SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 5, ISSUE 1
ISSN: 2054 -7420

TCSC: A Trustworthy and Communal Social Classifieds

Yung-Ting Chuang
Department of Information Management, National Chung Cheng University, Taiwan
ytchuang@mis.ccu.edu.tw

ABSTRACT

Today, everyone relies heavily on the Internet to access information, and it changes the way we live, including our use of social networking services and community-based classified systems. However, two major problems lurk behind both social networking and community-based classified systems. privacy and security. First, the current systems could be modified someday to filter, conceal or censor information. Second, the anonymity inherent in the community-based classifieds does not provide any safeguard to the users.

In order to address the privacy concerns and give better protection to the users, we present a trustworthy and communal social classifieds, named TCSC, in this paper. Our TCSC is a decentralized communal social classifieds infrastructure, and it contains: 1) distributed search and retrieval system, 2) reputation system, 3) protection system, and 4) randomized message routing system. We further demonstrate that our TCSC can accurately and retain significant utility even in the presence of malicious nodes.

Keywords: P2P; decentralized search/retrieval; probabilistic analysis; distributed systems; trustworthiness; statistical inference.

1 Introduction

Today, everyone relies heavily on the Internet to access information. The Internet-distributed, uncontrolled, unbiased and dispassionate-greatly facilitates the free flow of information, and share perspectives from across the world. So do both social networks and classified systems. Online social networks, such as Facebook and Twitter, allow people to share and exchange information with their friends. Online classified systems, such as Craigslist and eBay, help people to buy, sell, or trade items to others.

However, there are advantages and disadvantages to both social networks and classified systems. Facebook helps people to connect to their friends from all over the planet. However, Facebook is a centralized social network service in which all information is controlled by its administrators. Thus, we all trust its administrators to remain benign and to let us access, publish, and view the information on the website. Unfortunately, history shows how easily control over a centralized mechanism can be subverted [28]. According to Bamman et al. [2], some countries, such as China, block or restrict access to Facebook and Twitter in order to curtail protests and political discussions.

DOI: 10.14738/tnc.51.2510
Publication Date: 14th January 2017
URL: http://dx.doi.org/10.14738/tnc.51.2510

Online classifieds systems, such as eBay, are a type of worldwide classifieds allowing people to advertise a variety of goods and services over the world. However, the disadvantage of such worldwide classifieds is that they involve shipping time, listing fees, and delivery fees. Community-based classifieds including Craigslist and Freecycle were created to avoid these problems. On Craigslist, people can advertise their goods and services to their local communities, sorted by geographic area. The advantage of such community classified systems is that there is no shipping fee or delivery time involved, as sellers/buyers are all living in the same neighborhood. Moreover, such community-based classifieds enable people to better connect to their communities, as well as learning where to satisfy their needs in the immediate area. Nevertheless, such community-based classified systems with anonymity features have created security and privacy issues. According to Verma [32,33], the anonymity feature on Craigslist has led to many crimes such as kidnapping, threats, and prostitution. The problem is that there is no way for the buyer to know the seller, or vice versa, and thus the entire transaction contains an element of risk.

For this reason, we present a trustworthy and communal social classifieds, named TCSC, to eliminate the drawbacks of both social network services and community-based classified systems. First, we implement a distributed search and retrieval system to get rid of the centralized control. Next, we develop a reputation system, so that a user can obtain the feedback of other peers before they start the interaction/transaction. Moreover, we investigate the lower bounds and the probabilistic analysis of a match. After that, we develop a protection system to enhance our TCSC. Furthermore, we implement a randomized message routing system that enables a user to communicate with other users directly. Finally, we evaluate the performance of TCSC in terms of the accuracy of the protection system and match probabilities. The main contribution of TCSC is to allow users to freely publish classifieds without being censored by the centralized administrator, as well as providing better protection to all users of the community.

2 Related Work

2.1 Peer-to-Peer Network

Mischke and Stiller [21] provide taxonomy of distributed search mechanisms in peer-to-peer networks. The distributed publish/subscribe approach is categorized as either structured or unstructured.

The structured approach, such as Bianchi et al. [3], Gupta et al. [16], requires the nodes to be organized in an overlay structure, based on Distributed Hash Tables (DHTs), trees, rings, etc. The structured approach is more efficient than the unstructured approach, but it has the following issues: 1) administrative control, 2) additional overhead for maintaining the overlay network, 3) churn or malicious disruptions.

The unstructured approach typically uses gossiping and randomization, and requires the subscriber and the publisher nodes to find each other by exchanging messages over existing links. Gnutella [14] uses flooding of requests to find information. Freenet [8] learns from previous requests. Ferreira et al. [13] use a random-walk strategy in an unstructured network to replicate both queries and data. BubbleStorm [29] is a probabilistic system for unstructured peer-to-peer search that replicates both queries and data, and combines random walks with flooding. Pub-2-Sub [31] uses directed routing to distribute subscription and publication messages to the nodes. Similarly, our TCSC system uses the unstructured approach.

2.2 Feedback/reputation strategies

Damiani [11] allows nodes to collect reputation information by gathering reports from large numbers of nodes for both the resources and the nodes that provide access to those resources. Buchegger and Le Boudec [4] investigate a Bayesian approach to evaluate a node's reputation from second-hand reports obtained from other nodes. Hu [17] presents a reputation system that resists malicious attacks, such that nodes develop reputations of their neighbors from observations of their neighbors' behaviors. Likewise, in TCSC, we adapt the similar reputation-based strategy as the work presented above, such that a node would obtain reputation based on their neighbors' point of view.

2.3 Distributed Online Social Networks

A number of systems have proposed to provide better privacy and trustworthiness in current online social networks. Diaspora [30] allows users to choose where to store their data with a number of different providers without a centralized control. Similarly, other systems such as Safebook [10] and PeerSoN [5], let users store their data on their own machines or on friends' machines. PrPl [26] and Vis-A-vis [27] offer systems where users can migrate their data between trusted machines or via another trusted third-party infrastructure. Other commercial applications, such as Mislove et al. [22], 2Peer [1], were presented to allow users to connect and share information with their friends in a decentralized way. [19] likewise proposes an integrated algorithm to protect the privacy threats. Similarly, TCSC provides better trustworthiness by having users to share and control their data in a decentralized way.

