Automated conflict resolution in collaborative data sharing systems using community feedbacks

Fayez Khazalah a, Zaki Malik a,⇑, Abdelmounaam Rezgui b

aWayne State University, Detroit, MI 48202, USA
bNew Mexico Tech., Socorro, NM 87801, USA

Article info
Article history:
Received 10 February 2014
Received in revised form 12 September 2014
Accepted 21 November 2014
Available online 5 December 2014

Keywords:
Data sharing
Conflict resolution
Trust
Reputation

Abstract
In collaborative data sharing systems, groups of users usually work on disparate schemas and database instances, and agree to share the related data among them (periodically). Each group can extend, curate, and revise its own database instance in a disconnected mode. At some later point, the group can publish its updates to other groups and get updates of other ones (if any). The reconciliation operation in the CDSS engine is responsible for propagating updates and handling any data disagreements between the different groups. If a conflict is found, any involved updates are rejected temporarily and marked as deferred. Deferred updates are not accepted by the reconciliation operation until a user resolves the conflict manually. In this paper, we propose an automated conflict resolution approach that depends on community feedbacks, to handle the conflicts that may arise in collaborative data sharing communities, with potentially disparate schemas and data instances. The experiment results show that extending the CDSS by our proposed approach can resolve such conflicts in an accurate and efficient manner.

1. Introduction

A collaborative data sharing system facilitates users (usually in communities) to work together on a shared data repository to accomplish their (shared) tasks. Users of such a community can add, update, and query the shared repository [17] (please see [37,11,2,45,20] for examples of some collaborative projects). While the shared database evolves over time and users extend it continuously, it may contain inconsistent data, as users may have different beliefs about which information is correct and which is not [18]. While a relational database management system (RDBMS) can be used to manage the shared data, RDBMSs lack the ability to handle such conflicting data [17].

In most scientific communities [26,44,21,25,28], there is usually no consensus about the representation, correction, and authoritiveness of the shared data and corresponding sources [25]. For example, in bioinformatics, various sub-communities exist where each focuses on a different aspect of the field (e.g., genes, proteins, diseases, organisms, etc.), and each manages its own schema and database instance. Still these sub-disciplines may have sharing links with their peer communities (e.g., a sharing link between genes and proteins sub-communities). A collaborative data sharing system thus needs to support these communities (and associated links), and provide data publishing, import, and reconciliation support for inconsistent data.
Traditional integration systems usually assume a global schema such that autonomous data sources are mapped to this global schema, and data inconsistencies are solved by applying conflict resolution strategies ([36,6,4,35,5,34] are example systems). However, queries are only supported on the global schema and these systems do not support any kind of update exchange. To remedy this shortcoming, peer data management systems [3,22] support disparate schemas, but are not flexible enough to support the propagation of updates between different schemas, and handling data inconsistency issues. In contrast, a collaborative data sharing system (CDSS) [26,44,21,25] allows groups of scientists that agree to share related data among them, to work on disparate schemas and database instances. Each group (or peer) can extend, curate, and revise its own database instance in a disconnected mode. At some later point, the peer may decide to publish the data updates publicly to other peers and/or get the updates from other peers. The reconciliation process in the CDSS engine (that works on top of the DBMS of each participant peer) is responsible for propagating updates and handling the disagreements between different participant peers. It publishes recent local data updates and imports non-local ones since the last reconciliation. The imported updates are filtered based on trust policies and priorities for the current peer. It then applies the non-conflicting and accepted updates on the local database instance of the reconciling peer. For the conflicting updates, it groups them into individual conflicting sets of updates. Each update of a set is assigned a priority level according to the trust policies of the reconciling peer. The reconciliation process then chooses from each set, the update with the highest priority to be applied on the local database instance, and rejects the rest. When it finds that many updates have the same highest preference or there is no assigned preferences for the updates in a set, it marks those updates as “deferred”. The deferred updates are not processed and not considered in future reconciliations until a user manually resolves the deferred conflicts.

1.1. Problem description

The administrator of each peer in a CDSS is usually responsible for declaring and managing trust policies. While the administrator can be expected to define trust policies for a small number of participant peers, the same is not true for a large number of participants. In addition, assuming that a community of hundreds or thousands of members can authorize a user or a group of users to define trust policies for their community may not be plausible. Moreover, a CDSS does provide a semi-automatic conflict resolution approach by accepting the highest-priority conflicting updates, but it leaves for individual users the responsibility of resolving conflicts for the updates that are deferred. However, the assumption that individual users can decide how to resolve conflicting updates is not strong, as users of the community may have different beliefs and may agree or disagree with each other about which conflicting updates to accept and why (i.e., on which bases). Therefore, the challenge lies in providing a conflict resolution framework that requires minimal or no human intervention.

1.2. Major contributions

In light of the above discussion, we propose a conflict resolution approach that uses community feedbacks to handle the conflicts that may arise in collaborative data sharing communities, with potentially disparate schemas and data instances. The focus is to allow the CDSS engine to utilize the feedbacks for the purpose of handling conflicting updates that are added to the deferred set during the reconciliation process. We list our primary contributions below:

- We define a novel conflict resolution approach that extends the CDSS to automate the resolution of conflicts in the deferred set of a CDSS’s reconciling peer. We define a distributed trust mechanism to compute the weight for each conflicting update.
- We provide results for a Java-based implementation of our approach that mimics a community of CDSS peers with disparate schemas and sharing needs.
- We compare our approach with similar techniques to show its applicability for real-world scenarios.

The remainder of the paper is organized as follows. Section 2 presents an overview the related works, followed by the proposed approach for automated conflict resolution in a CDSS (in Section 3). An illustrative example of the proposed approach is introduced in Section 4, which is further used in Section 5 to present an experimental evaluation of the proposed approach. We then conclude the paper in Section 6 and provide brief directions for future work.

2. Related work

In this section, we provide a brief overview of related literature on conflict resolution and trust management in peer-oriented environments. Approaches for the problem of inconsistent data have been described in detail in the context of traditional data integration systems. For instance, [36,6,4,35,5,34] described different approaches to conflict resolution while integrating heterogeneous database sources (see [7] for a comprehensive survey about conflict classifications, strategies, and systems in heterogeneous sources).

Approaches for handling conflicts in community shared databases, based on the concept of multi-versioned database, are described in [17,39,18]. In [17], a BeliefDB system enables users to annotate existing data or even existing annotations, by adding their own beliefs that may agree or disagree with exiting data or annotations. A belief database contains both base...
data in the form of tuples and belief statements that annotate these tuples. It also represents a set of belief worlds, where each world belongs to a different user. Moreover, a belief-aware query language is introduced to represent queries over a belief database. This query language can be used to retrieve facts that are believed or not believed by a particular user. It also can be used to query for the agreements or disagreements on particular facts between users. An algorithm is also proposed in [17] to translate belief database queries into equivalent relational database queries.

Ref. [18] describes an automatic conflict resolution based on trust mappings between users. A user usually has trust relationships with other users in the community. A user also assigns different trust priorities for different trusted users. To resolve a conflicting data, a user accepts a data value that comes from the most trusted user. Thus, each user is shown his own consistent version of the shared database based on his trust mappings and priorities with other users.

Ref. [39] handles inconsistent data by allowing users to rate data. Updates done by users are stored in a shared, uncertain database, where all versions of conflicting updates are inserted into the database in parallel. In other words, all update operations, whether insertion, replacement, or deletion, are treated as insertion operations. Users in [39] can update, query, and even rate the quality of updates, based on their own beliefs. The rating is usually weighted according to the reputation of the user who does the rating. Conflicting updates are usually various versions of the same tuple, sharing the same key, but having different values for non-key attributes. For each version of a tuple, the ratings of different users are collected, and the average rating for this version is computed. The reputation of a user who initiates the rated update can be then computed by comparing aggregate ratings of his updates to aggregate ratings of others. The computation of a user’s reputation is incrementally, such that a new reputation value is computed for the user each time a new rating arrives. For answering a query from a user, the average rating of each consistent version (or world) of the database is computed, and the best rated world is found. After that, a user query is answered according to this consistent version of the database.

The work done in [39] is similar to that of [17,18], in that all apply a multi-versioned database model to resolve conflicts. However, each user in [17,18], based on his own beliefs or trust mappings, sees his own consistent version of the shared database. In contrast, all users in [39] see the most consistent version of the database which has the best rating. Our approach is similar to [39] in that it also deploys community feedback to resolve conflicts. However, we only deploy the community feedback for the purpose of resolving conflicts between the updates of conflict groups in the deferred set of a local peer. Moreover, our approach is based on the CDSS, where each participant peer maintains a relational and consistent database instance, where conflicts between data are not allowed due to the restrictions of the relational DBMS. On the other hand, [39] deploys the concept of uncertain and multi-versioned database, such that all conflicting updates are kept permanently in the same database, and users’ queries are answered based on the combination of updates that have the best rating.

