Thesis Proposal — Distributing Information for Collaborative Filtering on Usenet Net News

David Maltz

November 15, 1993

As part of the "Information Revolution," the amount of information available to computer users has increased as never before. Unfortunately, there has been a corresponding jump in the amount of unrelated information users must search through in order find information of interest. Harnessing the power of multiple users to form a collaborative filter may provide a robust way of helping direct users to the information that will be most useful to them. To test this idea, we will create a large scale collaborative filtering system tuned to help users extract information from Usenet Net News.

1 Background

The Usenet is a loosely organized network of heterogeneous computers stretching across all 7 continents. These computers communicate via a number of protocols such as **uucp**[7] and TCP/IP data streams. A primary use of the Usenet network is the exchange of Usenet Net News.[12] Net News functions like a huge distributed bulletin board system (BBS) such that messages created at one site are eventually seen at all sites on Usenet.

Information on Net News takes the form of *articles* written by individual users. Users enter articles into the Net News system by *posting* them. Each site which participates in Usenet Net News runs a program called a *news server* that exchanges articles with news servers at other sites using the Network News Transfer Protocol (NNTP).[4] The news server stores all the articles it receives for a number of days and deletes older articles to make room for the new. Until the articles are deleted, the news server makes them available to be read by users at the site. Users request articles from the news server using any of a number of *news reader* programs which provide an user interface for reading and browsing the articles.

To help users find articles they are interested in reading, the articles are arranged into a hierarchy of *newsgroups* which is organized by topic. To keep discussions on topic, some newsgroups are *moderated*. Any article posted to a moderated newsgroups is first sent to a human moderator who decides whether or not the article will be distributed across the Usenet.

Usenet is currently growing to an enormous size. Estimates show that there are over 2.6 million users of Net News at some 87 thousand sites throughout the world. These users generate over 26 thousand new articles a day, amounting to 57 Mbytes of data.[10] In the past users attempted to control the number of articles they had to read a day by only subscribing to newsgroups on topics they are interested in. However, the continuing increase in the number of newsgroups and the number articles posted daily has had the result that many users are presented with far more messages a day than they can possibly read. To cope with this overload of incoming data, users have adopted reading strategies for dealing with the flood, but these strategies often sacrifice any chance the user has of finding useful information in the data. In fact, one of the few things the users of Net News seem to agree on is that they need a simple way of filtering messages to find ones they will be interested in.[3]

Many filtering systems of different types have been created, but none of them have all the characteristics that are needed for filtering Net News. Most current filtering systems are designed for use by a single user. These systems attempt to extract patterns from the observed behavior of the user, and then predict which other items from the information stream will be selected or rejected based on the patterns. An example of such systems is the LyricTime project by Shoshana Loeb which compiles a profile of musical tastes for each user of an on-line jukebox.[6] LyricTime uses the user's profile to filter songs from a collection of available music. As the user accepts and rejects songs suggested by the system, the system refines the user's profile with the goal of delivering only songs which the user will approve of. Such systems suffer from a "cold-start" problem in that new users start off with nothing in their profile and must train a profile from scratch. A better system would allow new users some type of access to the experiences of current users to help create an initial profile.

A more general problem with systems that rely on user profiles is that the user can become circumscribed by their profile. The profile only selects articles similar to the ones the user has already read, and new areas which might be of interest can be missed completely. Further, if the user tries to explore areas outside the knowledge domain of the profile, the profile can provide little if any filtering ability, so the user will again face the cold-start problem and be left facing a staggering amount of data to search through manually. To support the exploration of new areas what is needed is a method of tapping the knowledge of users currently well versed in those areas.

A collaborative filtering system supports such exploratory searching by providing users with information derived from the experiences of previous users. Collaborative filtering is based on the observation that people are good editors and filters for their friends. Currently, when a user reads an article that he thinks will be of interest to a friend, he copies the article and sends it to the friend. Often the friend does not even know that the information contained in the article is available, and so could not have constructed a filter or query to extract the information directly.

To experiment with collaborative filtering, the Tapestry group at Xerox PARC developed a system for retrieving documents from a growing corpus. The system selects documents to retrieve for a user by searching for keywords present in the documents or annotations added to the documents by other readers.[5] Tapestry's ability to answer queries which take into account the opinions of other users is sufficient to enable collaborative filtering. Unfortunately, the system used by Tapestry suffers from two distinct problems which prevent it from being widely applied to Usenet Net News. First, the system requires users to specify requests for information in the form of queries in an SQL-like language. Writing such a query requires the user to have a firm sense of what types of articles he wants to read; this is a hindrance to exploration of new areas. Second, the architecture of the Tapestry system incorporates a commercial database program and can not be scaled to the size of the whole Usenet. For this thesis we propose a collaborative filtering system for Net News that does not need the full power of a commercial database or SQL.

