An Architectural Approach to Composing Reputation-based Trustworthy Services

Suronapee Phoomvuthisarn1, 2, 4, Yan Liu1, 2, Jun Han3

1 National ICT Australia (NICTA), Australia,
2 School of Computer Science and Engineering, University of New South Wales, Australia,
3 Faculty of ICT, Swinburne University of Technology, 3122 Hawthorn, Australia
4 Mahanakorn University of Technology, Bangkok, Thailand

{suronapee.phoomvuthisarn, jenny.liu}@nicta.com.au, jhan@ict.swin.edu.au

Abstract
In SOA, Reputation-Based Trust (RBT) mechanism is applied to achieve trust management. RBT enables services to assess the trust level of other services based on the reputation accumulated from user recommendations. A key challenge to apply RBT is to prevent the inciting behavior of users when they provide recommendations -- they might give an unfair rating to benefit themselves. In this paper, we propose a novel architectural approach to integrating auction mechanisms into the trust framework to prevent benefits from untruthful incentives. In this architecture we define an auction-based trust negotiation protocol and realize it in the trust framework. The contribution of our architecture is that it scales and produces accurate results to achieve protection against untruthful incentives, especially when a majority of ratings are unfair, without the potential increase in a computation overhead. An example on a travel agent scenario is devised to collect empirical evidence.

Key Words- Service Oriented Computing, Trust, Auction Mechanism, Software Architecture

1 Introduction

Trust plays an important role in service oriented computing. It facilitates secure service interactions, especially when services have little or no trustworthy information about others. One of the trust mechanisms commonly used is Reputation-Based Trust (RBT) mechanism [1]. Such mechanism assumes that services’ (i.e., software agents on behalf of users) can provide recommendations or ratings for others. These ratings can be aggregated into a meaningful reputation that can assist services to decide whether or not to interact with a particular service in the future.

Quality of service (QoS) is the major factor of decision making in service oriented environments [9]. Currently, most of the known trust frameworks [1], [5] allow services to rate the QoS of each other. However, raters might provide ratings for their own benefits. For example, a service provider might collude with other services to rate its offered service as positive in order to enhance the provider’s reputation or it could be targeted by some other services to deliberately lower the provider’s reputation by negative ratings. Hence, the reputation gathered from these ratings is often compromised and cannot be used with full confidence to determine the trustworthiness of individual services.

To address this concern, a number of approaches have been developed to detect unfair ratings. They are mostly based on statistic methods and treat unfair ratings as outliers of the samples [2], [3], [7]. This statistical data is automatically learned to recognize complex patterns and make intelligent decisions which would help reputation systems to foresee the trend of untruthful behavior.

However, the problem of using these approaches is that raters may not have a direct incentive, or may lose motivation in providing ratings. Raters may easily feel irritated or reluctant to go through tedious rating steps to provide online feedback, or even abandon visiting this service, which finally would misrepresent the reputation due to insufficient ratings captured.

To solve this problem, researchers have been, instead, working on developing preventive mechanisms. Their aim is to encourage honesty for raters to faithfully provide ratings. Rather than relying on a statistical learning process, preventive mechanisms give raters some incentive (e.g., digital currency or credit) so that truthful reporting maximizes the raters’ expected revenue and no one would like to deviate from reporting the truth.

However, these preventive mechanisms cannot ensure truthful reporting when the majority of raters lie. This is because these approaches assume the majority of ratings provided must be fair, and unfair ratings are in the minority as outliers. For example, the side payment scheme [13] assume that a rater providing a truthful rating will be rewarded and get paid only if its rating is the same as the next rating of the same rated service provided by

---

1 We assume that all services follow common rationality and irrational behavior is not considered.

2 We refer the term “raters” as services providing ratings for other services.
another rater. Thus the raters might receive a payment if they give similar ratings to many others, which is unfair.

