P2P Meta-Recommenders: Aggregated Diversity Maximization as a Bulwark against Attacks on Reviewers

Khalid Alhamed [a,c], Markus Zanker [b] and Elmane Shakre and Marius C. Silaghi [c]
[a] *Information Technology Department, Institute of Public Administration, Riyadh, Saudi Arabia*
E-mail: hamedk@ipa.edu.sa
[b] *Alpen-Adria-Universität Klagenfurt*
E-mail: markus.zanker@aau.at
[c] *Computer Sciences and Cybersecurity Department, Florida Institute of Technology, 150 W. University Blvd. Melbourne, FL 32901*
E-mail: msilaghi@fit.edu

Abstract. We focus on the problem of selecting reviewers (or raters) that are considered by a recommender system (or a user) under the aspect of security. Malicious reviewers can exert unreasonable influence, and can bias online consumers unfairly against an attacked item or competitor. This paper proposes an approach where a meta-recommender maximizes the aggregated diversity of reviewers when deciding which reviews should be considered by a recommender system or an online consumer. This problem can be of interest in many domains where producers or service providers may seek advantages by compromising competitors with fake reviews or ratings such as tourism and hospitality industries or even free open-source software. A solution is proposed for users linked in social networks, such as unstructured P2P societies. In order to evaluate the proposed solution, we describe a mechanism of selecting reviewers of software updates such that not all end-users of a software are impacted by a potentially malicious strict subset of all available reviewers, and we experimentally assess resistance to attacks.

Keywords: Recommender system, peer-to-peer, open-source, security, attack, software update

1. Introduction

Ratings and reviews play an important role in motivating users into selecting a specific item or doing some action such as listening to a specific song or downloading a specific software. In collaborative recommender systems, these ratings and reviews are driven or estimated based on the opinions of other users, referred to as reviewers or raters [3]. Different techniques have been used to associate/link reviewers/raters to a user in order to provide personal recommendations and to increase the accuracy of the recommendations. In this paper, we focus on the problem of selecting reviewers (or raters) that are evaluated by a recommender system (or a user) under the aspect of security. Attackers can take over reviewers/raters of products to influence their adoption. For example, reviewers of restaurants and hotels can bias customers against competitors of the attacker. Maximizing the diversity and thereby the independence of reviewers, can help make such attacks more difficult. This problem can be of interest in many domains where producers or service providers may seek advantages by compromising competitors with fake reviews or ratings. In this paper, we investigate this problem by providing an example in the software updates domain. Reviewers can play an essential role in the mechanism for auto-updating agent software based on free open-source software (FOSS). Software updates can drop essential features, can introduce attack vectors, and sometimes have led even to immediate system failure (e.g.,

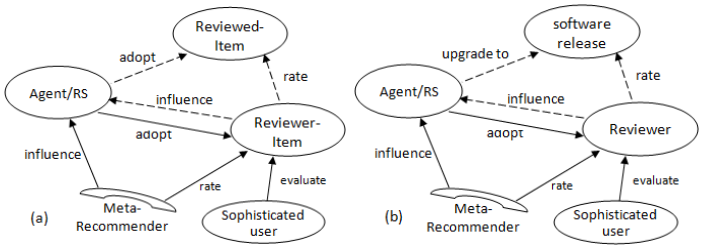

Fig. 1. a) Meta-recommendation, b) Updates case

Apache takeover in 2001). For some sensitive FOSS systems a review process can be put in place where experts act as reviewers/raters. Attackers can take over a centralized recommendation system of software updates, with the same ease with which they can take over the development. The goal of this research is therefore to provide a distributed meta-recommender that addresses security concerns by a rating and advertising mechanism for reviewers. Reviewers are thus treated as items (illustrated in Figure 1). The proposed technique can be readily integrated with peer-to-peer FOSS.

The rest of this paper is organized as follows. In the next section, we survey related work. Next the motivation for the aggregated diversity maximization is explained and we describe its implementation in a P2P meta-recommender. Finally, results from an experimental assessment on a synthetic dataset are reported.

2. Related Work

Recommender systems are investigated in academia and industry [2,15]. In this paper, we present a technique that can help to avoid attacks that can bias the recommendations. The proposed solution encompasses four areas related to recommender systems: collaborative filtering, distributed recommendations, diversity of recommendations, and attacks on recommender systems.

