IMSCAN: An Algorithmic Framework for Mining Instant Messaging Networks

John Resig
Department of Computer Science
Laboratory for Applied Computing
Rochester Institute of Technology
Rochester, New York 14623
jer5513@cs.rit.edu

Ankur Teredesai
Department of Computer Science
Laboratory for Applied Computing
Rochester Institute of Technology
Rochester, New York 14623
amt@cs.rit.edu

Christopher Homan
Department of Computer Science
Laboratory for Applied Computing
Rochester Institute of Technology
Rochester, New York 14623
cmh@cs.rit.edu

ABSTRACT

Collecting and analyzing data in large-scale instant messaging (IM) networks is a mostly unexplored problem. IM networks are data-rich environments where one can obtain information at various levels of granularity, from simple status-change logs to detailed, text-based conversations. In this paper, we present an architecture for collecting user status changes over IM networks. Then we define in terms of a number of distinct similarity measures a general pattern analysis framework for discovering sets of users that show similar instant messaging behavior. In particular, this framework helps us answer two natural queries: "Who behaves like user X?" and "What is the probability that user X is in status S $\in$ {Online, Away, Busy, Off-line} at time T?" Each measure we use can be efficiently computed. The highlight of this paper is that, without collecting any explicit user-specific association information such as buddy-lists, we are able to obtain reliable answers in the form of user-behavior clusters. Using a dataset consisting only of the status-change behavior of 207 users collected over a 67 day period we experimentally evaluate the different properties of the measures our framework uses.

Categories and Subject Descriptors

D.2.8 [**Database Applications**]: Data Mining ; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Instant Messaging, Pattern Analysis, Clustering, Social Communications, Counter-Terrorism Analysis

1. INTRODUCTION

Instant Messaging, abbreviated IM, is a type of communications service that enables you to create a kind of private chat room with another individual in order to communicate in real time over the Internet, analagous to a telephone conversation but using text-based, not voice-based, communication. Typically, the instant messaging system alerts you whenever somebody on your private list is online. You can then initiate a chat session with that particular individual.

In 1968, Doug Engelbart's presentation at the Fall Joint Computer Conference, was a live online hypermedia demonstration of the pioneering work that Engelbart's group had been doing at SRI [5]. In this presentation, Engelbart showed how the computer could be used to deal with everyday tasks. The majority of consisted of him using the computer to plan out a set of things. All of this information was in simple hypertext which contained many different methods of organization, each appropriate to the task at hand. That "Mother of All Demos" (on display in the Exhibit on The Information Age at the Smithsonian Museum of American History) perhaps was the beginning of human-computer interaction as we see it today: Pervasive and Ubiquitious. Instant messaging was invented way before ICQ or other chat programs flooding the Internet in the early 1990's. We know for a fact that it existed in the form of the command *talk* on UNIX. IM networks are fast becoming a de-facto way for personal communication. IM technology has significantly advanced to let users communicate accross networks, in remote areas and in a highly-pervasive, highly-ubiquitios manner. Ironically, specifically of interest, to the knowledge discovery community, will be the "ideal" way this form of communication generates large amounts of data that so far (to the best of our knowledge) not been effectively analyzed. It is of significant interest to industrial and goverment organizations to understand the broad knowledge-sharing networks that exist within the organization and IM communication is fast becoming a standard platform for such knowledge sharing. Apart from this fundamental interest in knowing "Who IM's whom"; a non-intrusive analysis such as the one we present

in this paper will prove very helpful as a testbed in social-network analysis, finding interpersonal relationships, behavioral analysis and general high dimensional cluster analysis. The objective of this paper is to highlight the framework for data-collection and present a sampling of the kind of challenges that arise in instant message mining. We were able to achive exciting results using relatively simple analysis techniques and outline the issues we faced during this data collection and analysis.

The technical contributions of this paper are organized in three parts: a) IMSCAN data collection architechture b) Pattern analysis and c) Status prediction. Within pattern analysis, we describe techniques and resuls for mutli-user pattern analysis, two-user pattern analysis, and single user usage analysis. On the status prediction front, we provide a simple probailistic tool to determine the time vs. possible status for a given user. We have developed a novel data collection framework to track and collect informaton on IM networks to determine the individual status changes. We then try to highlight the need for developing effective mining techniques and metrics to analyze and assimilate information from such status change logs. We describe several user-grouping metrics for association rules based clusters, PageGather [10] based grouping and edit distance based similarity to generate groupings of users.

In the next section we have tried to outline the previous work in a variety of domains that we felt was related to the concept of mining instant messaging networks. Section 11 describes the data collection framework which is an imporant contribution of this paper. A discussion on the scalability and privacy issues for IMSCAN is also included. In sections 3 and 4 we have tried to separate the algorithmic description of the analysis technique from the results obtained where possible, but at times, where we felt a simple result of the technique would be the best explaination we have taken the liberty to do so. Section 5 describes the experiments conducted on the dataset and presents the corresponding results. An interesting observation can be made: Between the elapsed time intervals 5000-5700 and 5900–6500 no activity appears to occur. This time period for data collection corresponds to the blackouts that occured throughout Northeastern United States last Summer and no activity could be recorded. A flavor of the general nature of the dataset can be obtained by viewing Figure **??**. The histograms in this figure outline the overall usage summary of various users in the four possible states. It was a very challenging task to design algorithmic solutions to find the correlations between two and more users to discern patterns in user behaviour. At the same time it was very interesting to observe that simple regression models could effectively predict the user probability for a status change as shown in figure **??**.

NOTE to Reviewers: a) We would like to emphasize that the data collection for the purpose of this study was non-obtrusive and did not collect any personal or demographic information about the participants.

2. RELATED WORK

We are unaware of any precedent for instant message mining. Recently, Getoor summarized nicely that a key challenge for data mining is tackling the problem of mining richly structured datasets, where the objects are linked in some way [6]. Links among the objects may demonstrate certain patterns, which can be helpful for many data mining tasks and are usually hard to capture with traditional statistical models. IM mining presents one such application domain that generates a relatively large number of such richly structured datasets. Largely the interest in web and hypertext mining has steadily grown, and so has the interest in mining social networks, security and law enforcement data. Newsgroup mining has also recieved considerable attention recently as a problem for social behaviour analysis as reported by Agrawal et al [1].