In addition, these preventive mechanisms currently in use [11], [13], [14] usually incur a high computation overhead. Such mechanisms require detecting all actual ratings submitted by raters and respectively long computation time to produce complete and accurate estimates of all the raters’ payments. Thus, the computation overhead caused by integrating such approaches might outweigh the gain obtained from preventing unfair ratings. Therefore, novel techniques should be devised to offer both minimal costs of computation and support the prevention against inciting behavior, especially when the majority of raters provide unfair ratings.

Auction mechanisms designed for marketplaces have already addressed the problems to prevent inciting behavior of market trading participants [4]. Unlike other approaches, the auction mechanisms aim to make sure “lie does not gain” where the measure of gains for each bidder does not rely on the majority of bids from others [6]. This property makes it in the best interest of one participant to report the truth no matter how others act, making it possible to prevent unfair raters gaining benefits when the majority of ratings are unfair.

As a result of the auction mechanisms’ impressive properties, the costs of computation are minimal because: (1) the behavior of participants is simple and easy to implement; and (2) it saves the complex knowledge representation of how other participants will behave. Recognizing the merits of auction mechanisms, we incorporate them into a trust negotiation protocol. The aim is to prevent ratings from being exploited by unfair raters while ensuring an affordable computation overhead.

In this paper, we propose an architectural approach to integrate an auction mechanism within the existing trust framework [5] to prevent inciting behavior. More specifically, we utilize the well-known Vickrey-Clark-Groves (VCG) mechanism that can theoretically force participants to truthfully reveal their private information, otherwise they lose their gains or even receive penalties.

Our contribution is threefold: (1) to introduce a novel architecture leveraging an auction mechanism to prevent inciting behavior in rating services; (2) to ensure the computation overhead incurred is minimal; and (3) to promote a direct incentive for raters to provide fair ratings.

The structure of this paper is as follows: Section 2 discusses how to prevent unfair ratings using VCG and its challenges. Section 3 presents a general use case that VCG is applied. Section 4 proposes the architecture that supports VCG. Section 5 demonstrates our approach via an example on a travel agent scenario and evaluates the overall architecture with empirical experiments. Section 6 describes related work. The paper concludes in Section 7.

2 Background

2.1 Trust level calculation

The degree of trust is usually represented as trust level, which is a collective measure of entity’s trustworthiness. The trust level can be calculated from the summation of the weighted direct trust (i.e., Beta distribution value $\varepsilon$ [0..1] (a real number between 0 and 1) [7]) that one entity has on a targeted entity and the targeted entity’s reputation provided by a neutral party as shown in Eq.1.

$$\text{Trust Level} = (\alpha \times \text{Direct trust}) + (\beta \times \text{Reputation}) \quad \text{(Eq.1)}$$

where

$\alpha$ and $\beta$ represent weights $\varepsilon$ [0..1], which is subjectively determined by the entity depending on how important each source of trust is ($\alpha + \beta = 1.0$).

A neutral party can acquire the reputation of each entity by collecting ratings from other entities that have previously interacted with a targeted entity. The reputation of one entity can be calculated by aggregating all values of ratings submitted into one percentage measure $\text{Reputation} \varepsilon [0..1]$, each of which is weighted by its rater’s reputation as shown in Eq.2.

$$\text{Reputation} = \sum_{i=1}^{n} w_i \times \text{rating}_i \quad \text{(Eq.2)}$$

where

$w_i$ represents a weighted reputation $\varepsilon [0..1]$ of a rater.

$\text{rating}_i$ represents a rating rated by a rater $i$.

$n$ represents a number of raters.

2.2 Motivating example

In this section, we illustrate the problem of unfair ratings in a trust-based service environment, which motivate the need for VCG within the architecture.

Consider the case of a travel agent scenario, where a travel agent service intends to provide complete travel information to customers by combining room reservation services from several hotels that matches its functional
requirements. The required functional features are the ability to book complete vacation packages including room bookings, tour bookings and transportation services to/from the airport. However, a key challenge lies in how to choose the best deals available among all the hotels offering the same function.

In this business-to-business integration, the travel agent and each hotel are independent of the services or resources they provide. Hence, each service’s interaction has some risk of unsuccessfulness such as not reaching customer satisfaction in terms of bad quality resources, not providing a service in a promised timeline, or failing to provide a service at all. Therefore, the overall QoS attributes including the amount of satisfaction, delivery time, and availability can be used by the travel agent to select relevant hotels from a list of hotels offering comparable functions.

