A Unified Abstraction for Messaging on the Semantic Web

Dennis Quan, Karun Bakshi, David R. Karger
MIT AI Laboratory/LCS, 200 Technology Square, Cambridge, MA 02139 USA
{dquan,karunb,karger}@ai.mit.edu

1. OVERVIEW

Since its inception, the Internet has been a hotbed of several successful communications channels, starting off with e-mail, IRC and newsgroups and more recently adding web annotation, instant messaging (IM), and news feeds. Although these channels were developed fairly independently, in many cases their respective functionalities and uses have grown to overlap significantly. For instance, users often have separate identifiers for e-mail, chat, and instant messaging, and clients for these systems all have their own implementations of threaded message views. Furthermore, multiple modalities may be involved in the completion of a single task: to notify a friend that you are going to be out for the day, you may start off using IM but switch to e-mail if he or she is not online. We believe these peculiarities stem from a lack of a common data model and user interface.

The opportunity exists to take a "big picture" look at the situation and to recast the problem in terms of a broader messaging abstraction. Once the existing systems are unified under a common model, we can realize a number of synergies that result from the reduction of overlap and the fine-grained control users are given over message composition, transmission, storage and retrieval. We can also enhance all forms of messaging by incorporating features that are currently present only for specific messaging paradigms. In this paper we use basic concepts from the Semantic Web to unify and model these seemingly disparate messaging paradigms.

2. EXISTING MESSAGING SYSTEMS

Our messaging model is built upon generalizations of the major Internet messaging systems. One recurring theme is synchronicity. In asynchronous communication, the sender does not wait for a response, conversations are generally carried out over longer periods of time, and each party has the luxury of formulating a well thought out response. On the other hand, users exchange information relatively rapidly in synchronous communication to facilitate an active dialog. Communication paradigms may also be grouped based on whether they support public access and dissemination of information to *a priori* unknown recipients, or whether they are intended for private communication where the participants are known and can be selected.

Previous work has uncovered the core strengths and limitations of each system. E-mail, serving as a generalized asynchronous communication mechanism for social interaction and work-related collaboration, is perhaps the most widely used mode of digital communication. Whittaker et al. report that e-mail has evolved from an asynchronous communication mode to a focal point for task management and information organization simply because it serves as a mechanism for assigning and tracking work, as well as a receptacle of various kinds of information [5]. Furthermore, it makes available a single convenient, accessible, long-term archiving mechanism allowing easy filing for items in the inbox, or letting the inbox itself be the archive.

Like e-mail, newsgroups also afford automatic persistence, which enables knowledge to be captured for future reference but also allows extraneous information to add "noise" to the information environment, making it difficult to obtain and attend to important information [3]. As present, newsgroup users resort to e-mail as a means of privately continuing conversations that start as public posts. Whittaker argues that newsgroups should allow users to easily change the communication mode (e.g. instant messaging when answers to urgent queries are needed) [3].

Finally, although the individual messages themselves may be short, immediate and rarely persisted, IM enables maintenance of longer term sessions that allow awareness of presence of other parties, thereby facilitating longer term context maintenance and allowing continuation of the conversation [2]. However, in synchronous systems such as IM, there is no way to ensure that a response appears in the right context since the display is a temporal sequence rather than a topical hierarchy [3].

3. APPROACH

In attempting to unify messaging, we have developed a robust infrastructure that rests on a well-defined ontology for messaging based on the Resource Description Framework (RDF). This robust infrastructure in turn facilitates addressing many of the UI problems and overlaps that exist. To realize our RDF data model, we are building support for unified messaging into Haystack, an information management platform for the Semantic Web [1].

In order to apply RDF to the problem at hand, we give ontological specifications of how to represent messages, conversations, and people using RDF Schema. These representations generalize the notions of sender, recipient and reply threads and form the basis of our messaging data model. This model allows us to aggregate arbitrary types of messages, thereby supporting the types of medium interchanges people often make (e.g. switching from a public post to a private e-mail discussion), while at the same time capturing the entire conversation in order to maintain message context that is so crucial in activities such as task management [5] [2]. Furthermore, by casting messages and conversations into RDF, we also gain a persistent description to which we can add additional metadata that will improve searches and other information retrieval techniques. Hence, users will be better able to reduce "noise" and manage information overload by being willing to file information and not worrying about not being able to find it later. Finally, a unified messaging paradigm implies that all information is collocated and hence conveniently accessible, which is crucial from a usability perspective [5].

Another problem with messaging today is that user interfaces are not equipped to handle the huge volumes of messages that are often encountered. By creating higher-level organizational concepts such as conversations, we are able to consolidate messages

WWW 2003, May 20-24, 2003, Budapest, Hungary.
ACM xxx.

with similar topics together and reduce the clutter in users' inboxes, while giving users more intuitive ways to navigate through their messages.

4. UNIFYING THE DATA MODEL

Our ontology is designed specifically to work with existing messaging systems. As a result, the base of our messaging infrastructure consists of a series of drivers capable of sending and receiving messages over protocols such as POP3, SMTP, and Jabber. In our system, messaging drivers are described as having type `msg:MessageSendService`. Messaging drivers are also responsible for emulating functionality that is normally not available in the underlying protocol.

