RC22850 (W0307-148) July 22, 2003
Other

IBM Research Report

THREAD ARCS: An Email Thread Visualization

Bernard J. Kerr
IBM Research
One Rogers Street
Cambridge, MA 02142

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

THREAD ARCS: An Email Thread Visualization

Bernard Kerr
IBM Research
One Rogers Street
Cambridge, MA 02142
+ 1 617 693 4373
bernard_kerr@us.ibm.com

ABSTRACT

This paper describes *Thread Arcs*, a novel interactive visualization technique designed to help people use threads found in email. Thread Arcs combine the chronology of messages with the branching tree structure of a conversational thread in a *mixed-model* visualization [10] that is stable and compact. By quickly scanning and interacting with Thread Arcs, people can see various attributes of conversations and find relevant messages in them easily. We tested this technique against other visualization techniques with users own email in a functional prototype of *ReMail,* our research group's experimental email client. Thread Arcs proved an excellent match for the types of threads found in users' email and the qualities users wanted in small-scale thread visualizations.

KEYWORDS

Thread, email, electronic mail, information visualization, tree structures, discussions, conversations.

1. INTRODUCTION

The importance of conversation threads in email and tools for inspecting them has been well documented [6, 10, 12]. The main advantages are seen as giving users a greater context of the messages they are reading, reminding users that a conversation is in progress, recording the state of a discussion, automatically collating related messages, reducing messages displayed in their inbox, and allowing users to perform actions such as reading or deleting on a group of messages [4, 6, 10, 11, 12].

Threads are defined as a collection of individual messages related to each other by the reply function in email. Commonly, the first message in a thread is called the *root*, any message that is being replied to is called the *parent* of that message, and any replies to a message are called *children* of that message. The *generational depth* of a message is defined as the number of reply-to relationships away it is from the root message. For example, all messages that are replies to the root message have a generational depth of 1. The branching reply-to relationships of threads have explicit meaning that is significantly different from an *ad hoc* or automatically-collected set of messages.

Email threads differ from public discussion threads, such as those found in Usenet, in a number of ways. Public discussion threads are often large [8]. In contrast, email threads are relatively small. Fisher [4] found that threads of greater than one message account for 35% of average users' mailboxes and that most threads, 87%, had 4 messages or less.

Public discussions also have a formal reply mechanism, while email users often use the last messages they received from a correspondent as a convenient way to start a new message. By using existing messages and the reply function as a makeshift address book they are, in effect, breaking the formal use of the reply function. This means that chronology is an important attribute to consider for threads found in email.

Chronology is also important because email is often time-sensitive, and it is the way most users see messages arrive in their inboxes. Email threads are often read backwards starting with the most recent message, because in email the last message sent often determines the thread's "conversational status" [12] by summing up the current state of the conversation or by containing questions or tasks that are still outstanding.

Large scale visualization tools for public discussion threads, such as Netscan's "visual dashboards" [8], Loom [3] and Conversation Map [7], consider a different set of qualities when presenting thread information than that needed for email. They contain social structures and conversations that are very different from egocentric nature of email. For example, email users probably know all of the people involved in an email thread personally, while Usenet is more public and anonymous.

Email threads on the other hand have a different set of qualities to consider. We believe small-scale, compact visualizations embedded into personal email clients could enhance the user experience of email.

In email one of the challenges in displaying these conversational threads is that they have two conflicting properties: the arrival sequence of messages and their reply-to relationship [10]. We describe a visualization technique that can effectively communicate both of these properties at once, and explain how we tested this design in an email prototype, on real threads found in users' own email.

2. KEY QUALITIES

Here is a list of the seven qualities we considered key to visualizing threads in an email client and why they are important.

1. **Chronology**: Show the arrival sequence of messages that created this thread. Which message came first? Which is the most recent message? The importance of this quality is explained in the introduction.
2. **Relationships**: Make all of the "reply to" relationships between messages visible at a glance [10]. What are the direct relationships to other messages in the thread? What is this message a response to? Which chain of messages led to this one? Which messages subsequently

responded to this one? This gives the context for each message in the thread.

3. **Stability**: As a thread grows, have each message appear in the same location. This allows one to return to the thread in the future and find the same message or to see easily if any new messages have been added.
4. **Compactness**: Be compact, since the visualization will be competing with other space required for email functionality.
5. **Attribute Highlighting:** One is able to highlight specific message attributes in a thread, including all messages sent by a particular person, unread messages, or all messages sent on a particular day. This helps one find particular messages or assess the state of a thread.
6. **Scalability**: Work for small threads as well as larger threads found in email. Does the clarity of the visualization degrade gracefully as more messages arrive? Can it still be interpreted when it is large and complex? Since the vast majority of email threads are typically between 2 and 20 messages [4], a visualization does not have to scale to hundreds or thousands of messages.
7. **Sense/Scanability**: From scanning the visualization give a sense of the type of conversation present in a thread. For example, is this thread a back-and-forth reply chain between two people, or is it a request for information and responses from a group?

The Thread Arc visualization was designed with these key qualities in mind.

3. VISUALIZATION

Thread Arcs have a linear layout of message nodes connected by relationship arcs. In Thread Arcs each circular node represents a message in the thread. Because the *chronology* of the thread is so important we encoded this by position. For example, for the six-message thread in Figure 1, each node is equally spaced horizontally in the order of its arrival, with the first message on the left. This layout also makes for a *compact* visualization that is *stable*.

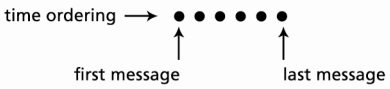

Figure 1. Chronology of message nodes in a line of 6 messages.

Figure 2 adds the *relationship* between messages. Here we draw arcs connecting each message node to its parent in the thread.