However, the arbitrary process for each hotel to reveal its QoS properties makes the travel agent vulnerable in selecting hotel services with inaccurate QoS information. Some hotels might exaggerate their QoS capability on purpose to attract any interested services for some benefits (e.g., for money). Hence, to ensure that the QoS of each hotel is the same as it claims, the trust level of a hotel can be used to determine the probability that the hotel will fulfill its guaranteed services.

A modeling of trustworthiness for hotels can be based on the travel agent’s past personal experience (or historical records) with the hotels. However, in our case the travel agent does not have much history interacting with all the hotels and it involves risk when making decisions based solely on his own experience. Thus the travel agent needs some additional advice from a neutral party that can provide public opinion about the hotels’ QoS information. To acquire this public opinion, the third party collects QoS ratings from other services (e.g., hotels, car rentals, or customers) that have previously interacted with the hotels into a collective evaluation of a group opinion called reputation.

For simplicity in explanation, we nominate the term “QoS rating” ε [0..1] to represent the average value of QoS attributes submitted by each rater, including the amount of satisfaction ε [0..1], delivery time ε [0..1] and availability ε [0..1], such that a quality of 1 denotes the best possible service as shown in Eq.3.

$$\text{QoS rating} = \frac{\text{the amount of satisfaction} \ast \text{delivery time} \ast \text{availability}}{3}$$  (Eq.3)

The problem then arises when the reputation relies on the aggregation of each QoS rating given by an individual rater. Some raters might provide unfairly high or low QoS ratings for their own benefits (e.g., boosting their partners or lowering competitive services’ reputation), resulting in the reputation compromised due to the unfair ratings captured. This unreliable reputation would lead the travel agent to interact with some hotel service, which indeed is completely untrustworthy. As a result, the travel agent might experience some unexpected QoS (e.g., slow response time), which itself leads to an unwanted service or even failure to serve customers in time.

To solve this problem, some mechanisms need to be integrated with the neutral party to ensure the robustness and accuracy of QoS ratings accumulated. The integrated mechanisms should force raters to give only fair ratings.

### 2.3 The VCG auction mechanism

VCG can play a role in the context of above problem. In theory, VCG can force each rater to faithfully reveal his rating. It is a specific type of auction. It is known simply as the second-price sealed-bid auction (or Vickrey) in which bidders place their bid on the items and hand them (sealed) to the auctioneer. The winner of the auction is the individual who places the highest bid, and pays a price equal to the exact amount of the second-highest bid.

In the single-item VCG auction, the utility gain for each bidder is the difference between the true value each bidder places on an item and his payment (i.e., a monetary value each bidder gains after auction) defined as follows:

$$u_i = \begin{cases} v_i - \max_{j \neq i} b_j & \text{if } b_i > \max_{j \neq i} b_j \\ 0 & \text{otherwise} \end{cases}$$  (Eq.4)

where

- $v_i$ represents bidder $i$’s true value for an item
- $b_i$ represents bidder $i$’s bid for an item

According to Eq.4, the winner gets a utility which is equal to the different between its valuation and the second highest bid, while the losers lose nothing (i.e., utility = 0).

This set of VCG rules govern the interaction of self-interested participants with preferences to obtain limited resources through auction and guarantee that individuals not telling the truth would not gain [6]. This principle of VCG ensures that bidding something other than the bidder’s true valuation is never beneficial and sometimes was detrimental with penalties.

### 3 The approach

As discussed above, VCG can be potentially exploited to resolve the unfair-rating issue in a reputation-based trust mechanism. In this section, we first discuss the challenges of applying VCG to solve the problem of unfair ratings and second present a general approach that VCG is applied in a trust-based service environment using the travel agent scenario as an example for illustration.