2 Goal of the Thesis

Simply put, the goal of this thesis is to create and install a working system for collaborative filtering on the Usenet. At a minimum this system will allow users of a selected set of news readers to vote for articles on a numerical scale and thus provide future readers of the articles a way of using the already recorded votes to filter their news. The system will be scalable so that it can grow from some small number of test sites to encompass the size of the entire Usenet.

The construction of this system will pull together several elements from projects such as the World Wide Web (gopher file transfer and URL)[1, 2], NNTP (quick, robust transport) [4] and Tapestry (collaborative annotations; separate storage of user annotations on document)[5]. The evaluation of the project will yield quantitative information about how Net News users spend their time reading, and qualitative information about the users' reading behaviors and strategies. Most importantly, the project will leave a working system to help the users of Net News filter their news.

The architecture of the system should be robust against attempts to "stuff the ballot box." Ideally, the system will allow each user to cast only one vote per article. At the very least, the system will make it difficult to cast multiple votes for the same article.

Under the current Net News system only a few people can be elected to moderate a newsgroup. Our collaborative system will provide a means whereby any user who chooses to can act as a newsgroup moderator. For example, if Joe announces that he will moderate a newsgroup, other users who trust Joe's opinions will be able to make requests of the form, "Show me only the articles in this newsgroup that Joe liked." Users who do not trust Joe's evaluations will continue to see the articles in the newsgroup as usual.

We assume that there is a large class of users who would be willing to vote, but for a multitude of reasons might not want others to find out what they have voted for. For these users the collaborative system should provide a level of anonymity at least as good as the current Net News system. To make the collaborative system comfortable for the most users, this will be the default level of privacy.

3 Behaviors of Net News Users

Given that the intent of the thesis is to create a system that will be useful for filtering Net News, it is crucial that the system be designed in such a way that the system supports the users. In order to support the users, we first needed to find out how they currently go about reading Net News so our system can be consistent with their demands. Based on informal surveys, we formed several hypothesis to begin the investigation.

We claim that since Net News is being used primarily for a casual, serendipitous search for information, users are only willing to spend a certain amount of time using Net News. The number of newsgroups a user will read is tightly limited by the work she must do to process the articles in those groups. If a user can't find articles she is interested in, or she feels overwhelmed by the amount of searching she must do to find interesting articles, she will give up the search in that newsgroup and move on to an other.

To substantiate this claim, we instrumented the **nn** and **xrn** news readers at PARC to record information about how users read news. Analysis of the collected data highlighted the reading behavior of the users as a group, and suggests some limits on the length of time that users are willing to spend reading news. Any system that tries to make Net News more readable will have to operate within these time limits, and bring the user more useful information during that time span.

3.1 Obtaining the data

In order to obtain data which would give an accurate picture of the behavior of most users, we wanted to collect data from all the Net News users at PARC. We chose not to ask for a group of volunteers to let us collect data on their reading habits for fear of biasing our survey with a self-selected sample group. Unfortunately, this choice meant that since we did not have explicit permission from each individual to record information about him or her, we had to record data in such a way that no individual could be identified from collected data.

To preserve the anonymity of the user community all the data was recorded in a global append-only log file. The log file consists of a series of records; each record representing one *session* with a news reader — a session being defined as the time between when a user starts a news reader program to when he exits it or has a four hour period of inactivity. For every newsgroup the user looked at during the session an entry was made in the log file. We considered an article to have been "read" if the text of the article was displayed to the user. An excerpt of raw data from the global log file is shown in figure 1. Because **nn** and **xrn** have different user interfaces, they recorded similar, but not identical, types of information for each newsgroup.

xrn begins the record for each session with a line identifying the

```
-- xrn entry -- Mon Jul 19 15:49:06 1993
  GRP: AVAIL: 1   READ: 1   TIME: 125 *
  GRP: AVAIL: 2   READ: 1   TIME: 1 *
  GRP: AVAIL: 11   READ: 3   TIME: 69 *
  GRP: AVAIL: 7   READ: 3   TIME: 89 *
-- nn entry -- Mon Jul 19 18:16:21 1993
  GRP: AVAIL: 33   READ: 0   TIME_READ: 0   TIME_SCAN: 4665
  GRP: AVAIL: 44   READ: 0   TIME_READ: 0   TIME_SCAN: 15
  GRP: AVAIL: 3   READ: 0   TIME_READ: 0   TIME_SCAN: 6
```

Figure 1: Excerpt of raw log file data from **nn** and **xrn**.

record as coming from **xrn** and the time at which the record was entered into the log. The record contains one line beginning with `GRP:` for each newsgroup the user was shown. For each newsgroup, **xrn** recorded: 1) How many articles were available to be read in the group(`AVAIL:`). 2) How many articles were actually read (`READ:`). 3) How much time, in seconds, was spent both reading and selecting articles (`TIME:`). 4) Whether one of the "catch up" buttons was used to ignore a set of articles with one action. The use of a catch up button is shown by an asterisk.

nn begins the record for each session with a line identifying the record as coming from **nn** and the time at which the record was entered. The record contains one line beginning with `GRP:` for every newsgroup the user was shown. For each newsgroup, **nn** recorded: 1) How many articles were available to be read in the group. 2) How many articles were actually read. 3) How much time was spent reading the selected articles (in seconds). 4) How much time the user spent scanning the subjects of available articles choosing which articles to read (in seconds).