Collaborative filtering (CF): Collaborative recommender systems estimate the utility of items for a particular user, based on the items previously rated by *other users* who *shared* the same interests in these items in the *past* (i.e., it is based on the opinions of other users) [17,20]. In general, algorithms for collaborative recommendations can be classified into two groups [3]: (1) *Memory-based* (or heuristic-based) [3, 20] and (2) *Model-based* [12,3,11,16]. The core difference between heuristic-based and model-based approaches in collaborative filtering systems is that the heuristic-based techniques predict ratings based on heuristic rules that use the entire collection of previously rated items by the users for estimating the rates for unrated items, while, model-based techniques are based on a *model* learned by using machine learning and statistical techniques that analyze the underlying data (e.g., users' ratings, users' preferences, or items' features.). In the proposed solution, the suggestion of reviewers (meta-recommendations) to a user is based on other users (peers) opinions. It is a personalized recommendations in the sense that not all users (peers) get the same recommendations. However, the proposed technique of suggesting reviewers to users is different than previous techniques in the literature of the RS field. We emphasize more on increasing global stability and security rather than on increasing the accuracy level of the recommendations. Global stability and security are important qualities for some domains i.e., recommending software updates for peer-to-peer open-source projects (Section 5).

Distributed Recommendations: The distributed recommendation appears in peer-to-peer based recommender systems that distribute the *recommendations handling* between peers in the network. Each peer is equipped with a distributed recommender module that receives the underlying data (e.g., user ratings, user profiles, and item attributes.) supplied from the source peers (neighbor peers), and delivers data to the destination peers (neighbor peers) after some certain processing or decision making [23]. One example of such systems is proposed in [6,22], using a distributed collaborative filtering algorithm (PipeCF) based on a distributed hash table to build a scalable distributed recommender system. Another example of a distributed collaborative filtering algorithm is the one used by the PocketLens system [13], which introduced five peer-to-peer architectures to find neighbors. PEOR is a P2P collaborative filtering-based recommendation system for multimedia recommendations [9]. A distributed

recommender system (MUADDIB) based on community partitions is introduced in [18]. A distributed recommender system has also beed used for P2P knowledge sharing among collaborative team members [23].

In order to avoid that attackers take over a centralized recommendation system, we propose an approach of recommending reviewers to users based on distributed heuristic-based recommendations in a P2P environment.

Diversity of Recommendations: A way of evaluating the recommendation quality is based on the diversity of recommendations. The idea here is to suggest varied recommendations to users and to avoid restricting all users to the same sets of similar items [1]. This is unlike most of the algorithms used by recommender systems which aim to improving recommendation accuracy quality only, i.e., the Netflix Prize competition (Netflix.com). For example, in Social Network-Based Recommender Systems (SNRS) [8], depending on accuracy quality only can lead to a *social filter bubble effect* [14]. The term ”filter bubble” is used in social network to refer to users being surrounded only by friends who have similar interests, which leads them to adopt a similar sets of items (in our case, similar sets of reviewers). This effectively isolates users from adopting items out of their current social neighborhood. In order to reduce this effect, we enhance our recommendation model to give priority to reviewers outside a user's current social neighborhood. Thus, we use a *diversity heuristic* rule as a mechanism to improve recommendation diversification.

Among many different aspects that cannot be measured by accuracy metrics alone, we focus on the diversity of recommendations. In general, there are two ways to measure the diversity of recommendations: individual and aggregate [1]:

This paper proposes an approach where a meta-recommender maximizes the aggregated diversity of reviewers when deciding which reviews/ratings should be considered by a recommender system or an online consumer.

Attacks in recommender systems (RS): A survey of research into making recommender systems (RS) robust against attacks is described in [5]. Obviously malicious reviewers of software projects could be seen as a profile-injection attack in the sense of entering attack profiles to the software ecosystem. The presented work distinguishes itself from existing research on attacks in RS by lifting the recommendation one level higher and treating reviewers as items that can be recommended by a meta-recommender. We propose a diversity maximization strategy for meta-recommenders to increase stability and emphasize security.

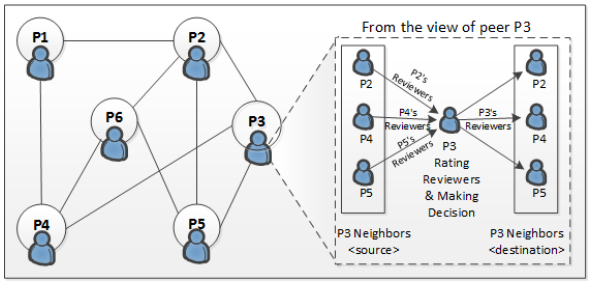

Fig. 2. Exchanging Reviewers Information in the Proposed Peer–to-Peer Recommender System

3. Motivation: Properties

The independence of the reviewers is the main principle that can help to maximize the security of the proposed system. In this section, we introduce heuristic rules that can enhance the independence of the reviewers.