2.4 Social Network, Web Search and Classifieds

Some social-networking websites have recently begun providing a framework for third party classified applications to combine their efforts with its existing social graph. For example, Facebook Marketplace [15], powered by Oodle, is an application that uses the power of social media to provide consumers a friendly local marketplace to buy, sell and trade items with others. Similarly, Verma [32,33] presents an interesting application called Serefind, which combines social networking and online classifieds. Serefind is a social classifieds site that tries to address security and privacy constraints by asking users to create their own accounts, so that the administrators can trace individuals to reduce community crime. However, none of the above applications addresses the problem of distrust of the centralized site - not the way our TCSC does. In TCSC, we don't tackle the security problems by having an administrator to trace individuals. Rather, we utilize feedback mechanisms to allow users to acquire more information about others before a transaction is started.

2.5 Exponential Weighted Moving Average

Several network security researchers like [34] use the EWMA to determine the λ(c in our TCSC) in order to detect anomalies in the network based on a known window size. Our TCSC protection system consider all of the results from the time a node joins the system, and utilizes EWMA algorithm to calculate our final match probabilities.

2.6 Protecting against subverted nodes

Some prior work has been focused on trustworthiness, in particular for detecting and protecting subverted nodes. Jesi et al. [18] identify malicious nodes in an overlay network based on gossiping, and place such nodes on a blacklist. Condie et al. [9] present a protocol where peers improve the trustworthiness of the network by forming connections, based on local trust scores defined by past

transactions. Similarly, [24] conducts a survey of black-hole attack in wireless ad-hoc network. Our TCSC does not use gossiping, but rather, identify malicious nodes based on its request requests.

Morselli et al. [23] describe an adaptive replication protocol with a feedback mechanism that adjusts the number of replicas according to the mean search length, in order to determine whether an

object is replicated sufficiently. Leng et al. [20] present mechanisms for maintaining the desired degree of replication in BubbleStorm, when each object has a maintainer node. However, our protection system uses different techniques to maintain the desired degree of replication of requests.

3 Design of TCSC

Our TCSC combines a community-based online classified system and a social-networking service without centralized control, such that it can protect privacy and provide better security to all the people in the neighborhood. Fig. 1 shows the flowchart of TCSC. In Fig. 1, a user begins the process by informing our TCSC whether he/she wants to search for an advertisement, or communicate with others. If a user chooses to search for an advertisement, our TCSC directs to search and retrieval system. Once the search and retrieval process is complete, our TCSC implements the reputation system to give buyers better purchase protection. After that, our TCSC runs protection system to further protect against malicious attackers. Finally, our TCSC applies randomized message routing system for better assurance of communication security. Our TCSC then goes back to the beginning of the process, and waits for the next input. However, if a user wishes to communicate with others, our TCSC would jump to the randomized message system, and then goes back to the beginning of the process. The detail explanations for these four systems are describes in following subsections.

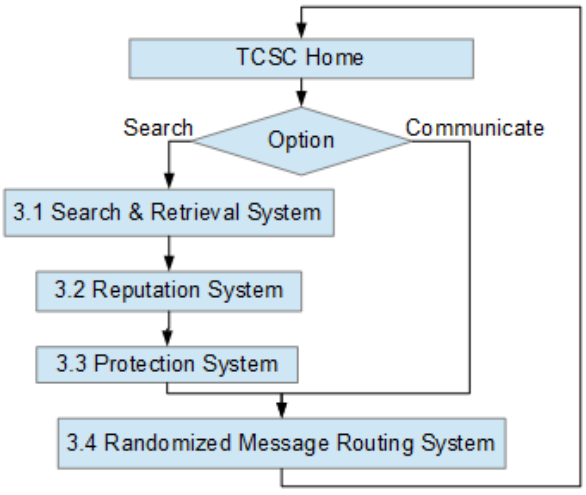

Figure 1 Flowchart of TCSC

3.1 TCSC Search and Retrieval System

The steps for the TCSC search and retrieval system are described below and are illustrated in Figures 2-4.

1. A source node wishes to distribute an advertisement first produces metadata that describes its advertisement and its URL, then distributes that metadata to a subset of nodes chosen at random.

2. The requesting nodes that seek the advertisement first generate requests that contains list of searching keywords, and then distribute the requests to a subset of nodes chosen at random.
3. All nodes that receive such requests from the requesting node compare this requests against the metadata that they hold. If there is a node that has metadata from both the source node and the requesting node, it returns the URL of the source node back to the requesting node.

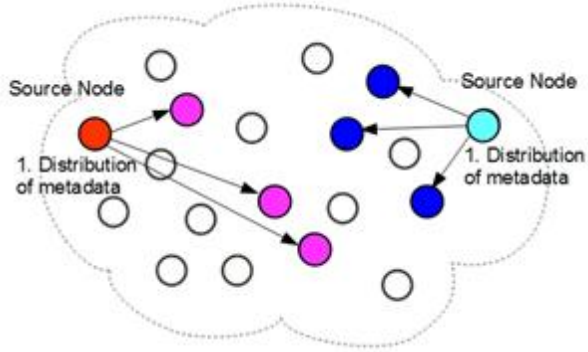

Figure 2. Source node distribute metadata for its advertisements to randomly selected nodes

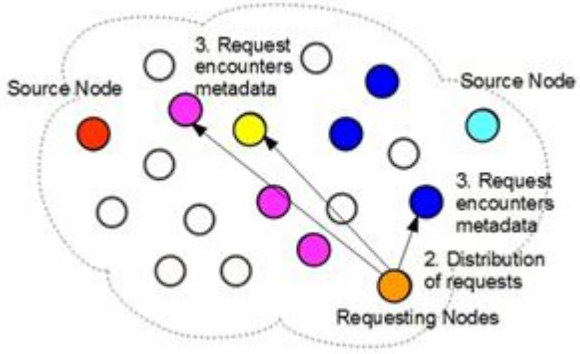

Figure 3 Requesting node distributes requests to randomly selected nodes

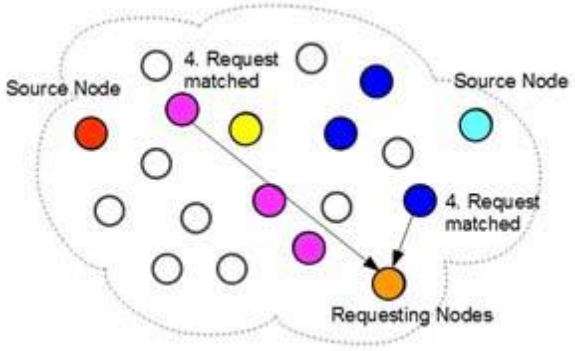

Figure 4 The matching nodes supply the URLs of the source nodes back to the requesting node.