Over the years, several research initiatives have worked on the modeling, data collection, data storage, communication, and assessment related problems for reputation management. These efforts have not been limited to the field of computer science. To name a few, economics, marketing, politics, sociology, psychology, etc. have all studied reputation in one context or the other [16,13,40,19]. In the recent past, these research activities have gained momentum.

In computer science, reputation has been studied both in theoretical areas and practical applications [24,47,41,12]. Theoretical literature mainly focuses on studying the properties of systems based on reputation. For example, results from game theory demonstrate that there are inherent limitations to the effectiveness of reputation systems when participants are allowed to start over with new names [51]. In [23], the authors study the dynamics of reputation, i.e., growth, decay, oscillation, and equilibria. Practical literature on reputation is mainly concerned with the applications of reputations. Major applications where reputation has been effectively used include e-business, peer-to-peer (P2P) networks, grid computing systems [1], multi-agent systems [42], Web search engines, and ad hoc network routing [8,29]. In the following, we give a brief overview of a few reputation management frameworks for P2P systems and Web services since these are closely related to our approach.

PeerTrust [51] is a P2P reputation management framework used to quantify and compare the trustworthiness of peers. In PeerTrust, the authors have proposed to decouple feedback trust from service trust, which is similar to the approach undertaken in this paper. Similarly, it is argued that peers use a similarity measure to weigh opinions of those peers highly who have provided similar ratings for a common set of past partners. However, this may not be feasible for large P2P systems, where finding a statistically significant set of such past partners is likely to be difficult [33]. Consequently, peers will often have to make selection choices for peers which have no common information in the system.

In [27], the EigenTrust system is presented, which computes and publishes a global reputation rating for each node in a network using an algorithm similar to Google’s PageRank [38]. Each peer is associated with a global trust value that reflects the experiences of all the peers in the network with that peer. EigenTrust centers around the notion of transitive trust, where feedback trust and service trust are coupled together. Peers that are deemed honest in resource sharing are also considered credible sources of ratings information. This is in contrast with our approach and we feel this approach may not be accurate. Moreover, the proposed algorithm is complex and requires strong coordination between the peers. Another major limitation of EigenTrust is that it assumes existence of pre-trusted peers in the network.

PowerTrust [53] is a “distributed version” of EigenTrust. It states that the relationship between users and feedbacks on eBay follow a Power-law distribution. It exploits the observation that most feedback comes from few “power” nodes to construct a robust and scalable trust modeling scheme. In PowerTrust, nodes rate each interaction and compute local trust values. These values are then aggregated to evaluate global trust through random walks in the system. Once power nodes are identified, these are used in a subsequent look-ahead random walk that is based on Markov chain to update the global trust values. Power nodes are used to assess the reputation of providers in a system-wide absolute manner. This is in contrast with
our approach where each consumer maintains control over the aggregation of ratings to define a provider’s reputation. Moreover, PowerTrust requires a structured overlay (for DHT), and the algorithms are dependent on this architecture. In contrast, service-oriented environments or the Web in general do not exhibit such structure.

The XRep system proposed in [14] uses a combination of peer-based reputations and resource-based reputations to evaluate a peer’s honesty. This scheme, storage overheads are substantially high while incorporating resource-based reputations, as the number of resources is significantly more than the number of peers. Moreover, the experiments consider a Zipf (non-uniform) distribution of resources and peers. However, it may not be practical to consider a single resource to be widespread enough to have a sufficient number of ratings in the system. Similar to our approach, XRep uses cluster computing to weigh feedbacks and detect malicious parties. However, no formalized trust metric is discussed in the paper.

REGRET [42] is a reputation system that adopts a sociological approach for computing reputation in multi-agent societies in an e-commerce environment. Similar to our approach where the nature of the community effects the service’s reputation, REGRET employs both individual and social components of social evaluations where the social dimension refers to reputation inherited by individuals from the groups they belong to. However, the proposed scheme requires a minimum number of interactions to make correct evaluations of reputation. It is likely that partners will not interact the minimum number of times to provide a reliable result. Moreover, the problem of malicious raters is not studied.

In [32], a distributed model for Web service reputation is presented. The model enables a service’s clients to use their past interactions with that service to improve future decisions. It also enables services’ clients to share their experience from past interactions with Web services. Agents are associated with each Web service, that act as proxies to collect information on and build a reputation of a Web service. The authors present an approach that provides a conceptual model for reputation that captures the semantics of attributes. The semantics includes characteristics, which describe how a given attribute contributes to the overall rating of a service provider and how its contribution decays over time. A similar reputation-based model using a node’s first hand interaction experience is presented in [41]. The reputation building process in [41] is similar to our approach. However, the proposed reputation model may not be completely robust and may not provide accurate results. First, the individual experience takes time to evolve over repeated interactions. Second, no distinction is made between the node’s service credibility in satisfying consumer requests and its rating credibility. It may be the case that a node performs satisfactorily but does not provide authentic testimonials. We provide an extensive mechanism to overcome these and similar inadequacies.

In [31], a trust model based on a shared conceptualization of quality of service (QoS) attributes is presented. The model shares the need for ontologies with our presented model. However, it lacks some important features that are central to our proposed model. The proposed reputation model lacks complete automation of feedback reporting. Human participation is necessary for rating Web services. Moreover, all agents that report reputation ratings are assumed to be trustworthy. Similarly, the common agencies to whom these ratings are communicated for sharing/aggregation are also expected to behave honestly. In our model, no such simplifying assumption is made. We calculate the reputation of a provider based on the testimonies of both trusted and malicious raters. We provide an elaborate method to measure the credibilities of service raters. The credibility-based scheme allows us to assign more weights to the trustworthy testimonies as compared to untrustworthy ones. This feature was deemed as “future work” in [31]. Another feature that is absent in the previous models, but is present in ours is the incorporation of “local historical information” with the “assessed reputation view”.

3. Automated conflict resolution in CDSS

In this section, we discuss our approach for resolving conflicts in the set of conflict groups of updates that are added to the deferred set of a CDSS’s participant peer during its reconciliation operation. Fig. 1 shows the general architecture of a CDSS participant peer using the proposed approach. Before further discussion, we need to define the key entities/players of the CDSS: (i) Provider Peer is the entity that shares its data updates with other peers in the CDSS. (ii) Consumer/Reconciling Peer is the entity that receives (possibly conflicting) updates on the same data from multiple providers. (iii) Remote/Rater Peer is the entity that helps the consumer in the reonciliation process by providing ratings about the provider. (iv) Multiple Users (which may be human) are registered with one peer in a mutually exclusive manner. In the proposed approach, after the reconciliation operation of the consumer adds a new conflict group to the deferred set, the following steps are taken:

1. The reconciliation operation inquires other remote peers (i.e., remote raters) about their past experiences with the provider peers that have conflicting updates in this conflict group. The following sub-steps are then taken to compute the remote assessed reputation of each provider:
   (a) After receiving all replies from remote raters, the credibility values of responding raters are (re)computed based on the majority rating and the aggregation of the previously, computed remotely assessed reputations of this provider.
   (b) Reported ratings provided by remote raters are then weighted according to the new credibility values. The credibility value of a remote rater represents to what degree the reconciling peer trusts the rating value reported by the remote rater.
   (c) Weighted reported reputation values are then aggregated for each update in the conflict group. This aggregated value represents the remotely assessed reputation of a particular provider peer as viewed by the reconciling peer.
2. The reconciliation operation informs local users of the reconciling peer to rate updates in this conflict group. Whenever this conflict group is rated by a number of users more than a predefined threshold, then it is marked as closed. Local users are thence not allowed to rate this closed conflict group or change their previous rating. The following sub-steps are then taken to compute the local assessed reputation of each provider peer:

(a) Whenever a conflict group is marked as closed, then for each provider peer that has an update in this conflict group, the credibility values of users that rate the updates of this provider peer, are (re)computed based on the majority rating and the aggregation of the previously computed locally assessed reputations of this provider.

(b) The reported ratings provided by users are then weighted according to the new credibility values. The credibility value for a user represents to what degree the reconciling peer trusts the provided rating for the update of a particular provider peer.

(c) Weighted reported ratings are then aggregated for each update in the conflict group. This aggregated value represents the locally assessed reputation of a particular provider as viewed by the reconciling peer.

3. The assessed reputation of each provider peer that is involved in the closed conflict group is computed by weighting both computed remotely and locally assessed reputations of this provider peer. The weights that are given for both computed values depend on the reconciling peer's administrator. The administrator may assign the local reputation of a provider higher weight than the remote reputation of a provider, or vice versa.