Principle *[Decentralization] The recommendation procedure should not be under the control of a strict subset of users.*

Without this principle, an attacker controlling the recommender system can filter only reviewers that she controls. Even with a decentralized recommender, the recommendation criteria can be exploited to focus on a few reviewers (since they are few, they are easier to attack). A heuristic to help distribute the trust away from a small kernel is to take into account proximity.

Heuristic *[Proximity] Give priority to reviewers that are close to the user, in terms of some social network.*

Another heuristic gives priority to reviewers that are used by fewer neighbors, as a mechanism to improve diversification.

Heuristic *[Diversity] Give priority to reviewers that are used by fewer neighbors, in terms of some social network.*

4. Priority of Diversity for Security

The goal of the most widely used recommender systems is to automatically identify the items that best fit users' personal tastes [5]. Even when aggregated diversity is used as a target property, commonly it was used in the context where its importance was not significantly exceeding the accuracy [1,21].

Due to the security requirements for diversity in the type of problems addressed by our research, the aggregate diversity can have significantly more relevance than the accuracy. Our goal is to recommend items (reviewer_items) that increase diversity to increase the degree of the decentralization of items (reviewer_items) in the whole system. In other words, we focus here on the impacts of the recommendations (distribution of items) on the society of users, not only on individuals.

Next we investigate an example in the software updates domain which shows how the proposed P2P meta-recommendations diversifies reviewers/raters and limits the impact of attacks. However, the proposed solution is not limited to a single domain, it can be implemented in different domains such as business recommendation domain.

5. Example in Software Updates Domain

When accepting automatic updates for sensitive open source software, one can evaluate options based on reviewers. The security concern we address is that such reviewers can be biased by an attacker, and our goal here is to maximize the aggregate diversity of reviewer-items to increase the difficulty of such attacks. Aggregate diversity here is a heuristic for maximizing the independence of reviewers.

There are various ways to build systems that recommend reviewers to end-users. We investigate a distributed collaborative recommender CF system designed such that it favors the independence of the reviewers. As mentioned above, to address the single point of failure presented by centralized systems to attacks via corrupted reviewers, we propose a peer-to-peer based recommender system. This system distributes *recommendation management* between expert peers in the network and automatically assigns reviewers to unsophisticated users. For sophisticated users, the system simply recommends a set of reviewers but lets users manually select the reviewers that they use. Each peer is equipped with a recommender module that receives ratings and information about reviewers, supplied from the neighbor peers. It then delivers the reviewer's information along with ratings to the neighbor peers after a local re-rating and decision making (see Figure 2).

For peer-to-peer systems, such as PeerSoN [4], there exists an intrinsic social network as defined by the connections of each peer (e.g., peers and social-links in PeerSoN). Such a social network is supposed to provide the base of the proposed recommender system, which uses a heuristic neighborhood-based CF technique for rating prediction. In one such scheme, reviewers used by a peer are recommended to neighboring peers (directly linked peers) as shown in Figure 2.

The recommendation made to a peer for a reviewer has a weight given as a function on the weights coming on all its links (neighbors). *Sophisticated users* are those that overwrite this default for themselves by increasing or decreasing the weight manually (user-assigned ratings). Their recommendation is forwarded only if the user manually accepts using the recommended reviewer.

6. P2P Meta-Recommender Specifications

Our approach of recommending reviewers to users is based on distributed heuristic-based recommendations in a P2P environment. In this section, we formally identify the exchanged messages between peers and then describe the processing algorithms and decision making applied by each peer.

To exchange the information about reviewers between peers, we are now formalizing the concept of the **reviewer-item**. We assume that global identifiers (GIDs) are used to uniquely specify peers and reviewers (as done by common peer platforms).