Each messaging protocol currently maintains its own address scheme. For example, SMTP servers are programmed to route e-mail messages according to recipients' e-mail addresses. These addresses are represented directly in our ontology as URIs, e.g., `mailto:karunb@ai.mit.edu`. We specify a base class called `msg:Address` that represents addresses handled by messaging drivers. This allows the system to recognize resources as being addresses without relying on the syntactic form of a resource's URI (e.g., whether the URI starts with "mailto:"). We then derive classes such as `msg:EmailAddress` and stipulate that e-mail address resources be asserted to have this type.

Our unified messaging ontology necessarily distinguishes between identity and address since the same person may have a different address for message delivery depending on the message type. People are represented directly by the `hs:Person` class. Recipients and senders are specified by instances of the `msg:AddressSpecification` class. Address specifications can specify either a specific address or a person resource. Addresses can be associated with people using the `msg:hasAddress` property. Furthermore, some protocols support the notion of presence, allowing users to tell when their contacts are online. We model this notion by annotating `msg:Address`'s with the `msg:onlineStatus` property.

In order to incorporate the various forms of messaging available, we define the class `msg:Message` in a very general manner. In our system a message is a unit of expressive communication transported from one or more senders to one or more recipients. This definition allows us to unify the concepts of instant messages, e-mails, newsgroup postings, annotations, chat, and even articles delivered via news feeds.

Built up from messages are higher level aggregations that model patterns of communication: threads and conversations. Threads typically indicate a stream of messages on a very specific topic. We indicate threading by the `msg:thread` property, linking a message to a `msg:Thread` object. Threading is not directly supported by most e-mail protocols, but one heuristic for constructing `msg:Thread` objects is to reconstruct threads based on `msg:inReplyTo` connections corresponding to the "in reply to" relationships present in e-mail and newsgroups.

Conversations consist of heterogeneous collections of messages, but the connection between messages in a conversation tends to be more loosely defined by a more generalized topic than those in a thread. Conversations in our ontology have type `msg:Discussion`. Conversations also maintain state in order to help facilitate changes in the interaction, e.g., a record of the current participants, whether the conversation is currently public or private and synchronous or asynchronous.

5. EXAMPLE SCENARIO

The scenario in our poster features a dialog between John and Mary coordinating the completion of a report using Haystack.

1. Mary notices that a key report is almost due and types up a message to John telling him to finish the report.
2. She receives a reply in which John has asked her about what is pending. She realizes that the conversation might require several turns and decides to start tracking the conversation by selecting "View Discussion" from the context menu.
3. In the conversation view she notices that John is online and changes the conversation mode to instant. She then has a short dialog with John explaining what needs to be finished.
4. Later Mary receives a message from John saying that an important chart is missing. She begins typing a response when she realizes that she needs to CC her secretary. Mary clicks "Compose message full screen" in order to take advantage of the full screen message editor, which includes the advanced functionality (e.g., CC) she needs for this particular message.
5. The next day she checks the conversation again and finds a meeting invitation sent by her secretary. Unfortunately, she is running late and sends John an IM concerning her tardiness.

The example conversation interleaves both synchronous (IMs) and asynchronous (e-mail) communication. Users can select the synchronicity when composing new messages by means of the Conversation Mode widgets. Also of note is the way messages of different types—here, textual and machine-processable structured messages (e.g., meeting requests, invitations, etc.)—can be combined within the same conversation. Finally, our system supports viewing the same conversation in a variety of presentation styles, e.g., threaded messages, reply graphs, etc.

6. FUTURE WORK

Although we have asserted a unified messaging model to be useful, user studies are required to understand whether users prefer a unified user interface or would rather keep the distinction present in current messaging paradigms. Also, we hope to further exploit RDF's capability to encode arbitrary metadata in order to realize other benefits of the Semantic Web. For example, if a piece of information being sent has already been characterized on the sender's end, the recipients' systems may be able to automatically file the information into the proper categories or process requests embedded within the information, such as meeting invitations.

7. ACKNOWLEDGMENTS

This work was supported by the MIT-NTT collaboration, the MIT Oxygen project, a Packard Foundation fellowship, and IBM.

8. REFERENCES

[1] Huynh, D., Karger, D., and Quan, D. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. Semantic Web Workshop, WWW2002.

[2] Nardi, B., Whittaker, S., and Bradner, E. Interaction and Outeraction: Instant Messaging in Action. Proceedings of CSCW '00.

[3] Smith, M., Cadiz, J., and Burkhalter, B. Conversation Trees and Threaded Chats. Proceedings of CSCW '00.

[4] Whittaker, S. Talking to Strangers: An evaluation of the Factors Affecting Electronic Collaboration. Proceedings of CSCW 1996.

[5] Whittaker, S. and Sidner, C. E-mail Overload: Exploring Personal Information Management of E-mail. Proceedings of CHI 96: Human Factors in Computing Systems.