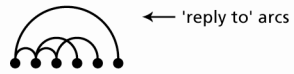

Figure 2. Relationships are shown with 'reply to' arcs connecting nodes above the line.

The density of lines and the intersection of arcs make this image hard to read. To alleviate this confusion we draw some of the arcs below the line, as shown in Figure 3, below.

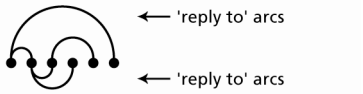

Figure 3. Relationships are shown with 'reply to' arcs connecting nodes above and below the line.

Figure 4 shows the advantages of this technique for a variety of threads with 5 messages.

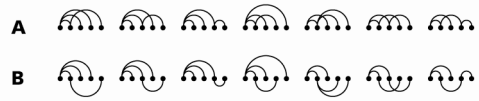

Figure 4. The relationship between messages are clearer when arcs are draw above and below nodes (B).

From this visualization one can see thread qualities such as the size of the thread (number of nodes, which also corresponds to the length of the visualization) and number of responses per message (the number of arcs leaving a message node). Threads that have messages which receive two or more replies are described as *bushy* while threads that have messages that get only one reply per message are called *narrow*. This helps one visually *scan* and get a *sense* of threads that have similar structures to threads one has seen before.

Figure 5. Bushy threads have messages which recieve two or more replies. Narrow threads recieve only one reply per message.

The width of a Thread Arc is a linear function of the size of the thread it portrays. We can make a more *compact visualization* if we can constrain the height of the arcs so they are flattened out when they are over a certain height, as shown in Figure 6. This means that the visualization will only grow horizontally, and therefore it is easier to fit inside the space constraints of an email client.

Figure 6. Constraining the maximum height of the arcs makes the visualization compact.

Figure 7 shows the effect of this technique on a larger thread containing 16 messages.

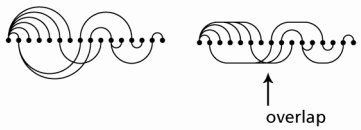

Figure 7. Unconstrained and constrained arc heights for a thread of 16 messages.

This technique creates some ambiguity when arcs overlap. This problem is alleviated by the *selection highlighting* that is

described under "Interaction" later in this paper, along with the other *attribute highlighting* schemes.

The general technique to make Thread Arcs can be summarized by the pseudo code shown in Figure 8.

```
To make a Thread Arc

sort all messages chronologically
find the generation depth of each message

for each message
    if the message is the root message then
        place the node at the starting position
        don't draw an arc
    else
        place the message to the right of the last message
        if the message generation depth is odd then
            draw an arc above the line to the message's parent
        else
            draw an arc below the line to the message's parent
next message
```

Figure 8. Pseudo code for drawing Thread Arcs.

Mathematically, for a thread of size *n* messages, the total number of possible Thread Arcs '*t*' that could be constructed is t(*n*) = (*n*-1)! Although the theoretical number of possible threads is enormous, in practice the actual number of threads found in email is a small subset. Figure 9 shows all the possible Thread Arcs that can be built with 2 to 5 messages.

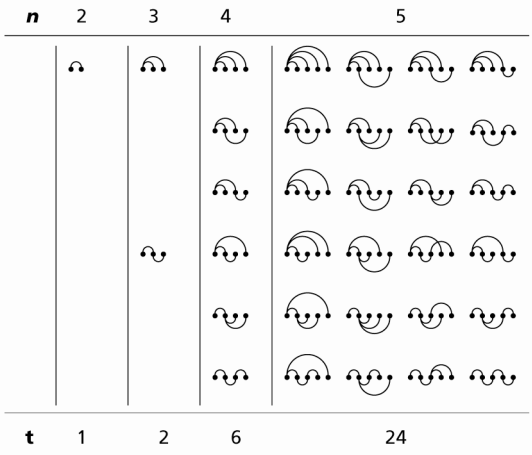

Figure 9. Distinct Thread Arcs for 2 to 5 messages (*n*).

Thread Arcs are designed to be optimal for bushy and narrow threads of 2 to 20 messages and to degrade gracefully for larger threads.

The time ordering sequence allows one to see the evolution of the thread as it grows. For example, Figure 10 shows the growth of a Thread Arc from 1 to 8 messages. In addition, it keeps each message in a *stable* position so that one can return to the exact location where that message was last seen, despite recent growth of the thread.

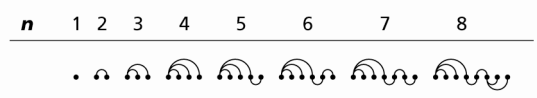

Figure 10. Evolution of a Thread Arc from 1 to 8 messages (*n*).

When a new message is added, the height of its arc reflects how far back in the thread its parent message is, relative to the most recent message. This helps one see any divergence from the most recent branch of the discussion.

4. EXISTING VIZUALIZATION TECHNIQUES

Figure 11 shows the evolution of a thread from 1 to 6 messages, and compares the Thread Arcs visualization to three other visualization techniques: Tree Diagrams, Tree Tables and Compact Chronological Tree Tables. Tree Diagrams (B) are a common way to represent threads. Unlike Thread Arcs, Tree Diagrams do not show *chronology* and are not *stable*. Instead they emphasize the generational nature of a thread. In a Tree Diagram, each row is a new generation of messages, and message nodes are moved to economically pack the nodes as it grows. For example, as message 4 is added to the Tree Diagram, message 3 is moved horizontally to make room for message 4. If one did not return to this thread until it contained 6 messages, it would be unclear which message was now message 3. A Tree Diagram is also not *compact* because it can grow very wide and/or tall, making it hard to dedicate a fixed space for it in an email client. The same problems apply to the Tree Table (C).