Definition 1 (Reviewer-Item) *A reviewer-item is a tuple* $\langle \tau, \mathcal{A}, \mathcal{W}, d, \mathcal{P}, \mathcal{S} \rangle$*, where* τ *is the GID (public key) of the reviewer,* $\mathcal{A}$ *is the address where the reviewer can be contacted for retrieving her data,* $\mathcal{W}$ *is the recommendation weight (trust coefficient) associated with* τ *and made by the sender* $\mathcal{P}$ *(this weight has been automatically calculated by the sender's agent or manually assigned by the sender),* d *is the timestamp of the given weight,* $\mathcal{P}$ *is the GID of the sending peer, and* $\mathcal{S}$ *is the digital signature with which the sender authenticates the provided weight:*
$\mathcal{S} = SIGN(SK(\mathcal{P}), \langle \tau, \mathcal{A}, \mathcal{W}, d \rangle)$.

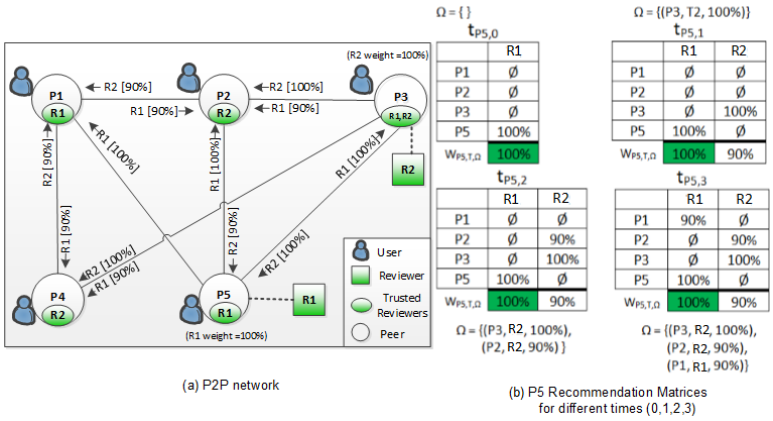

Fig. 3. Overall Architecture of the P2P Recommender System: Only applying the *Proximity Heuristic*

Here $SK(\mathcal{P})$ denotes the secret key of $\mathcal{P}$. Only the newest reviewer-item is stored by a peer for a given pair $\langle \tau, \mathcal{P} \rangle$, as per timestamp d.

6.1. Applying the Proximity Heuristic

At a user, each reviewer is associated with a weight (trust coefficient). Reviewers manually introduced by the user are given a fixed weight (e.g., 100%). This weight can decrease with each level of forwarding (using an *amortization coefficient*). This amortization is a mechanism to implement a version of the aforementioned heuristic, namely of giving priority to reviewers who are close to the user in terms of the social network (i.e., *proximity heuristic*). Proximity, in this context, plays the role of the social influence in social networks where such influence can be defined as a user behavior that leads users to adopt items as a result of a friend's adoption [10,19].

At a certain time $t_{p,m}$ (where p is the name of the peer and m is a logical time based on its local clock), peer p aggregates the received recommendations Ω and computes the new weight for each received reviewer τ_i based on Equation 1.

$$\mathcal{W}_{p,\tau_i,\Omega} = f \cdot \max_{j \in \Omega} \{\mathcal{W}_j | \tau_i = \tau_j\}, \tag{1}$$

where $\mathcal{W}_{p,\tau,\Omega}$ is the estimation of the weight given or calculated by peer p for reviewer τ based on the received recommendations Ω, f is the amortization factor, $\{a|b\}$ denotes the set of elements a for which the condition b holds, and $\max(A)$ is the maximum numerical element of the set A. In other words, peer p selects the maximum weight for reviewer τ received from neighbors of p (source peers) and then reduces the selected weight with f (the amortization factor). Based on this scheme, peer p gives high priority to reviewers who are introduced by close neighbors and low priority to those who are introduced by far neighbors (close and far neighbors are measured based on the underlying social network). Then peer p updates its recommendation matrix (see Figure 3.b). After calculating the new weight for all received reviewers, peer p will send only its used reviewers to its neighbors (destination peers).

For example, assuming the amortization coefficient is 0.9 (i.e., the trust coefficient is reduced by 10% for each new link in the chain of recommendation), the obtained recommendation in the agent system is shown in Figure 3. There are five peers (P1,...,P5) that use two reviewers: R1 and R2. The user of P5 introduces and uses R1 as a trusted reviewer and she has started giving R1 a 100% as weight. P3 introduces and uses the reviewer R2, to whom she also assigns a weight of 100%. Both P3 and P5 pass their selected reviewers information to neighboring peers (destination peers). In Figure 3, P3 announces R2 to her neighbor peers P2, P4, and P5. Also, P5 recommended R1 to her neighbor peers P1, P2, and P3. Based on these recommendations, each peer updates its recommendation matrix and decides which reviewer to use. For instance, Figure 3.b shows the recommendation matrix of peer P5 as it is changed (updated) from time:0 to time:3 (green cells in the matrix show reviewers used by the peer). At time $t_{P5,0}$, peer P5 had only one trusted reviewer R1 with weight = 100%, i.e., $\mathcal{W}_{P5,R1,\Omega} = 100\%$ (manually assigned by P5), where $\Omega = \{\}$ (P5 has not yet received any recommendations from neighbors). Later on, P5 updated its matrix in time $t_{P5,1}$ based on recommendations received from its neigh-