#### 3.1 The challenges in applying VCG in RBT

To solve the problem of unfair ratings, VCG can theoretically force each rater to faithfully reveal his rating. However, using VCG in a trust-based service environment is not straightforward due to several aspects. One aspect is the measure of gains. The outcome of VCG,
which is the utility gain for each bidder, is represented as money. Bidders are aware of losing money, which leads to truth-telling behavior. However, money may not be a direct measure or imply any trust based information for non-human users such as computing services.

In RBT, instead, reputation can be used to represent the perception of raters’ trustworthiness, and can guide the interaction of future encounters. Therefore, utility gain should be converted into a reputation that reflects each rater’s own interests. If raters get negative utility gains when they give an unfair rating, their reputation will be degraded. As a result, these unfair raters are eventually forced out of business due to their low reputations.

However, if raters get positive utility gains when they give a fair rating, their reputation will be increased. This encourages raters participating in the auction by giving them a chance to enhance their reputation as a reward. Therefore, one way of making a utility gain of interest to raters is to use their reputation, instead of money, for bidding in the auction.

3.2 Prevention of unfair ratings using VCG

In the travel agent scenario, by utilizing the VCG capability, a neutral party can manipulate the auctions so that all raters (i.e., services providing QoS ratings to hotels) would not deviate from reporting truthful ratings.

To enforce the VCG property, the reward and punishment process is conducted to promote truthful incentives for raters. This is achieved by using reputation as the measure of gains. At the end of the auction, the winning rater will enhance their reputation proportional to the utility gain/loss, which is then accumulated with the winning rater’s existing reputation.

However, this newly updated reputation would be meaningless if it cannot be stored as a reference and made publicly known to others. This requires VCG to coordinate with service capabilities (i.e., a service registry) to manage the reward and punishment process in each rater’s reputation as depicted in Figure 1.

![Figure 1. The problems of unfair ratings](image)

3.3 Auctioning process

To acquire truthful ratings through auctions, the trust framework needs to generate auction services (for each hotel) in order to get QoS ratings\(^2\) from other services that have previously interacted with the hotels. These QoS ratings are then used to calculate each hotel’s reputation.

Figure 2 shows the key elements of the auctioning process enhanced with the VCG capability in order to force raters to truthfully reveal their QoS ratings. Our auctioning process includes a service requester (e.g., the travel agent), services (e.g., raters), a trust framework, and auction services (e.g., for all hotels being rated).

![Figure 2. The auctioning process](image)

The process of eliciting truthful QoS ratings starts when the travel agent first inquires about the hotels through the trust framework. Once receiving a list of the hotels, the trust framework then generates auction services to elicit QoS ratings from other raters.

To participate in the auctions, raters can give QoS ratings of the hotels into one or more than one auction according to their past experience with the hotels. The QoS ratings submitted can be interpreted as the resources in terms of current bidding reputation (measured as credits) of each rater.

The credit that a rater can spend for the auctions can be calculated by the multiplication of the rater’s current reputation (i.e., a percentage measure between 0 and 1) stored in the service registry and its total number of transactions previously conducted as shown in Eq.5.

\[
\text{Credit} = CRep \times T \quad \text{(Eq.5)}
\]

where
- \(CRep\) represents a current reputation of a rater.
- \(T\) represents the total number of previous transactions.

At the end of the auction, the trust framework then coordinates with the service registry to either enhance or degrade the winning rater’s reputation based on the utility gain or loss (using Eq.4), respectively. The service registry then later makes the newly updated reputation publicly known for future encounters.

To update a rater’s reputation with the calculated utility gain, a rater’s newly updated reputation is calculated by accumulating the rater’s utility gain with the rater’s current reputation, each of which is weighted by their total number of previous transactions shown in Eq.6.

\[
\text{Update Reputation} = \frac{(Utility \times 1 + CRep \times T)}{T + 1} \quad \text{(Eq.6)}
\]

where
- \(Utility\) represents the utility gain of a rater after auction.
- \(CRep\) represents a current reputation of a rater.
- \(T\) represents the total number of previous transactions.
- The number of transactions for an auction is equal to 1.