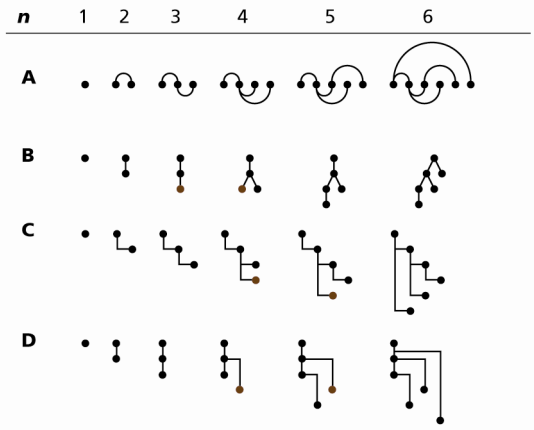

Figure 11. Stability of Thread Arc (A) in comparison to a Tree Diagram (B), a Tree Table (C), a Compact Chronological Tree Table (D), as a thread grows from 1 to 6 messages.

The Compact Chronology Tree Table (D), described in [6,7], does show chronology, but it is unstable. Nodes must move in the horizontal dimension when some new messages arrive, for example, as message 5 arrives message 4 is moved to its right. Compact Chronology Tree Tables can grow wide and/or tall and therefore do not fit the *compact* visualization requirements. In addition, positioning the first child of any message directly below its parent (for example 2 and 3) gives a disproportionate weight to the first child, making the sibling relationships less obvious, for example the siblings 2 and 6.

Because Tree Diagrams and Tree Tables don't show the chronological attributes of a thread, they also over simplify their representation. For example, Figure 12 shows how eight different Thread Arcs (A) are equivalent to one Tree Diagram (B) or one Tree Table (C).

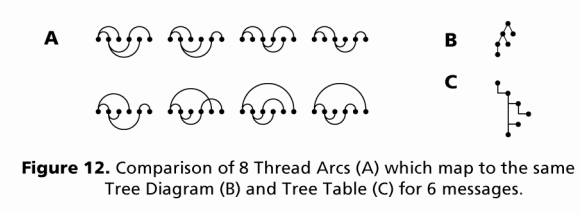

Figure 12. Comparison of 8 Thread Arcs (A) which map to the same Tree Diagram (B) and Tree Table (C) for 6 messages.

As a result, Thread Arcs are more likely to make individual conversations more distinctive and therefore easier for one to recall the content and the position of important messages from the shape of the Thread Arc alone.

When we visualize threads in email we need to represent each message node equally. Other visualization techniques such as Tree Rings, Icicle Plots, and Tree Maps change the size of a message in relation to other messages in the thread based on its position in the tree structure, thereby putting a lot of visual emphasis on specific messages. These visualizations are also limited in their ability to be *compact* [1].

It should be noted that Thread Arcs are visually similar to Arc Diagrams [11], which show repeating structures in a linear list. Thread Arcs, however, differ from Arc Diagrams in three important respects: first, Thread Arcs show tree structures. Second, the arcs in Thread Arcs represent a "reply to" rather than "contains the same sequence as". Third, they use arcs above and below the line to help reduce the number of crossovers, making it easier to see conversation paths.

5. INTERACTION

Thread Arcs in the context of an email client also have interactive components that allow one to highlight and inspect thread and message *attributes* dynamically. This capability allows one to decide which attributes are relevant to the task at hand and display them when needed. For example, when one selects a message to read, the unrelated messages fade out, and the selection is highlighted with a bright blue hollow circle. Its parent appears in a lighter blue highlight, and its children are a darker blue. In Figure 13 below, the children of each selection (S1, S2, S11) are highlighted as dark blue nodes. From this one can see that S2 and S11 each have two children. This highlighting shows specific *relationships* relative to the selection. By clicking on the selected message, one can toggle between this selected state and the highlighting scheme for the thread.

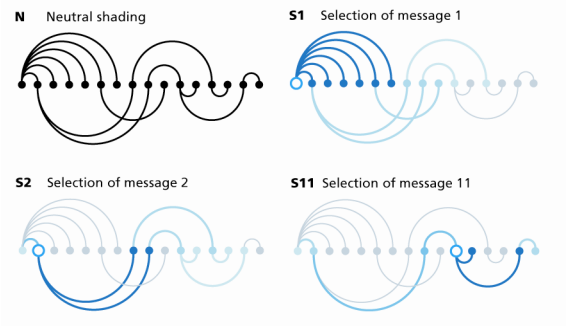

Figure 13. Selection of messages in a thread.

Highlighting schemes show other message attributes, for example one's own contribution to a thread or the contributions of a set of "important people" one has specified. These attributes can also be derived from the thread as a whole. For example, all messages received on the same day as the last message of the thread can be shaded the same way. Other attributes that could be useful to visualize include messages with alerts, positive search results, drafts, calendar information, attachments, or unread messages. Some of these attributes can be shown simultaneously, while others conflict with each other. One can dynamically expose different attributes, depending on what one is looking for. For the *Remail* prototype we used two highlighting schemes: *People Highlighting* and the *Attribute Shading.*

The *People Highlighting* scheme allows one to highlight one's own contributions, the contributions of a set of "important people", or any person from the list of the *contributors* in the currently selected thread with hollow circles. From the *Attribute Shading* schemes one can choose shades to show the times when messages came in and the generational depth of each message, or colors showing each contributor to the thread. These schemes can be activated independently or in combination, with *People Highlighting* superceding the *Attribute Shading* if there is a conflict. The hollow circles, shades and colors used for these schemes are shown in Figure 14 below.

Figure 14. People Highlighting and Attribute Shading schemes.

Figure 15 below, illustrates some of these highlighting schemes for the same thread. P, Personal highlights, shows one's contribution to the thread. This attribute is important because one's own messages often represent to-do's [2, 12].

