**RC22951 (W0310-175) October 30, 2003**
**Computer Science**

# IBM Research Report

## THREAD ARCS: An Email Thread Visualization

**Bernard Kerr**
IBM Research Division
T. J. Watson Research Center
One Rogers Street
Cambridge, MA 02142

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

**LIMITED DISTRIBUTION NOTICE**: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. Ithas been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distributionoutside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g, payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

# THREAD ARCS: An Email Thread Visualization

Bernard Kerr
Collaborative User Experience Group
IBM Research
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 [Venolia and Neustaedter 2003] 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 email client. Thread Arcs proved an excellent match for the types of threads found in users' email and for the qualities users wanted in small-scale visualizations.

**CR Categories:** H.5.2 User Interfaces, H.5.3 Group and Organization Interfaces, I.3.6 Methodology and Techniques

**Keywords:** conversations, discussions, electronic mail, email, information visualization, threads, tree structures, user interfaces.

## 1. Introduction

The importance of conversation threads in email and tools for inspecting them has been well documented. The main advantages are: allowing users to see a greater context of the messages they are reading, reminding users that a conversation is in progress, recording the state of a discussion, collating related messages automatically, reducing messages displayed in inboxes, and allowing users to perform actions such as reading or deleting on a group of messages [Fisher and Moody 2001; Rohall et al. 2001; Venolia and Neustaedter 2003; Whittaker and Sidner 1996].

Common terminology used to discuss threads is defined here briefly. A *thread* is defined as a collection of individual messages related to each other by the reply function in email. In a thread, the message to which a reply is sent is called the *parent* of that message. Any replies to a message are called *children* of that message. The first message in a thread is called the *root*. The *generational depth* of a message is the number of "reply to" relationships between a message and its root. For example, all messages that are replies to the root message have a generational depth of one.

Email threads differ from public discussion threads, such as those found in Usenet, in a number of ways. Public discussion threads are often large [Smith and Fiore 2001]. In contrast email threads are relatively small. Fisher and Moody [2001] found that threads of greater than one message account for 35% of their users' mailboxes, and that 87% had four messages or fewer.

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

Chronology is also important because email is often time-sensitive; 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" [Whittaker and Sidner 1996] 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" [Smith and Fiore 2001], Loom [Donath et al. 1999] and Conversation Map [Sack 2000] consider a different set of qualities when presenting thread information than those needed for email. They contain social structures and conversations that are very different from the egocentric nature of email. For example, email users usually know all of the people involved in an email thread, while Usenet is more public and anonymous. Email threads 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 [Venolia and Neustaedter 2003]. We describe a visualization technique that can effectively communicate both of these properties at once and can explain how we tested this design (in an email client prototype) on real threads found in users' own email.

## 2. Key Qualities

Outlined here are seven qualities we consider to be essential in effectively visualizing threads found in email, along with a brief discussion of their value.

1. **Chronology**: One must show the arrival sequence of messages which create a thread. This illustrates the evolution of a thread, including which messages came first, and which is the most recent message.
2. **Relationships**: One should make all of the "reply to" relationships visible at a glance. This allows the user to see

the direct relationships of a particular message to all others in a thread. For example, one should be able to examine a particular message, and should be able to trace back through the chain of earlier messages which lead back to the root of the thread. In addition, one should be able to see which messages subsequently respond to a particular message clearly. These relationships give important contexts for each message in a thread.

3. **Stability**: As a thread grows, it is essential to 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**: Since the visualization will be competing with other space required for email client functionality, it is imperative that any visualization be small in size without compromising clarity.
5. **Attribute Highlighting:** One must be 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 aids in assessing the state of a thread.
6. **Scale**: A visualization should work for small threads as well as larger ones found in email. The clarity of the visualization should degrade gracefully as more messages arrive. It must still be clean when threads are large and complex. Since the vast majority of email threads are typically between two and twenty messages [Fisher and Moody 2001], the visualization does not have to scale to hundreds or thousands of messages.
7. **Interpretation/Sense**: A visualization must give a sense of the type of conversation present in a thread, i.e. whether a thread is a back-and-forth reply chain between two people, or 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*.