bors $\Omega = \{(P3, R2, 100\%)\}$. Therefore, the new matrix has a new column R2 and based on Equation 1; R2 is recommended with 90%, i.e., $\mathcal{W}_{P5,R2,\Omega} = 90\%$ (automatically predicted by the system). At that time, P3 had decided to trust only one reviewer R1. In time $t_{P5,2}$, P5 received new recommendations and updated its matrix based on recommendations received from its neighbors $\Omega = \{(P3, R2, 100\%), (P2, R2, 90\%)\}$ (note that $\mathcal{W}_{P5,R2,\Omega}$ still equals 90%). P5 keeps using only one trusted reviewer R1 for all time (time=0 to time=3). On the other hand, P4 has decided to use R2 as a trusted reviewer and forward R2's information to P1. In addition, P1 has evaluated R1 and R2, then decided to use R1 as a trusted reviewer (P1 had the choice to use R1[90%] or R2[81%] or both). However, P3 has decided to use R1 as a trusted reviewer beside R2.

6.2. Applying The Diversity Heuristic

Applying proximity through an amortization factor f as described in Equation 1 can lead to a *social filter bubble effect* [14]. The term "filter bubble" is used in social networks to refer to users being surrounded only by friends who have similar interests, which leads them to adopt a similar sets of items (in our case, reviewers). This effectively isolates users from adopting items out of their current social neighborhood. In order to reduce the *social filter bubble effect*, we give priority to reviewers less utilized in a user's current social neighborhood. Thus, we use a *diversity heuristic* rule as a mechanism to improve recommendation diversification.

Given a set Ω of n reviewer-items of the form $\langle \tau_i, \mathcal{A}_i, \mathcal{W}_i, d_i, \mathcal{P}_i, \mathcal{S}_i \rangle$ received from distinct neighbors of peer p, the current weight of the recommendation made to the user of peer p for reviewer τ_i is computed as:

$$\mathcal{W}_{p,\tau_i,\Omega} = f \cdot \left(1 - \frac{\#_{j \in \Omega} \{j | \tau_i = \tau_j\}}{K \cdot n}\right) \cdot \max_{j \in \Omega} \{\mathcal{W}_j | \tau_i = \tau_j\}, \quad (2)$$

where f is an amortization factor used to enforce proximity, $\#$ is the cardinality function, and K is a factor modeling the trade-off between proximity and diversity ($K \geq 1$).

For example, if all neighbors recommend only one and the same reviewer, based on Equation 2 that reviewer receives an aggregated recommendation with a weight given by half of the maximum weight received from neighbors (when $K = 2$), amortized with the factor f.

In another example, if the user has many neighbors and a given reviewer is recommended by only one of them, the received recommendation is a large fraction, $\frac{Kn-1}{Kn}$, of the weight received from that neighbor, amortized with f. This technique also can solve the *new-item* problem as appears in many recommender systems [7], where the new introduced reviewer can be distributed with a high weight.

Note: With the described mechanism, once a user influences many of her neighbors to switch to the same reviewers as she selected, then she will be recommended different reviewers (for diversification), and the recommendation she has for the original reviewers is decreased. To avoid a cycle of switching reviewers, we implement the probabilistic replacement mechanism described further in the next section. Also this does not lead to an iterative decrease to 0 of all recommendation weights, as some weights are pinned at certain values by users manually setting values for them.

6.3. The Frequency of Reviewer Replacement

Recommendation for a peer p is done by sorting all received reviewer-items in a list called "Known Reviewers List" ($L_K(p)$) based on their recommendation scores in decreasing order and recommending the top-N items as a list called "Used Reviewers List," which is denoted as $L_U(p) = \{\tau_1, \cdots, \tau_N\}$. This process is detailed below:

6.3.1. Known Reviewers List

All the reviewers known by a peer are stored in a list $L_K(p)$, sorted by their weight. Based on the Equation 2, each peer p evaluates the weight of the reviewers in $L_K(p)$ based on the recommendations received from links (neighbors):

$$L_K(p) \xleftarrow{sort} \forall \tau \in \Omega, \quad \mathcal{W}_{p,\tau,\Omega} \quad (3)$$

$L_K(p)$ items are stored as a pair $\langle \tau, \mathcal{W} \rangle$.