Since there are potentially very large number of users in an IM network the IMSCAN framework needs to provide a robust solution for finding associations between large user-sets. This problem is similar in many ways to frequent item-set mining (in fact we aptly name the term user-set in the same vein). There are inherent problems with the concept of finding rules based simply on their support and confidence. Silverstein et. al [12] describe the pitfalls of using support as the threshold for pruning the rule-set. This problem become severe in environments such as IM mining. Due to the fact that relationships between user-sets (analogous to item-sets) can exist in-spite of the fact that the user-sets may not be present in significant number of transactions. In this paper we have adopted an interactive density based approach to mining association rules where we first plot all the association rules on a support vs. confidence graph. Then based on the density of a particlar region, we investigate the rules to generate the relationships.

There are several related areas in pattern analysis for large datasets that display properties similar to the problem of mining instant messaging networks to determine the relationships between users. Social network analyis and graph based techniques come to mind immediately. Several problems in web-mining are also closely related including finding related but unlinked pages to generate index pages automatically or “auto-indexing” [10]. We make use of the PageGather algorithm in a similar setting for clustering users based on their time-¿status logs.

Since September 11, 2001, and the terrorist attacks against the United States, the area of Counter-Terrorist Data Mining has seen a surge of interest and papers relating to applications of old data mining techniques to a new field of study. Most papers attempt to utilize the study of Social Network Analysis in order to find potential links between suspicious groups of people. Two such, current, works include the Mapping of Terrorist Cells [8] and Assymetric Plan Detection [4].

A discussion on finding direct correlations for Counter-Terrorist activities was provided in the manuscript 'Mapping Networks of Terrorist Cells' by Krebs [8] and directly concerns the social network analysis surrounding the Sept. 11 attacks. In this paper we outline the issues we faced while attempting to find such direct correlations between users based on their online-activity.

The work on Assymetric Plan Detection[4] attempts to co-

ordinate a social network of people and places with links of seemingly trivial actions using a CBR modelling technique. They succinctly outline the majority of issues that any 'elegant' algorithm faces in such domains : Massive data sets, and noise. Mining instant messaging networks also displays these characteristics:

- Massive data sets consisting of an evergrowing number of users changing states at varying intervals of time leading to a stream of status change information filtering-in by the second.
- Data Collection in instant messaging networks is noisy and state changes are affected by noise resulting from any unintensional reflection of a status change. There is no way of determining if a given status change was intensional or the result of a loss of power or other interfering factors.

The analysis framework we describe has applications in such information rich, yet noisy data environments such as counter-terrorism. We believe that IMSCN provides a new realm of data collection and analysis which has gone mostly unexplored so far. We reported a basic outline of the data collection framework as pertinent to the data mining efforts for counter-terrorism in a recent submission [11]. In this paper we present the pattern analysis algorithms, status prediction algorithms, the corresponding results, and the resulting discussion on creating a scalable and robust IMSCAN framework.

3. STATE PREDICTION

The first problem that we wanted to look at was that of calculating user availability expectation. In order to acheive this we looked at a user's usage time for a particular service over several weeks in an attempt to determine certain habits that a user may have. In addition to the particular times that users are expected to be online and offline, we can also develop a metric to estimate how long it will take a user to move to an online state, given that they are currently unavailable.

In order to find a user's availability, we separated the IM data set into two different categories, available and unavailable. A user that is online was said to be available, while any other status for this user is marked as being unavailable (e.g. Idle, Away, and Offline). We then recorded the unavailable times for all users, our data structures recording the start, end, and duration of each users' unavailability. These unavailability values gave us information about the user over the 10 weeks in which the data was collected. In Figure 1, the bars in the diagram represent times when a user is unavailable, the gaps between the bars represent when a user is available.

For each interval of time, during a week period, we looked at the averages of the total 10 weeks of user activity. Based upon the average unavailability, and the average duration of time which the user was unavailable, a probability is calculated for that time block. A representation of this data can be found in Figure 1.

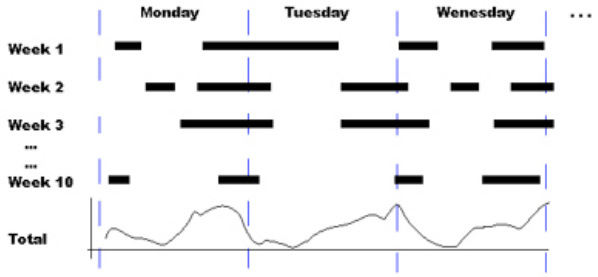

Figure 1: Estimating user usage time; the black bars represent time periods a user is unavailable, the gaps between represent the period a user is online. The total represents the time period it will take a user to return given that they are unavailable.

1. $Avail_{u,x,w}$ = 1 if User u is unavailable at Time x in Week w, 0 if not.
2. $Until_{u,x,w}$ = Number of time blocks until User u becomes available at Time x in Week w.
3. $Status_{u,x} = \frac{\sum_{i=1}^{10} Avail_{u,x,i}}{10}$
4. $Duration_{u,x} = \frac{\sum_{i=1}^{10} Until_{u,x,i}}{10}$

Using this method it becomes possible to predict, with a certain degree of probability, if a user will be online at a certain time block within a week - along with how long it will be until the user is likely to return.

When the data file is read in, sparse data structures are created to store the time intervals which users are offline for. The final structure contains 10 lists for each of the 10 weeks we collected data for. Each list contains the start, duration, and end times of each users offline status. Using this sparse design we were able to load the entire dataset into memory, and ended up consuming only 1.2MB of memory. This design is highly scalable and would accommodate for a couple years of data to remain in memory before the need of serializing data to disk should arrise.

The algorithm, as implemented, would benefit greatly from more user data to process. It is important to note that while 10 weeks of data can provide interesting usage patterns, periods of a year or more could expose monthly, as well as seasonal, patterns that users follow.

4. PATTERN ANALYSIS

4.1 Frequent User Set Mining

One of the initial attempts at finding associations between two given users was through utilizing Assocation Rule Mining. The first thing that was attempted was trying to find a correlation between the status of one user against the status of another. Association Rule Mining seemed the obvious solution for such a situation. In order to generate the transactional data sets required to utilize ARM across our data, the collected information has to be discretized into time 'buckets'. Each transaction, Equation 2, represents a

bucket of time and a status, holding the set of users who have the specified status for a majority of the given time, Equation 1.

$$S_{stx} = \{x,t\} -> s \quad (1)$$
$$T_{st} = \{S_{st1}, S_{st2}, S_{st3}, ..., S_{stn}\} \quad (2)$$

Where:

n = The id of the largest user being evaluated.
s = The user status being evaluated.
t = The time interval ('bucket') being evaluated.
x = The id of the specific user being evaluated.

