Information Systems Frontiers 5:1, 15–28, 2003

Shock: Aggregating Information While Preserving Privacy

Eytan Adar and Rajan Lukose
Information Dynamics Lab, HP Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA
E-mail: eytan.adar@hp.com
E-mail: rajan.lukose@hp.com

Caesar Sengupta
Encentuate Pte. Ltd., 151 North Buona Vista Road #02-45, Singapore 139347, Republic of Singapore
E-mail: caesars@cs.stanford.edu

Josh Tyler
Information Dynamics Lab, HP Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA
E-mail: jtyler@cs.stanford.edu

Nathaniel Good
School of Information Management and Systems, UC Berkeley, 102 South Hall, Berkeley, CA 94720
E-mail: ngood@sims.berkeley.edu

Abstract. *An important problem facing large, distributed organizations is the efficient management and distribution of information, knowledge, and expertise. In this paper we present the design and implementation of a low-cost, extensible, flexible, and dynamic peer-to-peer (P2P) knowledge network that helps address this problem. This system, known as* Shock*, is designed to protect the privacy of user's personal information, such as email, web browsing habits, etc., while making that information available for knowledge management applications. It reduces participation costs for such applications as expert-finding, allows highly targeted messaging, and enables novel kinds of* ad hoc *conversation and anonymous messaging. The system is tightly integrated with users' email clients, taking advantage of email as habitat.*

Key Words. *privacy, peer-to-peer networks, expertise location, anonymous messaging, recommendation systems, collaborative filtering, knowledge management, computer-supported cooperative work*

1. Introduction

The pervasiveness of information technology in the work practices of organizations has raised two issues that are in tension with each other. The first issue arises out of the recognition that work in many organizations is information and knowledge intensive and that much of an organization's continuing value resides in its so-called intellectual capital or knowledge assets. As a result, knowledge management applications such as expertise location (Streeter and Lochbaum, 1988), capturing an organization's memory (Ackerman and Halverson, 1998), etc., have received much attention. The effectiveness of these applications is in part determined by the information made available to them. An organization interested in maximizing the value of its knowledge assets must make as much information as possible available for knowledge management applications.

The second issue is the workplace privacy of users. With so much of our work practice being in the form of email, web browsing, instant messaging, etc. the possibilities for easy electronic monitoring are manifold. While employee privacy rights in the United States are limited (though still uncertain (Lewis, 2001)), European laws are much more strict in this regard and constrain what information an employer may gather (Brittenden, http://www.ictur.labournet.org/Online.htm). Regardless of the legal implications, employee-company relationships and employee morale depend

on a company's privacy policy (Weisband and Reinig, 1995)

From the point of view of knowledge management, however, a user's electronic trails hold great potential value. For example, information about keywords or phrases in a user's email, what web-based resources and documents they access, and even with whom they communicate and how frequently they do it can aid in the location of expertise and implicit knowledge within the organization.

Thus, in order for knowledge management solutions to be effective, organizations must balance these two seemingly conflicting demands. On the one hand they must have as much information as possible available to them, especially user specific data closely associated with work practice. On the other hand the organization must provide users with a solution they can be comfortable with, and make allowances for privacy concerns. One potential solution is to ask users to screen the information themselves. This solution has very high participation costs for users. A much better solution would be to have a system designed to protect privacy from the beginning, eliminating the need for explicit screening, and as a result, radically reducing participation costs.

It is worth emphasizing that with respect to privacy, we are not here concerned with protecting the privacy of users from monitoring by those who administer the infrastructure. This is not possible in practice. Instead, we are concerned with protecting the privacy of users by making use of (potentially private) information without revealing that information to other users in the organization or to a centralized store of data. As previous research has argued (Ackerman, 1994; Grinter, 1997), collaborative systems must pay heed to the needs and concerns of users if they are to be used, and so we are attempting to make users feel comfortable, through a technological solution, that their privacy is protected.

In this paper, we will describe the architecture and implementation of a system called Shock (Social Harvesting of Community Knowledge) that takes as one of its primary design goals the protection of user privacy while retaining the power afforded by local observation of users. Shock is client software, downloaded onto users' computers, that automatically forms a detailed profile of a user's behavior and environment and stores it locally on the user's computer. The clients are able to talk to each other in a decentralized manner through a peer-to-peer network architecture. The resulting system forms a low-cost, extensible, flexible, and dynamic knowledge network within an organization that allows users to find others with specific expertise, send highly targeted messages, and engage in novel kinds of *ad hoc* conversations with the option of true anonymity.

We will argue that Shock improves upon many prior expert-finding systems by increasing the detail and richness of data available to the system and by reducing participation costs through automatic profiling. Both of these benefits derive from the privacy-preserving peer-to-peer architecture since users would be reluctant to allow access to their detailed information or automatic profiling without it. In addition, Shock's architecture allows some novel features such as truly anonymous and secure questions, responses, and conversations. Another important feature of our implementation is tight integration with our user base's current tools, specifically the Microsoft Outlook client, which further reduces participation costs by taking advantage of how users treat their email client as their habitat (Ducheneaut and Bellotti, 2001).

In the next section, we discuss related work in more detail and show how Shock compares to other systems. We then present a system walk-through, followed by a presentation of the design rationale, and description of the main system components. Next, we present a preliminary review of the system based on a large pilot test currently underway, and we conclude with a summary and directions for future work.

2. Related Work

All expert-finding systems must operate by taking a description of the expertise sought and matching it against profile information associated with possible experts. Expert-finding systems most often differ in the kinds of information available for the experts' profiles and the way that profile information is gathered. These choices strongly impact user participation costs and the richness of available data for profile generation.

Most prior systems have a centralized architecture, in which profile storage and matching occurs at central servers. For example, expert databases, such as Microsoft's SPUD (Davenport and Prusak, 1998), HP's CONNEX[1] and SAGE[2] contain repositories of manually entered expertise data. While providing a repository of expert knowledge, manual entry requires constant maintenance and overhead to maintain usefulness. In addition, these databases generally

contain knowledge that is too broad to answer the specific queries that are frequently used in expert finding searches.