6.3.2. Used Reviewers List

Each peer maintains a list of used reviewers, $L_U(p)$. Advanced users populate this list manually while remaining users get their $L_U(p)$ list populated automatically by the distributed recommender system based on

Equation 4.

$$L_U(p)[N] \xleftarrow{Pr} L_K(p) \left[\underset{i, L_K(p)[i] \notin \widehat{L_U(p)}}{\operatorname{argmin}} \mathcal{W}_{p, L_K(p)[i], \Omega} \right] \tag{4}$$

With the software updates problem, the more different reviewer configurations are adopted by a peer, the higher is the risk that a configuration controllable by an attacker is eventually selected. In order to reduce the number of configurations with different sets of reviewers, a peer only switches at most a single reviewer at a time, only to a higher ranked one, and only with a probability Pr. This mechanism increases the *stability* of the system by reducing the switching frequency of reviewers in $L_U(p)$, and by helping to reach convergence. Formally, each peer p updates its trusted reviewers list $L_U(p)$ based on the following procedure. Note that there are two cases of populating the $L_U(p)$ list:

1. When the current used reviewer list $\widehat{L_U}(p)$ is empty or is not fully populated (the size of $\widehat{L_U}(p)$ is less than the threshold N). Here $L_U(p) \leftarrow \widehat{L_U}(p)$ and the missing positions in $L_U(p)$ are filled with the top elements in $L_K(p)$.
2. When $\widehat{L_U}(p)$ already has N reviewers. Here $L_U(p) \leftarrow \widehat{L_U}(p)$, except that the reviewer $\widehat{L_U}(p)[N]$, the one with the lowest weight, is a candidate for being replaced by the highest weight reviewer $L_K(p)$[i], as long as $L_K(p)$[i] does not exist in $\widehat{L_U}(p)$. Switching reviewers from list $\widehat{L_U}(p)$ is done with a certain probability, Pr (e.g., if $Pr = 0.5$, then the chance of replacing $\widehat{L_U}(p)[N]$ with a new reviewer from $L_K(p)$ is 50%).

6.4. Reviewer's Life-Cycle

Each reviewer in the system has a life-cycle that starts by having him introduced to the system and ends when there are no more recommendations of that reviewer distributed across the system. There are three stages:

1) Introducing reviewer stage: each reviewer is introduced to the system with mechanisms that depend on the given application. The reviewer information is distributed (disseminated) between users in the P2P evaluation network only after it is adopted by some sophisticated user (introducer peer).
2) Reviewer-item distribution stage: a reviewer-item is passed from one peer to another in the system and the weight of the reviewer is changed each time by the receiver peer, based on Equation 2. This weight can increase or decrease to satisfy the system constraints/properties (e.g., diversity, proximity, ...).
3) Termination stage: A reviewer can eventually be discarded by the system if the reviewer's expiration date is passed. Each reviewer is tagged with an expiration date that can be updated by retrieving the reviewer status from the reviewer's reference URL.

7. Evaluation And Experiments

In order to evaluate the performance of our distributed recommender system for reviewers, we simulate instances with the following characteristics:

- 10000 peers ($P_0..P_{9999}$) divided into 100 neighborhoods (of 100 peers each). For example, $P_0..P_{99}$ are in neighborhood N_1, $P_{100}..P_{199}$ in neighborhood N_2, etc.
- Each peer is linked to 50 other peers; 48 of these links are within the neighborhood of the peer (only two of the links are out of the neighborhood).
- 100 reviewers ($T_0..T_{99}$), each introduced by a peer as follows: P_{100i} manually selects T_i for usage with weight 100%.
- Each peer not manually selecting its reviewers automatically uses at most 10 reviewers (i.e. $N = 10$), as per our mechanism.
- We evaluate the behavior for multiple values of the parameters f, k, and p.

Our simulation works in rounds (intuitively the decisions taken in one round corresponds to the decision taken at a fixed interval by each peer, e.g., each day). In each round of the simulation, all peers synchronously evaluate recommendations from their links and decide new usage. We ran 100 such rounds for each experiment and then analyzed the results.