3.2 TCSC Reputation System

The steps for the TCSC reputation system are described below and are illustrated in Figures 5-6.

1. Once a node creates its advertisement, our system first asks the seller the duration of this post. A seller can delete the post, or edit the duration of the post at any time. The post will then expire by the time that a seller has defined. However, the seller may also extend the listing time if the transaction hasn't been completed by the specified time.
2. Once an advertisement reaches its expiration date, or when the seller deliberately ends the posting, a system then initiates a survey for this transaction. The survey includes the buyer for this transaction, a rating for this buyer, the price for the final deal, the date that the transaction was completed, and a space for comment about this buyer.
3. After the system collects the survey data from the seller, it now knows the buyer for this particular transaction; thus, the system further sends a survey for this transaction to the buyer.
4. After the buyer finishes the transaction survey, the system saves this information to the buyer's database.
5. Before a requesting node calculates the reputation score for a particular advertisement, as shown in Figure 5, it produces a reputation request containing the source node, and distributes that

request to a subset of nodes chosen at random. The requesting node might get more than one source node from the matching nodes; in that case, it will produce the reputation request that contains all the source nodes.

6. All nodes that receive such reputation requests from the requesting node check their databases and determine whether they have appropriate reputation scores and reputation comments for these source nodes, as shown in Fig. 6. If so, the reputation information is sent back to the requesting node. Otherwise, the node simply replies N/A.
7. Once the requesting node finishes collecting all the reputation information, it calculates and displays the overall reputation scores and comments to the user. If there is more than one source node matching the request, the results will be displayed from highest reputation to lowest. The user can see the advertisement along with its reputation score, and determine whether he or she would like to retrieve the advertisement based on the reputation scores provided. Once the user makes his or her selection, the requesting node contacts the selected node to retrieve the full advertisement if necessary. The primary parameters that determine the reputation of a particular advertisement are:
   - W_{ct}: Weighting factor of the completed transactions.
   - W_{td}: Weighting factor of the transaction date.
   - W_{tp}: Weighting factor of the transaction price.
   - W_{ss}: Weighting factor of the seller score.
   - CT: The completed transactions of a node. In our system, a node can be a seller, a buyer, or both. Thus, the CT refers the total number of completed purchase and sale transactions of a node, which is retrieved from the database. We classify completed transactions into five ranks using logarithmic scales, where the five ranks PF are classified as follows:

$$PF = \begin{cases} 1/5 \text{ if } 0 \leq CT \leq 100 \\ 2/5 \text{ if } 100 < CT \leq 250 \\ 3/5 \text{ if } 250 < CT \leq 500 \\ 4/5 \text{ if } 500 < CT \leq 1000 \\ 5/5 \text{ if } 1000 > CT \end{cases}$$

   - MD: The maximum time decay of the transaction.
   - CD: The current date
   - ED: The end date
   - TD: The time decay of the transaction. We believe that the transactions that occur in different periods should calculate differently; old transactions should be of diminishing significance and fresh transactions should be weighted more as they are more important. Our equation sets TD = 0 if CD - ED > maxDate. The time decay TD is calculated as follows:

$$TD = \begin{cases} \frac{MD-(CD-ED)}{MD} & \text{if } (CD - ED) < MD \\ 0 & \text{else} \end{cases}$$

   - TP: The price for the final transaction. We also classify the transaction price into one five ranks, defined as follows:

$$TP = \begin{cases} 1/5 \text{ if } 0 \leq \text{transaction price} \leq 100 \\ 2/5 \text{ if } 100 < \text{transaction price} \leq 1000 \\ 3/5 \text{ if } 1000 < \text{transaction price} \leq 5000 \\ 4/5 \text{ if } 5000 < \text{transaction price} \leq 10000 \\ 5/5 \text{ if } 10000 > \text{transaction price} \end{cases}$$

- ratedScore: The seller score given from the buyer
- maxScore: The max score.
- SS: The seller score, which is calculated as: SS = ratedScore/maxScore
- RS: The final reputation score is calculated as:

$$RS = \omega_{ct} CT + \omega_{td} TD + \omega_{tp} TP + \omega_{ss} SS$$

- For the sake of simplicity, we set $w_{ct} = w_{td} = w_{tp} = w_{ss} = 0.25$ as the appropriate values for our experiments, and thus, making overall $w_{ct} + w_{td} + w_{tp} + w_{ss} = 1$. Although $w_{ct} = w_{td} = w_{tp} = w_{ss} = 0.25$ might be the appropriate values for our experiments, our TCSC system also allows users to choose different values of w_{ct}, w_{td}, w_{tp}, and w_{ss} on their own.

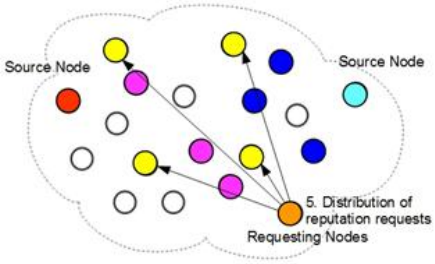

Figure 5. The requesting node distributes reputation requests to randomly selected nodes

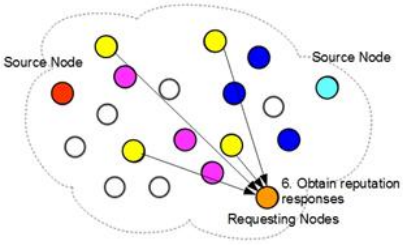

Figure 6 Nodes reply reputation scores and comments back. Requesting node display the final match results to the users.

3.3 TCSC Protection System

The objective of a malicious attacker is to prevent the matching of metadata, and preventing reports of matches from being returned to the requester. Detecting a malicious attack is as important as preventing a malicious attack from being effective. Because our system is a not a centralized mechanisms, every node must be able to detect a malicious attack. Thus, our proposed strategy supports such detection probabilistically.

The effect of such an attack might be that a large number of nodes behave normally except that they do not match requests and return responses. In our previous work [6,7] we demonstrate that our proposed algorithms can effectively detect and defend against malicious attacks. Our detection algorithm compares those empirical probabilities with the expected probabilities of the numbers of matches for various proportions of operational nodes, and estimates the actual proportion of operational nodes. On the other hand, our defensive algorithm then increases the number of nodes to which the requests are distributed, to maintain the same high probability of a match when some of the nodes are subverted as when all of the nodes are operational.