Automated expert finding systems such as Ackerman's Answer Garden (Ackerman and Malone, 1990), ER (McDonald and Ackerman, 2000), P2Pq,[3] Askme,[4] Tacit,[5] MIT's Expert Finder (Vivacqua and Lieberman, 2000), and also MITRE's Expert Finder (Mattox, Maybury, and Morey, 1998), are solutions to the high costs of maintaining expert databases. Such systems typically use some algorithm to classify information and/or expertise and then distribute this to the users seeking answers. Despite their automation, expert finding systems tend to limit themselves to mining very specific data sets. For example, MIT's Expert Finder recommends Java-programming experts based on profiles created from the users source code by comparing Java class usage to an external model. Similarly, the commercial Tacit product mines publicly available documents and e-mail available on centralized servers (Microsoft Exchange, for example). Referral Web (Kautz, Selman, and Shah, 1997) utilizes the "six degrees of separation" phenomena to capitalize on existing social networks inside of an organization to discover people who are experts and the paths of people to them by mining co-authorship graphs for published papers.

Each of these automated systems tends to use information for profile generation that is already available within the organization. For example, the ER system used data from the host organization's software development system and the call center help system. Systems that go further by using more personal data, such as email (e.g. Tacit), must address sensitive privacy concerns (Schwartz and Wood, 1992), usually through explicit user screening of published data. This explicit screening entails higher user participation costs. Furthermore, since they are centralized systems, they do not have access to other personal data such as web browsing or file viewing habits, which are easily had at the client.

Centralized expert systems also tend to remove control from the user about when they should be contacted. While systems such as P2Pq act as question brokers and route questions to users, the more traditional systems such as MITRE's Expert Finder and CONNEX simply list which experts a user can communicate with.

One of the main benefits of modern knowledge management systems over other solutions such as public mailing lists and newsgroups is the reduction of information overload. The attempt to connect individuals to the right information or information source is largely the subject of recommender systems (Konstan et al., 1997; Terveen et al., 1997; Shardanand and Maes, 1995). Our own system draws inspiration from the techniques used in recommender systems to reduce disruptions and increase productivity for individuals. Recommender systems have successfully employed various metrics to determine the information most relevant for a given user. For example, link analysis (Terveen et al., 1997), explicit (Konstan et al., 1997; Shardanand and Maes, 1995) and implicit (Morita and Shinoda, 1994) recommendations. While they provide valuable lessons, the specific techniques of such systems traditionally depend on public participation, active communities and sustained interest in a given topic to alleviate problems associated with data sparsity and bootstrapping. In addition, recommender systems utilize a centralized repository and thus are not compatible with our approach. Recent work in collaborative filtering and e-commerce looks into solutions to privacy issues (Ackerman, Cranor, and Reagle, 1999; Canny, 2002), but has yet to be implemented and tested in live systems.

Shock is similar to decentralized systems such as DEMOIR (Yimam, 2000) and Yenta (Foner, 1997). Although privacy concerns are addressed and incorporated into their architectures, they are fundamentally different from Shock in their design. Yenta (Foner, 1997) is a system created primarily for a passive user to gain value from a distributed network of users, by implicitly determining what items that user may find interesting. DEMOIR (Yimam, 2000) is a hybrid architecture proposal that allows users to share information, but lacks the ability to transfer and target expertise in a P2P manner or provide a level of privacy control and guarantees that Shock provides. For example, the proposed DEMOIR architecture does not make explicit allowances for anonymous questions and answers.

Finally, the majority of the systems mentioned require use of a secondary user interface. Shock, by contrast, is embedded in the user's email client, and provides rapid integration into existing tools and knowledge bases within an organization. Prior work has shown that email is a habitat for most office workers, and in designing Shock we sought to incorporate the tools for finding expert knowledge into the tools that people use on a daily basis.

While the prior work addresses pieces of the larger problem of finding experts to solve problems, we

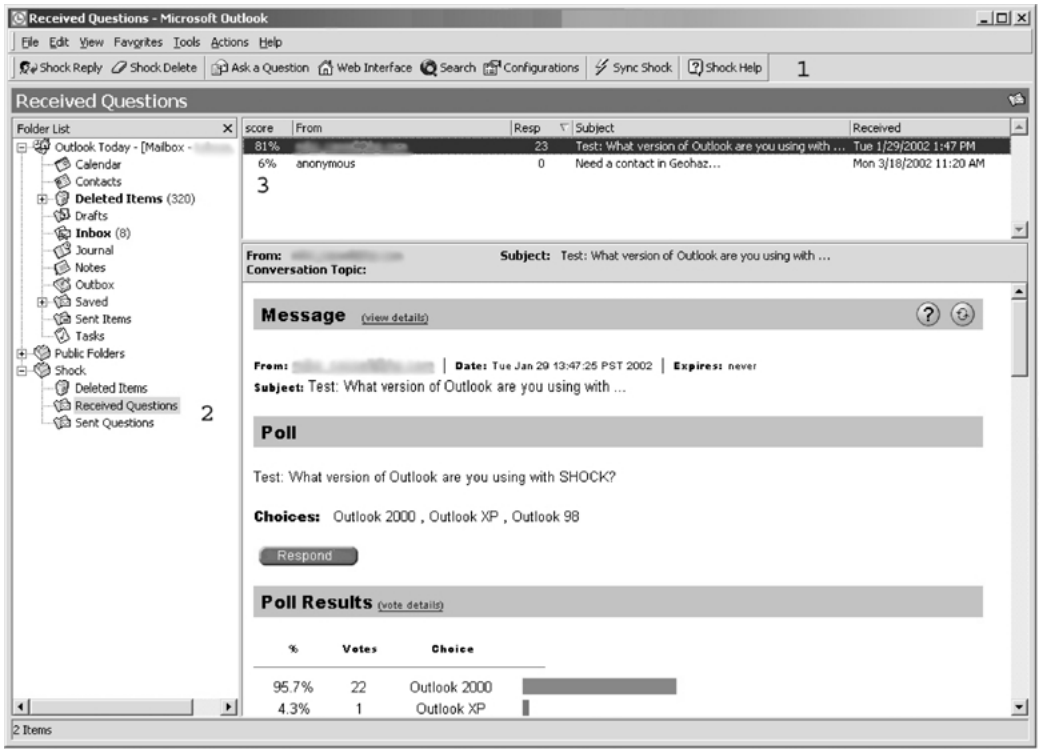

Fig. 1. *Outlook integration.*

believe that no system exists that preserves privacy, provides targeted profiling while being cost effective and simple to maintain.