4. Finally, the update which is imported from the provider peer with the highest assessed reputation value is applied to the reconciling peer's instance (making sure it does not violate its integrity constraints).

In the following, we describe in details how to compute both remote and local reputations of a provider peer. We assume a CDSS, where a group of autonomous peers share a single schema, and each one manages its own database instance. Every relation in the database has a key, and a tuple is an entry in the database identified by a key. Disagreement on the non-key values of a tuple leads to several versions of this tuple.

Table 1 lists the definition of symbols used henceforth.

3.1. Remote Reputation of a Provider Peer (RRPP)

When a new conflict group is added to the deferred set of a consumer peer, it needs to resolve the conflict by choosing a single update from the group, and reject others. This decision is based on the feedbacks collected from both, other remote CDSS peers, and the local user community (that form the consumer peer). In this section, we provide details on feedbacks collection from remote peers, while we discuss the feedbacks collected from the local user community in the next section.

In the proposed system, each CDSS participant peer records its perception of the reputation of the provider peer(s). This perception is called the personal evaluation of a provider peer in the consumer's view. In this study, we assume that a consumer peer computes this personal evaluation every time it needs to resolve a conflict for any conflict group added to its deferred set and only for provider peers that have their updates in this particular conflict group. Let $p_j$ be a provider peer and $p_s$ be a rater peer. $p_s$ maintains $\text{Rep}(p_j, p_s)$ that represents its personal evaluation of $p_j$'s reputation score. Other peers
may differ or concur with \( p_i \)'s observation of \( p_j \). A consumer peer \( p_j \) that inquires about the reputation of a given provider peer \( p_i \) from rater peers may get various differing personal evaluations or feedbacks. Thus, to get a correct assessment of \( p_i \), all the collected feedbacks about \( p_i \) need to be aggregated. The aggregation of all feedbacks collected from remote raters to derive a single reputation value \((RRPP)\) represents \( p_i \)'s remote assessed reputation as viewed by \( p_j \). Consumer peers may employee different aggregation techniques. Formally, the remote assessed reputation \((RRPP(p_j,p_i))\) of a provider peer \( p_j \) as viewed by a consumer peer \( p_i \) is defined as:

\[
RRPP(p_j,p_i) = f(\phi)\sum_{x}^{k}(Rep(p_j,p_x))
\]  

where \( L \) denotes the set of rater peers which have interacted with \( p_j \) in the past and are willing to share their personal evaluations of \( p_j \) with \( p_i \), \( Rep(p_j,p_x) \) is the last personal evaluation of \( p_j \) as viewed by \( p_x \), and \( f(\phi) \) represents the aggregation function, which can simplistically be the average of all feedbacks, or it can be a more complex process that considers a number of factors.

A major drawback of feedback-only based systems is that all ratings are assumed to be honest and unbiased. A provider peer that usually produces high quality updates may get incorrect or false ratings from different evaluators due to several malicious motives. In order to deal with this issue, a reputation management system should weigh the ratings of highly credible raters more than raters with low credibilities [15]. In our approach, the reputation score of the provider peer is calculated according to the credibility scores of the rater peers. The credibility score of a rater peer \( p_x \) is assigned by a consumer peer \( p_i \) determines to what degree \( p_x \) trusts the reputation value assigned by this rater to a provider peer \( p_j \). Taking into consideration the credibility factor, the \( RRPP \) of \( p_j \) is calculated as a weighted average according to the credibilities of the rater peers. Thus, the Eq. (1) becomes:

\[
RRPP(p_j,p_i) = \frac{\sum_{x=1}^{L}(Rep(p_j,p_x) \times C_{p_x})}{\sum_{x=1}^{L}C_{p_x}}
\]  

where \( C_{p_x} \) is the credibility of \( p_x \) as viewed by \( p_i \). The credibility of a rater peer lies in the interval \([0,1]\) with 0 identifying a dishonest rater and 1 an honest one. The overall rater credibility assessment process follows.

Evaluating Rater Credibility: To minimize the effects of unfair or inconsistent ratings we screen the ratings based on their deviations from the majority opinion (similar to other works in [9,50,46,48], etc). The basic idea is that if the reported rating agrees with the majority opinion, the rater's credibility is increased, and decreased otherwise. However, unlike previous models, we do not simply disregard/discard the rating if it disagrees with the majority opinion but consider the fact that the rating’s inconsistency may be the result of an actual experience. Hence, only the credibility of the rater is changed, but the rating is still considered. We use a data clustering technique to define the majority opinion by grouping similar feedback ratings together. We use the k-mean clustering algorithm [30] on all current reported ratings to create the clusters. The most densely populated cluster is then labeled as the “majority cluster” and the centroid of the majority cluster is taken as the majority rating (denoted \( MR \)). To obtain a better measure of the dispersion of ratings, we calculate the Euclidean distance between the majority rating (\( MR \)) and each reported rating (\( R \)). The resulting value is then normalized using the standard deviation (\( \sigma \)) in all the reported ratings. The normalization equation (to assess the change in credibility due to majority rating), denoted by \( MR_{+} \) is then defined as:

---

**Table 1**: Definition of symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>Set of CDSS’s participant peers ( {p_1, \ldots, p_n} )</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>Schema that represents the relations in the system</td>
</tr>
<tr>
<td>( h_i(S) )</td>
<td>Local database instance controlled by a peer ( p_i )</td>
</tr>
<tr>
<td>( t )</td>
<td>Reconciliation time counter</td>
</tr>
<tr>
<td>( p_i )</td>
<td>Peer who is reconciling</td>
</tr>
<tr>
<td>( p_j )</td>
<td>Remote peer</td>
</tr>
<tr>
<td>( G_i )</td>
<td>Particular conflict group in the deferred set of ( p_j )</td>
</tr>
<tr>
<td>( \delta_{ij} )</td>
<td>Particular conflicting update of ( G_i ), that is imported from remote peer ( p_j )</td>
</tr>
<tr>
<td>( t_c )</td>
<td>Closing time of the rating process for an unresolved conflict group ( G_i )</td>
</tr>
<tr>
<td>( p_i^\prime )</td>
<td>Local user ( x ) of ( p_i ) who participates in the rating process</td>
</tr>
<tr>
<td>( \sigma_i )</td>
<td>Threshold of ( % ) of raters for ( p_i ) to close the rating on updates of ( G_i )</td>
</tr>
<tr>
<td>( h_n )</td>
<td>Last ( n )-neutral rating by ( p_i^\prime ) to ( p_j )’s from already resolved conflict groups</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Smoothing factor in the interval ([0,1]) for determining the weights of recent ratings</td>
</tr>
<tr>
<td>( MR )</td>
<td>Value of the majority rating</td>
</tr>
<tr>
<td>( MR_{+} )</td>
<td>Change in credibility due to the majority rating</td>
</tr>
<tr>
<td>( RRPP )</td>
<td>Aggregation value of previously ( k ) assessed reputations of a particular peer</td>
</tr>
<tr>
<td>( RRPP_{+} )</td>
<td>Effect on credibility due to agreement or disagreement with ( RRPP )</td>
</tr>
<tr>
<td>( \Phi )</td>
<td>Credibility adjustment normalizing factor</td>
</tr>
<tr>
<td>( \Psi )</td>
<td>Amount of change in credibility</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Pessimism factor</td>
</tr>
<tr>
<td>( f(\phi) )</td>
<td>Aggregation function</td>
</tr>
</tbody>
</table>
\[ \mathcal{M}R_d = \begin{cases} 
1 - \frac{\sqrt{\sum_{k=1}^{n} (MR_k - R_k)^2}}{\sigma} & \text{if } \sqrt{\sum_{k=1}^{n} (MR_k - R_k)^2} < \sigma; \\
1 - \frac{1}{\sqrt{\sum_{k=1}^{n} (MR_k - R_k)^2}} & \text{otherwise.} 
\end{cases} \]

Note that \( \mathcal{M}R_d \) does not denote the rater's credibility (or the weight), but only defines the effect on credibility due to agreement/disagreement with the majority rating. How this effect is applied will be discussed shortly. There may be cases in which the majority of raters collude to provide an incorrect rating for a particular provider peer. Moreover, the outlier raters (ones not belonging to the majority cluster) may be the ones who are first to experience the deviant behavior of the providers. Thus, a majority rating scheme “alone” is not sufficient to accurately measure the reputation of a provider peer.

We supplement the majority rating scheme by adjusting the credibility of a rater based on its past behavior as well. The historical information provides an estimate of the trustworthiness of the raters [43,49]. The trustworthiness of a rater peer is computed by looking at the “last assessed reputation value” (for a provider peer \( p_j \), the present majority rating for \( p_j \), and the rater peer’s corresponding provided rating. We define a credible rater as one which has performed consistently, accurately, and has proven to be useful (in terms of ratings provided) over a period of time.