3.2 Trends in the data

Once analyzed, the collected data show the following trends which we believe are relevant to users' ability to find interesting articles in the Net News system:

1. Users read an average of 12 newsgroups each session.

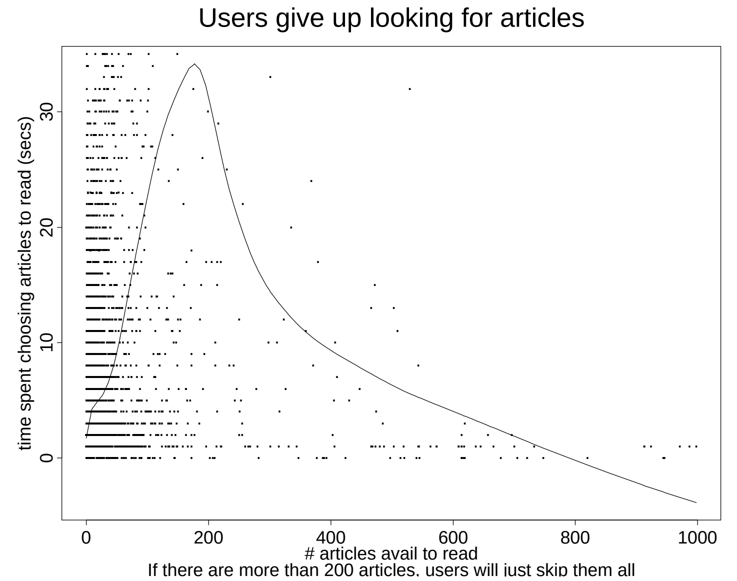

Figure 2: Plot showing the length of time users spent choosing articles to read versus the number of articles presented as available.

2. A plot of the length of time a user spends searching for articles to read in a newsgroup versus the number of articles available to be read in the newsgroup shows a telling breakpoint at around 200 articles (see figure 2). The graph also shows an elbow at around 50 articles. We infer from this graph that users reading a group with fewer than 50 some articles will take the time to actually read the subject lines of the available articles. In groups where there are 50 to 200 articles available to be read, users scan or in some way process the subject lines more rapidly. In groups with over 200 articles available, users seem to skip all the articles.

3. Far more people read Net News than post. Based on data gathered from 632 sites by the Network Measurement Project at the DEC Network Systems Laboratory, it is clear that for all groups, there are more "lurkers" than "posters."[9, 8] This is true regardless of the number of people who read the group or the number who post articles to it. Table 1 shows some sample data for groups with the largest readership, the smallest readership, the greatest traffic, and several others.

4. The time a user spends on each article, even after he or she has taken an action to display the full text of the article, is very short. The data taken so far show that users spend an average of 40 seconds reading an

Table 1: Posting volume and estimated worldwide readership for several news groups. (Data from Reid, USENET Readership report for Aug 93)

newsgroup	est. readers	# posts/month	comment
misc.jobs.offered	200,000	2,016	most readers
rec.humor	150,000	2,087	very popular
comp.unix.questions	130,000	1,062	
talk.abortion	50,000	1,590	controversial
rec.arts.tv.soaps	45,000	5,125	most traffic
bit.listserv.hellas	12,000	802	
aus.sport	2,500	238	fewest readers

article. Data broken down by article length and group will be available after the results of a second study are in.

3.3 Why people will vote

A common concern of those hearing about this project for the first time is, "Why would any user take the trouble to vote? Especially since you just showed how little time people are willing to spend on Net News. Won't you end up with no votes?" To test this hypothesis we constructed a simple prototype collaborative system at PARC by modifying the **nn** news reader so that its users could cast votes. Estimates from the log data described above show that there are about 40 users of **nn** a day. During a span of 54 days the prototype system recorded 222 votes from 25 users demonstrating that over half the population of **nn** users was willing to cast votes. It is possible that we are observing a tendency among PARC researchers to be willing to try some new feature for the novelty effect, or a desire to help another researcher's project. However, the data from PARC shows that it is plausible that the overall user community will be willing to take the effort to vote on articles.

4 Design Criteria

If any collaborative filtering system for Usenet Net News is ever to be successful, it will need to meet the requirements of two major communities.

First is the community of users, who will have to use the system and derive benefits from it. Second is the community of net news system administrators who will have to build, install and maintain the system. The experiments described above give us a feel for what the users' requirements will be. To determine the requirements of system administrators, we have turned to the informal opinions of administrators expressed on the news.admin.misc, news.admin.technical, and news.future newsgroups on Usenet itself.