3. *Usage Scenario (Walk-through)*

To demonstrate the Shock system, we present a walk-through of a likely scenario. A user, Alice, wishes to find out the experiences of others in her organization with the peer-to-peer system called Freenet. Fig. 1 shows how her Outlook client looks with Shock installed. A Shock toolbar, labeled (1) in Fig. 1, appears below the standard Outlook toolbar. In the left pane is a special folder labeled "Shock" (2) that contains sent and received questions. In this view, a received question is selected for viewing, and the message view pane has a column labeled "score" (3) which shows the relevance of the received questions. By asking a Shock question, Alice will be starting a conversation that will appear in her Sent Questions folder (2).

The current system allows many different kinds of questions, each with different features enabled in the interface (the implementation allows easy extensibility and flexibility for defining new types). After clicking the "Ask A Question" toolbar item, she selects a "Survey/Poll" type and is presented with the screen shown in Fig. 2. She enters a descriptive question (1), defines her multiple choices (2), allows responders the ability to reply with text comments in addition to their poll choice (3), decides not to be anonymous (4), and sets the message to expire in one week (5). In addition, she decides to target her message by including a "Filter" in it. Filters, which we discuss in detail later, allow the sender of a Shock message to (optionally) specify highly detailed criteria to help target the message. She decides that only users who have "Freenet" installed should receive her message (6).

Clicking on the send button results in the message being sent over the Shock network according to a protocol we describe in Section 4.2. Other Shock clients on the network score the message (described in Section 4.4). The score will depend on the text of the question as well as the filters. Each client has a threshold value, set by the user, which Shock message scores must exceed in order to be presented in the interface.

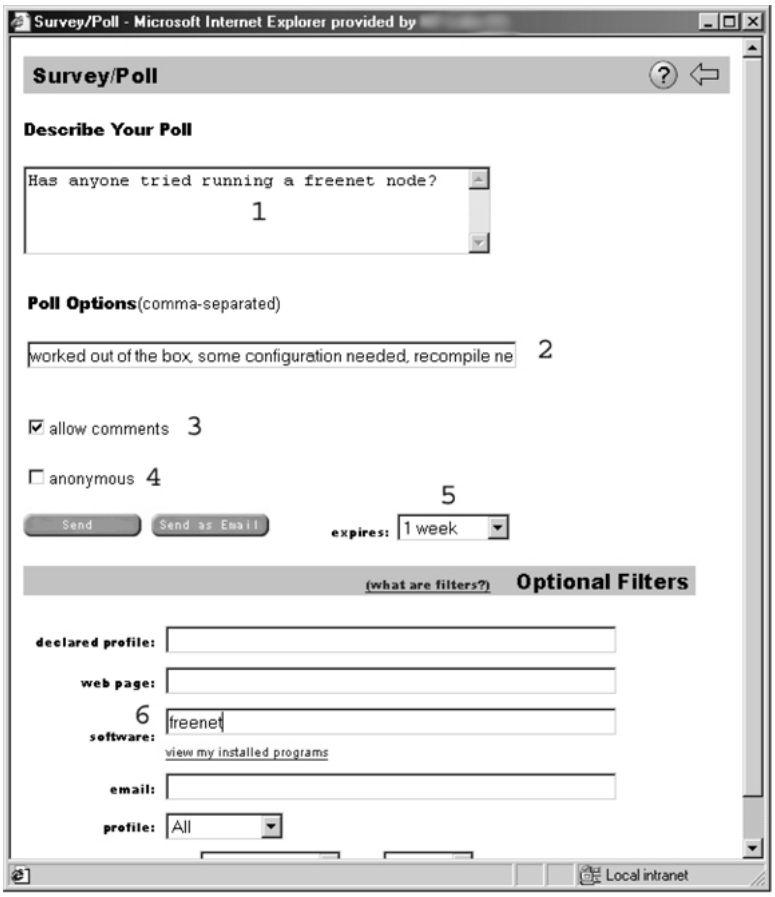

Fig. 2. *Asking a Shock question with a poll and a software filter.*

If a user, Bob, is presented a message, it appears as an Outlook item in the Received Questions folder (Fig. 1, label 2). The user can respond to the message through the interface shown in Fig. 3. Here the user selects an item from the list, enters optional comments, decides if the response should be anonymous, and also whether the reply is to be encrypted so that only the sender can see it. Note that the sender, Alice, will never know that the question, with its filters, was presented to Bob unless a reply is made and Bob chooses to explicitly divulge his identity in that reply. Back at Alice's client, we see a partial view of the results of several responses (Fig. 4). This dynamically generated page appears when she clicks on the corresponding Outlook item in her "Sent Questions" folder. Notice the graphical poll summary and the threaded conversation. To facilitate conversations, all users who do not filter the initial message will have this view and can participate in the discussion (excepting those messages which are encrypted for someone else).

4. System Architecture

The Shock architecture was designed to simultaneously address various (and traditionally conflicting) design principles. Specifically, clients were to be easily installed, maintained and used, privacy and anonymity were to be respected, automation was used where possible, and flexibility for additional features was built in. The most likely candidate that satisfied these various concerns was a P2P or P2P/server hybrid system where the bulk of work was done on each client.

This design allows Shock clients to automatically collect highly detailed information from the user's behavior while still maintaining privacy for that profile

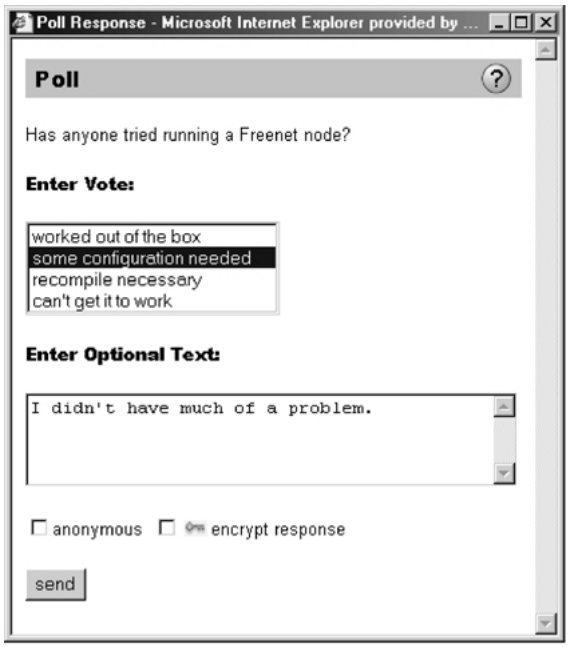

Fig. 3. *Responding to a Shock question with a poll.*

through local storage. That is, all data collected is as secure as the source data. Anonymity is optionally provided to users through the decentralized network topology and randomized laundering of messages. This is similar to the way Gnutella[6] and Crowds (Reiter and Rubin, 1999) operate and essentially implements an anonymous bulletin board.