T, Time shading, shows messages sent on the same day in the same shade of gray. The last five messages in this thread were sent two days after the thread started. This shading helps users see threads that have large intervals between messages. This type of thread has been characterized as one of the harder types of thread to keep track of [2] because in conventional email clients the older messages drift out of the inbox list view as other messages arrive.

C, Contributor shading, shows each contributor to the thread in a different color. In this example only three people were involved in this discussion.

G, Generational shading, uses a different shade for each generation of the thread, showing the depth of the conversation. This helps users see the branching nature of the Thread Arcs, which is less apparent from its linear layout. This shading scheme, with black nodes as the deepest generation, emphasizes the end of branches, which are the current state of the email conversations.

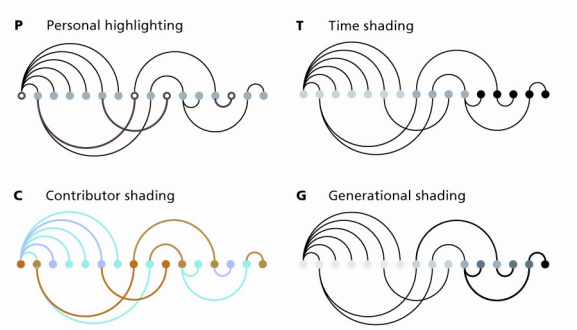

Figure 15. Different message attributes highlighting schemes.

6. STUDY

The goal for our study was to learn about the usefulness and effectiveness of email thread visualizations in users own email. In particular, the important qualities that users wanted thread visualizations to show, as described in "Key Qualities".

As part of this study we gathered statistics of the size and shape of threads found in users' email to give us a better understanding of the frequency and structure of threads that thread visualizations need to accommodate.

6.1 METHOD

We recruited 8 participants for our study, 4 male and 4 female. The participants were all software knowledge workers (Advisory Software Engineer, Human Factors/Usability Specialist, Software Engineer, Senior Development Manager, UI Designer, UI Design/Developer, and Usability Specialist) and were recruited internally. The participants had intermediate to advanced experience using Lotus Notes email client and had been using it for 3 to 10 years. Some had previous experience with large email conversations and discussions databases. None had any previous knowledge of the Thread Arc visualization. Each test took 1.5 hours.

6.2 PROCEDURE

The studies consisted of the following stages.

1. Asking questions about users' backgrounds and email habits.
2. Introducing users to the concept of a conversation visualization using paper images of Thread Arcs, Tree Diagrams and Tree Tables, as show in Figure 16 below.

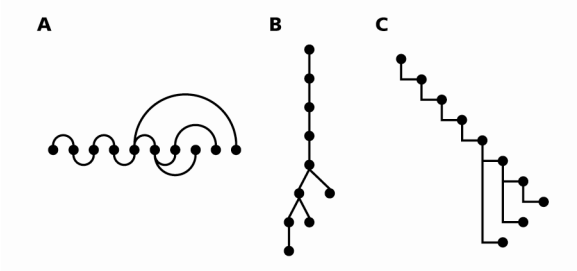

Figure 16. Three visualization techniques showing the same thread. Thread Arc (A), Tree Diagram(B) and Tree Table (C).

3. Conducting two card-sorting exercises for the key qualities and attribute highlighting.
4. Introducing subjects to the three visualization techniques used in the test via interactive screen exercises.
5. Having users explore the visualizations with their own email conversations in a simulated email experience.
6. Repeating the card sorting exercises.
7. Asking the subjects to rate the three visualizations against the key qualities.
8. Creating a series of large scale printed posters of all of the threads found in the users email database.

The test was conducted on an IBM ThinkPad laptop T-30 with an optical mouse. Users were asked to think aloud throughout the test. Audio was captured with a tape recorder and screen and mouse input was recorded using Camtasia screen capture software.

At the beginning of each test we used a Java program to traverse each user's email database, collate all of his or her threads, and output them as a set of XML files. This software implemented an improved version of the complex Zawinski's threading algorithm [13] developed originally for Netscape Messenger. The XML files contained each thread's structure, along with each message's basic email content such as the "to", "from", " subject", "time" and the first 100 characters of the "body". However we did not collect the read/unread status of messages. We used this data as the content for stage 5 of the test, where users experienced a simulation of an email client experience with their own email content. In addition we could use this information to get statistics on the size and structure of their email threads.

At stage 3 we asked users to perform two card sorting exercises to discover:

1. What key qualities users thought were important for a conversation visualization.
2. What attributes of a message in a conversational thread they would like to be able to highlight to help users interpret them.

We had users perform sorts on the key qualities (as previously described in "Key Qualities") and one sort for attribute highlighting schemes (as described in "Interaction" above). This was a priming exercise which gave the subjects a better understanding of what a visualization could represent and a common vocabulary for us to talk about visualization concepts as the test continued. It also gave us an understanding of their perceived priorities before the test. The cards contained titles and a brief textual explanation for reference and we introduced the cards in random order with a short verbal explanation. The first set of cards contained the following key qualities:

- Attribute Highlighting
- Chronology
- Compactness
- Relationships
- Scalability
- Sense/Scanability
- Stability
- Other (additional user suggestions).

The second set contained the attribute highlighting schemes:

- Contributor
- Generational
- Important people
- Marked important (by other people)
- Own contribution
- Time
- Unread
- Other (additional user suggestions).

The sorting tests were repeated in stage 6 to give us users' preferences after experiencing the potential of thread visualizations on their own email threads.

At stage 7 of the test we asked the users to rate each of the visualizations against the key qualities they had ranked earlier. A three-star system was used to determine their preferred method, 3 stars to the visualization that performed best for that quality, 2 stars for the second best, and 1 star for the worst performer.