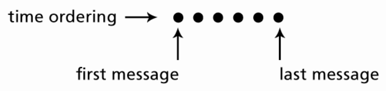


**Figure 1.** Chronology of message nodes in a line of six messages.

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

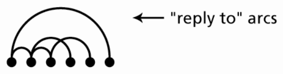


**Figure 2.** Relationships are shown with "reply to" arcs connecting nodes above the line.

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.

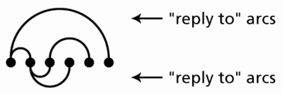


**Figure 3.** Relationships are shown with "reply to" arcs connecting nodes above and below the line.

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

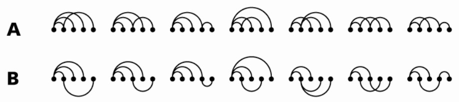


**Figure 4.** The relationship between messages are clearer when arcs are draw above and below nodes (B).

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*, as shown below in Figure 5.

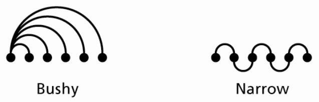


**Figure 5.** Bushy threads have messages which recieve two or more replies. Narrow threads recieve only one reply per message.

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 so, is easier to fit inside the space constraints of an email client.

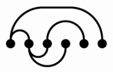


**Figure 6.** Constraining the maximum height of the arcs makes the visualization more compact.

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

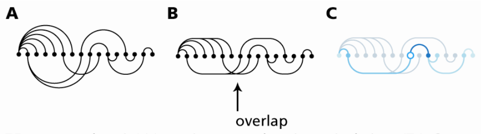


**Figure 7.** Unconstrained (A) and constrained arc heights (B) for a thread of sixteen messages. A selection highlighting scheme for the ninth message (C).

This technique creates some ambiguity when arcs overlap. This problem is alleviated by the *selection highlighting* that is described under Section 5, "Interaction", later in this paper, along with the other *attribute highlighting* schemes.

The 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 from *n* time-ordered messages is $t(n) = (n\text{-}1)!$ (i.e. thread variations are due to parent/child relationships only). 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.

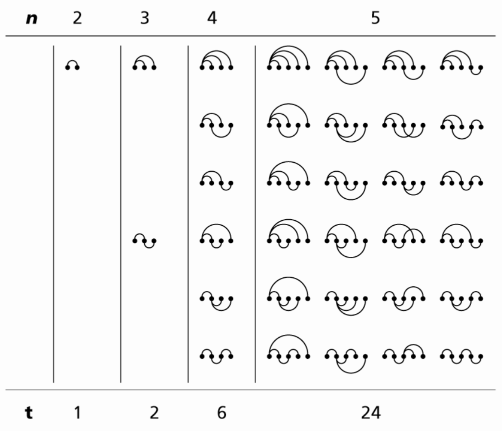


**Figure 9.** Distinct Thread Arcs for two to five messages (n).

Thread Arcs are designed to be optimal for bushy and narrow threads of around two to twenty messages. For larger threads, the visualization degrades gracefully because one can still see bushy and narrow structures within the total thread. This helps one scan large threads visually and *interpret* or get a *sense* of the conversations taking place.

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 one to eight 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.

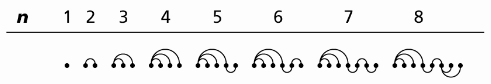


**Figure 10.** Evolution of a Thread Arc from one to eight messages (n).

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 Visualization Techniques

Figure 11 shows the evolution of a thread from one to six messages, and compares the Thread Arcs (A) visualization to two other visualization techniques, Tree Diagrams (B) and Tree Tables (C). Tree Diagrams 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 pack the nodes more economically as it grows. For example, as message 4 is added to the Tree Diagram (B), message 3 is moved horizontally to make room for message 4. If one did not return to this thread until it contained six messages, it would be unclear which message was now message 3. Tree Diagrams are also not very *compact* because they can grow very wide and/or tall, making it hard to dedicate a fixed space for them in an email client. The same problems apply to the Tree Table (C).