Fig. 5 abstractly illustrates a Shock client and the Shock network. All user interactions with the system are done through a *UI* module that includes both a web style interface as well as a Microsoft Outlook interface. The client serves the role of generating user profiles though *Observer* and *Bootstrapping* modules. These automate the process of building the user profile through indexing documents the user interacts with and cataloging other facts that are known about them or their systems (installed programs, for example). Additionally, the Shock client determines which incoming messages are to be shown to the user through a *Scoring* module. The *Network* module serves to route questions to and from other peers as well as interacting with the Shock message server. Each module is described further below.

The client software is primarily written in Java with some code in Visual Basic (VB) to interface to Microsoft Outlook and Internet Explorer. While currently being tested on Windows OS machines, we expect to port the software to other systems in the future.

4.1. System UI

In keeping with Shock's design goal of encouraging usage by reducing participation cost, Shock's user interface has been designed to seamlessly integrate into a normal user's work practice. The present Shock user interface contains functions for generating and responding to messages as well as general controls for Shock.

Since one of Shock's key functions is messaging, we decided to integrate Shock with the most common messaging platform being used by Shock's target demography, Microsoft Outlook. Therefore, the present version of Shock adds a special folder called 'Shock' to Microsoft Outlook (Fig. 1). All of Shock's messages are stored in this folder and this folder behaves like a regular Outlook folder (i.e. the user can view, sort, or delete messages in the same way). As mentioned earlier in the walkthrough, this folder contains two sub folders called "Sent Questions" and "Received Questions". The messages in the Received Questions contain an additional property called "Score" which indicates the relevance of the particular question for that user. (Scoring will be described in detail in Section 4.4).

Shock messages are inherently threaded in nature and hence each instance in the Shock folder is in reality a message thread. When the message is opened, it displays the complete discussion thread for that message (Fig. 4). This view is similar to that of a web-based news group. In practice, we have found this hybrid, email-newsgroup message representation model to be highly suited to Shock's nature.

Shock adds a special toolbar called "Shock" to Outlook (Fig. 1). This toolbar allows the user to easily send and reply to Shock and to change the various configuration options. Like many other Microsoft Windows products, Shock also presents an icon to the user in the System Tray. This icon changes to notify the user of new messages and can be used to access a special menu, which gives the user quick access to most Shock configuration parameters. Shock also provides a browser-based interface that can be used by users who prefer not to use the Outlook interface.

The use of these established and familiar models in Shock gives the novice user many affordances that help reduce the learning curve involved in using a new system and hence encourages usage.

In order to provide an interface between users who have Shock installed and those that do not, the system

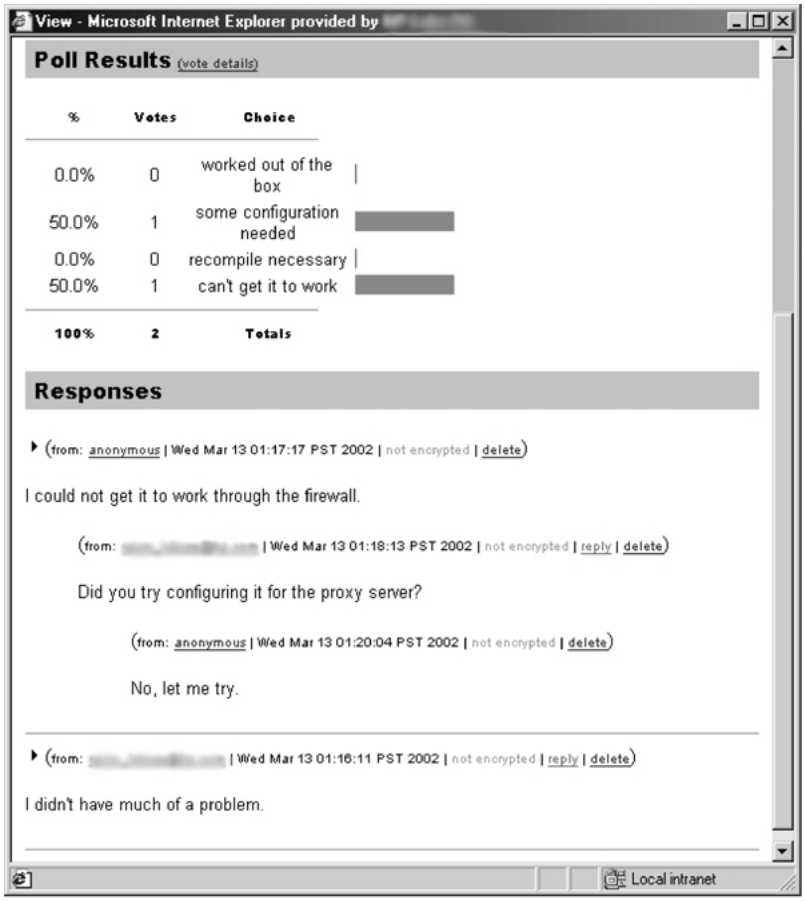

Fig. 4. *Partial view of a conversation summary for a Shock question. The summary dynamically tallies the associated poll, and shows the threaded group conversation.*