The posters created at stage 8 allowed us to quantitatively analyze the entire spectrum of threads present in a user's real email. These posters consisted of 9 different attribute highlighting schemes for each of the visualization techniques tested. Users were comfortable with us taking this data away as it showed only the structure, sizes and shapes of the threads with no text content, thereby ensuring their participant's privacy.

6.3 EMAIL PROTOTYPE

During stage 5 we let users explore these threads in a simulation of an email client built using Macromedia Director. Users were able to switch between Thread Arcs, Tree Diagrams and Tree Tables during the test. Each visualization used the same user controls, behaviors, colors and highlighting schemes, so access to and manipulation of each tree visualization was controlled for. We encouraged subjects to switch between the different visualizations and highlighting schemes to get a better understanding of the type of information each visualizations could convey, and what they found most useful. We asked the users to perform small tasks designed to get them to think about the key qualities and test each visualization against them. For example, they had to find the last message in this thread, or figure out how many responses a particular message received. Other exercises included letting users observe the stability of a visualization as new messages were added to a thread. The prototype allowed us to show the evolution of any of the threads they encountered to see how the visualization layouts changed as new messages arrived, from the first to the last message in the thread.

In the email prototype client, shown in Figure 17, two areas were dedicated to contextual information about threads, the *preview pane* and the *thread view pane*. When a message was selected in the inbox list, a thread visualization was displayed in both the *preview pane* (A) and the *thread view pane* (B). In the *preview pane* there was only a limited amount of space (180x50 pixels), so when the Thread Arcs were shown here they were *constrained.* The visualization in the *preview pane* showed a selection highlighting scheme while the *thread view pane* showed the current *attribute highlighting* scheme. Combined this gave the users more information about the entire thread than one scheme alone. These were also interactively linked, so that if a node in the *thread view pane* was selected this new selection would be seen in the *preview pane*, along with a preview of that message. Message nodes were 8 pixels in diameter and when users hovered over them with the mouse they would see the author, time, and subject for that message.

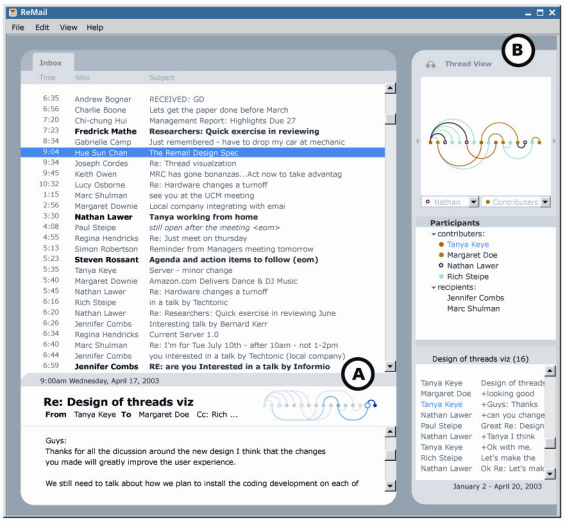

Figure 17. Thread Arcs integrated into email client prototype.

In the *thread view pane,* enlarged in Figure 18 below, the space allocated to the visualization was larger (200x200 pixels). When any visualization was larger than its allocated space, scrolling was provided. For example the Thread Arcs could show a thread with 16 messages before scrolling was needed (and the arc heights were constrained) (C). In addition there were two drop-down menus (D), which allow users to apply *attribute highlighting* schemes. Other contextual information below this area showed all the *participants* in the thread, a combination of the *contributors* and the *recipients* (E). The *contributors* were defined as the authors of the messages in the thread, while the *recipients* were people who received the messages but were not *contributors*. This list dynamically became a legend for the visualizations when an *attribute highlighting* scheme was activated, as shown for the "contributors" highlighting scheme by the colored nodes to the left of each name (E). At the bottom, there was also a list of all the messages in the thread with author and subject (F). Typically the subject lines of a thread were identical to the first message's subject line or would have a "RE:" followed by the subject of the first message. Instead of repeating this redundant information these subject lines contained a "+" and the first line of the body text of that message (this was also the text of the subject line that appeared on the hover over). The lifespan of the thread was shown at the bottom of this pane (G).

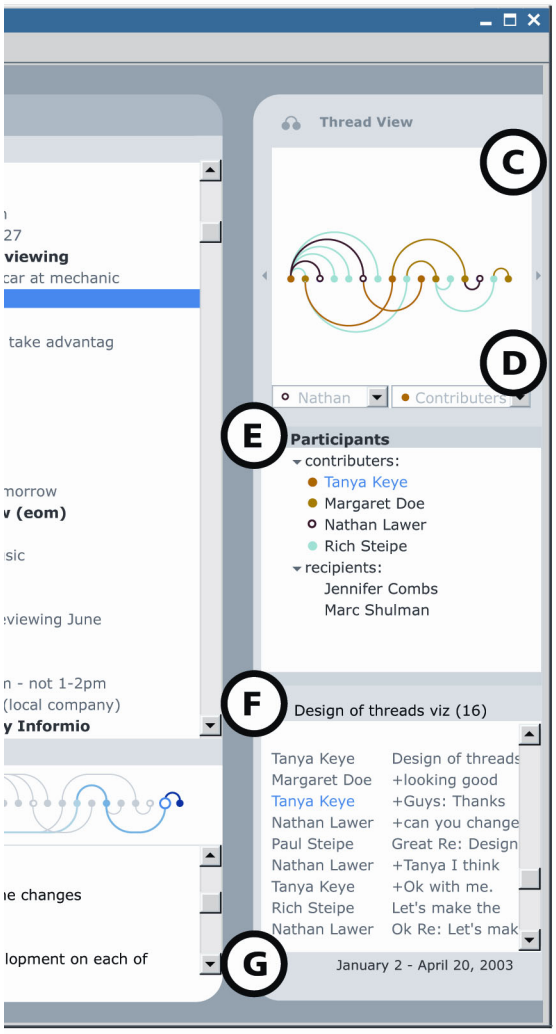

Figure 18. Detailed view of Thread view pane in email prototype.