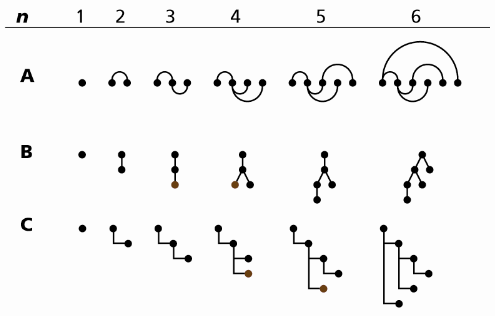


**Figure 11.** Stability of Thread Arc (A) in comparison to a Tree Diagram (B) and a Tree Table (C), as a thread grows from one to six messages.

Tree Diagrams and Tree Tables also oversimplify the conversational structure of threads because they do not show *chronology*. For example, Figure 12 shows that the Tree Diagram (A) and the Tree Table (B) are a simplification of eight potentially different conversations which may have occurred. The Thread Arcs (C) show all of the possible ways that the conversation may have occurred. For example, the last message in the thread could have been concluding "thank you" to the fifth message (top left Thread Arc (C)), or someone could be trying to take the entire conversation in another direction by responding to the first message (bottom right Thread Arc (C)). This chronological information makes it much easier to get a *sense* of the current state of the conversation.

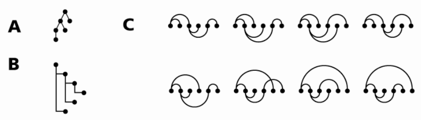


**Figure 12.** Tree Diagram (A) and Tree Table (B) oversimplify the conversational structure of a thread. The 8 Thread Arcs (C) reveal different ways the conversation may have evolved.

Individual conversations shown as Thread Arcs are more likely to be distinctive and, as a consequence, this makes it easier for one to recall the content and the position of important messages from the shape of the Thread Arc alone.

Tree Diagram [Smith and Fiore 2001] and Tree Table [Rohall et al. 2001; Venolia and Neustaedter 2003] techniques have been modified to emphasize chronology. Unfortunately, both modified techniques are not *stable* and can grow wide and/or tall depending on the structure of the thread, thereby becoming less *compact.* As shown in Figure 13 below, the Tree Diagram (B) sacrifices *compactness* when modified (C). The modified Tree Table (E) [Rohall et al. 2001; Venolia and Neustaedter 2003] attempts to stay *compact* by its layout technique. It positions the first child of any message directly below its parent. This spatial positioning gives a disproportionate weight to the first children of any generation and makes any other sibling relationships less obvious. As shown below (E), the nodes down the left hand side (the first branch) of the thread have a disproportionately strong visual presence compared to the other two nodes.

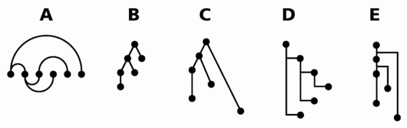


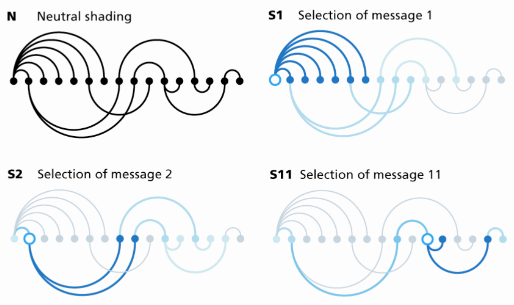
**Figure 13.** Thread Arc (A), Tree Diagram (B), Tree Table (D) with modified versions of the Tree Diagram (C) and Tree Table (E) that emphasize chronology.