Based on the behavioral data gathered from users and informal knowledge derived from reading the posts of various current system administrators, we have chosen the following as general requirements:

4.1 Architecture criteria

1. Avoid centralized servers or clearinghouses. Usenet is an anarchy with very little central control. There is no one site or entity with responsibility for all of Usenet, and no one site or entity is able to control, censor, or shut-down Usenet. This decentralization is useful from a legal standpoint as there is no organizational entity to sue, and important to administrators who need and want control over their sites.

2. Traffic and disk space requirements should not be unreasonable. We do not have a good estimate of what would be unreasonable. We believe that an extra ten percent overhead would be reasonable.

3. Minimize the amount of software re-writing that needs to be done. Unless collaborative filtering proves itself as a extremely useful, we assume we will be the only ones writing software to support it.

4. Provide a simple, yet sufficient, application programming interface (API) to the collaborative software. To encourage the writers of news reading programs to include our collaborative features, we must make it as easy as possible for them. We will provide the framework for the collaborative extensions in self-contained modules.

5. It is probably easier to convince administrators to change their news readers than their NNTP news server software. While we think many designs for a collaborative system could be simplified if they had some support from the news servers, informal opinions show that most news administrators spent large quantities of time and chewing gum sticking together and customizing their news servers for their particular sites. News readers, on the other hand, are reasonably easy to build or recompile. Unlike the news servers, news readers also contain relatively

little state so that if a new version fails it is easy to go back to the old one.

4.2 User interface criteria

1. Help the majority of Usenet users. There is a huge number of silent readers — all groups have far more lurkers than active posters. These lurkers are the people we believe collaborative filtering can help the most, and also the people who least want to wade through all the traffic on Usenet. These are also the people least willing to announce their presence (ie: they do not post), and most likely the ones who will least tolerate any extra overhead in their news reading.

2. Do not interrupt the existing flow. On average, users spend so little time reading most articles that any operation we expect a majority of users to perform must be exceedingly quick and consistent with the flow of the interface they already use. Voting for articles must be seamlessly integrated into the way the user processes Net News.

3. The filter must behave in a simple way that is easy to understand. If users can not easily understand how the system is trying to help them, the system will only be getting in the way of the user's tasks.

4. No articles may be suppressed. If a user desires to see all the articles posted to a group, the system must provide them. Further, the system may not alter in any way an existing article — to do so violates the rights of the article's author.

5 Architecture

The proposed architecture for the system is shown in figure 3. The fundamental object on which the system works is a vote. A vote represents a single user's evaluation of a single article. The architecture is broken up into two main pieces. The first piece is the interface module. This interface stands between the news reader programs and the collaborative system. The interface sends votes off into the collaborative system, and answers requests from the news reader for information about the popularity of an article. The interface obtains information about the popularity of an article by consulting various vote sources which hold collections of previous users' votes. The second piece of the architecture is the vote server. The vote server acts as a vote source