6.4 RESULTS

We surveyed a total of 42,000 messages in our user's email databases, which included sent as well as received messages. User C's email database contained 84 000 messages alone which was too large for our testing algorithm, so we took a sample 6 months worth of correspondences (3700 messages). Figure 17 below gives the message sizes collected for each user.

Users	Number of messages
A	4 636
B	2 435
C	3 716
D	7 651
E	7 084
F	4 201
G	8 249
I	4 014
Total	**41 986**

Figure 17. Total number of eMail messages in users database.

Figure 21 below, shows the percentage of user's email messages for each size of thread from 1 to 20. From this we see that only 38% of messages were singles or unthreaded (size 1 thread), the next most common thread size was 3 (16%) and as the size of the thread increases it percentage decreases. We did find a handful of threads larger that 20 for each of the users, the biggest was a thread of size 483.

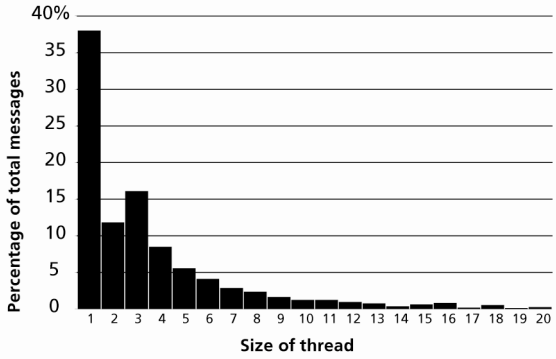

Figure 21. Percentage of total messages found in threads.

Plotting this data cumulatively, in Figure 22, we see that 50% of our user's messages are contained in threads of size 2 or less, and that 80% of all the messages in the study were of size 5 or less.

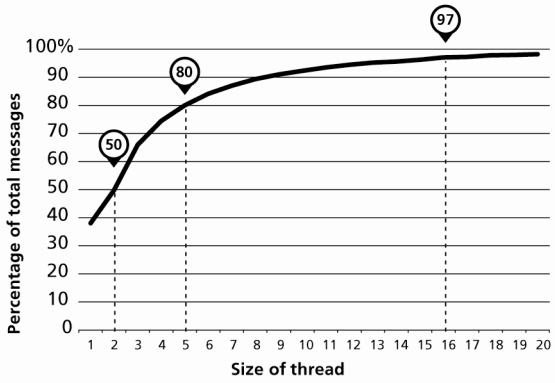

Figure 22. Cumulative percentage of total messages found in threads.

The percentage of distinct thread structures found in users' email for each thread size is shown in Figure 23 below. Other studies have suggested that email threads tend to be "narrow rather than bushy – that is to say that a message is much more likely to get one reply than two or more" [10]. From this data it appears that there is a high percentage of threads that are bushy and there are also a high percentage of threads that are narrow. For example, of threads of size 5, 26% were bushy and 20% were narrow. For our users' data we see this polarization trend continue for larger threads. The biggest threads found were either bushy or narrow. This means that the visualizations and the space dedicated for them needs to accommodate both bushy and narrow threads. The compact nature of Thread Arcs makes it particular good at achieving this goal.

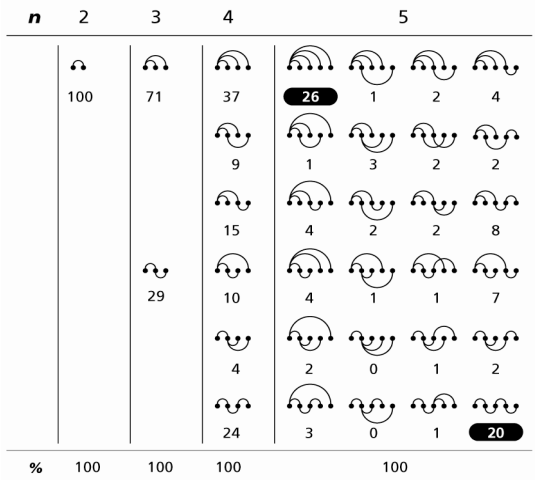

Figure 23. Distribution of distinct Thread Arcs for 2 to 5 messages (*n*) found in users' email (as a percentage). Highlights show the bushy and narrow threads found in threads of size 5.

Figure 23 also reveals another interesting aspect of thread structures in this study. We see that users have a tendency to reply either to the first or to the last message of a thread. For example, Figure 24 below shows the possible growth of a thread with 4 messages. When a new message arrives it will be a reply to any one of the 4 messages currently in the thread. These 4 possibilities are represented by the 4 columns in the n-5 section of Figure 23. The first column shows threads which have their latest message (5th message) as a reply to the 1st message while the fourth column shows messages that have their latest message as a reply to the 4th message. The median values for each of these columns are 3.5, 1, 1.5 and 5.5, which implies that the users have a tendency to reply to either the first or the last messages in a thread of size 4. This behavior is interesting because it tends to support the informal use of the reply function in email, discussed in the introduction.

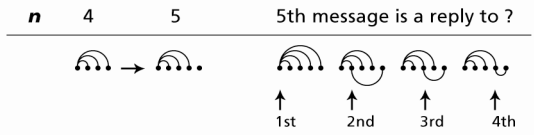

Figure 24. Possible growth of a thread as it grows from 4 messages to 5 messages.

Users found all of the key qualities they were asked to consider important for small-scale thread visualization. When forced to rank them after their test experience Relationships and Chronology came out as the most important, as seen in Figure 25.