In order to find the frequently occuring user sets, along with their associated support and confidence, an implementation of Apriori [3, 2] was utilized. Using the newly created transactions, assocations were found between one user at a given status and another. Using this association, correlations between the actions of one user's actions and another user's actions can be found.

4.2 Clustering

We cluster our data by partitioning the set of users into a collection of disjoint subsets that cover all users. Our goal is to maximize the number of nontrivial clusters (i.e., clusters of cardinality at least two). The reason for this is that instant messenging is primarily a person-to-person communication medium. Certainly, any two people who communicate via an instant messenging service must be online at the same time. More generally, we would like to know if with any confidence significant patterns of usage emerge between small groups of users.

4.2.1 General Clustering Algorithm

We used the following family of clustering algorithms that are parameratized by a distance function D, introduced in the description below, and a confidence threshold.

1. Choose a threshold γ, which is a real number whose range of possible values depends on D.
2. Let $A = \begin{bmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ be an $n \times n$ matrix, where n is the number of ids and each matrix element a_{ij} is equal to $D(i,j)$.
3. Let $B = \begin{bmatrix} b_{11} & \cdots & b_{n1} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$ be an $n \times n$ matrix, where each matrix element b_{ij} is defined as follows.
$$b_{ij} = \begin{cases} 0 & \text{if } a_{ij} \leq \gamma, \\ 1 & \text{otherwise.} \end{cases}$$
4. Each connected component in the graph induced by the adjacency matrix B is a cluster.

The function D is a parameter we can change in order to perform different clusterings. function. In our experiments, we always base D on an assumed probabilistic model for the data. (In order to symplify our notation, we use numbers to represent the various user states according to the following dictionary: 0 = offline, 1 = away, 2 = idle, 3 = online.) Let $E_{i,s}$ be the event that id i is in state s. As is commonly done, we model $E_{i,s}$ probabilistically and assign to $\Pr(E_{i,s})$ the frequency of $E_{i,s}$. Similarly, we assign to $\Pr(E_{i,s} \wedge E_{j,s})$ the frequency of $E_{i,s} \wedge E_{j,s}$.

We use two different basic sets of functions for F. The first set $\{D_{c0}, D_{c1}, D_{c2}, D_{c3}\}$, is based on conditional probabilities. For $k \in \{0, \ldots, 3\}$, D_{ck} is defined on inputs i and j as

$$D_{c3}(ij) = \begin{cases} 0 & \text{if } i = j, \\ \min\{\Pr(E_{j,s}|E_{i,s}), \Pr(E_{i,s}|E_{j,s})\} & \text{otherwise.} \end{cases}$$

Choosing D from this set results in an algorithm that is essentially the same algorithm used in Pagegather [10].

The next set $\{D_{l0}, D_{l1}, D_{l2}, D_{l3}\}$ is based on the lift between two ids i and j being in state s. One pitfall in using conditional probabilities alone, as we did in the previous section, is that, to determine associations, they do account for the relative independence between $E_{i,s}$ and $E_{j,s}$, and the relative independence between these two events is perhaps a more natural notion of what an assoication is. Recall that two events F and G are considered to be independent if $\Pr(F|G) = \Pr(F)$ and $\Pr(G|F) = \Pr(G)$. Using Bayes' rule, one can confirm that the ratio between $\Pr(E_{j,s} \wedge E_{i,s})$ and $\Pr(E_{i,s}) \cdot \Pr(E_{j,s}$, i.e., $\frac{\Pr(E_{i,s} \wedge E_{j,s})}{\Pr(E_{i,s}) \cdot \Pr(E_{j,s})}$ known as lift. If the lift has a value less than 1, it signifies a negative correlation between $E_{i,s}$ and $E_{i,s}$ whereas a value greater than 1 indicates a positive correlation.

For $k \in \{0, \ldots, 3\}$, D_{lk} is defined on inputs i and j as

$$D_{l3}(ij) = \begin{cases} 0 & \text{if } i = j \text{ or } \Pr(E_{i,s}) \\ \Pr(E_{j,s} \wedge E_{i,s}) / (\Pr(E_{i,s}) \cdot \Pr(E_{j,s}) & \text{otherwise.} \end{cases}$$

4.3 Frequent Action Set Mining

Another form of analysis that became apparent was the mining of frequent actions that a given user may act upon. By removing any element of time from the data set and simply focusing on the sequence of actions that a user takes provides a new level of analysis not available in the straight user-status comparison metrics. Two unique aspects of analyzing the sequence of user actions are: It may become possible to detect similar user patterns across time zones and geographical locations and it can be used to determine what a user's next action is going to be, based upon the series of previous actions (aiding the State Probability analysis).

$$S_u = \{A_1, A_2, A_3, ..., A_n\} \quad (3)$$
$$Sub_u = \{\{A_1\}, \{A_2\}, ..., \{A_1, A_2, ..., A_n\}\} \quad (4)$$
$$Large_{U_a,U_b} = \{x \subset Sub_{U_a}, Sub_{U_b} || x| > |all|\} \quad (5)$$
$$Dist_{U_a,U_b} = 1/|Large_{U_a,U_b}| \quad (6)$$

It was decided that attempting to find correlation between

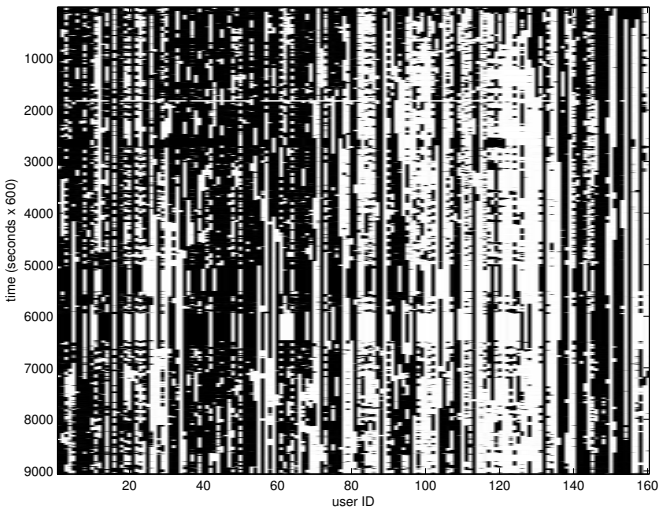

Figure 2: Intensity map showing the elapsed time each user spent offline. The columns represent the users by ID, and the rows are the time intervals. The lightness of the rectangle at each ID/time intersection represents the amount of time during that interval spent offline. Due the the scale of the data, only black (meaning no time was spent in time 0) and white (meaning the entire interval was spent in time 0) are evident.

two users' action sequences would provide an interesting comparison against the common state analysis. In order to compare the state analysis clustering against the this non-time based method, a metric had to be determined to compute the distance between any two users. The series of actions taken were as follows: Firstly, the sequence of user actions, Equation 3, had to be constructed. This was done simply by moving through the log of user actions and constructing the final non-time based sequence of actions. Then using these sequences, the set of all possible sub-sequences, Equation 4, were computed. Using these sets of sub-sequences, the longest common sub-sequence was found for each pair of users, Equation 5. Finally, the distance between two particular users was computed by normalizing the length of the largest common sub-sequence, Equation 6. Using this distance as a common metric of distance, they were placed into a matrix and fed into an implementation of the Pagegather algorithm [10].