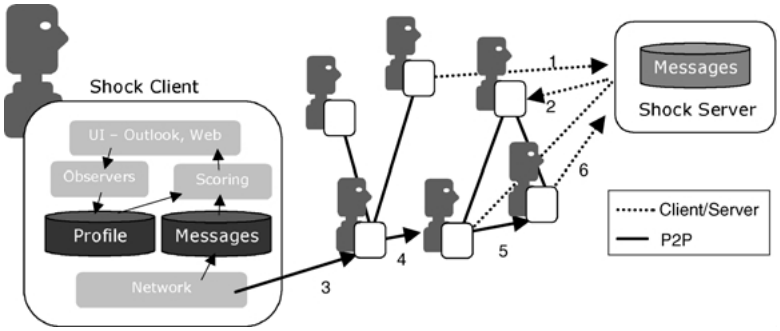

Fig. 5. *The Shock architecture.*

allows users to send Shock messages as email. Next to the "send" button in the interface is a "Send as Email" button (see Fig. 2). A Shock message sent this way is embedded in an email message that can be sent to any user. When the message arrives at a user's mail client, those with Shock installed will have the message automatically filtered and moved to the appropriate Shock folder. Users who do not have Shock installed will be

presented with a regular email message that contains the question and a link to the Shock install website. This feature encourages the growth of the user base, and can also facilitate the explicit creation of groups since only users who received the first message will be aware of and be able to participate in that message thread. In addition, this feature, combined with the robust profiling mechanism, offers a flexible implementation of computational email (Borenstein, 1992).

4.2. Shock network architecture

As discussed above, maintaining privacy and anonymity is a cornerstone of the Shock system design. In order to achieve these properties, Shock can function as a purely decentralized P2P system. However, we have created a hybrid design that extends the design and provides these same features while providing scalability and persistence for messages.

4.2.1. Peer-to-peer networking. In a peer-to-peer network, each node connects directly to other nodes; there is no central server. The most well known examples of true P2P networks are the file-sharing programs Gnutella[7] and Freenet (Clarke et al., 2000). In a typical P2P system, when a client is added to the network, it connects randomly to other peers. Messages are then sent through these links, eventually propagating to the other peers through a series of "hops" from peer to peer. In Shock, we randomize the way messages move from one peer to the next, making it practically impossible to detect a message's origin or destination. This randomization is similar to technique used by the Crowds system for anonymous web browsing (Reiter and Rubin, 1999).

In addition, a peer-to-peer network has the advantage that it has no single point of failure. If a client's connection client is terminated, the client will find a new client to which it can connect. This constantly adapting network topology ensures that an organization's knowledge base is never offline, and accommodates a mobile and dynamic organization.

Peer-to-peer networks have some limitations, however. In particular, they typically have difficulty scaling effectively to large numbers of users (Yang and Garcia-Molina, 2001). The inherent message redundancy and multiple retransmissions of each message (one for each "hop") add network overhead. The second problem with such a pure P2P model is the lack of message persistence. The network lacks the ability to remember messages that were sent and received, and as clients enter and leave the network, information is lost. While Shock can function in this mode, the preferred method is through the hybrid model illustrated in Fig. 5 and described in the following section.

4.2.2. Implementation details. Shock clients maintain both links to each other (typically 3-5 connections) as well as knowledge about a centralized server, that acts as a simple message store. Periodically clients can retrieve information from the server (Fig. 5, link 2). Messages arrive at the server in one of two ways. The first is through non-anonymous message transmission. In this mode the user sends a non-anonymous message and will simply deliver the message (1) to the server. The second method of delivery leverages the P2P aspect of the system to deliver anonymous messages. In this mode a client sending a message will pass the message to a randomly chosen neighbor client (3). That neighbor will randomly choose to transmit the message to the server or to another neighbor. The message is then passed through a number of clients (4 and 5), finally ending up at the server (6). Note that neither the intermediate clients nor the final server is able to determine from where the message originated. The initial client may query the server to ensure that the message arrived (if not, it can be resent).

For additional reliability, in the case where the server is not part of the shock network for some reason (e.g. it has crashed or is being rebooted), the clients will switch to operating in a purely P2P mode and the system will function as before.

4.2.3. Security. A final feature of the network architecture is the ability to provide private (and anonymous) communication between two clients (although this can be scaled to groups). To provide this feature, Shock automatically generates a public/private key pair for each new message. The public key is included with the message, and the private key is retained on the client's machine. When another user desires to send a secure response to a message, the response is encrypted with the original message's public key, so that it can only be decrypted by the owner of the message's private key, namely, the creator of the message. Key pairs are not reused so that a message sender cannot be identified by the user's public key, thus preserving anonymity.

By using both the encryption and message anonymization features, Shock uniquely provides support for secure, anonymous interactions. The public-key encryption scheme enables each party to

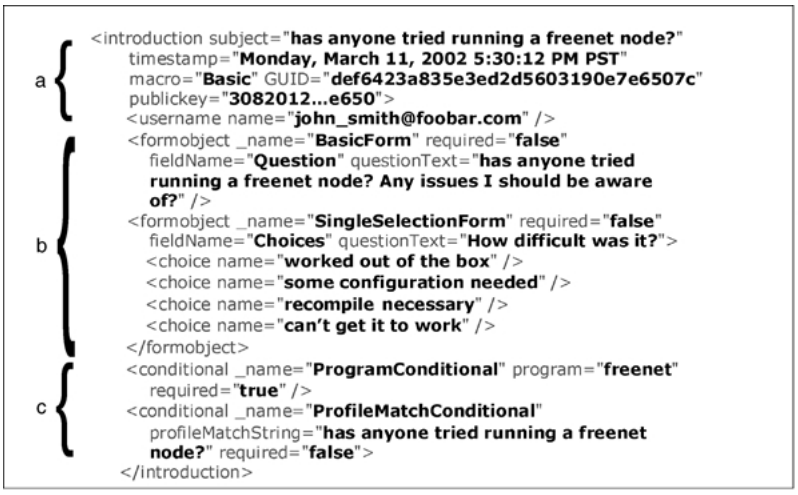

Fig. 6. *A sample message.*

guarantee that messages are sent only to the intended recipient, who can remain anonymous nevertheless. Thus, the two parties can communicate back and forth without revealing their identities. This feature helps complete the system's ability to allow the full spectrum of possibilities for conversations, from fully identified to completely anonymous and private.