However, our previous work [6,7] only focuses on having requesting nodes to distribute requests without dynamically changing the value of r', which is not very suitable for real-world scenarios. In addition, we estimate the value of x' with only five curves for different values of x, x=1.0, 0.8, 0.6, 0.4, and 0.2. In this paper, we refine our previous work, such that requesting nodes issue requests based on the dynamic

value of r'. Moreover, we estimate the value of x' with ten curves for different values of x, x=1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1, such that our extended scenario for this paper would be closer to reality. Lastly, we have slightly decreased the estimated x' to obtain a higher value of r' and match probabilities, when the value of currentMP drops below a certain threshold.

The parameters and variables for our protection system that determines the estimated proportion x of non-malicious nodes and number r of nodes to which a requesting node distributes its request are as follows:

- n: The number of nodes in a node's view of the membership at a given point in time.
- m: The number of nodes in a source node's view of the membership to which it distributes metadata.
- r: The number of nodes in a requesting node's view of the membership to which it distributes requests.
- x: The proportion of nodes in a node's view of the membership that are non-malicious.
- k: The number of participating nodes that report matches to the requesting node.
- x': The estimated proportion of non-malicious nodes in a node's view
- r': The estimated number of nodes in a requesting node's view to which it distributes requests.
- t: The number of successive estimates by the chi-squared test that indicates the same change in the value of x' before the protection system accepts that change and takes action to change the value of r'.
- o[l]: The actual number of observations that fall into the l_{th} bucket
- O: The array of observations.
- $\bar{Q}(k)$: In our protection system, we apply normalization equation to both the analytical probabilities and the collected observations before applying the modified chi-squared test, calculated as:

$$\hat{Q}(k) = \frac{Q(k)}{\sum_{k=1}^{K} Q(k)}$$

- P[x][k]: The probability of successes in a sequence of draws from a finite sample space without replacement. We adapt the hypergeometric distribution [12] and calculate the probability of k matches in r trials, where a trial corresponds to sending the request message to a node. The analytical probability of k matches is given by

$$P[x][k] = \frac{\binom{m}{k}\binom{n-m}{r-k}}{\binom{n}{r}} = \frac{\left(\frac{m}{k} \cdots \frac{m-k+1}{1}\right)\left(\frac{n-m}{r-k} \cdots \frac{n-m-r+k+1}{1}\right)}{\left(\frac{n}{r} \frac{n-1}{r-1} \cdots \cdots \frac{n-r+1}{1}\right)}$$

- y_0: The probability $y_0 = P(k \geqq 1) = 0.9817$ of one or more matches when $r = m = 2\sqrt{n}$ and x = 1.0.
- K: The upper bound on the number of buckets used for the responses to a request.
- x_2: The modified chi-squared goodness-of-fit test [25], which is used to compare the normalized analytical probabilities and the normalized observed probabilities for different values of X. The modified chi-squared statistic is given by

$$x^2 = \sum_{k=1}^{K} \frac{(o[k] - e[k])^2}{o[k] + e[k]}$$

- c: The weighting factor that is used for calculating the overall match probabilities, where 0 < c < 1.
- d: The number of requests in the initial transient for the exponential weighted moving average method, where the protection system starts to estimate the value of x' after d samples.
- AveR100MP: The average of 100 most recent match probabilities.
- AveOMP: The average of rest match probabilities, that consider all the match probabilities without the 100 most recent match probabilities.
- cMP: The final match probability, which is calculated using the Exponential Weighted Moving Average method, and is defined by:

$$\text{cMP} = (aveR100MP)c + AveOMP(1-c)$$

The pseudocode for the algorithm protecting against malicious attacks is given in Fig 7. The algorithm first calculates the empirical probabilities from equation (4). Next, the algorithm compares the normalized observed probabilities against normalized analytical probabilities. Then, the algorithm estimates the proportion of operational nodes, for which the chi-squared value is smallest using question (5). After that, the algorithm adjusts the proportion of operational nodes, based on the calculated currentMP using the Exponential Weighted Moving Average method. Lastly, the algorithm determines appropriate number of nodes r' which could compensate for non-operational nodes.

Protecting(O, n, m, r, y_0, c)
1 for $i \leftarrow 1$ to 15
2 $O[i] \leftarrow \frac{O[i]}{\sum_{k=1}^{15} O[i]}$
3 $r = 0$
4 for $j \leftarrow 1$ to 10
5 for $k \leftarrow 1$ to 15
6 $P[0.1 \times j][k] \leftarrow \frac{\binom{mx}{k}\binom{n-mx}{r-k}}{\binom{n}{r}}$
7 for $i \leftarrow 1$ to 15
8 $P[0.1 \times j][i] \leftarrow \frac{P[0.1\times j][i]}{\sum_{k=1}^{15} P[0.1\times j][i]}$
9 $chiSq[0.1 \times j] \leftarrow \sum_{l=1}^{15} \frac{(o[l]-P[0.1\times j][l])^2}{o[l]+P[0.1\times j][l]}$
10 $x' \leftarrow \min(chiSq[0.1], chiSq[0.2] \ldots chiSq[0.9], chiSq[1.0])$
11 $currentMP = (AveR100MP)c + AveOMP(1-c)$
12 if $currentMP > 0.9$ & $currentMP \leq 0.95$
13 $x' = x' - 0.1$
14 else if $currentMP > 0.85$ & $currentMP \leq 0.9$
15 $x' = x' - 0.2$
16 else if $currentMP > 0.8$
17 $x' = x' - 0.3$
18 if x' reaches t consecutive estimates
19 do
20 $r \leftarrow r + 1$
21 $y \leftarrow 1 - \frac{(n-mx')}{(n)}\frac{(n-mx'-1)}{(n-1)} \cdots \frac{(n-mx'-r+1)}{(n-r+1)}$
22 until $(y > y_0)$
23 $r' = r$

Figure 7 Method for protecting against malicious nodes

3.4 TCSC Message Routing System

Having nodes exchange messages with each other directly without passing through any intermediary is impractical and vulnerable to malicious attacks. Thus, we present a randomized message-routing system; every node has to adopt this system before it begin communicating with others. Our randomized message-routing system contains routing paths that are composed of a set of mutual nodes (i.e., the

nodes that know the sender and also the receiver) between the sender and the receiver. We call this randomized message routing system because we randomly pick nodes to check their mutual status. The steps for the routing protocol are described below and are illustrated in Figures 8-11.