5. EXPERIMENTS AND RESULTS

IMSCAN was tested on a data set consisting of 160 users, identified by the numbers 1–160, sampled over a 62 day (5437147 second) period. At each second, each users' state was sampled. Figure 5, for example, shows as an intensity map the elapsed time each user spends in state zero (intensity maps for the other states look similar to this one, but are sparser). One interesting feature of our data is that between the elapsed time intervals 5000-5700 and 5900–6500 no activity appears to occur. This is because the experiment was run during the blackouts that occured throughout Northeastern North America last Summer and data gathering was temporarily halted.

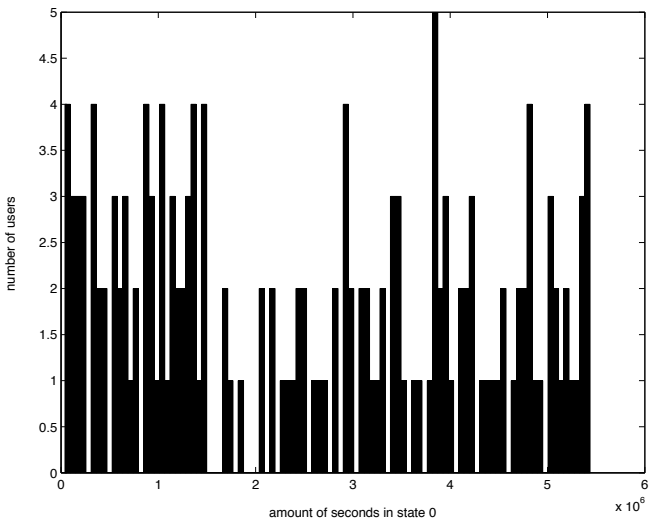

Figure 3: This histogram shows the number of users that are offline for the amount of time indicated on the x-axis.

5.1 Distribution of States

Figures 5.1 shows a histogram of the number of users that are offline for a given amount of time. Offline is by far the most popular state. The other states are less popular (with online being the least frequent state) but have similar shapes.

5.2 Distribution of Cooccurences

Figure **??** shows the number of seconds each unordered pair of users shares some state in common. The figure 5.2 shows a histogram of this data, and figures 5.2–5.2 show histograms of the common number of seconds shared in a single state. What is interesting about this data is that the distribution figure reffig:freq.tot appears to have a gamma function, but figures 5.2–5.2 do not (graphing figures 5.2–5.2 using a log-linear axis did not reveal a discernable pattern. This suggests that there is a least some independence between two users sharing two or more different states.

5.3 State Prediction

A sample result set from this algorithm can be found in Figure 9. In this figure two users are represented using the notation 'Series1' and 'Series2'. The vertical axis denotes the time until a user is expect to return to an available state while the horizontal axis represents the time, during an arbitrary day, at which the user is likely to be available. Within Series1 it becomes apparent that the user being represented by the line typically doesn't return to an online status until approximately time 630, while a constantly-increasing return probability is immediately visible. In Series2, the user takes a few trips offline throughout the day, apparent by the jagged increases in time until user is expected to return. Using this information it could be possible to develop plug-ins for existing Instant Messaging services in order to aide users in determining when members of their buddy list will be likely to return.

5.4 Frequent User Set Mining

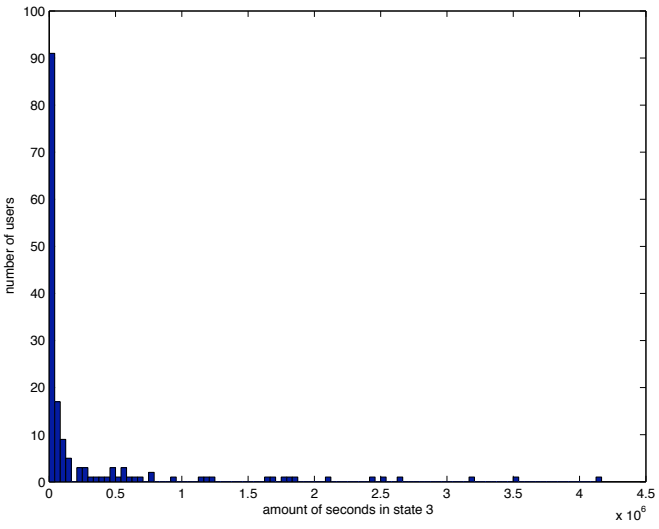

Figure 4: This histogram shows the number of users that are online for the amount of time indicated on the x-axis.

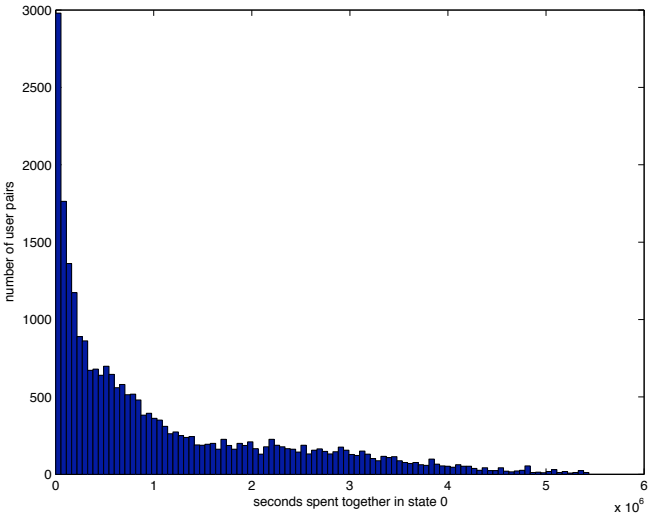

Figure 6: A histogram showing the distribution of seconds a pair of users share offline.