4.2.4. Message format. There are two primary message types in Shock, *Introduction* and *Response*. Introductions are sent to ask a question or start a conversation and Responses are the messages that follow. Fig. 6 illustrates a simple introduction. The message contains three types of fields: *general headers*, *conditionals*, and *form objects*. Response messages follow the same general structure but do not ordinarily contain conditionals. General headers (a) simply specify information such as the message subject, time stamp, a globally unique identifier (GUID), and an identifier for the user who sent the message (this may be "anonymous"). Messages include the previously mentioned public key in order to support private communications. Introductions may also include an expiration date (not depicted) which a user may attach to their question to indicate when to remove the question from the network. Response objects contain two additional fields indicating the GUID to which this is a response (we can have responses to responses for message threading purposes) as well as the GUID for the general thread (this must be the GUID of an Introduction message).

The form object section (b) of the message specifies the fields into which users can respond. This is analogous to an HTML form where the form programmer describes the different types of desired information. Specifically, in this instance the user is asking that a free form answer be filled out (the *BasicForm* line), and one item be selected from a list (the *SingleSelectionForm*). It is up to the interface to determine the best way to represent these fields to the user. However, in our case basic forms tend to take the form of text input boxes and single selection forms are drop down selection lists. A response object will contain a version of the form object that holds the response value.

Finally, the conditional section (c) specifies the rules on which a message should be scored and filtered. This message specifically states that the user must have the program "freenet" installed (the *ProgramConditonal*) and their profile should score high enough on the question (*ProfileMatchConditional*). It is the responsibility of the Shock profiling infrastructure to construct this profile (see Section 4.3) Additional conditionals and the exact scoring mechanism are described below (see Section 4.4).

4.3. Profiling

One of Shock's key features is its ability to generate rich user profiles. These profiles are generated by tapping multiple data points, by capturing static user data during bootstrapping, and by explicit user declarations. However, keeping these user profiles constantly

updated while ensuring that they are rich enough to encompass a user's implicit knowledge is a non-trivial task.

Shock's solution to this problem is to provide an extensible architecture for the addition of Observers and Bootstrapping modules and to allow the user to explicitly declare their interests. The primary role of the observer modules is to plug into various dynamic data sources and receptors and to tap into the data flowing through these points. Bootstrapping modules are used to capture static pieces of data that are unlikely to change often and to bootstrap the user's profile.

The current implementation of Shock includes Observers to tap into Microsoft Outlook and Internet Explorer to capture new information. In addition to these Observers, Shock provides corresponding Bootstrapping modules for pre-existing data (e.g. existing email and browsing history). Other Bootstrapping modules discover the programs installed on the user's machine and capture an employee's organizational profile (affiliations, location etc.) from the employee database. Shock also allows its users to augment their automatically generated profiles by explicitly declaring their expertise in a *Self Declared Profile*.

Initial user feedback as well as existing research (Herlocker, Konstan, and Riedl, 2000) suggested that users desire control over their profiles. However, this desire is often in direct opposition to the need to maintain unbiased profiles. Shock facilitates user control of profiles while at the same time preserving the independent and autonomous nature of the Shock profiler. Specifically, Shock allows users to (a) create a Self Declared profile, (b) turn off the profiler at will, (c) remove information from the profile through a search and delete interface, and most drastically, (d) delete their entire profile.

Privacy has been one of Shock's key design criteria. Each user's profile is stored locally on her computer and this profile never physically leaves the computer. The only information that leaves the user's local machine is through that user's explicit responses to questions that matched the profile. The choice of responding to these messages rests with the user and the ability to reply to messages anonymously ensures that users are able to participate on the Shock network while keeping their profiles completely private. These features, combined with Shock's architecture, provide the user with complete control over her profile's privacy.

4.4. Scoring and conditionals

As described above, Shock messages contain rules by which messages are scored and filtered. The user who originates the message may set conditions and, depending on the number of conditions that are met, and as a function of the independent scores, a total message score is generated. If the message exceeds a certain threshold set by the user, the message will be displayed in the user's interface.

Conditionals, which are the filter rules described above, come in two main varieties, boolean and fuzzy. Those that are boolean matches will either return a score of 1 or 0. A fuzzy conditional will return a number between 0 and 1 (inclusive). Furthermore, conditionals may be required or not required.

4.4.1. Fuzzy matching. The profile conditional is one of the most frequently used due to its versatility and ease of use. Documents that the user accesses are indexed in a full text index. The question text is then used to search the full text index for likely matches. Each matching document is then independently scored (using standard TFIDF (Salton, 1988) metrics) against the question text and the results are combined and normalized. This method attempts to model the likelihood of a user's interest in a question based on the number of matching documents as well as taking into account the user's other interests. Such a solution is necessary as the Shock clients do not have a global view and cannot compare one user absolutely to another. Additionally, Shock provides fuzzy matching against declared profiles.

4.4.2. Boolean matching. Currently Shock provides three boolean conditionals that allow targeting of users who visited specific web sites, have matching fields in the enterprise directory (department, location, etc.), and who have e-mailed a specific domain or user. These are abstractly similar in operation so we will only describe the web conditional in more detail.

Through this conditional, a user may specify which web sites or specific pages the recipient of the question should have seen. The result of this is simply a 0 or 1. Possible extensions of boolean query include allowing the asker the ability to specify that the recipient *not* have seen a website (e.g. "Please look at page x if you haven't seen it yet and tell me what you think."), and employing recency and frequency (e.g. "users who often visit web site x").

4.4.3. Scoring. The total score generated is a function that combines the score of each independent conditional. The scoring mechanism takes into account whether conditionals are required or not. If all required conditionals score above 0, the combined score is compared against the threshold (otherwise the message is filtered out since a requirement was not met).

Because of our object-oriented implementation strategy, new conditionals are easily and constantly added to the system. Additionally, we are currently experimenting with alternative scoring mechanisms including manipulation of scores not only in response to local scores but global behavior. For example, questions for which answers are observed on the network will have their score reduced (multiple users need not answer the same question). Alternatively, questions that receive no answers may have their scores boosted.