Key Qualities	Median Ranking
Relationships	2.0
Chronology	2.5
Attribute Highlighting	3.0
Sense/Scanability	3.5
Scalability	4.5
Stability	6.0
Compactness	6.0

Figure 25. Post test rankings of Key Qualities (1-7 scale, 1 best).

The post test ranking order of the attribute highlighting schemes are shown in Figure 26 below. It is significant that unread was considered the most important attribute to highlight even though we were not able to gather or show the unread status of messages for the test. The importance of time reinforces the fact that users want to see some form of chronology as a primary aspect of these visualizations.

Attribute Highlighting	Median Ranking
Unread	2.0
Time	2.5
Important People	3.0
Contributors	3.5
Marked Important	5.0
Generational	6.0
Own contribution	6.5

Figure 26. Post test rankings of Attribute Highlighting (1-7 scale, 1 best).

The ratings for each of the visualization techniques in relation to the key qualities are shown in Figure 27. A three star system was used to determine their preferred method with 3 stars to the visualization that performed best for that quality, 2 stars for the second best and 1 star for the worst performer. The attribute highlighting schemes were identical for each visualization so we did not ask users to rate it.

Key Qualities	Arc	Diagram	Table
Relationships	1.0	2.0	**2.5**
Chronology	**3.0**	1.5	1.5
Sense/Scanability	1.3	**2.0**	**2.0**
Scalability	1.0	**2.5**	2.0
Stability	**3.0**	1.5	1.3
Compactness	**3.0**	2.0	1.3

Figure 27. Median of ratings for each visualization (1-3 scale, 3 is best).

6.5 DISCUSSION

Thread Arcs did significantly better than the other visualizations in chronology, stability and compactness. And on balance they did a better job at satisfying all of the qualities that users valued in small scale thread visualization in email.

As mentioned in the introduction of this paper one of the challenges in displaying these conversational threads is that they have two conflicting properties: the arrival sequence of messages and their reply-to relationship. This study has confirmed that users find both of these qualities important, but in addition we have also shown that for small scale visualizations in the context of an email client there are other important qualities to be considered. In particular the compactness and the scalability of visualizations for the size and structure of threads found in users real email.

Chronology was clearly the most important element for threads in email, both in terms of key quality ranking and the attribute highlighting ranking. Thread Arcs had the best rating on this quality by all users.

It should be noted that the other visualization techniques could be modified to also emphasizes chronology, as others have done [8][10]. Unfortunately these techniques result in sacrifices in the compactness, stability and sense/scanability of the visualizations as shown in Figure 28 below.

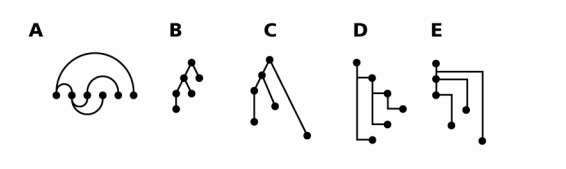

Figure 28. Thread Arc (A), Tree Diagram(B) , Tree Table (D) with modified versions of the Tree Diagram(C) and Tree Table(E) that emphasies chronology.

The comparatively low ranking for the Thread Arcs against the others for the Relationship quality in this study is the result of three factors. First, Thread Arcs' primary emphasis on Chronology results in less emphasis in the Relationships quality. Second, the users focused on large threads during the test. When users examined their smaller threads of size 3-5 (80% of their mail) the complexity of the visualization was trivial to interpret so they concentrated on examining larger threads, of size 8 and above. Third, users were more familiar with the other two techniques (tree tables and tree diagrams) used in the test. Therefore their interpretation was easier for users, and rating correspondingly higher.

Although rated low in the key qualities rankings, compactness was a big issue for all users whenever the visualization extended outside of its dedicated area. Thread Arcs could accommodate 16 messages (97% of all threads as seen in Figure 20) without the need to scroll in the preview pane or the thread view pane. In contrast the Tree Table needed to be scrolled in the preview pane for any threads with more than 5 messages. The Tree Diagram performed better than the Tree Table for wide threads but suffered if the threads were deep.

However, the need to see the relationships in larger threads (greater than 5 messages) should not be discounted as many of the larger threads encountered in this study were important conversations and visualization became a even more valuable tool to give context to users.

The poster printout tells us that the polarity of threads discussed in the distinct figure above continues as threads get bigger. The result is that one must design for narrow threads as well as bushy threads.

From the posters we also noted a large number of threads that had empty message nodes. This was caused by the Zawinski[13] algorithm used to find the threads in users' email. This algorithm takes two passes through users' databases. The first pass uses the message's reply reference to join related messages. The second pass then collates any of these threads which have matching "RE:" based subject lines at their roots. These empty message nodes will be created for one of two reasons: One is that an empty node is created on the first pass if a message, referred to by another message in a thread, is missing – either deleted or not saved when sent. Another is that, on the second pass, if two threads found on the first pass have the same "RE:" based subject line they are considered to be part of the same thread and if no root node is found an empty node is created. In this case the algorithm treats both messages as siblings of the newly created empty node. See Zawinski[13] for more details of this method.

The algorithm tends to be aggressive in threading messages in a bushy fashion, by creating empty nodes for missing messages. But overall thread distribution is consistent with others like Fisher and Moody's [4]. Their threading algorithm did not have a second pass subject-matching scheme and therefore their results would increase the number of size 1 threads and undercount longer threads by not taking into account any missing messages. More research into appropriateness of these empty nodes and their interpretation by users is required.

The important people and contributors highlighting schemes were ranked 3rd and 4th on the Attribute Highlighting rankings. Some users described this as the default setting that they would apply if they had this sort of visualization technique in their email client. The Marked Important and Generational attributes were ranked as much less important. We were surprised to see that one's own contribution had the lowest ranking. Users tended to believe that they did not need to see their own contribution because they already knew what they had written and there was no need to bring that to their attention. Users still placed some value on this attribute and we suspect that in use this would give context to their involvement in threads and allow them to quickly see any responses to their own contributions. Taken together one could consider important people, contributors and own contribution as a general people attribute highlighting scheme.