We believe that under controlled situations, a consumer peer’s perception of a provider peer’s reputation should not deviate much, but stay consistent over time. We assume the interactions take place at time \( t \) and the consumer peer already has record of the previously assessed RRPP, then:

\[ RRPP = f(\phi)_t \begin{Bmatrix} \frac{1}{4}RRPP(p_j, p_i)^t \end{Bmatrix} \]  

where \( RRPP(p_j, p_i) \) is the assessed RRPP of a provider peer \( p_j \) by a consumer peer \( p_i \) for each time instance \( t \). \( f(\phi) \) is the aggregation function and \( k \) is the time duration defined by each consumer peer. It can vary from one time instance to the complete past reputation record of \( p_i \). Note that \( RRPP \) is not the “personal evaluation” of either the rater peer or the consumer peer but is the “remote assessed reputation” calculated by a consumer peer at the previous time instance(s). If a provider behavior does not change much from the previous time instances, then \( RRPP \) and the present reported rating \( R \) should be somewhat similar. Thus, the effect on credibility due to agreement or disagreement with the aggregation of the last \( k \) assessed RRPP values (denoted \( RRPP_t \)) is defined in a similar manner as Eq. (3):

\[ RRPP_d = \begin{cases} 
1 - \frac{\sqrt{\sum_{k=1}^{n} (RRPP_k - R_k)^2}}{\sigma} & \text{if } \sqrt{\sum_{k=1}^{n} (RRPP_k - R_k)^2} < \sigma; \\
1 - \frac{1}{\sqrt{\sum_{k=1}^{n} (RRPP_k - R_k)^2}} & \text{otherwise.} 
\end{cases} \]

In real-time situations it is difficult to determine the different factors that cause a change in the state of a provider peer. A rater peer may rate the same provider peer differently without any malicious motive. Thus, the credibility of a rater peer may change in a number of ways, depending on the values of \( R, \mathcal{M}R_d, \) and \( RRPP_t \). The general formula is:

\[ C_{p_i} = C_{p_j} \pm \Phi \ast \Psi \]

where \( \Phi \) is the credibility adjustment normalizing factor, while \( \Psi \) represents amount of change in credibility due to the equivalence or difference of \( R \) with \( \mathcal{M}R \) and \( RRPP \). The signs \( \pm \) indicate that either \( + \) or \( - \) can be used, i.e., the increment or decrement in the credibility depends on the situation. These situations are described in detail in the upcoming discussion.

We place more emphasis on the ratings received in the current time instance than the past ones, similar to previous works as [10,49]. Thus, equivalence or difference of \( R \) with \( \mathcal{M}R \) takes a precedence over that of \( R \) with \( RRPP \). This can be seen from Eq. (6), where the \( * \) sign with \( \Phi \) indicates \( R \approx \mathcal{M}R \) while \( - \) sign with \( \Phi \) means that \( R \neq \mathcal{M}R \). \( \Phi \) is defined as:

\[ \Phi = C_{p_j} \ast (1 - \left| R_k - \mathcal{M}R \right|) \]

Eq. (7) states that the value of the normalizing factor \( \Phi \) depends on the credibility of the rater and the absolute difference between the rater’s current feedback and the majority rating calculated. Multiplying by the rater’s credibility allows the honest raters to have greater influence over the ratings aggregation process and dishonest raters to lose their credibility quickly in case of a false or malicious rating. The different values of \( \Psi \) are described next.

Adjusting Rater Credibilities: \( \Psi \) is made up of \( \mathcal{M}R_d \) and/or \( RRPP_d \), and a “pessimism factor” \( \rho \), which is used to normalize the change factor (for rater credibility). The exact value of \( \rho \) is left at the discretion of the consumer peer, with the exception that its minimum value should be 2. The lower the value of \( \rho \), the more optimistic is the consumer peer and higher value of \( \rho \) are suitable for pessimistic consumers (this value is inverted in Eqs. (10) and (11)). We define a pessimistic consumer as one that does not trust the raters easily and reduces their credibility drastically on each false feedback. Moreover, honest rater’s reputations are increased at a high rate, meaning that such consumers make friends easily. On the other hand, optimistic consumers tend to “forgive” dishonest feedbacks over short periods (dishonesty over long periods is still punished), and it is difficult to attain high reputation quickly. Only prolonged honesty can guarantee a high credibility in this case. \( R, \mathcal{M}R \), and \( RRPP \) can be related to each other in one of four ways, and each condition specifies how \( \mathcal{M}R_d \) and \( RRPP_d \) are used in the model. Note that the normalizing factor (\( \rho \) in our case) is common among all the four conditions. The difference
is in the different ‘amounts’, that are based on equalities or inequalities among \( R, M_R \), and \( RRPP \). In the following, we provide an explanation of each and show how the credibilities are updated in our proposed model using different values for \( \Psi \).

**Case 1.** The reported reputation value is similar to both the majority rating and the aggregation of the previously computed \( RRPP \) values (i.e., \( R \simeq M_R \simeq RRPP \)). The equality \( M_R \simeq RRPP \) suggests that majority of the raters believe that the quality of updates imported from a provider peer \( p_j \) has not changed. The rater peer’s credibility is thus updated as:

\[
C_{p_i} = C_{p_i} + \Phi \times \left( \frac{|MR_A| + \text{RRPP}_i}{\rho} \right)
\]  

Eq. (8) states that since all variables are equal, the credibility is incremented. We will see in the following that in the current case, the factor multiplied to \( \Phi \) is the largest (due to the variable equalities).

**Case 2.** The individual reported reputation rating is similar to the majority rating but differs from the previously assessed reputation, i.e. \( (R \simeq M_R) \) and \( (R \neq RRPP) \). In this case, the change in the reputation rating could be due to either of the following. First, the rater peer may be colluding with other raters to increase or decrease the reputation of a provider peer. Second, the quality of updates imported from the provider peer may have actually changed since \( RRPP \) was last calculated. The rater peer’s credibility is updated as:

\[
C_{p_i} = C_{p_i} + \Phi \times \left( \frac{|MR_A|}{\rho} \right)
\]  

Eq. (9) states that since \( R \simeq M_R \), the credibility is incremented, but the factor \( R \neq RRPP \) limits the incremental value to \( \left( \frac{|MR_A|}{\rho} \right) \) (not as big as the previous case).

**Case 3.** The individual reported reputation value is similar to the aggregation of the previously assessed \( RRPP \) values but differs from the majority rating, i.e. \( (R \neq M_R) \) and \( (R \simeq RRPP) \). The individual reported reputation value may differ due to either of the following. First, \( R \) may be providing a rating score that is out-dated. In other words, \( R \) may not have the latest score. Second, \( R \) may be providing a “false” negative/positive rating for a provider peer. The third possibility is that \( R \) has the correct rating, while other rater peers contributing to \( M_R \) may be colluding to increase/decrease the provider peer’s reputation. None of these three options should be overlooked. Thus, the rater peer’s credibility is updated as:

\[
C_{p_i} = C_{p_i} + \Phi \times \left( \frac{|MR_A|}{\rho} \right)
\]  

Eq. (10) states that since \( R \neq M_R \), the credibility is decremented, but here the value that is subtracted from the previous credibility is adjusted to \( \left( \frac{|MR_A|}{\rho} \right) \).

**Case 4.** The individual reported reputation value is not similar to both the majority rating and the calculated aggregation of assessed \( RRPP \) values, i.e. \( (R \neq M_R) \) and \( (R \neq RRPP) \). \( R \) may differ from the majority rating and the past aggregation of \( RRPP \) values due to either of the following. First, \( R \) may be the first one to experience the provider peer’s new behavior. Second, \( R \) may not know the actual quality of the provider peer’s imported updates. Third, \( R \) may be lying to increase/decrease the provider peer’s reputation. In this case, the rater peer’s credibility is updated as:

\[
C_{p_i} = C_{p_i} + \Phi \times \left( \frac{|MR_A| + \text{RRPP}_i}{\rho} \right)
\]  

Eq. (11) states that the inequality of all factors means that rater peer’s credibility is decremented, where the decremented value is the combination of both the effects \( M_R_A \) and \( RRPP \).