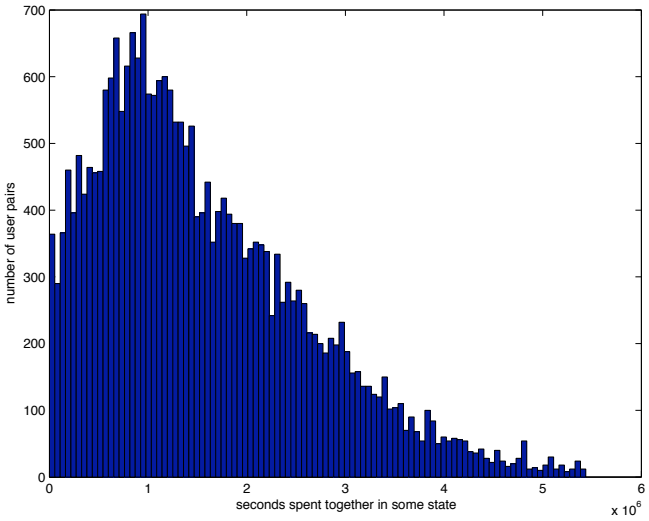

Figure 5: A histogram showing the distribution of seconds a pair of users shares a common state.

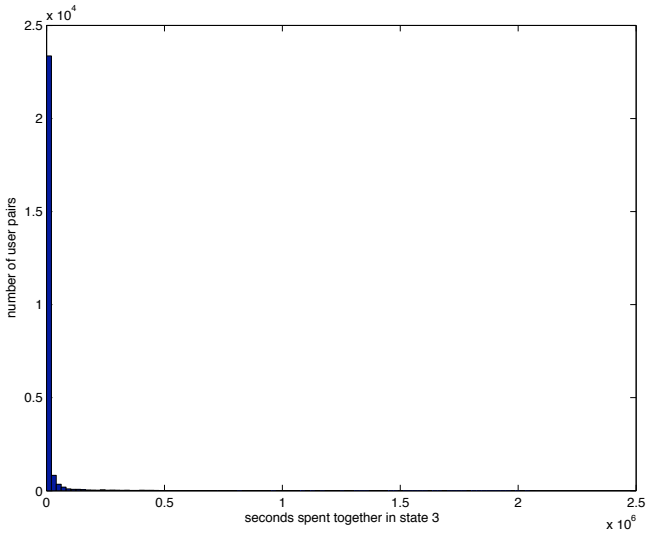

Figure 7: A histogram showing the distribution of seconds a pair of users share offline.

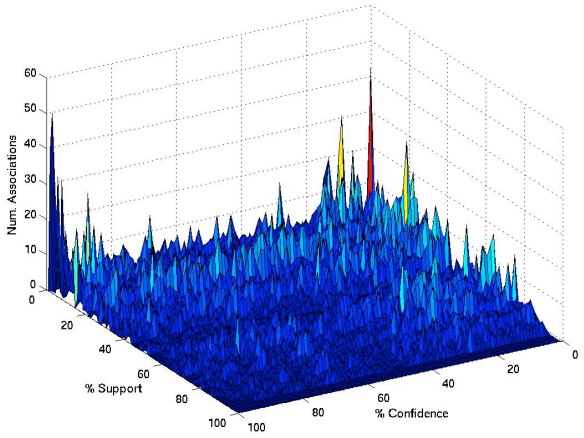

Figure 10: Online Associations - Horizontal Axis represents Confidence, Vertical, Number of Associations, Z-Axis, Support

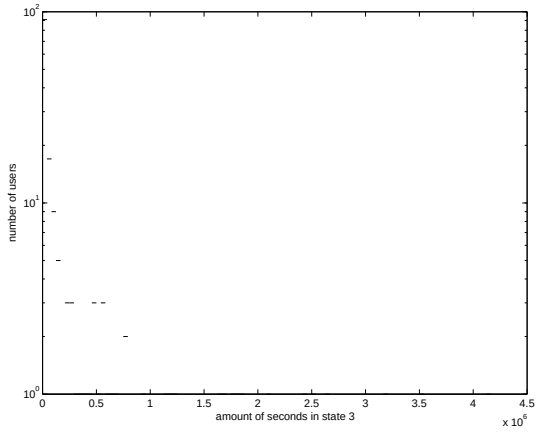

Figure 8: A log-linear plot of figure 5.2.

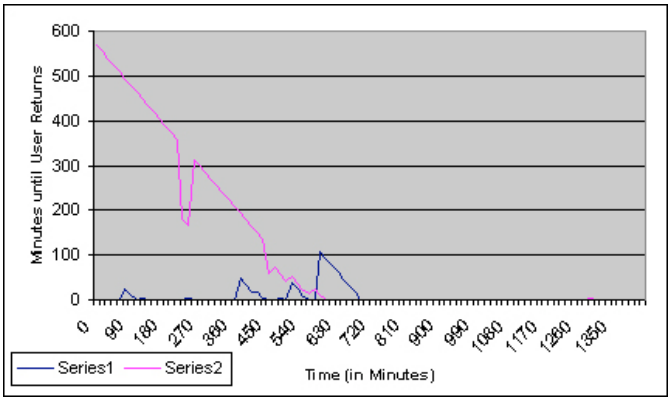

Figure 9: Two users usage graphs - Minutes until available vs. Time

In order to better view the distribution of the Association Rules generated various methods for visualization [9, 7, 14, 13] were analyzed for intuitivity and degree of information presented. CrystalClear [9] was chosen due to its ability to reveal the distribution of associations effecitvely throughout the total association rule set. From the Online Chart, Figure 10, the extreme density of associations with low support, but high confidence, becomes apparent. Some sample data from the Online Status, centered around the darkened area can be found in Table **??**. The sample data used has a minimum of 99% Confidence and maximum of 5% support. Within the data shown, the actions of User B are said to be associated with that of User A with the specified support and confidence. While in the Offline Chart, Figure 11, the associations exhibit a diminished level of confidence, perhaps implying the independence of associations between users having an offline status at the same time as one another.

User A - User B	Support %	Confidence %
29 - 81	1.5	100.0
133 - 81	1.5	100.0
71 - 81	1.5	100.0

Table 1: 10 Associations, ranked by confidence

User A - User B	Support %	Confidence %
59 - 160	94.5	10.1
138 - 160	94.5	8.5
39 - 160	94.5	8.9

Table 2: 10 Associations, ranked by support

5.5 Clustering

We first consider clustering using the conditional-probability-based distance functions D_{c0} and D_{c3}. In finding the optimal value for γ, i.e., the value that maximizes the number of nontrivial clusters, we ran our clustering algorithms on all γ from 0.1 to 0.99, stepping by 0.1 between each trial. Figures 5.5–5.5 show these results.