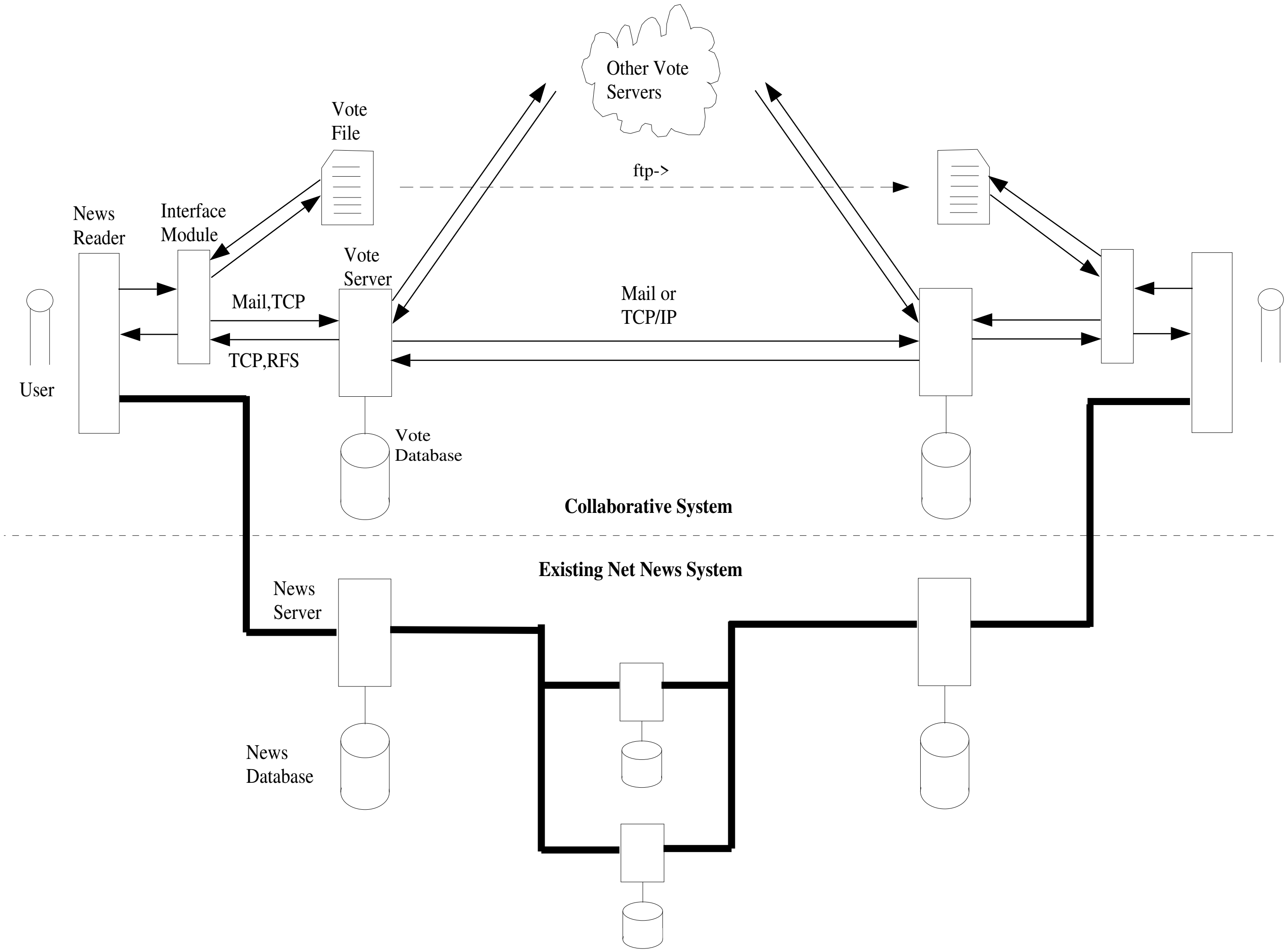

Figure 3: Block diagram for the proposed architecture.

and is responsible for maintaining a vote database which contains the collected votes of all Usenet. The vote server incorporates votes from its own site and other sites into its database, and exchanges the votes cast by local users with the vote servers at other sites. We will now describe each of these parts in more detail.

5.1 Votes

Users evaluate the articles they read to create objects we call votes. A vote contains the name of the newsgroup in which the evaluated article resides, the unique message-id assigned to the article, and the user's evaluation of the article. Evaluations are one of *terrible, ok, good,* or *great.* Although a single article can appear in more than one newsgroup, the vote only contains the name of the newsgroup in which the user read the article. This decision should make the accumulated votes more useful for filtering as newsgroups are organized around topics. An article cross posted to several newsgroups may be extremely relevant and important in one newsgroup, but totally irrelevant in the others. This decision will also discourage users from posting their articles to more than one newsgroup in hopes of garnering more votes for the article.

5.2 Vote sources

A collection of votes is termed a vote source. We plan to support two types of vote sources in order to support the two major classes of queries we expect users to make of the collaborative system. The first type of vote source will be a specially formatted vote file. The second type of source will be a vote server program.

Vote files are the more secure type of vote source. Much work has already been done to create filesystems that allow tight control over who can read or write a file. Votes written into a file can be guaranteed to both originate only from one of the people with write permission on the file, and to be read only by those with read permission on the file. The intention is that if an individual creates a vote file, other users can consult that vote source to determine how the individual evaluated an article.

The presence of these individualized vote sources makes it possible for users to create filters of the form, "Show me the articles that Jane Doe liked." The system can answer the query by consulting Jane Doe's vote file provided that Jane Doe has provided the user making the query with access to her vote file. Jane Doe can control which people have access to her opinions in the same way she currently controls who has access to her other files.

Vote servers as a source provide access to the evaluations of anonymous users throughout the Usenet. When an evaluation of an article is requested from a vote server, the returned response represents a summary of all the votes posted for that article by any user anywhere in Usenet. The information contained in the vote server supports the exploratory searching described earlier as presumably the best articles in a newsgroup will have received the most votes. New users exploring a newsgroup can quickly locate the articles previous users of the newsgroup thought were most relevant. Based on these articles, the users can decide whether or not to continue searching the newsgroup for information.

In order for the collaborative filtering system to work with multiple vote sources, there must be a naming convention to describe and identify each vote source. Uniform Resource Locators (URLs) [2] provide such a mechanism so vote sources are referred to by their URL. For example, a vote file with a local name of /usr/maltz/votes available by anonymous ftp from intrepid.xerox.com would have the URL *ftp://intrepid.xerox.com/usr/maltz/votes*. A vote server that listened for TCP connections at port 3636 of host news.xerox.com would be known as *telnet://news.xerox.com:3636*.

While sites are assumed to have a local vote server whose URL is well known at that site, some method must exist for informing users of the URL for a vote file. Any of several approaches could be used for informing other users where a vote file lives. The simplest mechanism for finding a person's vote file would be to ask them personally. This mechanism is identical to the way phone numbers are currently exchanged. A more complicated mechanism would be for users to include the URL for their vote file as an extra header line in their posts to Net News — many people already distribute their phone numbers and addresses in this fashion. News reader programs could take advantage of the extra header line by extracting the URL from articles the user liked. These URLs might then be added to the list of sources the user uses to filter Net News.