When we visualize threads in email, we 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 [Barlo and Neville 2001]. These visualizations do not show *chronology,* and are also limited in their ability to be *compact.*

It should be noted that Thread Arcs are visually similar to Arc Diagrams [Wattenberg 2002]. Thread Arcs, however, differ from Arc Diagrams in three important respects. First, Thread Arcs show tree structures of the reply relationship between a set of messages, while Arc Diagrams show repeating sequences in a linear list which contain the same sequence. Second, Thread Arcs show message-to-message relationships with lines, while Arc Diagrams show sequence-to-sequence relationships with area. Third, Thread Arcs use arcs above and below the message nodes 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 to 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 14 below, the children of each selection (S1, S2, S11) are highlighted as dark blue nodes. From this one can see that for the selected messages in S2 and S11, both 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.



N Neutral shading
S1 Selection of message 1
S2 Selection of message 2
S11 Selection of message 11

**Figure 14.** Selection of messages in a thread.

Highlighting schemes show other message attributes, such as 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 highlights, with hollow circles, 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. 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 one can use colors showing each contributor to the thread. These schemes can be activated independently or in combination. If there is a conflict, *People Highlighting* superceding the *Attribute Shading*. The hollow circles, shades, and colors used for these schemes are shown in Figure 15 below.

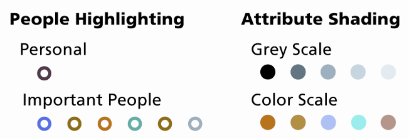


**Figure 15.** People Highlighting and Attribute Shading schemes.

Figure 16 below illustrates some of these highlighting schemes for the same thread. Personal highlights (P), shows one's contribution to the thread. This attribute is important because one's own messages often represent "to-do's" that one expects from others [Bellotti et al. 2002; Whittaker and Sidner 1996].

Time shading (T) shows messages sent on the same day in the same shade of gray. The last eleven messages in this thread were sent four days after the thread started. This shading helps emphasize 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 [Bellotti et al. 2002] because, in conventional email clients, the older messages drift out of the inbox list view as other messages arrive.

Contributor coloring (C) shows each contributor to the thread in a different color. In this example, only four people were involved in this discussion.

Generational shading (G) uses a different shade for each generation of the thread, showing the generational 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.

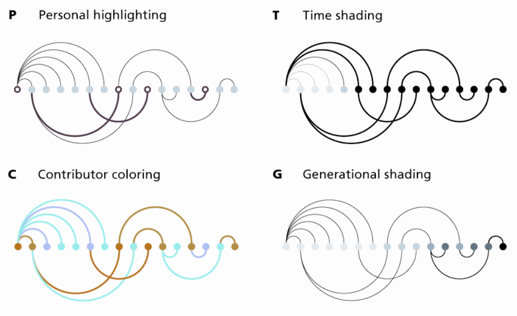


**Figure 16.** Different message attributes highlighting schemes.

## 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, we investigated which qualities users considered important in thread visualizations, as described in Section 2, "Key Qualities".

As part of this study, we gathered statistics on 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 eight participants for our study, four male and four female. The participants were all software knowledge workers, and were recruited internally. The participants had intermediate to advanced experience using the Lotus Notes email client, and had been using it for three to ten years. Some had previous experience with large email conversations and discussion databases. None had any previous knowledge of the Thread Arc visualization.

At the beginning of each test, we used a Java program to traverse each user's email database, then collated 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 [2002] originally developed for Netscape Messenger. The XML files contained each thread's structure, along with each message's basic email content, including the "to", "from", " subject", "time", and the first 100 characters of the "body". We used this data as the content for the study where users experienced a simulation of an email client with their own email content. In addition, we used this information to get statistics on the size and structure of their email threads.

At the conclusion of each user's session, we created a series of large scale posters of all of the threads found in their email database. This allowed us to analyze quantitatively the entire spectrum of threads present in a user's real email. These posters consisted of nine 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 privacy.

