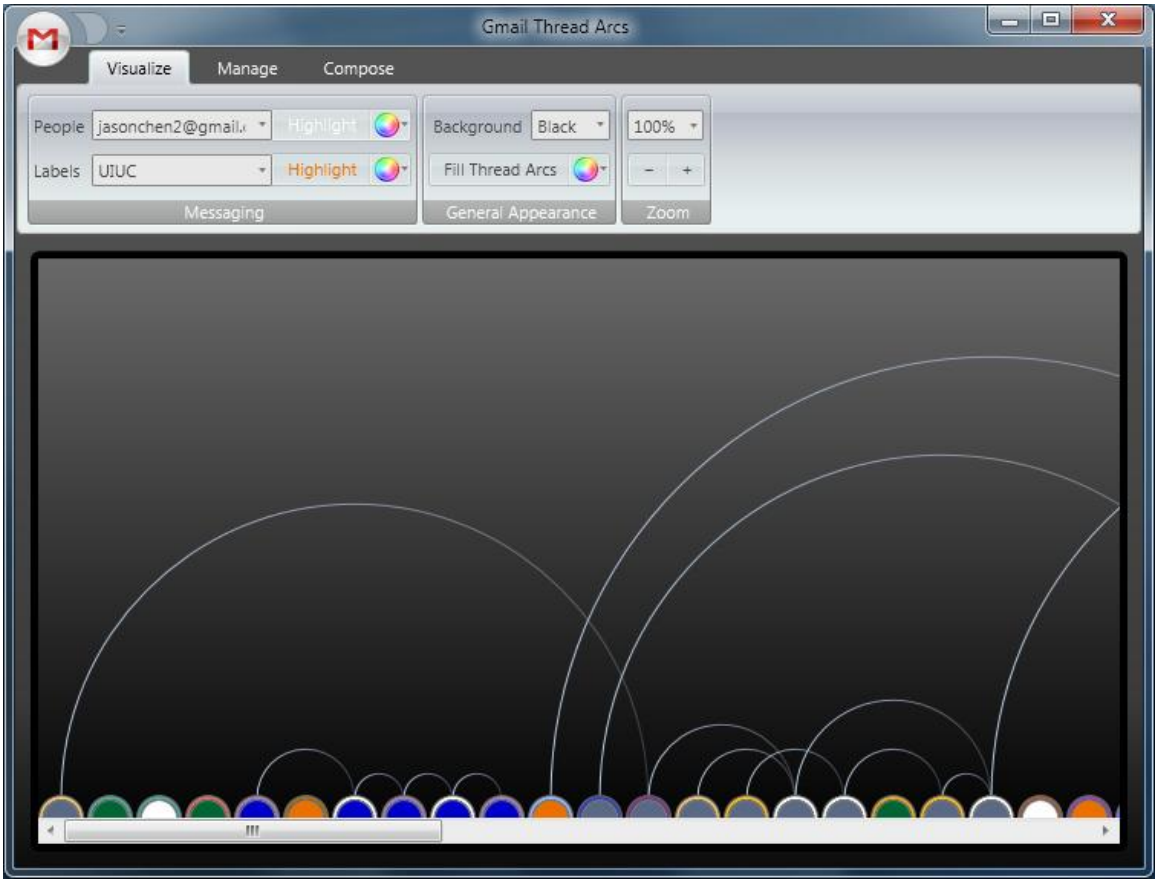

Figure 9: Many options were available in the "wrapper" for the visualization, and many related e-mail features were planned.

Final Visualization

The main principle behind the revision was keeping things simple and going back to basics. The result is, I hope, a more intuitively understandable visualization which provides pertinent information wherever appropriate and whenever demanded (e.g., on hover).