### 3.2. Local Reputation of a Provider Peer (LRPP)

In our proposed solution, users can rate deferred updates according to their own beliefs about which update is the most correct. The reconciliation operation in a consumer peer \( p_i \) notifies local users when a new conflict group of updates \( G_c \) is inserted into the deferred set \( \text{Deferred}(p_i) \). It also specifies the closing time \( (t_c) \) of the rating process for this unresolved conflict group. Local users of \( p_i \) rate the updates of unresolved \( G_c \) in \( \text{Deferred}(p_i) \). A user \( x \) \((x,t_l)\) of \( p_i \) assigns a probabilistic rating \( (r_{x,t_l}) \) in the interval \([0,1]\) to each update \( g_{c,l} \) of a provider peer \( p_j \) in \( G_c \), where \( 0 \) identifies the rater’s extreme disbelief and 1 identifies the rater’s extreme belief in an update. Moreover, a user can assign a neutral rating \((-1)\) to an update to express his lack of opinion about this particular update. A trigger is fired to inform the reconciliation operation when a voting period of unresolved \( G_c \) is ended. The reconciliation operation then checks whether this \( G_c \) is rated by a number of users exceeding a predefined percentage of the total number of local users \( (\sigma_l) \). If the number of users who rate this \( G_c \) exceeds \( \sigma_l \), the reconciliation operation marks this \( G_c \) as “closed” and users cannot rate this \( G_c \) anymore. Otherwise, the reconciliation operation extends the rating period of this particular \( G_c \) to (attain the threshold).

We adopt the same technique introduced in the Section 3.1 to compute the \( LRPP \) value. Each participant peer records the past computed \( LRPP \) values for each provider peer it works with. We also assume that a consumer peer computes a new \( LRPP \) value every time it needs to resolve a conflict for any conflict group added to its deferred set and only for provider peers which they have their updates in this particular conflict group \( G_c \). Then, \( \text{Rep}(p_j,G_c) \) represents the rating assigned by a local
consumer \( p_i^e \) to the update of provider \( p_j \) in \( G_c \). Formally, the LRPP of a provider peer \( p_j \) as viewed by a consumer peer \( p_i \), computed post closing a conflict group \( G_c \), is defined as:

\[
\text{LRPP}(p_j, p_i^e) = \frac{\sum_{x=1}^{L} (\text{Rep}(p_j, p_i^e) \ast C_{p_i})}{\sum_{x=1}^{L} C_{p_i}}
\]

where \( L \) denotes the set of local users who have rated \( p_j \)'s update in \( G_c \), \( \text{Rep}(p_j, p_i^e) \) is the rating of \( p_j \), and \( C_{p_i} \) is the credibility of a local user \( p_i^e \) as viewed by \( p_i \). This Equation is the same as Eq. (2). The only difference is that we here aggregate the summation of ratings given by local users, for the purpose of computing the LRPP value for a particular provider peer. The credibility of a local user assigned by a parent peer \( p_i \) determines to what degree \( p_j \) trusts the ratings assigned by a local user to a provider peer \( p_j \). As mentioned earlier, we follow the same approach discussed previously to compute the credibility of local users. We do not provide the details here to avoid redundancy as only minor changes are required. The only modification to Eqs. (1)-(11) is using ratings assigned by local users of a reconciling peer to provider peers’ updates in a closed conflict group. Notice that RRPP, defined in Eq. (4), here represents the aggregation of the past LRPP computed by \( p_j \) for \( p_j \), assuming that \( p_j \) keeps records of the previously computed LRPP.

### 4. Illustrative example

In this section, we provide a comprehensive example to illustrate the proposed approach. Let us consider a CDSS community of three participant peers \( (p_1, p_2, \text{and } p_3) \) that represent three bioinformatics warehouses (example adapted from [44]). The three peers share a single relation \( F(\text{organism, protein, function}) \) for protein function, where the key of the relation is composed of the fields \( \text{organism} \) and \( \text{protein} \). Peer \( p_1 \) accepts updates from both \( p_2 \) and \( p_3 \) with the same trust priority level. \( p_2 \) accepts updates from both \( p_1 \) and \( p_3 \), but it assigns a higher priority for updates that come from \( p_1 \). \( p_3 \) only accepts updates that come from \( p_2 \). For the purpose of the illustration, we also assume that there are 10 other participant peers (\( p_4 \) through \( p_{13} \)). In this example, we assign different roles for the participant peers. We consider peers \( p_2 \) and \( p_3 \) as provider peers for the rest of peers, \( p_1 \) as a consumer peer who imports updates from the provider peers and needs to reconcile its own instances. The remaining peers (\( p_4 \) through \( p_{13} \)) play the role of raters which are assumed to have interacted with the provider peers in the past and are willing to share their experiences with other consumer peers. Similar to [44], we illustrate the reconciliation operation of this CDSS example as shown in Table 2, taking into consideration our proposed modification for the system.

In the beginning (i.e., at time 0), we assume that the instance of relation \( F \) at each participant peer \( p_i \), denoted by \( I_i(F) \), is empty. At time 1, \( p_3 \) conducts two transactions \( T_{3,0} \) and \( T_{3,1} \). It then decides to publish and reconcile its own state (to check if other peers made any changes). Since the other two participant peers have not yet published any updates, \( p_1 \)'s instance, after the reconciliation operation is complete; \( I_1(F) \) denotes the result (the second transaction is only a modification to the first one). At time 2, \( p_2 \) conducts two transactions \( T_{2,0} \) and \( T_{2,1} \). It then publishes and reconciles its own state. Note that the resulting instance \( I_2(F) \) of \( p_2 \) contains only its own updates. Although there is a recently published update by \( p_2 \), which is trusted by it, \( p_2 \) does not accept \( p_3 \)'s published update because it conflicts with its own updates. At time 3, \( p_3 \) reconciles again. It

<table>
<thead>
<tr>
<th>( t )</th>
<th>( p_i )</th>
<th>( I_i(F) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( p_3 )</td>
<td>( { +F(\text{rat,prot1,cell-metab};3) } )</td>
</tr>
<tr>
<td>1</td>
<td>( p_1 )</td>
<td>( { +F(\text{rat,prot1,cell-metab};3) } )</td>
</tr>
<tr>
<td>2</td>
<td>( p_3 )</td>
<td>( { +F(\text{rat,prot1,cell-resp};2) } )</td>
</tr>
<tr>
<td>3</td>
<td>( p_1 )</td>
<td>( { +F(\text{rat,prot1,cell-metab};3) } )</td>
</tr>
<tr>
<td>4</td>
<td>( p_1 )</td>
<td>( { +F(\text{rat,prot1,cell-resp};2) } )</td>
</tr>
<tr>
<td>5</td>
<td>( p_3 )</td>
<td>( { +F(\text{cat,prot3,cell-metab};3) } )</td>
</tr>
<tr>
<td>6</td>
<td>( p_1 )</td>
<td>( { +F(\text{cat,prot3,cell-metab};3) } )</td>
</tr>
</tbody>
</table>

The remaining peers (\( p_4 \) through \( p_{13} \)) play the role of raters which are assumed to have interacted with the provider peers in the past and are willing to share their experiences with other consumer peers. Similar to [44], we illustrate the reconciliation operation of this CDSS example as shown in Table 2, taking into consideration our proposed modification for the system.
Table 3
The deferred set of peer p₁.

<table>
<thead>
<tr>
<th>G₁</th>
<th>Txn</th>
<th>p₁1</th>
<th>p₁2</th>
<th>p₁3</th>
<th>p₁4</th>
<th>p₁5</th>
<th>p₁6</th>
<th>p₁7</th>
<th>p₁8</th>
<th>p₁9</th>
<th>p₁10</th>
<th>Status</th>
<th>σ₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>G₁</td>
<td>T₁1</td>
<td>0.95</td>
<td>0.65</td>
<td>1.00</td>
<td>0.60</td>
<td>0.97</td>
<td>1.00</td>
<td>0.95</td>
<td>0.90</td>
<td>0.95</td>
<td>1.00</td>
<td>Closed</td>
<td>100%</td>
</tr>
<tr>
<td>G₂</td>
<td>T₂1</td>
<td>0.45</td>
<td>0.80</td>
<td>0.45</td>
<td>0.75</td>
<td>0.40</td>
<td>0.40</td>
<td>0.45</td>
<td>0.40</td>
<td>0.45</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

accepts the transaction T₂₁ that is published by p₂ and rejects p₁’s second update T₂₂₀ because it conflicts with its own state. At time 4, p₁ reconciles. It gives the same priority for transactions of p₂ and p₁. Thus, it accepts the non-conflicting transaction T₂₀, and it defers both the conflicting transactions T₂₁ and T₂₂.

p₁’s reconciliation operation forms a conflict group G₁ (shown in Table 3) that includes both deferred transactions that are added to the deferred set of p₁ during the reconciliation. p₁ first inquires other remote peers about their trust placed in the provider peers that have conflicting updates in G₁. Second, it notifies its local users that a new conflict group is added to Deferred(p₁), so they can start rating updates in this particular conflict group. The result of these two steps is the computing of RRPP and LRPP values for each provider peer that has an update in G₁ (p₂ and p₃ in this case). p₁ then computes the assessed trust for each provider peer who has update in G₁ by weighting the values of RRPP and LRPP according to its pre-defined preferences. Next, we provide the details of these steps.