### 6.2 Email Prototype

As part of the study, we let users explore their email 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 visualization was controlled.

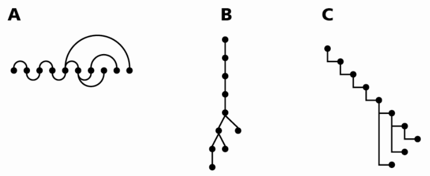
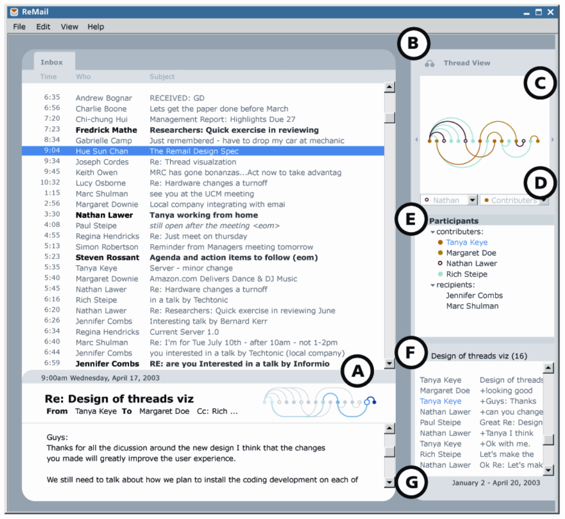


**Figure 17.** Three visualization techniques showing the same thread. Thread Arc (A), Tree Diagram (B) and Tree Table (C).

We encouraged subjects to switch between the different visualizations and highlighting schemes to get a better understanding of the type of information each visualization could convey so they could assess which one they found most useful. We asked the users to perform small tasks designed to get them to think about the *key qualities,* and to test each visualization against them. For example, they were asked to find the last message in a thread, or to figure out how many responses a particular message received. Another exercise involved 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. We could demonstrate how the visualization layouts changed as new messages were added from the start of the thread up to its current state.

In the email prototype client shown in Figure 18, 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 (185x50 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 linked interactively 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.



| Time | Who | Subject |
|---|---|---|
| 6:35 | Andrew Bognar | RECEIVED: GD |
| 6:56 | Charlie Boone | Lets get the paper done before March |
| 7:20 | Chi-chung Hui | Management Report: Highlights Due 27 |
| 7:23 | **Fredrick Mathe** | **Researchers: Quick exercise in reviewing** |
| 8:34 | Gabrielle Camp | Just remembered - have to drop my car at mechanic |
| 9:04 | Hue Sun Chan | The Remail Design Spec |
| 9:34 | Joseph Cordes | Re: Thread visualzation |
| 9:45 | Keith Owen | MRC has gone bonanzas...Act now to take advantag |
| 10:32 | Lucy Osborne | Re: Hardware changes a turnoff |
| 1:15 | Marc Shulman | see you at the UCM meeting |
| 2:56 | Margaret Downie | Local company integrating with emai |
| 3:30 | **Nathan Lawer** | **Tanya working from home** |
| 4:08 | Paul Steipe | *still open after the meeting <eom>* |
| 4:55 | Regina Hendricks | Re: Just meet on thursday |
| 5:13 | Simon Robertson | Reminder from Managers meeting tomorrow |
| 5:23 | **Steven Rossant** | **Agenda and action items to follow (eom)** |
| 5:35 | Tanya Keye | Server - minor change |
| 5:40 | Margaret Downie | Amazon.com Delivers Dance & DJ Music |
| 5:45 | Nathan Lawer | Re: Hardware changes a turnoff |
| 6:16 | Rich Steipe | in a talk by Techtonic |
| 6:20 | Nathan Lawer | Re: Researchers: Quick exercise in reviewing June |
| 6:26 | Jennifer Combs | Interesting talk by Bernard Kerr |
| 6:34 | Regina Hendricks | Current Server 1.0 |
| 6:40 | Marc Shulman | Re: I'm for Tue July 10th - after 10am - not 1-2pm |
| 6:44 | Jennifer Combs | you interested in a talk by Techtonic (local company) |
| 6:59 | **Jennifer Combs** | **RE: are you Interested in a talk by Informio** |

9:00am Wednesday, April 17, 2003

**Re: Design of threads viz**
**From** Tanya Keye **To** Margaret Doe Cc: Rich ...

Guys:
Thanks for all the dicussion around the new design I think that the changes
you made will greatly improve the user experience.