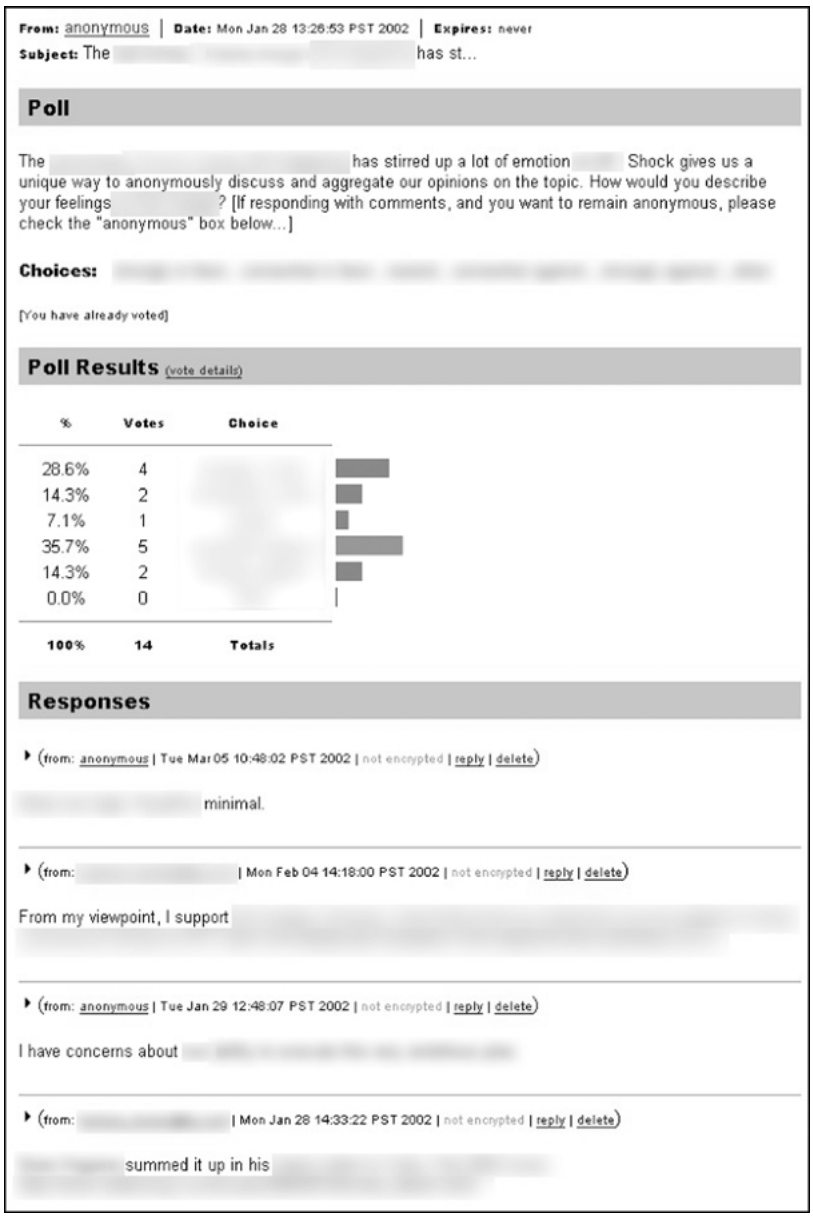

Fig. 7. *Anonymous discussion on sensitive issues.*

While Shock is highly flexible in the options available to users for filtering, we have designed the user interface to provide a more abstract view, which provides users with access to the filter fields appropriate for different tasks. For example, the "software announcement" question screen will only display the program conditional. Again, because of our object oriented implementation, new *Macros*, as we call them, can easily be constructed for specific tasks.

5. Pilot Study

At the time of this writing we are currently in the midst of a 3 month pilot study to test the usefulness of Shock. The pilot study is being conducted within a subgroup of Hewlett-Packard, that consists of security consultants, support personnel, and administrators from all over the world. A large number of people have been contacted to participate in the pilot study, from over 20 countries and at least 6 different departments worldwide.

A baseline survey indicates that the users currently use over 10 systems to access expert knowledge, including 6 in-house systems that are similar to expert databases. Despite this, the most popular methods of finding expert knowledge are through personal networks, as supported by Ackerman's field studies (McDonald and Ackerman, 1998), or by running a web search. The survey also showed that users asked questions frequently and found that access to expert information was important to their job. These results help justify the need for easy and natural approaches to finding expert knowledge.

At the conclusion of this study we hope to gain a better understanding of usage patterns, and converge on metrics to understand whether or not the pilot was successful. We are also planning to perform experiments comparing Shock to existing solutions to determine its impact on the pilot community.

It is still too early to report on how Shock is performing, but we are able to provide some evidence of Shock's usefulness. Specifically, we have seen users utilizing the unique features of Shock to achieve certain goals. For example, one user, interested in public transportation to work targeted her question to users in a specific geographical location. A user in that area received her message, and gave her advice.

Although most messages are not anonymous, preliminary observation of usage shows us that the people do use the anonymous features. As expected, the anonymous messages tend to be about more sensitive material, such as opinions on company policies and so forth. Fig. 7 shows a conversation within the company on a sensitive company issue. Shock helped users maintain a sense of security and privacy, while encouraging conversation on a sensitive topic.

In the beginning of the pilot study, it seems that Shock is accomplishing its goals of providing a private, low-cost flexible means for people within an organization to find expert knowledge and discuss sensitive issues with others in an organization without fear of reprisal or backlash. We are encouraged by our initial success, and look forward to further studies of Shock's effectiveness within the pilot group.

6. Conclusion and Future Work

Shock is a system designed to help resolve the tension between privacy and increased access to user information for knowledge management applications. It does this by employing locally stored automatic profiling for access to valuable information about users, close to their work practice. Shock allows users to make some use of the profile information of others by assembling Shock clients into a peer-to-peer network. Users can exchange a wide variety of message types and form *ad hoc* community discussions around specific topics. Shock also gives users the full ability to manage their identity by allowing truly anonymous messaging through the architecture. The resulting platform is low-cost, highly flexible and can support a variety of knowledge management applications, in a context that is tightly integrated with users' current email application.

We plan to analyze and evaluate Shock usage from the results of the ongoing pilot study. We are interested in how good this system is at uncovering hidden knowledge in an organization. We plan to explore issues of message quality and trust in anonymous or private settings, as well as the issues of public and private rewards for contributing to the network. We also hope to investigate the effects of anonymity for the facilitation of conversation inside an organization.