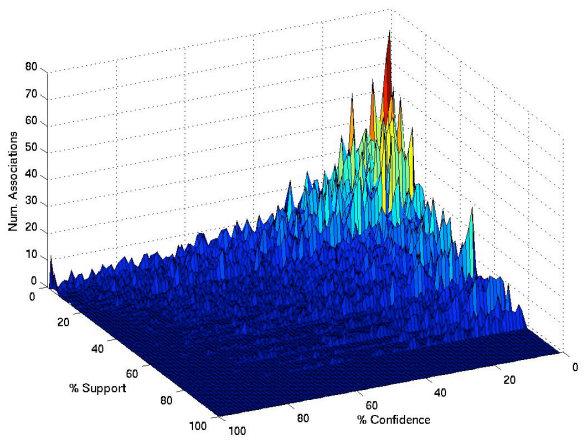

Figure 11: Offline Associations - Horizontal Axis represents Confidence, Vertical, Number of Associations, Z-Axis, Support

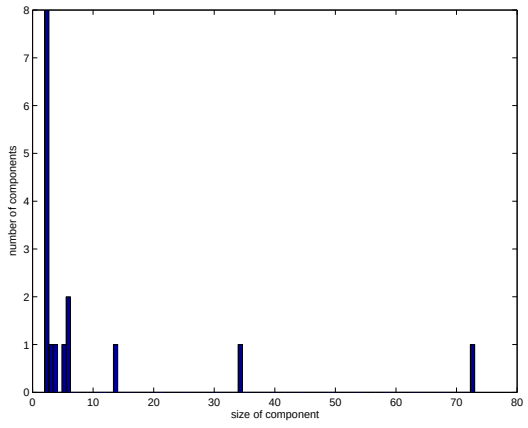

Figure 12: A histogram showing the distribution of D_{c0}-based clusters by size for $\gamma = 0.88$, which yields an optimal number clusters (16).

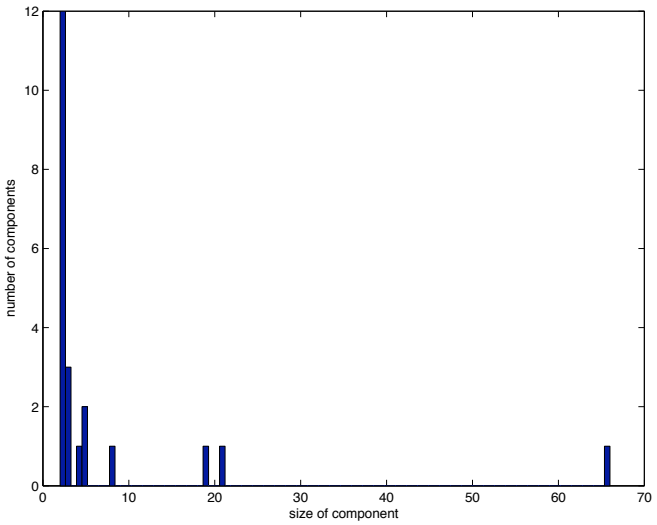

Figure 13: A histogram showing the distribution of D_{c3}-based clusters by size for $\gamma = 0.43$, which yields an optimal number clusters (22).

Next we look at the clusters obtained using the lift-based distance functions D_{l0} and D_{l3}. In finding the optimal value for γ, i.e., the value that maximizes the number of nontrivial clusters, we ran our clustering algorithms on all *gamma* from 1 to 358, stepping by 0.1 between each trial. Figures $5.5and5.5$ show these results.

6. FREQUENT ACTION SET MINING

Having computed the distance between users usign the metric described within Frequence Action Set Mining, the distances, Table 3, were placed into the Pagegather algorithm to find potential clusters. The clusters 4 that were found by the algorithm all exhibit very common patterns that often continue indefinitely.

User A	User B	Distance
70	100	0.663316582914573
13	70	0.663316582914573
4	70	0.663316582914573
20	118	0.949748743718593
37	118	0.949748743718593
100	118	0.949748743718593
13	118	0.949748743718593
4	118	0.949748743718593
20	37	1
20	100	1
37	100	1

Table 3: Sample Subset Distances

7. CONCLUSION AND FUTURE WORK

In this paper we describe a framework for collecting and mining status information from instant messenging networks. Our experiments show that the data gathered can be used to predict the state of a user and to co-relate user access patterns leading to clusters of users. In future we plan to perform time-series analyse to spot outliers in user behavior. In addition, instant messenging networks offer a variety of

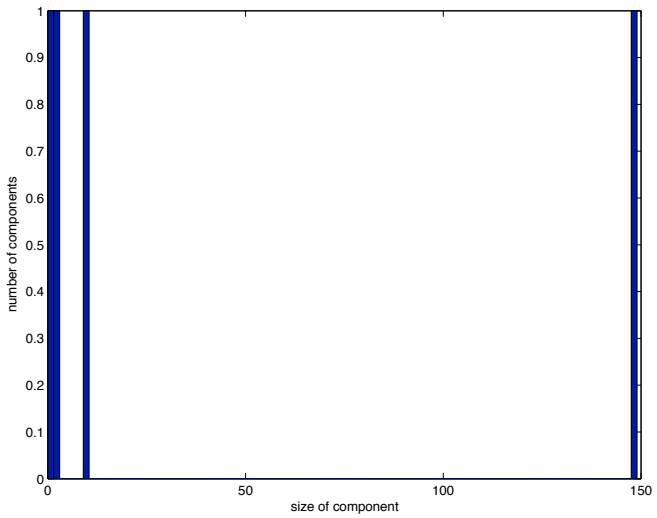

Figure 14: A histogram showing the distribution of D_{l0}-based clusters by size for $\gamma = 9.2$, which yields an optimal number clusters (3). This optimal number held for all test values of $\gamma \in (6.3, 12.1)$