5.3 Interface module

The interface module provides simple methods for a news reader program to send votes into the collaborative system, or to request information from the collaborative system.

Exactly how a user indicates his or her evaluation of an article is a user interface question left up the designer of the news reader program. The news reader designer can choose whatever mechanism fits in best with the existing user interface. When the news reader program invokes the interface module to enter a vote into the collaborative system, the news reader specifies where the vote should be sent. The vote can either be sent on to a vote server where the vote will be combined with other votes and distributed to the entire network, or the vote can be placed in a vote file.

When answering requests for information the interface module can combine the votes from several sources, each source being a vote server or a file of votes previously written by another interface module. Combining different vote sources in the interface module makes possible queries of the form: "Show me all the articles that Joe User or Jane User liked." The collaborative information returned by the interface module will consist of lists of articles and the articles' accumulated evaluations. The news reader program can use this information as it sees fit to help the user filter his or her news.

5.4 Application Program Interface

The application program interface (API) will provide a structured way for the news reader program to communicate with the interface module. The API has two main parts. The first section deals with how the news reader casts votes for articles, and the second section deals with how the news reader gains access to the collected information for an article. A specification for the procedures in the API can be found in table 2.

Casting a vote requires only that the news reader provide the interface with information required to create the vote: the name of the newsgroup in which the article appears, the message-id contained in the article, and the user's evaluation of the article. Before exiting, the news reader must call the procedure *collab_send_off_votes()* to send the votes to the collaborative system. If the news reader provides a file name as an argument to *collab_send_off_votes()*, any votes not cast anonymously will be written to that file.

Retrieving information about the votes available to help filter the articles in a news group requires first setting up a list of sources (eg: vote

Table 2: Specification of the API to the interface module.

```
/******* Routines for casting votes *********/
/* return status: 0 = failed, 1 = successful */
int collab_enter_vote_anonymously(/* char *message_id, vote, char *group_name */);
int collab_enter_vote(/* char *message_id, vote, char *group_name */);
int collab_send_off_votes(/* char *file_name*/);

/******* Routines for querying sources for votes and information *******/
char* collab_available_groups(/* int *num_groups */);
/* return a new string of the group names separated by \n characters;
   the number of groups will be left in the loc pointed to by num_groups */

vote_t* collab_get_votes_for_group(/* char *group_name */);
/* return a linked list of the votes for articles in group_name,
    return NULL if there are no votes available */

/**** change which sources we collect data from ****/
/* return status: 0 = failed, 1 = successful */
int collab_add_source(/* char *source_url */);
int collab_remove_source(/* char *source_url */);

/* free memory for the list returned from collab_get_votes_for_group */
void collab_destroy_votes(/* vote_t *votes */);

/* notify someone of problems in the collaborative system */
int issue_error(/* varargs : fmt, args */);
```

Table 3: Structure of vote_t record.

```
/****** Structure of a vote ******/
struct _vote_data_t {
  int  num_high;
  int num_med;
  int num_low;
  int num_terrible;
  char  *stuff;
}; typedef struct _vote_data_t vote_data_t;

struct _vote_t {
  struct _vote_t *next_vote;
  vote_data_t *vote_data;
  char *message_id;
  void *internal_magic;
}; typedef struct _vote_t vote_t;
```

files and/or a vote server) that should be searched for votes. The *collab_add_source()* and *collab_remove_source()* routines are provided for this purpose. The strings passed in as source names should be the URLs of the vote files or vote servers.

Once a list of sources has been set, a string containing the names of all the news groups which have votes in them can be obtained by calling *collab_available_groups()*. The votes for the articles in a group can be obtained by calling *collab_get_votes_for_group()*. This will return a linked list of *vote_t* structures (see table 3) each of which contains the vote information for an article. The vote information tells the number of *terrible*, *ok*, *good*, and *great* votes the article received. The linked list of *vote_t* structures should be destroyed when it is no longer needed by calling *collab_destroy_votes()*.

5.5 Vote server

The vote server is the part of the system responsible for accumulating, marshaling and redistributing votes from one site to another. Votes sent to the server by an interface module are accumulated into the server's vote database and marshaled into a packet with other votes to be sent on to neighboring vote servers. Since we have chosen to handle only numerical votes in this thesis, we can perform the accumulation by simply adding the new votes into running tallies of the number of *terrible*, *ok*, *good*, and *great* votes an article has received. If votes contained more diverse information (perhaps character string keywords) the accumulation process could be extended to include a process such as string concatenation.

The vote servers exchange information in a fashion similar to the news servers. At regular intervals each vote server contacts its neighbors and offers to exchange packets of votes with them. Before contacting other vote servers, a server combines all the votes received from local users since the last information exchange into a single packet. The vote server then offers to send each neighbor the packet of new votes and any packets the server has recently received from other neighbors.