The participants list was also seen as extremely useful because it allowed people to quickly identify all the people involved in the conversation and get a better sense of the context of the thread quickly.

One of the additional highlighting schemes that some of the users suggested was an indication of changes in the participants list within the thread. For example if new people were added to the "To" or "CC:" fields as the thread evolved. This would allow users to ensure that all participants were informed or quickly see when new people had been added to the conversation.

Users liked the subject line modification replacing it with the first line of the body if the subject was identical to the first message. Some observed that, for the Thread Arcs, having the nodes in a line makes it much easier to hover over the entire set of messages in a single horizontal motion rather than hunting around a branching structure.
Perhaps one of the most useful aspects of the visualization's interactions was the ability of users to quickly navigate, by clicking on nodes, to other messages in the thread without having to use their inbox.

All, but one, of the users said that they would like a thread visualization in their future email clients.

7. FUTURE RESEARCH

Further users studies could provide insights into the types of tasks users want to perform with threads and modify some of the design criteria for thread visualizations. We would

predict that different users with different work practices would produce different thread structures.

Text analysis of the message's contents could help to expose other attributes of a thread, for example the highlighting of search results. We have also been exploring designs for a *thread reading pane*, which would give more contextual thread information when reading a set of thread messages.

Improvements in the Zawinski[13] algorithm are being considered to improve the accuracy of the threads that are built on the second pass of the algorithm. Another issue to resolve is what users would prefer to see when the algorithm joins threads with missing messages and/or encounters empty nodes.

Different message sort orders for the layout of the messages in Thread Arcs have potential benefits. Instead of laying out the message nodes in strict time sequences we have placed them in hierarchal and generational orders, as shown in Figure 29. This removes the strong chronology characteristics of the Thread Arc but reveals these other attributes of the thread such as the sub branches of a thread or the generational depth of the threads. We have found that for very large threads (greater than 100 messages) chronological sorting becomes less useful and that these other types of sorts can reveal useful properties of the thread. In particular it reduces the number of overlaps in big discussions such as Usenet.

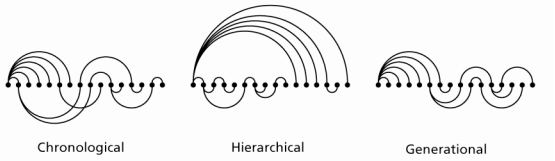

Figure 29. Comparison of the same thread with different message sorting techniques.

8. CONCLUSION

The main contribution of this paper is the description of a tree visualization technique that can display chronological and relationship properties simultaneously. Thread Arcs show the "reply to" *relationship,* with an emphasis on the *chronology* of when the messages arrived. The *relationships* between messages are shown with arcs that connect each message to its parent. By laying out the message nodes linearly, the visualization not only emphasizes *chronology,* but also renders if *compact* and *stable*. This stability allows one to observe the evolution of a thread over time. Other useful properties of Thread Arcs include the ability to see, at a glance, the size of a thread and the number of responses to any specific message in it. There are a number of attribute highlighting schemes that can be applied to Thread Arcs which help one find important messages or predict the types of conversations present. We have also compared Thread Arcs to existing techniques and described the *key qualities* that we consider important to thread visualizations in email. Our user study has confirmed the importance of chronology in email threads. The study also showed that Thread Arcs are well suited for the size and type of conversations found in users' real email.

ACKNOWLEDGEMENTS

I would like to thank all the people in my research group. In particular Steven Rohall, Martin Wattenberg, Kushal Dave, Li-Te Cheng, Eric Wilcox and Maida Eisenberg. I would also like to thank Deb Maurer and Sandra Kogan for their assistance with the user testing.

REFERENCES

1. Barlo, T., Neville, P., "A Comparison of 2-D Visualizations of Hierarchies," Proceedings of the IEEE Symposium on Information Visualization 2001.

2. Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. "Taskmaster: recasting email as task management," Workshop: "Redesigning Email for the 21st Century", CSCW 2002.

3. Donath, J., Karahalios, K., Viegas, F., "Visualizing Conversation," Proceedings of the Hawaii Internationals Conference on System Sciences 32, January 1999.

4. Fisher, D and Moody, P. "Studies of Automated Collection of Email Records." University of Irvine, Technical Report, UCI-ISR-02-4, 2001.

6. Rohall, S.L., Gruen D., Moody P., and Kellerman S., "Email Visualizations to Aid Communications," Late Breaking, Hot Topic Proceedings of the IEEE Symposium on Information Visualization, San Diego, CA, October 22-23, 2001, pp. 12-15.

7. Sack, W., "Conversation Map: A Content-Based Usenet Newsgroup Browser", Proceedings of IUI'00, New Orleans, LA, January 9-12, 2000, pp. 233-240.

8. Smith, M. A., Fiore, A. T., "Visualization Components for Persistent Conversations", Proceeding of CHI 01, 31 March-5 April, 2001.

10. Venolia, G. and Neustaedter, C. "Understanding Sequence and Reply Relationships within Email Conversations: A Mixed-Model Visualization," CHI 2003 Paper, Nov 25, 2002

11. Wattenberg, M. "Arc Diagrams: Visualizing Structure in Strings," Proceedings of the IEEE Symposium on Information Visualization, Boston, MA, October 28-29, 2002 pp. 110-116.

12. Whittaker, S. and Sidner C., "Email Overload: Exploring Personal Information Management of Email," Proceedings of CHI'96, Vancouver, B.C., April 13-18, 1996, pp. 276-283.

13. Zawinski, J. http://www.jwz.org/doc/threading.html