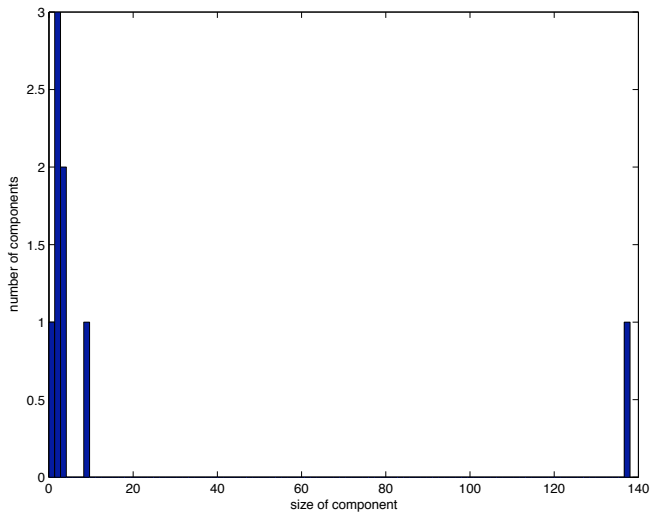

Figure 15: A histogram showing the distribution of D_{l3}-based clusters by size for $\gamma = 328.9$, which yields an optimal number clusters (7). This optimal number held for all test values of $\gamma \in (302.2, 355.6)$

- Users: 1, 49, 71, 81, 92, 114
  Pattern: 12121212121212121212121212121212121212121212...
- Users: 4, 13, 19, 20, 37, 70, 100, 118
  Pattern: 30303030303030303030303030303030303030303030...
- Users 23, 74, 79, 80, 140
  Pattern: 20202020202020202020202020202020202020202020...

Table 4: Sample Subset Clusters, via Pagegather

other information which includes the user profiles and their away messages. This text data can be utilized to further co-relate users.
We believe that ample amount of information can be gathered from instant messenging networks for identifying groups of users and relations between their behavior patterns.

8. ACKNOWLEDGMENTS

We thank members of the Laboratory for Applied Computing and the Data Mining Research Group at Rochester Institute of Technology for their support.

9. ADDITIONAL AUTHORS

Additional authors: Vineet Chaoji (Department of Computer Science, email: `vsc2002@cs.rit.edu`).

10. REFERENCES

[1] R. Agrawal, S. Rajagopalan, R. Srikant, and Y. Xu. Mining newsgroups using networks arising from social behavior. In *Proceedings of the twelfth international conference on World Wide Web*, pages 529–535. ACM Press, 2003.

[2] C. Borgelt. Efficient implementations of apriori and eclat, 2003.

[3] C. Borgelt and R. Kruse. Induction of association rules: Apriori implementation, 2002.

[4] E. R. Daniel Fu and J. Eilbert. A cbr approach to asymmetric plan detection. In *Proceedings of the Workshop on Link Analysis for Detecting Complex Behavior (LinkKDD2003)*. ACM Press, 2003.

[5] D. C. Engelbart and W. K. English. A research center for augmenting human intellect. volume 33, pages 395–410. Morgan Kaufmann Publishers, Inc., San Mateo, December 1968. Republishd in 1982 in Computer Supported Cooperative Work: A Book of Readings, Irene Greif [Editor].

[6] L. Getoor. Link mining: a new data mining challenge. *ACM SIGKDD Explorations Newsletter*, 5(1):84–89, July 2003.

[7] M. C. Hao, U. Dayal, M. Hsu, T. Sprenger, and M. H. Gross. Visualization of directed associations in E-Commerce transaction data. pages 185–192.

[8] V. Krebs. Mapping networks of terrorist cells, 2002.

[9] K.-H. Ong, K.-L. Ong, W.-K. Ng, and E.-P. Lim. Crystalclear: Active visualization of association rules.

[10] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *AAAI/IAAI*, pages 727–732, 1998.

[11] T. A. Resig J. A framework for mining instant messaging services. In *Workshop on Link Analysis, Counter-terrorism and Privacy*, 2004.

[12] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, 1998.

[13] P. C. Wong, P. Whitney, and J. Thomas. Visualizing association rules for text mining. In *INFOVIS*, pages 120–123, 1999.

[14] H. Zhang. Mining and visualization of association rules over relational dbmss.

11. APPENDIX

The medium of Instant Messaging on the Internet is a well-established means by which users can quickly and effectively communicate with one another. Long utilized by the public as a quick form of free communication, data mining tasks have not been attempted over Instant Messaging. Additionally, on a corporate or government level, people are just beginning to take notice of the potential that IM provides in terms of the type of information that can be collected from these networks. Many large software or internet based corporations have started Instant Messaging networks of their own, generally open to the public after registration, including Time Warner, Yahoo, and Microsoft. Currently, some of the most popular Instant Messaging networks are run by some of the aforementioned companies:

- AOL Instant Messenger
- Yahoo! Instant Messenger
- MSN Instant Messenger
- Various IRC Networks

Interestingly enough, even with all the variety of networks available, their physical communication structures (client-server architecture) and communication protocols (information packets) are very similar to one another. Currently, IMSCAN is best suited to collect data from the AOL instant messenger (AIM) and IRC networks.

Online	The user's client is connected to the central server and the user is active (currently typing or moving the mouse on his computer).
Offline	The user's client is not connected to the messaging server at this time.
Idle	The user's client is connected to the central server, but the user is not active. Additionally, how long a user has been idle can be determined from their status.
Away	The user is logged on but away from the station. Sometime users specify a text message that can be viewed by anyone who wishes to get more information about where they are or why they are away. (e.g. "Out to lunch.", "Watching TV.") In fact a user can be either idle, or active, while an away message is explicitly up.

Table 5: Possible user statuses. As shown above an IM client can be in one of the above statuses at a given time.

Most Instant Messaging networks follow a Client-Server model in which a server (or a cluster of servers) is maintained by a service provider who controls traffic coming to and from the server. Users who wish to utilize a certain network generally register themselves with the service provider, then download a provider-approved client for use on their network. Using this client, users can connect to the central server in order to be able to send and receive messages and collect account information. A *friend* is generally another registered user (the term *friend* is server-specific, but exists on almost all messaging networks). The concept is that a user may maintain a *Buddy List* under which a listing of their immediate

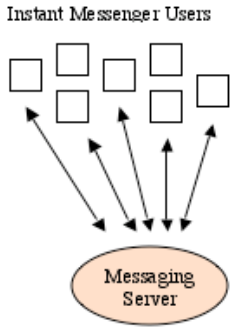

Figure 16: Existing Instant Messaging networks follow a client-server communication architechture. Each messaging client when *"online" has an individual communiation thread with the messaging server. In turn the server communicates the status of user's "friends" to each client.*