1. Figure 8 shows Node A, Node B, and Node C with its friends in the network.
2. Node A decides to communicate with Node C. Therefore, Node A creates a "seeking" request containing Node C, and distributes its requests to a subset of nodes selected at random. In Figure 9, we can see that one of the seeking requests is sent to Node B.
3. All nodes that receive such requests from Node A determine whether Node C is in their membership. If there is a node that has Node C in its membership list, it replies "yes" to Node A. In Figure 10, Node B replies "yes" to Node A to announce that Node C is in its membership. If none of the nodes replies "yes" to Node A, Node A picks up another subset of nodes selected at random and distributes its "seeking" requests again.
4. Once Node A finishes collecting all the responses, it determines the final routing path to deliver its message to Node C (See Figure 11). It's possible that Node A might discover multiple mutual nodes between itself and Node C. In such a case, Node A will randomly choose the mutual peers to deliver the message from it to Node C. If however Node A does not receive any mutual node, it will go back to Step 2 to distribute its seeking requests to another subset of nodes. If however Node A cannot discover any mutual nodes after repeating Step 2 for 16 times, it will give up this discover process, and randomly select one of its peers to deliver the message to Node C.

The pseudocode for discoverPath() that discovers mutual nodes and creates a communication path is given in Fig 12, and the input values for the discoverPath() are described as follows:

- n: The total number of nodes in the network
- h: The number of hops
- f: The number of nodes in a node's view
- senderView: The sender's membership view
- receiver: The recipient that the sender is looking for

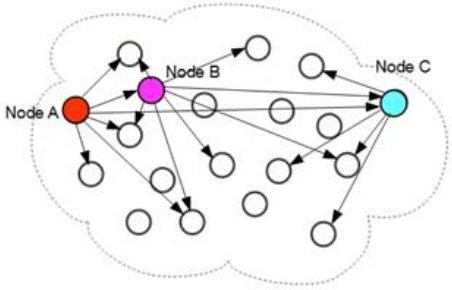

Figure 8 Nodes with its friends in the network. Node B is a mutual friend of both Node A and Node C.

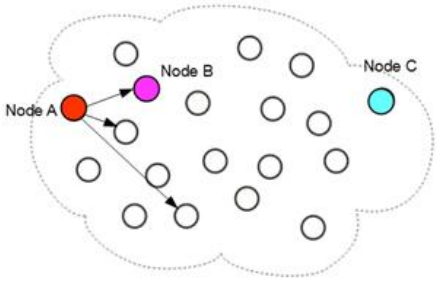

Figure 9. Nodes A first creates a seeking request containing Node C, then distributes its requests to a subset of nodes

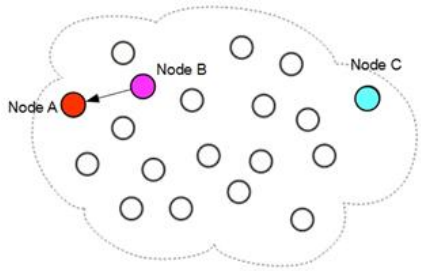

Figure 10 Node B replies back to Node A and announces that Node C is in its membership

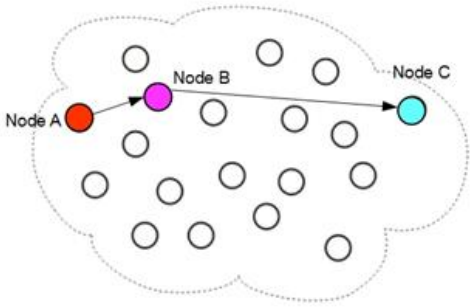

Figure 11 Subsequent messages sent from Node A will first pass through Node B, then deliver to Node C.

4 Tuning the Parameters

4.1 Determine appropriate r and m

Table 1 shows the probability p of one or more matches obtained from Equation 4 and its lower bound, when the metadata and the requests are distributed to [√n],[√2n],"and" [2√n] nodes, respectively. We first notice that the probability p of matches is low when metadata and requests are distributed to √nne nodes. Even though the probability p of matches increases when we increase r = m = √2n, the results are still not good enough. When r = m = 2√n, we notice that the probability p of matches achieves high values for all cases of n. Therefore, based on the result obtained from this table, we decide to set r = m = 2√n for our later experiments. In addition, it's worth noting that we only need to distribute metadata and requests to a small number of nodes such as r = m = 2√n, but can still obtain a probability of matches exceeding 0.9818.

```
discoverPath(n, f, h, senderView, receiver)
1  for i ← 1 to h
2    for j ← 1 to 2√n
3      for k ← 1 to 16
4        chosenNode ← random(senderView)
5        if chosenNode knows receiver
6          pushArray(chosenNode, responseArray)
7          break
8    if responseArray not empty
9      mutualNode = random(senderView)
10   else
11     mutualNode = random(responseArray)
12   pushArray(mutualNode, path)
13   receiver ← mutualNode
14 return path
```

Figure 12 Method for discovering mutual nodes and communication path

Table 1 Probability *p* when distributing to $r=m=[\sqrt{n}],[\sqrt{2n}]$, and $[2\sqrt{n}]$ nodes.

n \ r=m=	$[\sqrt{n}]$	$[\sqrt{2n}]$	$[2\sqrt{n}]$
20	0.6244	0.9225	0.9997
200	0.6507	0.8915	0.9896
2000	0.6450	0.8711	0.9842
20000	0.6325	0.8674	0.9828
200000	0.6326	0.8651	0.9819
2000000	0.6323	0.8649	0.9818
Lower Bound	0.6323	0.8649	0.9818

4.2 Determine appropriate r and m

In section 3.4, we employ the discoverPath() algorithm introduced from section 3.4 to three different scenarios, and determine the appropriate number of hops h to balance response time, message costs, and success rate. We apply the following performance metrics for our evaluations:

- Success rate: the success rate for discovering mutual nodes for the communication path. For every iteration, if a sender successfully creates its communication path, we set it as 1; otherwise we set it as 0. Our program then sums up the number of success rates divided by the total number of iterations.
- Message cost: The average number of message costs communicated for discovering mutual nodes and creating a communication path.
- Communication delay (ms): The communication delay that a node needs for it to discover mutual nodes and create a communication path, measured in milliseconds (ms).

We perform our experiments using five different values of h (h = 0; 1; 2; 3; 4) and three different values of f (f = 1999; 999; 99). We obtain mean success rate, message cost, and mean communication delay for the algorithm when the number of hops is high (i.e. h = 5), when the number of hops is low (i.e. h = 0), when every node has a complete view of the membership (i.e. f = 1999), and when every node has relatively few nodes in its membership view (i.e. f = 99). For these experiments, we set total number of nodes in the network as n = 2000.