4.1. Computing the RRPP

We assume here that the local peer p₁ maintains a table of all the previously assessed reputation values of provider peers that it interacts with. For instance, the last 10 RRPP values previously computed by p₁ for provider peers p₂ and p₃ are {0.58, 0.55, 0.56, 0.62, 0.60, 0.63, 0.59, 0.51, 0.53, 0.55} and {0.95, 1.00, 0.94, 0.89, 0.90, 0.94, 0.85, 0.87, 0.96, 0.92} respectively. Similarly, as mentioned earlier, p₁ maintains a credibility value for each rater peer that responds to its request for any p₁’s rating.

After a new conflict group G₁ is added to the deferred set of p₁, assume that p₁ gets back responses from rater peers p₄, p₅, ..., p₁₃. The received responses (in-order) for p₁ are {0.70, 0.65, 0.50, 0.46, 0.52, 0.67, 0.55, 0.43, 0.47, 0.90}, and for p₂ are {0.98, 0.88, 0.93, 0.96, 0.99, 0.91, 0.90, 0.89, 0.95, 0.45}. Using this information, p₁’s reconciliation operation performs the following steps for each provider peer in G₁:

1. p₁ computes the values of MR, M₄, RRPP, and RRPP, factors for each provider peer in G₁. The computed values for p₂ are (0.57, 0.59, 0.67, 0.67) and for p₃ are (0.92, 0.88, 0.68, 0.67), respectively.
2. p₁ computes the new credibility values for each rater peer, as shown in Table 4, who has provided their ratings for p₂. Then, it takes the new computed credibility values as an input to compute the new credibility values for consumer raters who provides their ratings to p₁, as shown in Table 5, assuming that each consumer rater has provided his rating for all provider peers that appear in the conflict group F₁. We provide more details about the computations done in Table 4 (and 5) in the following:
   (a) The first row of Table 4, titled (Cₚ₁(old)), shows the current credibility values for rater peers (p₄, p₅, ..., p₁₃).
   (b) In the second row of Table 4, the values of Φ variable are shown after Eq. (7) is applied.
   (c) The rows (3–5) show the equalities between the factor pairs (R ∼ MR), (M₄ ∼ RRPP), and (R ∼ RRPP), for each consumer rater. Here, we assume that the two compared factors are equal if the amount of difference between them is equal or less than 0.20. Otherwise, they are considered not to be equal. If we look at Table 4, we see that all pairs are considered equal, except for the consumer rater p₁₃. For those raters who have (R ∼ MR ∼ RRPP), Case (1) conditions are met, and thus we apply Eq. (8) for computing the new credibility values. For p₁₃, we have (R ∼ MR) and (R ∼ RRPP). Thus, Case (4) is met, and we apply Eq. (11) for computing the new credibility value. Since the reported rating value by p₁₃ is not similar to both the majority opinion and the aggregation of the previously computed RRPP values of provider peer p₂, p₁₃, is penalized (by decreasing its credibility and giving a less weight for its reported rating).
   (d) The rows (6–8) of Table 4 show the matched case, the value of Ψ, and the new computed credibility value (Cₚ₁(new)), for each rater.
   (e) The last row, titled Rₚᵣ, shows the weightage of reputation values received from the different raters.
   (f) Based on the last two rows of Table 4, p₁’s reconciling operation computes the RRPP for provider peer p₂ (RRPP(p₂, p₁) = 0.58) by applying Eq. (2).

Table 5 values are obtained in the same manner as defined above, and the RRPP for p₁ is computed as RRPP(p₁, p₁) = 0.89 by applying Eq. (2). Note that the new credibility values computed in Table 4 are used as inputs to compute the new credibility values for consumer raters who provided their reputation values for provider peer p₂. Again, credibilities of all consumer raters are altered according Case (1), except for consumer rater p₁₃ where its credibility is altered according Case (4).
Table 4
Computing $p_1$’s RRPP and the new credibility values for remote raters who respond to the inquiry regarding the reputation of the provider peer $p_2$.

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_4$</th>
<th>$p_5$</th>
<th>$p_6$</th>
<th>$p_7$</th>
<th>$p_8$</th>
<th>$p_9$</th>
<th>$p_{10}$</th>
<th>$p_{11}$</th>
<th>$p_{12}$</th>
<th>$p_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{i}(X)_{old}$</td>
<td>0.95</td>
<td>0.97</td>
<td>0.89</td>
<td>0.88</td>
<td>0.97</td>
<td>0.90</td>
<td>0.95</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>0.84</td>
<td>0.91</td>
<td>0.81</td>
<td>0.77</td>
<td>0.91</td>
<td>0.82</td>
<td>0.92</td>
<td>0.79</td>
<td>0.83</td>
<td>0.65</td>
</tr>
<tr>
<td>$R \approx \text{MR}$</td>
<td>0.12</td>
<td>0.07</td>
<td>0.09</td>
<td>0.13</td>
<td>0.06</td>
<td>0.09</td>
<td>0.03</td>
<td>0.16</td>
<td>0.12</td>
<td>0.32</td>
</tr>
<tr>
<td>$\text{MR} \approx \text{RRPP}$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>$R \approx \text{RRPP}$</td>
<td>0.13</td>
<td>0.08</td>
<td>0.07</td>
<td>0.11</td>
<td>0.05</td>
<td>0.10</td>
<td>0.02</td>
<td>0.14</td>
<td>0.10</td>
<td>0.33</td>
</tr>
<tr>
<td>Case(1−4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.02</td>
<td>0.01</td>
<td>0.10</td>
</tr>
<tr>
<td>$C_{i}(X)_{new}$</td>
<td>0.96</td>
<td>0.97</td>
<td>0.90</td>
<td>0.89</td>
<td>0.97</td>
<td>0.91</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.88</td>
</tr>
<tr>
<td>$R_w$</td>
<td>0.67</td>
<td>0.63</td>
<td>0.45</td>
<td>0.41</td>
<td>0.51</td>
<td>0.61</td>
<td>0.52</td>
<td>0.41</td>
<td>0.45</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Table 5
Computing $p_1$’s RRPP and the new credibility values for remote raters who respond to the inquiry regarding the reputation of the provider peer $p_3$.

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_4$</th>
<th>$p_5$</th>
<th>$p_6$</th>
<th>$p_7$</th>
<th>$p_8$</th>
<th>$p_9$</th>
<th>$p_{10}$</th>
<th>$p_{11}$</th>
<th>$p_{12}$</th>
<th>$p_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{i}(X)_{old}$</td>
<td>0.96</td>
<td>0.97</td>
<td>0.90</td>
<td>0.89</td>
<td>0.97</td>
<td>0.91</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.88</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>0.87</td>
<td>0.97</td>
<td>0.85</td>
<td>0.82</td>
<td>0.87</td>
<td>0.88</td>
<td>0.94</td>
<td>0.94</td>
<td>0.89</td>
<td>0.50</td>
</tr>
<tr>
<td>$V \approx \text{MR}$</td>
<td>0.10</td>
<td>0.00</td>
<td>0.05</td>
<td>0.08</td>
<td>0.11</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
<td>0.07</td>
<td>0.43</td>
</tr>
<tr>
<td>$\text{MR} \approx \text{RRPP}$</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>$V \approx \text{RRPP}$</td>
<td>0.06</td>
<td>0.04</td>
<td>0.01</td>
<td>0.04</td>
<td>0.07</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.47</td>
</tr>
<tr>
<td>Case(1−4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.21</td>
</tr>
<tr>
<td>$C_{i}(X)_{new}$</td>
<td>0.97</td>
<td>0.98</td>
<td>0.90</td>
<td>0.89</td>
<td>0.98</td>
<td>0.91</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.78</td>
</tr>
<tr>
<td>$V_w$</td>
<td>0.95</td>
<td>0.86</td>
<td>0.83</td>
<td>0.86</td>
<td>0.97</td>
<td>0.83</td>
<td>0.86</td>
<td>0.84</td>
<td>0.90</td>
<td>0.35</td>
</tr>
</tbody>
</table>