7.0.1. Evaluation of Distribution

In the first reported experiment, we show the distribution (usage) of reviewers among peers after running the simulation with 100 rounds. In this experi-

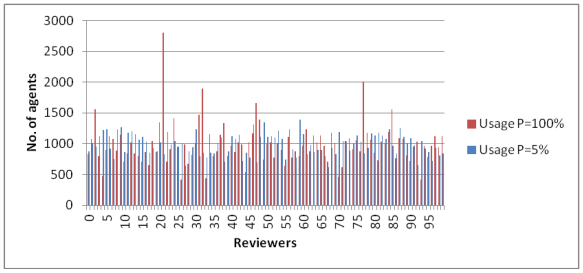

Fig. 4. Number of users trusting each reviewer.

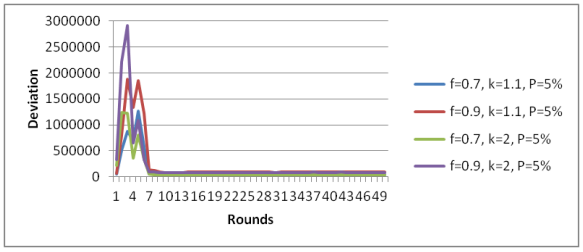

Fig. 5. Global Stability

ment, each peer switches to the top 10 reviewers recommended to her in the round. The other parameters are $k = 2$ and $f = 0.9$. As one can see in Figure 4 the standard deviation for reviewer usage is 331. For example (in the worst scenario), assuming an attacker peer introduced the malicious reviewer T_{21}, one ends up with 2803 peers using T_{21} (maximum usage in this experiment). We also ran an experiment with $p = 5\%$, $k = 2$, and $f = 0.9$. One can see in Figure 4 that the standard deviation for reviewer usage is reduced to 173. Now (in the worst scenario) when an attacker peer introduces a reviewer T_{59}, then T_{59} is used by 1393 peers (maximum usage in this experiment), which reduces its negative effects by more than 50%. Also, the usage of each reviewer tends to be distributed over several neighborhoods. For example T_{86} is used by 846 peers where only 50 peers are from the same neighborhood N_{86}.

7.0.2. *Global Stability*

In this experiment, we compute the local deviation between reviewers' weights in the previous list of used reviewers $(L_U(p_j)_{time(k-1)})$ and the new $(L_U(p_j)_{time(k)})$ for peer p_j:

$\left|L_U(p_j)_{time(k)} - L_U(p_j)_{time(k-1)}\right|$. Then we compute the global deviation by summing up all local deviations for each peer $p_j \in P$, where P is a set of all peers in the system, as in Equation 5. For example, let's assume that the previous list of used reviewers for peer p_j is as follows: $L_U(p_j)_{time(k-1)} = [(T1 : 40\%), (T2 : 70\%), (T3 : 20\%)]$ and the new list is as follows: $L_U(p_j)_{time(k)} = \{(T1 : 50\%), (T2 : 60\%), (T4 : 80\%)\}$. In this example, the local reviewers' weight deviations for peer p_j can be calculated as follows: |(T1:50 - T1:40) + (T2:60 - T2:70) + (T3:0 - T3:20) + (T4:80 - T4:0)| $= |(10) + (-10) + (-20) + (80)| = |60|$. Simulations with 50 rounds each are performed for the parameters $f = 0.7$ and $f = 0.9$. Here we use $p = 5\%$. In Figure 5 we report results for $k = 2$ and $k = 1.1$. The experiment suggests that the best parameter values in these situations are $f = 0.7$ and $k = 2$.

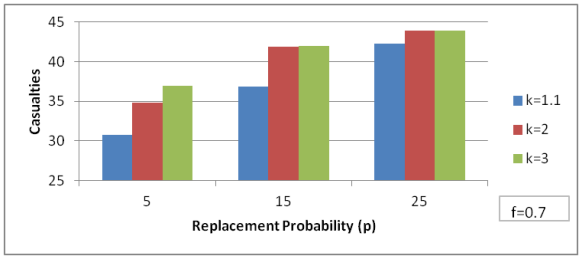

Fig. 6. Casualties after 50 rounds where f=0.7

$$\sum_{j=0}^{n-1} \left|L_U(p_j)_{time(k)} - L_U(p_j)_{time(k-1)}\right| \qquad (5)$$