We still need to talk about how we plan to install the coding development on each of

**Participants**
- contributers:
  - ● Tanya Keye
  - ● Margaret Doe
  - ○ Nathan Lawer
  - ● Rich Steipe
- recipients:
  - Jennifer Combs
  - Marc Shulman

Design of threads viz (16)

| | |
|---|---|
| Tanya Keye | Design of threads |
| Margaret Doe | +looking good |
| Tanya Keye | +Guys: Thanks |
| Nathan Lawer | +can you change |
| Paul Steipe | Great Re: Design |
| Nathan Lawer | +Tanya I think |
| Tanya Keye | +Ok with me. |
| Rich Steipe | Let's make the |
| Nathan Lawer | Ok Re: Let's mak |

January 2 - April 20, 2003

**Figure 18.** Thread Arcs integrated into email client prototype.

In the *thread view pane* the space allocated to the visualization (C) was larger (185x185 pixels). When any visualization was larger than its allocated space, scrolling was provided. In addition, there were two drop-down menus (D) which allowed users to apply *attribute highlighting* schemes. Other contextual information below this area showed all the *participants* in the thread, both the *contributors* and the *recipients* (E). The *contributors* were defined as the authors of messages in the thread, while the *recipients* were people who received the messages but were not *contributors*. This list was a legend for the visualizations which changed dynamically when an *attribute highlighting* scheme was activated. For example, the "contributors" highlighting scheme shows 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 they had 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. The lifespan of the thread was shown at the bottom (G).

## 6.4 Results

We surveyed a total of 42,000 messages in our users' email databases, which included "sent" as well as "received" messages. Figure 19 below shows the percentage of users' email messages for each size of thread from 1 to 20.

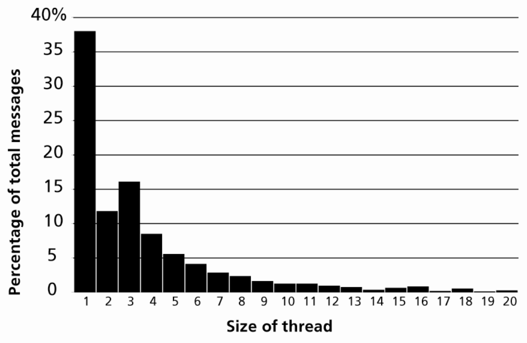


**Figure 19.** Percentage of total messages found in threads.

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 increased, its percentage decreased. We did find a handful of threads larger than 20 for each of the users; the biggest was a thread of 483 messages. Plotting this data cumulatively, we see from Figure 20 below, that 50% of our users' 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.

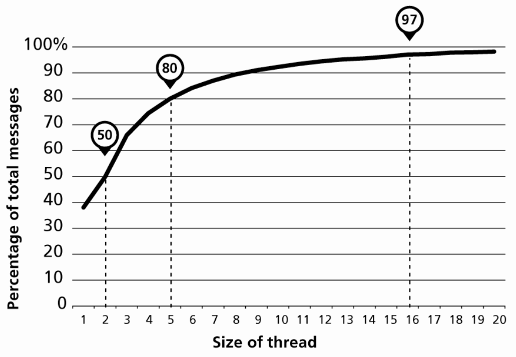


**Figure 20.** Cumulative percentage of total messages found in threads.

The percentage of distinct thread structures found in users' email for each thread of size 2-5 is shown in Figure 22 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" [Venolia and Neustaedter 2003]. From this data, it appears that there is a high percentage of threads that are bushy, and there are similarly a high percentage of threads that are narrow. For example, of threads of size five, 26% were bushy and 20% were narrow.