Figure 13 shows the mean success rate, mean message cost, and mean communication delay for f = 1999; 999, and 99, respectively. At the top of Fig. 13, we see that the mean success rate for f = 99 decreases dramatically as the number of hops h increases, and this is because all the nodes have a mere view of the membership. In addition, the success rates for f = 999 and f = 1999 all remain at 1 for all the cases of h.

In the middle of Fig. 13, we see that the message cost increases as the value of h increases. In addition, the curve for f = 1999 increases much faster than the curves for f = 999 and f = 99. The reason is that, for the scenario with f = 1999, every node has a complete view of the membership, and therefore most of the nodes that receive discovery requests would inform the sender that they know the searching node. As a result, the message costs for f = 1999 would be greater. Conversely, for the scenario with f = 99 where every node only knows 99 nodes in its membership, few nodes would reply with matches, causing lower message costs. In addition, we discover that the message costs are too high for the values of h = 4.

The bottom of Fig. 13 shows that the communication delay increases as the number of hops h increases. In addition, we also notice that the communication delay for f = 99 almost remains 0, and the communication delay for f = 999 and f = 1999 grows linearly as h increases.

Based on these results, we conclude that the number of hops h = 2 is the appropriate number, such that it can ensure communication security and still achieve a high success rate, reasonable message cost, and low communication delay. Therefore, we have chosen h = 2 for our further experiments

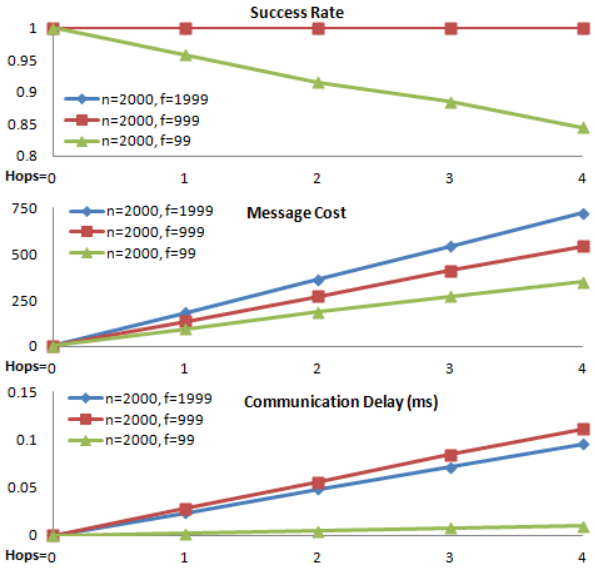

Figure 13 Successful rate, message cost, and communication delay for various number of hops h.

4.3 Determine appropriate t

As we discussed in section 3.3, t represents the number of successive estimates that indicates the same change in the value of x' before our protection system accepts that change and takes action to change the value of r'. We employ the following performance metrics for our evaluations:

- Accuracy: The number of correct estimates of operational nodes x' divided by the total number of estimates within an interval of requests.
- Response time: After a change in the actual proportion x of operational nodes, the number of requests before the system changes the estimated proportion x of operational nodes.

Table 2 Mean accuracy and response time for t = 1, 5, 6

t=	1	5	**6**
Accuracy	0.6344	0.8992	0.7527
Response time	27.4	42.5714	63.6667

Table 2 shows the mean accuracy and the mean response time when the action is taken immediately (t = 1), and when the action is delayed (t = 5 and t = 6). The values of other parameters for these experiments are: n = 2000, m = 89, r = 89, c = 0.97, d = 30, and K = 15, and with x = 1.0 initially. In our experiments, we evaluate our accuracy and response time with a scenario that begins with 500 requests where x = 1.0, then decreases to x = 0.9, x = 0.8, x = 0.7, x = 0.6, x = 0.5, x = 0.4, and finally decreases to x = 0.3, with another 500 requests for each decrease in the value of x.

In table 2, we see a useful increase in the mean accuracy as t is increased from 1 to 5, but a decrease in the accuracy as t is further increased to 6. The reason is that when we increase t to 6, it causes a delay in the correct estimate value of x', and overall accuracy is consequently reduced. In addition, we also see that there is an increase in the response time as t increased from 1 to 5, and a larger increase in the response time as t is further increased to 6. We see that the improvements in accuracy are more substantial when t = 5, and the increase in response time is reasonable. Thus, we chose t = 5 for our further investigation and experiments.

5 Performance Evaluation

In the performance evaluation, we consider the probability of a match using both analysis and emulation based on our HTTP implementation.

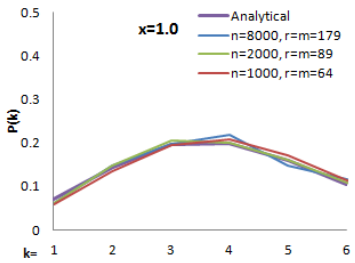

Figure 14 Analytical model vs. Emulation results for x=1.0

5.1 Evaluating robustness of TCSC

We adapted HTTP implementation and performed emulation experiments to validate Equation (4). In our emulation, we used libCURL to collect the match probabilities. In addition, all the requests are distributed and processed concurrently. The nodes are assumed to have enough memory to store the source files, metadata, and messages, which are then delivered reliably. In addition, we assume that all the participating nodes in the TCSC network have the same membership set, and all the nodes are online all the time.

Before we run our emulation program, we delete all resources and data from the SQLite databases. Next, the program adds all the nodes to the membership. Once all the nodes are added to the membership, we supply m number of nodes for distribution of metadata, r number of nodes for requests, and the x proportion of operational nodes, to the emulation program. Next, we have the source nodes to upload files and create the corresponding metadata. Then the program selects the nodes chosen at random for metadata distribution.

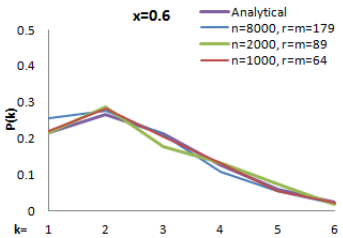

Figure 15 Analytical model vs. Emulation results for x=0.6

Next, the program randomly selects the nodes for the requests distribution. If one or more nodes returns a response, that means there is a match and the simulation program returns 1; otherwise, the simulation program returns 0. Our program then obtains the probability p that a node has a match using Equation 4.

Figure 14 and Figure 15 show the analytical and emulation results for probability p of k matches with n = 1000, n = 2000, n = 8000 with x = 1.0 and x = 0.6 values. We observe that the emulation results are very close to all the analytical results for both x = 1.0 and x = 0.6. Hence, we have demonstrated that our TCSC retains significant utility in a range of circumstances, even when a substantial proportion of the nodes is non-operational, which might be the circumstances in which the information is most needed.