7.0.3. *Casualties*

For this experiment, we introduced 20 malicious reviewers (controlled by an attacker). We want to count the number of peers who eventually end up making decisions relying on these malicious reviewers. Here, we consider that any peer who has more than half of its Used Reviewers List $(L_U(p))$ from the introduced malicious reviewers is an *affected peer*. This indicates that the majority of its reviewers are controlled by the attacker. An experiment based on simulations with 50 rounds each, for the parameter $f = 0.7$, is used to detect the best value for the parameters p and k. The results in Figure 6 are averaged over 10 instances for each of the values 5%, 15%, and 25% for the parameter p and the values 1.1%, 2%, and 3% for the parameter k. The experiment suggests that the lowest number of affected peers can be achieved with low values for parameters p and k in this situation is $p = 5\%$ and $k = 1.1$ as shown in Figure 6.

8. Conclusions

We design and propose a distributed meta-recommender system for advertising reviewers based on heuristics of

proximity and diversity meant to improve the chances of independence of the used reviewers. The independence of the used reviewers is essential for the resistance to attacks where reviewers can be biased to influence end users.

A set of metrics is defined for quantifying the promise of the investigated distributed recommendation system. These are: Distribution, Local Stability, Global Stability, and Casualty Rate. It is also shown how using simulations we have evaluated the parameters of the proposed mechanisms for distributed meta-recommendation of reviewers for software updates.

References

[1] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE TKDE*, 2012.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[4] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*. ACM, 2009.

[5] R. Burke, M. O. Mahony, and N. Hurley. Robust collaborative recommendation. In F. Ricci, L. Rokach, and B. Shapira, editors, *Handbook on Recommender Systems*, chapter 28, pages 969–999. Springer, 2015.

[6] P. Han, B. Xie, F. Yang, and R. Shen. A scalable {P2P} recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, 27(2):203 – 210, 2004.

[7] S. Hauger, K. Tso, and L. Schmidt-Thieme. Comparison of recommender system algorithms focusing on the new-item and user-bias problem. In *Data Analysis, Machine Learning and Applications*. Springer Berlin Heidelberg, 2008.

[8] J. He and W. Chu. A social network-based recommender system (snrs). In N. Memon, J. J. Xu, D. L. Hicks, and H. Chen, editors, *Data Mining for Social Network Data*, volume 12 of *Annals of Information Systems*, pages 47–74. Springer US, 2010.

[9] J. K. Kim, H. K. Kim, and Y. H. Cho. A user-oriented contents recommendation system in peer-to-peer architecture. *Expert Systems with Applications*, 34(1):300 – 312, 2008.

[10] W. Lu, S. Ioannidis, S. Bhagat, and L. V. Lakshmanan. Optimal recommendations under attraction, aversion, and social influence. In *ACM 2014 International Conference on Knowledge Discovery and Data Mining (SIGKDD 2014)*, 2014.

[11] B. M. Marlin. Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul, and B. SchÃűlkopf, editors, *Advances in Neural Information Processing Systems 16*, page None. MIT Press, Cambridge, MA, 2003.

[12] D. B. Michael. Learning collaborative information filters, 1998.

[13] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, July 2004.

[14] T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan. Exploring the filter bubble: The effect of using recommender systems on content diversity. In *the 23rd Intl. Conf. on WWW*. ACM, 2014.

[15] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *EXPERT SYSTEMS WITH APPLICATIONS*, 39(11), Sept. 2012.

[16] D. Y. Pavlov and D. M. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *In Proceedings of Neural Information Processing Systems*, pages 1441–1448. MIT Press, 2002.

[17] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Mar. 1997.

[18] D. Rosaci, G. M. L. Sarné, and S. Garruzzo. Muaddib: A distributed recommender system supporting device adaptivity. *ACM Trans. Inf. Syst.*, 27(4), 2009.

[19] S. Shang, P. Hui, S. R. Kulkarni, and P. W. Cuff. Wisdom of the crowd: Incorporating social influence in recommendation models. *CoRR*, abs/1208.0782, 2012.

[20] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[21] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 109–116, New York, NY, USA, 2011. ACM.

[22] B. Xie, P. Han, F. Yang, R.-M. Shen, H.-J. Zeng, and Z. Chen. Dcfla: A distributed collaborative-filtering neighbor-locating algorithm. *Information Sciences*, 177(6):1349 – 1363, 2007.

[23] L. Zhen, Z. Jiang, and H. Song. Distributed recommender for peer-to-peer knowledge sharing. *Information Sciences*, 180(18):3546 – 3561, 2010.