When a server receives a packet for the first time, the server combines the votes from the packet into the server's vote database. The server records the packet as having been accepted so the server will not accept the same packet twice. The server continues to store the packet however, so that the packet can be offered to the server's other neighbors. Once the server has offered the packet to all its neighbors, the packet will be deleted.

5.6 Transport of information

So that the collaborative system will work at as many sites as possible, we will attempt to keep the transport mechanisms as general as possible. For example, communication between the collaborative interface and the vote server can take place via a remote file system abstraction; TCP/IP connections; or email. Any site receiving Net News will support at least one of these communication mechanisms. Likewise, information sent between vote servers can be in the form of email or TCP/IP.

To retrieve vote files created by users at other sites, the interface module will use the URL provided as the name of the source to create a local copy of the vote file. For this thesis we will support the afs and ftp protocols for accessing vote files. See [2] for more information about these protocols.

5.7 Security and privacy offered by the system

The primary security issue the collaborative system must protect against is ballot-box stuffing. If a large enough group of people start using the system to filter their incoming news, stuffing the ballot-box for or against articles will become a way of harassing and censoring groups of users on the net. To prevent the opinions of any individual from having undue effect on the outcome of the collaborative filtering process, the collaborative system should ensure that each individual can only cast one vote for each article. Unfortunately, there is an inherent trade off between the desire to allow users to vote anonymously, and the desire to prevent ballot-box stuffing. If every vote is transported through the system authenticated with a digital signature[11] from the user casting the vote, it is possible to prevent ballot-box stuffing completely as the final consumer of the votes can ensure that each user votes only once. The computational and organizational costs of such an authentication system are very high however, and gone is the ability for a user to vote anonymously. If the vote distribution system is completely anonymous, then there is no way of stopping a person from voting as many times as they want since there is no way to tell who has voted and who has not.

We propose a system which is not perfect, but should provide protection against the most common means of attack. If users have their votes posted to vote files, then the existing systems for file protection can insure that only authorized users read or write votes to the vote file. We assume that users will not deliberately alter their vote files to stuff the ballot box. If some users do alter their vote files, we rely on other users noticing this tampering and exposing the tamperer. Since vote files are associated with individual people, the number of people who use a vote file to filter their Net News will be a strong function the reputation of the vote file's owner. Once a person has been exposed as keeping a dishonest vote file, other users will likely exclude the vote file from the list of sources they use for collaborative information. If no one uses the altered vote file to filter Net News, it can not do any damage.

Preventing a user from sending multiple votes for a single article to a vote server is harder problem to solve. Since the votes distributed by the vote servers are anonymous, there is no way to detect if single user has voted multiple times once the votes are entered into the stream of distributed votes. However, some security *can* be provided if the communication between an interface module and a vote server is not anonymous. An honest vote server can enforce a one vote per article per user rule by requiring users to authenticate themselves to the vote server before transmitting any votes. The local vote server can record which users have cast votes for each article, and prohibit the same user from casting a second vote for the article. Under this system the local vote server will know which articles a user has read.

Since the vote server is local to the user's site, it can be trusted to obey the local policy determining the privacy of personal information in electronic form. Many sites now have such policies which declare who on the system can legitimately access system logs containing personal information.

Users will authenticate with their vote server by sending their user identification number (uid) along with their votes to the vote server. This will prevent casual attempts at ballot-box stuffing, but would not stop a person who writes a program which masquerades as an interface module to the vote server. This stuffing program could generate an arbitrary sequence of numbers for uids — effectively casting votes for all the users on the system.

Vote servers can be configured to only exchange votes with specific peers, so there is little risk of an individual inserting fake vote packets into the stream of vote packets exchanged between honest vote servers. Unfortunately, there is little that can be done to stop dishonest vote servers from generating fake vote batches to stuff the ballot-box. Once a dishonest vote server is identified (most likely by human operators), the only option to secure the system is for peers to derecognize the dishonest vote server and cease exchanging votes with it. This approach is currently used on Net News itself to control users who forge messages.

5.8 Type of information exchanged by vote servers

Will numerical rankings from users provide enough information to help other users filter articles? Is there other information we could gather inoffensively which would be cheap to transport and useful for filtering? This is an important area of inquiry for tuning the collaborative filtering system and making it work well. However, this project will not directly address these issues. The vote transport system we create will be flexible enough to encompass extra information as an add on. Once some system is operational, it will be possible to experiment with exchanging different types of information.

5.9 Comparison of architecture and design criteria

The proposed architecture meets the requirements set out in the design criteria section. The vote servers keep the vote information in a separate stream from the news articles so that system administrators do not have to modify their current news servers. By placing a vote server local to every participating site and providing a means for them to exchange information between themselves, there is no one central vote server authority. Since votes are stored separately from the articles, there is no change made to any article. Any user who wants to see all the articles posted to a newsgroup in their original form can do so by simply ignoring the votes, so there is no issue of censorship. The API provides an interface between the collaborative system and the news reader program, so modifying existing news readers to use collaborative filtering should be easy. Integrating collaborative filtering into existing news reader programs will both prevent users from having to learn a new user interface, and allow news readers to use the collaborative filtering data in whatever way the program's user interface supports best.