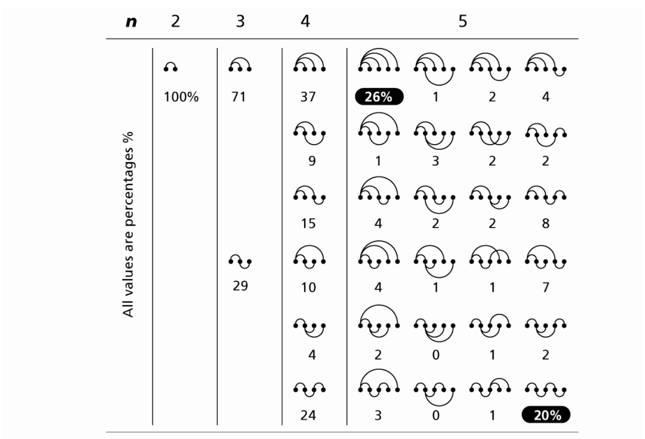


**Figure 21.** Distribution of distinct Thread Arcs for two to five messages (n) found in users' email (as a percentage %). Highlights show the bushy and narrow threads found in threads of size five.

## 6.5 Discussion

Thread Arcs proved to be a better technique for showing *chronology*, and were both more *stable* and more *compact* than either Tree Diagrams or Tree Tables. On balance, Thread Arcs did a better job at satisfying all of the *key qualities* that users valued.

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. Showing the *chronology* as part of the visualization came through as a strong theme. In addition, we found that for small-scale visualizations in the context of an email client, there are other important qualities to be considered, specifically the *compactness* and the *scale* of visualizations needed for the size and structure of threads found in users' real email.

Compactness was a big issue for all users whenever the visualization extended outside of its dedicated area. Thread Arcs could accommodate sixteen 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 five messages. The Tree Diagram performed better than the Tree Table for bushy threads but suffered if the threads were narrow.

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

For our users' data, we saw that the polarity of bushy and narrow threads discussed above in Figure 21 continues as threads get larger. From the posters that were generated at the end of tests, we saw that the biggest threads found were either bushy or narrow. This means that visualizations and the space dedicated for them need to accommodate both bushy and narrow threads for small and large numbers of messages. The compact linear nature of Thread Arcs makes it particularly good at achieving this goal.

The posters also revealed a number of threads that had empty or missing messages caused by the Zawinski [2002] algorithm. This algorithm, used to find the threads in users' email, tends to be aggressive in threading messages in a bushy fashion. It does this by creating empty messages for any missing messages. The algorithm takes two passes through a user's 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. Empty messages will be created for one of two reasons. An empty message 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. Alternatively, 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 message is created. In this case, the algorithm treats both messages as siblings of the newly created empty message. See Zawinski [2002] for more details of this method.

However, the overall thread distribution is consistent with other studies like that of Fisher and Moody [2001]. It should be noted that 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 would undercount longer threads by not taking into account any missing messages. More research into the appropriateness of these empty messages and their interpretation by users is required.

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

Users liked the subject line modification in which the first line of the body replaces the subject line if the subject is identical to the first message. Some users observed that arranging the nodes in a line in Thread Arcs made it much easier to hover over the entire set of messages in a single horizontal motion compared to a branching structure where users had to "hunt around" to find messages.

Perhaps one of the most useful aspects of the visualization's interactions is the ability of users to navigate quickly to other messages in the thread by clicking on nodes without having to use their inbox list.

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

## 7. Future Research

Further user studies could provide insights into the types of tasks users want to perform with threads and could modify some of the design criteria for thread visualizations. For example, text analysis of a message's contents could help to expose other attributes of a thread. We also predict that different users with different work practices would produce different thread structures.