5.2 Evaluating robustness of TCSC

We plot our result in Figure 16, in which n = 2000, m = 89, r = 89, c = 0.97, d = 30, t = 5, and K = 15, with x = 1.0 initially. The red curve represents the actual value of x, and the blue curve represents the estimated value of x'. The scenario starts with 500 requests where x = 1.0, and then decreases to x = 0.9, x = 0.8, x = 0.7, x = 0.6, x = 0.5, x = 0.4, and then finally decreases to x = 0.3, with another 500 requests for each decrease in the value of x.

In Fig. 16, we first note that when the actual value of x decreases, there is a short interval during which this change is not yet detected. In addition, we note that the change of incorrectly estimating the value of x', the mean response time, and the mean time to recover from an incorrect estimate are all small. Moreover, we discover that the incorrect estimate of x' is within plus or minus 0.1 of the actual value of x. Overall, these experiments demonstrate that our protection system can effectively detect malicious nodes, and that false alarms rarely occur.

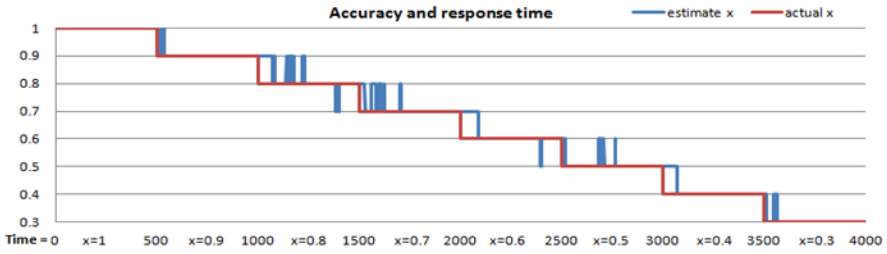

Figure 16. Accuracy and response time for the protection system for various values of x with n = 2000, m = 89, r = 89, c = 0:97, d = 30, t = 5, and K = 15

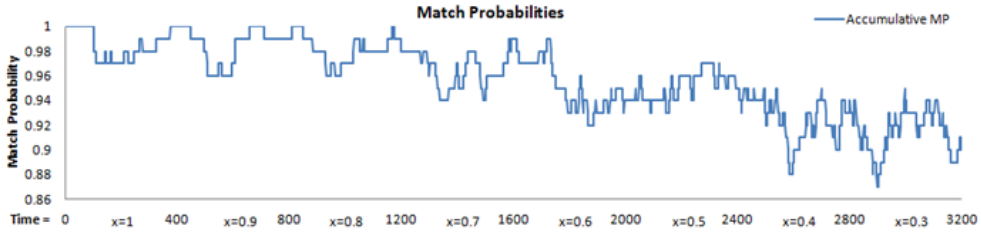

Figure 17. Accumulated match probabilities for the protection system for various values of x with n = 2000, m = 89, r = 89, c = 0:97, d = 30, t = 5, and K = 15

Next, we investigate the overall match probabilities p of our protection system. We consider the extended scenario in which n = 2000, m = 89, r = 89, c = 0.97, d = 30, t = 5, and K = 15, with x = 1.0 initially. The scenario starts with 400 requests where x = 1.0, and then decreases to x = 0.9, x = 0.8, x = 0.7, x = 0.6, x =

0.5, x = 0.4, and finally decreases to x = 0.3, with another 400 requests for each decrease in the value of x. Figure 17 shows the overall accumulated match probability p, which is calculated from Equation 6.

In Figure 17, we notice that there are some fluctuations in the match probabilities, and this is because the match probabilities for the intervals are bounded by changes in the value of x and the subsequent changes in the value of x'. In addition, the accumulated match probability p is quite high, and we can still obtain the probability of a match to exceed 0.88, even when we have x = 0.3 proportional of operational nodes in the system. Furthermore, the average of the match probabilities for this experiment is about 0.9576, which is relatively high.

Overall, our experiments demonstrate that our protection system is effective in detecting malicious nodes, and the accuracy for detecting the change in x' is high. Moreover, our protection system can effectively adjust the number of nodes to which the requests are distributed to maintain a high probability of a match. With appropriate choices in parameters, we can obtain high accuracy, reasonable response time, and high accumulated match probability

6 Conclusion

We have presented the architecture of TCSC, a trustworthy and communal social classifieds system, along with four system to eliminate the centralized control of both social network services and community-based classified systems. Our TCSC involves distributing of metadata and requests, matching of requests and metadata, retrieving information corresponding to the metadata, obtaining feedback of other peers beforehand, protecting against malicious attacks, and communicating to other peers with no direct communication.

Through several experiments, we have proved that our TCSC retains significant utility in circumstances in which a substantial proportion of nodes are non-operational, and have shown that our protection system can detect the proportion of nodes that are malicious, and can adaptively adjust the number of nodes to which the requests are distributed to maintain high match probability.

In the future, we plan to investigate the scalability of the TCSC system to thousands of nodes and, then, to extrapolate those results to millions of nodes. In addition, we plan to investigate other possible attacks on TCSC, and countermeasures to such attacks.

ACKNOWLEDGMENT

This research was supported in part by MOST 103-2410-H-194-064 and NSC 102-2410-H-194-118 of Ministry of Science and Technology, Taiwan. In addition, we would like to thank two undergraduate students, Wei-Cheng Li and Jian-Xun Huang, for setting up and developing the user interfaces of the TCSC system.

REFERENCES

[1] 2Peer (n.d.), http://2peer.com. Accessed: Mar. 20, 2015.

[2] Bamman, D., O'Connor, B. & Smith, N. (2012), 'Censorship and deletion practices in chinese social media', First Monday 17(3).

[3] Bianchi, S., Felber, P. & Gradinariu, M. (August 2007), Content-based publish/subscribe using distributed r-trees, in 'Proceedings of Euro-Par', Rennes, France, pp. 537–548.

[4] Buchegger, S. & Le Boudec, J.-Y. (2004), A robust reputation system for peer-to-peer and mobile ad-hoc networks, in 'Proceedings of P2PEcon', Vol. 2004.

[5] Buchegger, S., Schioberg, D.,Vu, L.-H.&Datta, A. (2009), Peerson: P2p social networking: early experiences and insights, in 'Proceedings of the Second ACM EuroSys Workshop on Social Network Systems', ACM, pp. 46–52.