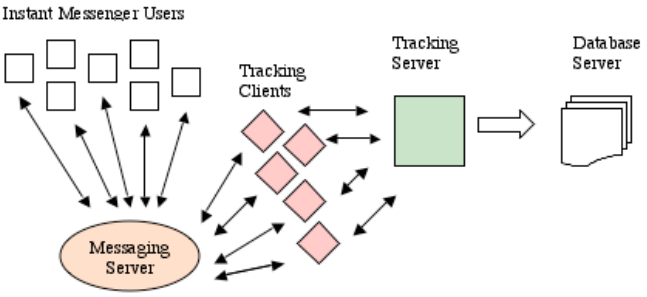

Figure 17: The IMSCAN Data Collection Framework: In addition to the IM network components, we use a set of tracking clients that monitor the messaging server notifications and notify the tracking server. The tracking server pre-processes the data and sends it for storage.

friends may exist. Using this, the server then sends a client updates based upon the statuses of their friends. Once the connection process has completed, the server performs all future communication in the form of *Update Packets*. An update packet is sent from the server to a client whenever an action occurs that is associated with him. For example, when a friend performs a status change or if a message is being sent to a user's client. An unfortunate consequence of the server maintaining such buddy lists is that it can impose restrictions upon the maximum number of friends which a user is allowed to maintain (this number is generally around 200). Since a client does not directly communicate with any other connected client, and only the server, the server is then in charge of disseminating any potentially useful information from one client to another. Once such piece of critical information is a user's status. Table 5 describes a list of possible statuses that a client can be in. Status is an attribute generally associated with a user's client and often indicates how a user responds to an Instant Message. Whenever a user's status changes, an update packet is relayed by the central server to everyone who has the user on their buddy list.

Another important aspect of communication flow within an Instant Messaging network is the traffic of messages between users. The amount of information revealed concerning instant messages is generally limited to the information which is directly related to a user. Such information paths include chat rooms (a group discussion area where multiple people can communicate with one another stimultaneously) and private Instant Messages (messages sent directly from one user to another).

Between the various information resources provided by Instant Messaging networks, there are a number of valuable resources available to the average user. The data generated in turn is very useful for data mining to analyze user behavior. However, in order to utilize the flow of information offered by these networks, a data collection framework need will have to be established. This paper proposes one such framework which has been developed. Information distributed by the Instant Messaging networks can be broken down into two simple groups: User status-change and communication-flow (Instant Messages, Chat Rooms).

The first item collected, user status change, can be achieved relatively simply as the current structure of Instant Messaging networks support the collection process. One interesting feature, previously discussed, of Instant Messaging networks is that of 'Buddy Lists' - lists of friends of a user. The direct benefit of this feature is the fact that whenever a buddy (a member of a user's buddy list) performs a status change, the client is immediately notified of it by the server. Utilizing this feature set, one could set up a client of their own, with an arbitrary buddy list, and begin collecting information about their 'buddies' resulting actions. This is significant due to the fact that most Instant Messaging networks don't require that someone actually be a friend of another user in order to watch their status changes.

Using this standard model, it is relatively simple to set up a tracking client whose only job is to collect pertinent information about users that are on its buddy list - aptly named, in this framework, *Tracking Client*. In order to maintain a tracking client a *Tracking Server* is constructed which manages the actions of its associated tracking clients. The *Tracking Server* marshalls communication between an arbitrary number of tracking clients and the database server. Whenever a new *Tracking Client* spawns and connects to the *Tracking Server* the server attempts to determine which Instant Messaging users need to be tracked, from a list of potential users. Due to the restrictions imposed by the various Instant Messaging networks as to the size of a user's buddy list this distributed *Tracking Client* structure is required in order to be able to track the maximum number of people at any given time. An advantage to this distributed network is that no one client is dependant upon for all tracking efforts or network bandwidth usage. Each *Tracking Client* within the network watches a given number of other clients in order to verify that they are, in fact, still connected to the network - if not then a communication is sent to the tracking server and another client is spawned to cover the users not being tracked by its disabled peer. As information packets come in from the server to each tracking client, the client attempts to determine if the packet should be re-transmitted to the

server for storage in the central database.

Another tracking effort that is currently being explored is that of monitoring inter-user communication. One resource offered by most Instant Messages networks (and exclusively by others, see IRC) is that of a public chat room. A tracking client has the ability to connect to one of these rooms as a spectator, simply to view the flow of conversation. Similar to how the server performed by sending data packets concerning a user's status change, the server will also send packets detailing messages being publicly sent from one user to another within this chat room setting. As with status changes these packets are verified for integrity and then passed along to the tracking server for subsequent storage. Packet integrity is verified by checking the information against the previously collected packets, making sure that no duplicate packets are transmitted to the server.

An advancement has recently been made by the AOL Instant Messaging network to allow a user to connect to the network from multiple locations using multiple clients. Using this pseudo-proxy, AOL displays only the users most-active (The order of activeness being: Online, Idle, Away, Away and Idle) connection to other users of the network. However, clients at equal states of activity receive all incoming communications. This advancement is very important due to the fact that now it is possible to spawn tracking clients for willing users of the network and provide them additional intelligent services on top of their normal Instant Messaging experience. It is expected that other Instant Messaging services will soon follow suit with a similar feature - due to which additional services can then be provided to the users of those networks.

To our understanding there are two major issues that need to be resolved for IMSCAN:

- Scalability: The IMSCAN framework is a distributed tracking client framework with a centralized tracking and database servers. Potentially, there indeed are scalability concerns when tracking a massive number of users in large IM networks. We have tried to address this problem as 'elegantly' as currently possible and are looking into other effective models. In the current implementation, a tracking client is capable of montoring a maximum of 200 individual users. Each user needs to be tracked by only one tracking client (unlike the IM client where each client has to monitor the status of all other clients in the buddy-list). This significantly reduces the number of tracking clients required. Adding more users will entail adding more tracking clients, but since very few users actually have the maximum allowed (200) friends, it has currently not been an issue. The database server needs to handle a batched update from each tracking client on a regular basis (say every 1 minute), but this has not been a bottleneck since an update is needed only if there is a status change indicated and much of the data can be filtered out.
- Privacy: The data collection for the purpose of this study was non-obtrusive and did not collect any personal or demographic information about the participants. We have utilized the functionality provided by the current IM networks where tracking user status is based on an open communication protocol. The current IMSCAN architechture currently supports AIM and IRC and users for Yahoo! and Microsoft MSN-Messenger have to opt-in and add our tracking client to their buddy list. We also do not track any inter-user chat initiations and conversations. Our objective was to purely discern and explore the kind of information that can be obtained from simple status logs.

NOTE to Reviewers: We can provide the experimental details on how many IMSCAN tracking clients can be supported on a single server before the final submission.