6 Evaluation Criteria

We propose to evaluate the success of this project along two dimensions. The first dimension will be the level of acceptance the system achieves in the Usenet community. The second dimension will be based on the improvement the project provides to users.

To evaluate to acceptance of the system we have several obvious quantitative measures. How many sites are using our system? How many votes are being cast? Does the number of votes increase or decrease over time?

Evaluating the ability of the system to provide a useful service will depend on both quantitative and qualitative measures. We plan to take informal surveys of users to find a rough estimate of user happiness with the filtering system. To quantitatively evaluate any improvement made by the system, we will look for changes in the user behaviors measured before implementation of the system. Specifically, we hope to see an increase in the number of newsgroups a user reads (on the assumption that the user now has an easier time finding relevant articles in a newsgroup and so can broaden his or her search to include more newsgroups). We also hope to see an increase in the number of groups in which the user reads some articles (this will demonstrate users are not being overwhelmed by the number of articles available in the group).

7 Timeline for Thesis Work

At present we have a small, working system consisting of a vote server and two news reader programs which have been modified to send and request votes. This current system is deficient from the proposed one in four major ways. First, the interface between the collaborative system and the news readers is *ad hoc* and poorly formed. Second, the system does not permit votes to be stored in or read from files. Third, the system does not scale up well. Fourth, the system provides little of the privacy or security discussed in the architecture proposal. We plan to work on these problems in roughly the order listed while integrating into the design new insights we gain from the operation of our existing system.

Since the available time may not permit the implementation of the entire system, we will strive hardest to develop the vote server to the point that it can be released to the Usenet network. Once this server is available for release, it will be possible to make some evaluation of the system no matter how *ad hoc* the other components of the system are.

By Late October The API for the news reader programs will be frozen. The interface module will be able to save votes to files and retrieve and combine votes from several sources. We will have extended the system to include the MIT Athena environment.

By Late November A vote server which can exchange articles in the fashion described by the architecture design will be ready for release. We will begin exporting the system to any site that will accept it.

Through December Perform damage control and create bug fixes for problems in the system unearthed by the release. Re-release of software if required. Begin writing the text of the thesis.

Through January Continue thesis write up. Evaluate system performance according to evaluation criteria.

By May Continue to maintain released system, handing it off to others if possible. Complete thesis writing to the satisfaction of David Goldberg and Karen Sollins.

8 Conclusion

The system presented in this thesis proposal will become the first attempt at providing collaborative filtering to users across the Usenet. We believe our architecture will be accepted by the Net News community as its design is based on the observed behaviors of Net News users and system administrators. Once operational, the system will provide an effective way for users to locate information of greatest interest to them.

References

[1] Anklesaria, F.; McCahill, M. et al.; "The Internet Gopher Protocol," RFC 1436, University of Minnesota, March 1993.

[2] Berners-Lee, Tim; "Uniform Resource Locators," working draft, CERN, July 1993. available via ftp from info.cern.ch as /pub/www/url6.[txt,ps]

[3] Jolicoeur, Lucie, Newsreaders survey RESULTS (long!), Usenet news.software.readers, 10 Aug 93, Message-ID: <1993Aug10.205558.18192@IRO.UMontreal.CA>; [185 lines].

[4] Kantor, Brian; Lapsley, Phil; "Network News Transfer Protocol," RFC 977, U.C. San Diego, February 1986.

[5] Goldberg, David et al.; "Using Collaborative Filtering to Weave an Information Tapestry," *Communications of the ACM,* December 1992, Vol 35, No 12, pp. 61-70.

[6] Loeb, Shoshana; "Architecting Personalized Delivery of Multimedia Information," *Communications of the ACM,* December 1992, Vol 35, No 12, pp. 39-48.

[7] O'Reilly, Tom; Todino, Grace; *Managing uucp and Usenet,* O'Reilly & Associates, Inc., 1992.

[8] Reid, Brian; USENET Readership report for Aug 93, Usenet news.lists, 9 Sep 1993, Message-ID: <26m3ae$n5k@usenet.pa.dec.com>; [2342 lines].

[9] Reid, Brian; USENET READERSHIP SUMMARY REPORT FOR AUG 93, Usenet news.lists, 9 Sep 1993, Message-ID: <26m39n$n4l@usenet.pa.dec.com>; [394 lines].

[10] Reid, Brian; USENET READERSHIP SUMMARY REPORT FOR MAY 93, Usenet news.lists, 2 Jun 1993, Message-ID: <1uibjb$8de@usenet.pa.dec.com>; [402 lines].

[11] Rivest, R.; Shamir, A.; Adleman, L.; "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM,* Feburary 1978, Vol 21, No 2, pp. 120-126.

[12] Weinstein, Sydney; "Special Issue: Network News," *The C Users Journal,* February 1991, pp. 109-116.