In the future, we plan to build upon this platform by including reputation management features, exploring other applications such as privacy-preserving collaborative filtering, as well as novel economic incentive mechanisms.

In addition, Shock may become an unobtrusive way to gather interesting data relevant for understanding the social networks within organizations, while preserving the privacy of participants.

Acknowledgments

We would like to thank Lada Adamic, Bernardo Huberman, and Marie-Jo Fremont for their ideas and encouragement.

Notes

1. http://www.carrozza.com/atwork/connex/.
2. http://sage.fiu.edu/MegaSource.htm.
3. http://www.p2pq.net.
4. http://www.askme.com.
5. http://www.tacit.com.
6. http://www.gnutelliums.com/.
7. http://www.gnutelliums.com/.

References

Ackerman M. *Augmenting the Organizational Memory: A Field Study of Answer Garden*. Irvine, CA: Department of Information and Computer Science, University of California, 1994.

Ackerman MS, Cranor LF, Reagle J, Privacy in e-commerce: examining user scenarios and privacy preferences. In: *Proceedings of the first ACM Conference on Electronic Commerce*, Denver, Colorado, United States, 1999:1–8.

Ackerman MS, Halverson C. Considering an organization's memory. In: *CSCW*, ACM, 1998.

Ackerman M, Malone T. Answer Garden: A tool for growing organizational memory. *ACM SIGOIS Bulletin* 1990;11(2):31–39.

Borenstein NS. Computational mail as network infrastructure for computer-supported cooperative work. In: *Conference Proceedings on Computer-Supported Cooperative Work*, Nov. 01–04, Toronto, Ontario, Canada, 1992:67–74.

Brittenden S. (ed.) *Online Rights—The Law in Europe.* http://www.ictur.labournet.org/Online.htm.

Canny J, Collaborative filtering with privacy. In: *IEEE Conf. on Security and Privacy*, Oakland, CA, May 2002.

Clarke I, Sandberg O, Wiley B, Hong TW. Freenet: A distributed anonymous information storage and retrieval system. In: *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

Davenport TH, Prusak L. *Working Knowledge: How Organizations Manage What They Know*. Boston, MA: Harvard Business School Press, 1998.

Ducheneaut N, Bellotti V. Email as habitat: An exploration of embedded personal information management. *Interactions* 2001;8(5):30–38.

Foner, L Yenta. A multi-agent, referral based matchmaking system. In: *Proceedings of the First International Conference on Autonomous Agents (Agents '97)*, Marina del Rey, California, United States, 1997:301–307.

Grinter RE. From workplace to development. In: *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work: The Integration Challenge*, Phoenix, Arizona, United States, 1997:231–240.

Herlocker J, Konstan J, Riedl J. Explaining collaborative filtering recommendations. In: *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work*, Dec. 2–6, 2000.

Kautz H, Selman B, Shah M, Referral Web. *Communications of the ACM* 1997;40(3):63–65.

Konstan J, Miller B, Maltz D, Herlocker J, Gordon L, Riedl J. GroupLens: Applying collaborative filtering to usenet news. *Communications of the ACM* 1997;40(3):77–87.

Lewis NA. Plan for web monitoring in courts dropped. 2001, New York Times (http://www.nytimes.com/2001/09/09/technology/09COUR.html).

Mattox D, Maybury M, Morey D. *Enterprise Expert Knowledge and Discovery*. MITRE Corporation, 1998.

McDonald D, Ackerman M. *Just Talk To Me: A Field Study of Expertise Location.* In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '98)*, 1998:315–324.

McDonald D, Ackerman M. Expertise Recommender: A Flexible Recommendation System and Architecture. In: *CSCW*, 2000:231–240.

Morita M, Shinoda Y. Information filtering based on user behavior analysis and best match text retrieval. In: *Proceedings of SIGIR Conference on Research and Development*, 1994:272–281.

Reiter M, Rubin A. Anonymous web transactions with Crowds. *Communications of the ACM* 1999;42(2):32–38.

Salton G. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1988.

Schwartz MF, Wood DCM. Discovering shared interests among people using graph analysis of global electronic mail traffic. Communications of the Association for Computing Machinery, 1992.

Shardanand U, Maes P. Social information filtering. In: *Conference Proceedings on Human Factors in Computing Systems*, Denver, Colorado, United States, 1995:210–217.

Streeter LA, Lochbaum KE. Who knows: A system based on automatic representation of semantic structure. In: *RIAO '88*. 1988, MIT, Cambridge.

Terveen L, Hill W, Amento B, McDonald D, Creter J. PHOAKS, *Communications of the ACM* 1997;40(3):59–62.

Vivacqua A, Lieberman H. Agents to assist in finding help. In: *ACM Conference on Human Factors in Computiong Systems* (CHI 2000), 2000:65–72.

Weisband SP, Reinig BA. Managing user pereceptions of email privacy, *Communications of the ACM* 1995;38(12):40–47.

Yang B, Garcia-Molina H. Comparing hybrid peer-to-peer systems. In: *Proceedings of the 27th VLDB Conference*, Roma, Italy, 2001.

Yimam D. Expert finding systems for organizations: Domain analysis and the demoir approach. In: *Beyond Knowledge Management: Sharing Expertise*, Boston: MIT Press, 2000.

Eytan Adar is a research scientist in the Information Dynamics Lab at Hewlett-Packard Laboratories, Palo Alto, CA. He holds a BS and Meng in Computer Science from the Massachussetts Institute of Technology.

Rajan Lukose is a research scientist in the Information Dynamics Lab at Hewlett-Packard Laboratories, Palo Alto, CA. He holds a PhD. in physics from Stanford University, granted in 1999.

Caesar Sengupta is a member of technical staff at Encentuate Inc. in Singapore. He holds a Masters in Computer Science with Distinction in Research from Stanford University.

Josh Tyler is a member of research staff at Hewlett-Packard Labs in Palo Alto, CA. He holds a Masters in Computer Science from Stanford with a specialization in Human-Computer Interaction and a BS in Computer Science from Washington University.

Nathaniel Good is a graduate student in SIMS at the University of California, Berkeley. He holds a BS in Computer Science from the University of Minnesota.