4.2. Computing the LRPP

Note that the local peer $p_1$ (the reconciling peer in our running example) maintains a table of all previously assessed LRPP values of provider peers that it interacts with. For instance, the last 5 LRPP values for $p_2$ and $p_3$ are $(0.41, 0.43, 0.58, 0.52, 0.38)$ and $(0.90, 0.89, 0.89, 0.94, 0.90)$ respectively. Similarly, it maintains a credibility value for each local user (remember each peer is composed of $n$ users) that has provided reputation ratings regarding different conflicts in the past. The credibility values change according to the new assessed LRPP values of provider peers computed by the local peer $p_1$. Let us also assume here that all 10 users of $p_1$ (denoted $p_1^1, p_1^2, \ldots, p_1^{10}$) have participated in the rating of all the updates in conflict group, and the rating process is considered to be closed, as illustrated in Table 3. When this requirement is met, $p_1$’s reconciliation operation marks the conflict group $G_i$ as closed to inform users to stop giving new ratings to updates of this conflict group. After $G_i$ is marked as closed, $p_1$’s reconciliation operation performs the same steps as in computing the RRPP value above for each provider peer in $G_i$. We omit the LRPP computation steps (and associated tabular results) here to avoid redundancy as they are very similar to the above mentioned steps that we follow to compute the RRPP value. Instead, we only summarize the outcome of these steps as follows: (i) $p_1$ computes the values of $\text{MR}_i, \text{MR}_i', \text{LRPP}$, and $\text{LRPP}_i'$ factors for each provider peer in $G_i$. The computed values for $p_2$ are $(0.47, 0.50, 0.68, 0.67)$ and for $p_3$ are $(0.90, 0.90, 0.67, 0.67)$, respectively. (ii) Based on the local user credibilities, and reported ratings $p_1$’s reconciling operation computes the LRPP for provider peers $p_2$ ($\text{LRPP}(p_2, p_1) = 0.48$) and $p_3$ ($\text{LRPP}(p_3, p_1) = 0.91$), respectively.

4.3. Conflict resolution

After the conflict group $G_1$ is closed, and RRPP and LRPP values are computed for each provider peer in $G_1$, $p_1$’s reconciliation operation computes the assessed reputations of provider peers $p_2$ and $p_3$. The assessed reputation of a provider peer is computed by weighing the RRPP and LRPP values. As mentioned earlier, the administrator of the reconciling peer $p_1$ is responsible for defining the appropriate weightages. For our example, let us assume that the weight given for the RRPP is 40% and for the LRPP is 60%. Thus, the assessed reputation of $p_2$ will be $\text{Rep}(p_2, p_1) = 0.58 \times 40\% + 0.48 \times 60\% = 0.52$, and the assessed reputation of $p_3$ will be $\text{Rep}(p_3, p_1) = 0.89 \times 40\% + 0.91 \times 60\% = 0.90$. Since $p_3$ has the higher reputation value, the transaction $T_{3,1}$ of $p_3$ is considered in the next reconciliation operation, and applied to the local instance of peer $p_1$, as it does not violate its local state or does not conflict with other accepted transactions during the reconciliation. However, the transaction $T_{2,1}$ of $p_2$ is rejected, and it is not considered in the next reconciliations.

In continuation of the scenario as illustrated in Table 2, at time 5, $p_3$ applies a new transaction $T_{3,2}$. It then decides to publish and reconciles its own state, and it ends with the instance $I_3(F)|5$. At time 6, $p_1$ decides to reconcile. It ends up with
applying the transaction \(T_{3,1}\), resulting from the ratings on the updates of conflict group \(G_{1}\), to its local instance. It also accepts and applies the new published transaction \(T_{3,2}\) of \(p_{3}\). Hence, \(p_{1}\) ends up with \(I_{1}(F)/6\).

5. Implementation model and results

In this section, we illustrate the implementation details of the proposed approach using the above mentioned scenario. We modeled the different entities (as defined in Section 3) as Java-based Web services to see how the algorithms perform with large number of conflicts and different qualities of providers and raters. Users are individual services, while Peers are modeled as service compositions. The experiments are conducted in a closed environment, where we can capture the actual behavior of providers and raters. The validity of the proposed approach can thus be measured by observing the difference between the actual behavior of the providers and raters, and their computed reputation values and credibilities, respectively. We used the WSdream QoS-Dataset [52] to model the different service quality behaviors. This data-set contains around 150 Web services distributed in computer nodes located all over the world (i.e., distributed in 22 different countries), where each Web service is invoked 100 times by a service user. Planet-Lab is employed for monitoring the Web services. The service users observe, collect, and contribute the QoS data of the selected Web services. This data is used in modeling rater credibilities, and provider quality patterns. The provider CDSS updates are created in a semi-automated manner, to follow one of five classes of providers (details in Section 5.2). Similarly, the percentages of honest and dishonest raters are changed to see their impact on the proposed approach.

5.1. One consumer and multiple providers

In the first set of experiments, we developed a CDSS with three participant peers. \(p_{1}\) is the reconciling peer, whereas \(p_{2}\) and \(p_{3}\) are the provider peers. \(p_{1}\) has 100 local users. The provider peers are initially assigned degrees of quality or behavior randomly on a scale of \([0, 1]\) where 0 denotes the lowest quality and 1 the highest. For \(p_{2}\), the value lies between 0.1 and 0.7, and between 0.7 and 1.0 for \(p_{3}\). We further divided the One consumer and Multiple providers case into two sets of experiments. In the first set, 80% of users are high quality users (i.e., they provide accurate rating values in the range \([0.8, 1]\)), and 20% of them are low quality users (i.e., they provide poor rating values in the range \([0.1, 0.4]\)). In the second set, we keep the quality level of rating for both groups of users the same as in the first set, but we only increase the percentage of dishonest raters (to 50%) and decrease the percentage of honest ones (to 50%). At the beginning of the simulation, we assume that all local users of the reconciling peer have credibility of 1. Each time during the simulation, \(p_{2}\) and \(p_{3}\) generate identical tuples (i.e., tuples that have the same key but differ in values of the non-key attributes) and then publish their updates. When \(p_{1}\) reconciles (i.e., imports the newly published updates from both \(p_{2}\) and \(p_{3}\)), a conflict is found in the pair of updates with the same key but imported from different providers. The conflict is resolved by either accepting the update of \(p_{2}\) or \(p_{3}\), according to the weighted ratings of users. The simulation ends when \(p_{1}\) resolves the conflict numbered 3600.

Fig. 2 shows the results for the above mentioned experiment sets. For conciseness, the average of 10 rounds of experiments is shown. In the first set (denoted by A), honest raters out-number dishonest ones. Fig. 2(A) shows the effect of this inequality in calculating raters’ credibilities, providers’ reputations, and thus the number of accepted updates per each provider. The average credibility of each group of users is shown in Fig. 2(A.1) with increasing number of conflicts, while Fig. 2(A.2) represents the average reputations of providers peers, and the number of accepted updates from each provider is shown in Fig. 2(A.3). Because there are more honest raters, we can see that the average assessed reputation for each provider is almost identical to their actual behaviors. Moreover, the average credibility of honest raters is always high compared to that of dishonest group where it is drastically decreasing for consecutive conflicts. The result of the second set where the number of honest and dishonest raters are equal is shown in Fig. 2(B). This equality results in the dishonest raters’ ratings forming the majority rating on several occasions. Therefore, we see an increase in the number of updates of \(p_{2}\) being accepted by \(p_{1}\). This causes a degradation in the credibility of honest raters, since their opinion differs from the majority opinion, and an increment in the dishonest raters’ credibilities (Fig. 2(B.1)).

5.2. Multiple consumers and providers

In the second set of experiments, we developed a CDSS of 40 participant peers, where 20 of them are only providers, and the other 20 peers are only consumers, with each consumer peer having 20 local users. We have divided the provider peers into 5 different behavioral groups that represent the real life scenarios: providers that always perform with consistently high quality (i.e., their updates are correct and of high value), providers that always perform with consistently low quality, providers that perform high at the beginning but start performing low after the time instance 200, providers that perform low at the beginning but they start performing high after the time instance 200, and the final group of providers that perform in a random manner, oscillating between high and low performance quality. We ran several experiments to cover the above mentioned CDSS cases, where each experiment is run multiple times for each scenario, and the averaged results over those runs are presented in the following.

The experiment rounds starts at time instance 0 and finish at time instance 400. The databases of all peers are empty at time instance 0. At the beginning of each time instance, all provider peers insert a new single update to their local instances,
and then they publish their most recent update to others. The inserted updates at each time instance are almost identical. In other words, they all have the same value for the primary key attribute, but they have different values in at least one non-key attribute. In the same way, after all providers publish their most recent updates at a particular time instance, each consumer peer reconciles its local instance with the recently published updates. As all providers will publish conflicting updates at each time instance, a consumer peer will find that all imported updates conflict with each other at each reconciliation point. Thus, a new conflict group that contains all imported updates in this particular time instance is added to the deferred set of the reconciling peer. Provider peers are assigned degrees of quality or behavior in the following manner: it is in the range $\frac{1}{2}:0 - 1$:0/C138 for the first group, $\frac{1}{2}:0 - 1$:0/C138 for the second group, $\frac{1}{2}:0 - 1$:0/C138 for the third group in the first half of the experiment run and $\frac{1}{2}:0 - 1$:0/C138 in the second half, and $\frac{1}{2}:0 - 1$:0/C138 for the last group.