[6] Chuang, Y. T., Melliar-Smith, P. M., Moser, L. E. & Lombera, I. M. (March 2015),'Maintaining censorship resistance in the itrust network for publication, search and retrieval', Peer-to-Peer Networking and Applications 8(2), 1–18.

[7] Chuang, Y. T., Melliar-Smith, P. M., Moser, L. E. & Lombera, I. M. (September 2012),'Protecting the iTrust information retrieval network against malicious attacks', Journal of Computing Science and Engineering 6(3), 179–192.

[8] Clarke, I., Sandberg, O.,Wiley, B. & Hong,T. (July 2001), Freenet: A distributed anonymous information storage and retrieval system, in 'Proceedings of the Workshop on Design Issues in Anonymity and Unobservability', Springer, Berkeley, CA, pp. 46–66.

[9] Condie, T., Kamvar, S. D. & Garcia-Molina, H. (August 2004), Adaptive peer-to-peer topologies, in 'Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing', Zurich, Switzerland, pp. 53–62.

[10] Cutillo, L. A., Molva, R. &Strufe, T. (2009), 'Safebook: A privacy-preserving online social network leveraging on real-life trust', Communications Magazine, IEEE 47(12), 94–101.

[11] Damiani, E., di Vimercati, D. C., Paraboschi, S., Samarati, P. & Violante, F. (2002), A reputation-based approach for choosing reliable resources in peer-to-peer networks, in'Proceedings of the 9th ACM Conference on Computer and Communications Security', pp. 207–216.

[12] Feller, W. (1968), An Introduction to Probability Theory and Its Applications, Vol. I, John Wiley & Sons.

[13] Ferreira, R. A., Ramanathan, M. K., Awan, A., Grama, A. & Jagannathan, S. (August 2005), Search with probabilistic guarantees in unstructured peer-to-peer networks, in 'Proceedings of 5th IEEE International Conference on Peer-to-Peer Computing', Konstanz, Germany, pp. 165–172.

[14] Gnutella, http://en.wikipedia.org/wiki/Gnutella. Accessed: Mar. 20, 2015.

[15] Guo, S., Wang, M. & Leskovec, J. (2011), The role of social networks in online shopping: information passing, price of trust, and consumer choice, in 'Proceedings of the 12th ACM conference on Electronic commerce', ACM, pp. 157–166.

[16] Gupta, A., Sahin, O. D., Agrawal, D. & El Abbadi, A. (2004), Meghdoot: Content-based publish/subscribe over P2P networks, in 'Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware', Toronto, Canada, pp. 254–273.

[17] Hu, J. & Burmester, M. (2009), Cooperation in mobile ad hoc networks, in 'Guide to Wireless Ad Hoc Networks', Springer, pp. 43–57.

[18] Jesi, G. P., Hales, D.&Van Steen, M. (July 2007), Identifying malicious peers before it's too late: A decentralized secure peer sampling service, in 'Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems', Boston, MA, pp. 237–246.

[19] Kavianpour, S., Ismail, Z., & Mohtasebi, A. (2011). Preserving Identity of Users In Social Network Sites By Integrating Anonymization And Diversification Algorithms. International Journal of Digital Information and Wireless Communications (IJDIWC), 1(1), 32-40.

[20] Leng, C., Terpstra,W.W., Kemme, B., Stannat,W. & Buchmann, A. P. (2008), Maintaining replicas in unstructured P2P systems, in 'Proceedings of the 4th International Conference on Emerging Networking Experiments and Technologies', Madrid, Spain, pp. 19–30.

[21] Mischke, J. & Stiller, B. (2004), 'A methodology for the design of distributed search in P2P middleware', IEEE Network 18(1), 30–37.

[22] Mislove, A., Post, A., Haeberlen, A. & Druschel, P. (2006), Experiences in building and operating epost, a reliable peer-to-peer application, in 'ACM SIGOPS Operating Systems Review', Vol. 40, ACM, pp. 147–159.

[23] Morselli, R., Bhattacharjee, B., Srinivasan, A.&Marsh, M. A. (July 2005), Efficient lookup on unstructured topologies, in 'Proceedings of the 24th ACM Symposium on Principles of Distributed Computing', Las Vegas, NV, pp. 77–86.

[24] Om, S., & Talib, M. (2011). Wireless Ad-hoc Network under Black-hole Attack. International Journal of Digital Information and Wireless Communications (IJDIWC), 1(3), 591-596.

[25] Press, W. H., Teukolsky, S. A.,Vetterling,W. T.&Flannery, B. P. (2007), Numerical Recipes in Fortran: The Art of Scientific Computing, Cambridge University Press, Cambridge, United Kingdom.

[26] Seong, S.-W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S. K., Chu, R., Dodson, B. & Lam, M. S. (2010), Prpl: a decentralized social networking infrastructure, in Proceedings of the 1st ACMWorkshop on Mobile Cloud Computing & Services: Social Networks and Beyond', ACM, p. 8.

[27] Shakimov, A., Lim, H., Caceres, R., Cox, L. P., Li, K., Liu, D. & Varshavsky, A. (2011), Vis-a-vis: Privacy-preserving online social networking via virtual individual servers, in 'Communication Systems and Networks (COMSNETS), 2011 Third International Conference on', IEEE, pp. 1–10.

[28] Story, L. & Stone, B. (2007), 'Facebook retreats on online tracking', The New York Times 30.

[29] Terpstra, W. W., Kangasharju, J., Leng, C. & Buchmann, A. P. (2007), Bubblestorm: Resilient, probabilistic, and exhaustive peer-to-peer search, in 'Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications', Kyoto, Japan, pp. 49–60.

[30] The Diaspora Project (n.d.), https://diasporafoundation.org/. Accessed: Mar. 20, 2015.

[31] Tran, D. A. & Pham, C. (August 2010), 'Enabling content-based publish/subscribe services in cooperative P2P networks', Computer Networks: The International Journal of Computer and Telecommunications Networking 52(11), 1739–1749.

[32] Verma, P. (2013), Serefind: a social networking website for classifieds, in 'Proceedings of the 22nd international conference on World Wide Web companion', International World Wide Web Conferences Steering Committee, pp. 289–292.

[33] Verma, P. (2014), Serefind: a crowd-powered search engine, in 'Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing', ACM, pp. 297–300.

[34] Viinikka, J. & Debar, H. (2004), Monitoring IDS background noise using EWMA control charts and alert information, in 'Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection', French Riviera, France, pp. 166–187.