Designs for a *thread reading pane* are also being explored which would give more contextual thread information when reading a set of thread messages.

Improvements in the Zawinski [2002] 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 hierarchical and generational orders, as shown in Figure 22. This removes the strong chronological characteristics of the Thread Arc but reveals other attributes of the thread, such as the sub-branches or the generational depth. We have found that, when testing Thread Arcs on large discussion conversations (greater than 100 messages), chronological sorting becomes less useful, and that these other types of sorts may reveal useful properties of the thread. The hierarchical sort emphasizes the same relationship qualities of Tree Tables, while the Generational sort emphasizes the generational depth in a similar way to Tree Diagrams.

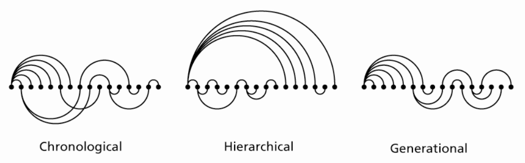


**Figure 22.** Comparison of the same thread with different message sorting techniques.

## 8. Conclusion

This paper describes Thread Arcs, a visualization technique that can display chronological and relationship properties simultaneously. Thread Arcs show the "reply to" relationship with an emphasis on the chronology of messages. The relationships between messages are shown with arcs that connect each message to its parent. By laying out the message nodes in a linear fashion, the visualization not only emphasizes chronology, but also renders it 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 the size of a thread and the number of responses to any specific message in it at a glance.

There are a number of attribute highlighting schemes that can be applied to Thread Arcs to 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 showing chronology in email thread visualizations. The study also showed that Thread Arcs are well suited for the size and structure of conversations found in users' real email.

Thread Arcs provide a unique tool for interpreting email threads by elucidating the context of each message and by offering insight to the structure and evolution of email conversations. Thread Arcs illustrate these complex relationships in an elegant and concise form.

## 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

BARLO, T. AND NEVILLE, P. 2001 A Comparison of 2-D Visualizations of Hierarchies, *Proceedings of the IEEE Symposium on Information Visualization.*

BELLOTTI, V., DUCHENEAUT, N., HOWARD, M., SMITH, I. 2002 Taskmaster: recasting email as task management, *Workshop: Redesigning Email for the 21st Century, CSCW.*

DONATH, J., KARAHALIOS, K., VIEGAS, F. 1999, Visualizing Conversation, *Proceedings of the Hawaii Internationals Conference on System Sciences 32.*

FISHER, D AND MOODY, P. 2001, Studies of Automated Collection of Email Records. *University of Irvine, Technical Report, UCI-ISR-02-4*

ROHALL, S.L., GRUEN D., MOODY P., AND KELLERMAN S. 2001, Email Visualizations to Aid Communications, Late Breaking, Hot Topic *Proceedings of the IEEE Symposium on Information Visualization*, San Diego, CA, pp. 12-15.

SACK, W. 2000, Conversation Map: A Content-Based Usenet Newsgroup Browser, *Proceedings of IUI'00*, New Orleans, LA, pp. 233-240.

SMITH, M. A., FIORE, A. T. 2001, Visualization Components for Persistent Conversations, *Proceedings of CHI 01*, AMC Press, pp. 136-143

VENOLIA, G. AND NEUSTAEDTER, C. 2003, Understanding Sequence and Reply Relationships within Email Conversations: A Mixed-Model Visualization, *Proceedings of CHI*, pp. 361-368

WATTENBERG, M. 2002, Arc Diagrams: Visualizing Structure in Strings, *Proceedings of the IEEE Symposium on Information Visualization*, Boston, MA, pp. 110-116.

WHITTAKER, S. AND SIDNER C. 1996, Email Overload: Exploring Personal Information Management of Email, *Proceedings of CHI'96*, Vancouver, B.C., pp. 276-283.

ZAWINSKI, J. 2002, http://www.jwz.org/doc/threading.html.