We further divided the experiments to model the different percentages of honest and dishonest users. In the interest of space, we present two cases in the following. In the first one, 90% of the users are high quality users (i.e., are honest), with values in the range $\frac{1}{2}:0 - 1$:0, and 10% of them are low quality users in the range $\frac{1}{2}:0 - 1$:0/C138. In the other set, the percentage of high quality users is set to 60%. A high quality rater generates a rating that differs at most 10% from the actual value. In contrast, a low quality rater generates a value that differs at least by 75% from the actual rating value. At the beginning of the experiment rounds, we assume that all local users of the reconciling peers and all consumer peers have their credibility values set to 1.0 (i.e., the maximum credibility value).

The plots (A–E) in Figs. 3 and 4 show the effect of the size of low quality raters in calculating the reputation values of each provider group. Each plot, from A to E, shows the comparison between the average of actual provider group quality and then they publish their most recent update to others. The inserted updates at each time instance are almost identical. In other words, they all have the same value for the primary key attribute, but they have different values in at least one non-key attribute. In the same way, after all providers publish their most recent updates at a particular time instance, each consumer peer reconciles its local instance with the recently published updates. As all providers will publish conflicting updates at each time instance, a consumer peer will find that all imported updates conflict with each other at each reconciliation point. Thus, a new conflict group that contains all imported updates in this particular time instance is added to the deferred set of the reconciling peer. Provider peers are assigned degrees of quality or behavior in the following manner: it is in the range $\frac{1}{2}:0 - 1$:0/C138 for the first group, $\frac{1}{2}:0 - 1$:0/C138 for the second group, $\frac{1}{2}:0 - 1$:0/C138 for the third group in the first half of the experiment run and $\frac{1}{2}:0 - 1$:0/C138 in the second half, and $\frac{1}{2}:0 - 1$:0/C138 for the last group.

We further divided the experiments to model the different percentages of honest and dishonest users. In the interest of space, we present two cases in the following. In the first one, 90% of the users are high quality users (i.e., are honest), with values in the range $\frac{1}{2}:0 - 1$:0, and 10% of them are low quality users in the range $\frac{1}{2}:0 - 1$:0/C138. In the other set, the percentage of high quality users is set to 60%. A high quality rater generates a rating that differs at most 10% from the actual value. In contrast, a low quality rater generates a value that differs at least by 75% from the actual rating value. At the beginning of the experiment rounds, we assume that all local users of the reconciling peers and all consumer peers have their credibility values set to 1.0 (i.e., the maximum credibility value).

The plots (A–E) in Figs. 3 and 4 show the effect of the size of low quality raters in calculating the reputation values of each provider group. Each plot, from A to E, shows the comparison between the average of actual provider group quality
(GroupX-R) and the average of assessed provider group reputation (GroupX-A). Similarly, plot F shows the comparison between the average credibility values of high and low quality user groups in all consumer peers. The last plot (G) shows the average number of updates accepted by all consumer peers from each group of providers.

It can be seen from Fig. 3 that when the percentage of low quality users is only 10% of the total number of local users, the computed assessed reputation values are almost equal to the original provider behavior. This is expected because Low quality users’ behavior is captured and their credibilities are thus reduced (Fig. 3(F)), which means that their provided ratings are also decreased. Fig. 3(G) shows the average number of updates accepted by reconciling peers from each group. Number of updates accepted from group \(G_1\) is around that of both groups \(G_3\) and \(G_5\). The chance of accepting updates from groups \(G_1\) and \(G_3\) are the same at the first half of the simulation time, while it is the same for groups \(G_1\) and \(G_4\) at the second half. We can see that there are no updates accepted from either group \(G_2\) or group \(G_5\), as the reputation values for members of these groups are low most of the time.

Fig. 4 shows the result of the second set, where 40% of users are low quality. We see from Fig. 4(F) that credibilities of both low and high quality users are decreased, and thus the difference between actual and assessed reputation is high. But within the same time, credibilities of low quality users are still decreased more, which reduces the difference between actual and assessed reputation. The simulation results show that our approach can effectively assess the reputation of providers even when the percentage of low quality users reaches 40% of the total number of users.
5.3. Trust engine comparison

To further evaluate the effectiveness of our proposed approach, we compare its accuracy with the conventional approach (in which rater credibilities are ignored and reputations are mere averages of all ratings), and a variant of the PeerTrust approach (a popular heuristics-based approach for P2P systems that also considers rater credibilities) [51]. We model malicious behaviors by experimenting under two settings, namely: “with no collusion” and “with collusion” (similar to the approach in [51]). In the setting with no collusion, malicious peers provide incorrect updates during transactions, and some raters (malicious) provide dishonest ratings. In the collusive setting, malicious peers perform similarly to the previous setting, and in addition, collude with other peers to increase/decrease some provider’s reputation (i.e., by attesting to an incorrect update). We change the percentage of malicious raters (denoted $S_m$) in steps of 10%, and consider a transaction as successful if post-transaction completion, the re-conciliated update is close to the ‘true’ update. Thus, transaction success rate ($TR$) is defined as the total number of successful transactions over total number of transactions in the community.

Fig. 5 shows the effects of changing $S_m$ for our proposed Automatic Conflict Resolution approach (ACR), PeerTrust-V, and the normal case where no trust system is used, for the two settings. We can see that since the raters provide dishonest ratings all the time, ($TR$) drops at a consistent rate, and the two settings exhibit similar results. However, ACR clearly provides slightly better results. In the collusive setting, ACR is fairly able to withstand the dishonesty till 45% of the raters are malicious, but the success rate drops thereafter. Since ACR relies on rater testimonies, when majority of the ratings are dishonest,
it becomes difficult for the system to assess the truth. Incorrect (majority) ratings are considered credible in each time instance and $TR$ drops. PeerTrust-V performs in a similar manner, i.e., with increasing $Sm$, $TR$ is brought down.

5.4. Execution time comparison

We also evaluate the execution time of our approach in comparison to the Orchestra system [44] (a primary CDSS). In orchestra, reconciliation is not trust-based, and transactions are applied (i.e. reconciled) if they satisfy a given set of requirements. Others are either deferred or rejected. Fig. 6(a) shows the execution times for an average peer with one transaction. Here we assume that a distributed storage scheme is followed, where "requests to follow antecedent transaction chains dominate the running time" [44]. We can see that ACR’s running time is slightly higher than Orchestra, due to the number of trust-messages exchanged in addition to the normal updates. In either case, frequent reconciliations put a heavier load on overall system resources, potentially reducing performance. Similarly, Fig. 6(b) shows the execution times with increasing number of participant peers. We can see that with a higher number of peers, more transactions need to considered and compared. This automatically increases the number of trust-messages across the network, and thereby the total reconciliation time. However, we posit that the automated reconciliation that ACR provides, with better accuracy, justifies the slightly higher running times.

6. Conclusion and future work

We presented an approach to resolve conflicts that may arise due to the propagation of updates among related peers in a CDSS. The focus is to resolve conflicts in the deferred set (of a CDSS’s reconciling peer) by collecting feedbacks about the quality of the conflicting updates from the local community (i.e., local users) and remote peers. When a new conflict group is added to the deferred set of a reconciling peer, it first inquires the participant remote peers about their experience while dealing with the provider peers that have updates in this particular conflict group. Then, for each provider peer in the conflict group, the reconciling peer aggregates the rating values received from remote raters to compute the remote assessed reputation value ($RRPP$) of the provider peer. Second, after a new conflict group is added to the deferred set of a reconciling peer, local users also rate the provider peers that have updates in this particular conflict group according to the quality of their updates. It then computes the local assessed reputation ($LRPP$) for each provider peer in the conflict group. Last, the assessed reputation of each provider peer in the conflict group is aggregated by weighting both $RRPP$ and $LRPP$ values. Thus, the
reconciling peer can resolve conflicts in a conflict group by accepting and applying the update that comes from the provider peer with the highest reputation value to its local instance, provided it does not violate its state. All other updates in the conflict group are rejected. Experimental results suggest that the CDSS can be extended with very little overhead (in terms of execution time) to automatically and efficiently resolve conflicts that may arise during the reconciliation operation of a participant peer. We plan to extend this work, to utilize community feedbacks not only to resolve conflicts for the updates in the deferred set, but also to deploy community feedbacks for the purpose of automatically defining trust policies for the local peer, thereby omitting the role of the administrator in defining trust policies.

References