\(^{2}\) We assume that raters submit QoS ratings independently of others.
4 The architecture

The architecture aims to provide a loosely coupled solution that encapsulates the VCG mechanism in components. We first discuss the challenges in devising the architecture when integrating with VCG. We then present the architecture with its composed layers, and introduce the key components designed to prevent inciting behavior. The auction-based trust negotiation protocol that guides the interactions between the key components is presented at the end of this section.

4.1 Architecture challenges

Using VCG to reveal RBT information poses a number of challenges from the architecture point of view, especially on integrating with VCG.

The first challenge is to achieve good separation of concerns. Communication relationships amongst trust-based components involve the use of protocols. As a result, existing components are tightly coupled with pluggable components (e.g., VCG’s components) designed to enhance the capability of the original trust framework. Thus, there is the possibility that a poorly-designed architecture might introduce tight couplings between VCG components and the original trust framework. Trust and VCG components should overlap in functionality as little as possible so that modifications to the trust components or replacement of the VCG components can be maintained without reengineering other parts of the architecture.

The secondary challenge is to capture the key characteristics of the deployed trust-based scenario. Each rater has its own preferences to choose their strategies that maximize his or her utilities. For example, one rater might have strategies to undertake fake transactions to give positive ratings for inflating a partner service’s reputation or negative ratings for lowering a competitive service’s reputation. Hence, the architecture should serve to handle varying preferences for all strategies to ensure that submitting unfair ratings cannot provide benefits to unfair raters. In addition, the resources each participant put into the auction depends on each strategy the participant performs. This requires the relations between the strategies and their required resources captured by the architecture.

4.2 Architecture layers

Conceptually, the architecture has three layers to compose the key components in Figure 3 namely service layer, trust layer and service metadata layer.

4.3 Key components

The key elements in the trust framework are the Trust Engine and the Reputation Engine. The reputation engine produces the reputation-related metrics and inputs to the trust engine. The details of each engine are shown in the architecture depicted in Figure 3.

4.3.1 Trust engine

The trust engine is first used to authenticate a service before it can start eliciting other services’ trust level. The authentication process verifies a requester’s identity based on the requester’s credential whether it has sufficient trust to elicit the trust level of other services. After the authentication process is completed, the trust engine then calculates other services’ trust level requested by a service requester.

The trust engine consists of two major components, namely the Credential Manager and the Trust Computation. The credential manager component is used to check the existence and validity of credentials. The trust computation component computes the trust level of certain services requested by a service requester.

4.3.2 Reputation engine

Once the authentication process is done, the trust computation component then invokes the reputation engine to provide the reputation of services requested.

The reputation engine is structured with three loosely coupled components, namely Reasoning Manager, Auction Engine, and Controller module.

The reasoning manager is responsible for providing reputation and updating the service registry. To compute a
reputation, the reasoning manager instructs the auction engine to initialize auction services and gather all values of ratings from the controller module to calculate the reputation. After the computation is completed, the reasoning manager stores each rater’s newly updated reputation to the registry.

The reasoning manager also works with the controller to detect the ratings provided by raters that have insufficient reputation to bid. Once receiving the ratings submitted from the controller, the reasoning manager then performs checking the corresponding raters’ resources (i.e., reputation) with the service registry. The ratings from the raters that do not have enough reputation to bid are then excluded from the auction. At the end of the auction, the reasoning manager then sends all values of the ratings that are not excluded to the auction engine.

The auction engine is the key component for maintaining the auction logic. It consists of the VCG component to manipulate the auction based on the VCG logic embedded. The VCG component computes the utility gain of the raters participating in an auction based on the utility function defined in Eq. 4. This utility would be used as outputs to the reasoning manager in order to give a reward to fair raters or make some punishment to unfair raters.

To calculate this utility, the true value of the rating submitted by the winning rater is required. This is achieved by the monitoring process made by the controller. As we cannot know a priori winner in the auction, the true value can only be measured after the auction. This can be achieved by using the controller to dispatch the monitoring service to monitor a rated service. The monitoring service itself becomes a normal client (with the same environments as the winning rater) of the rated service, and requests the rated service with the sole purpose of rating it. When the controller completes its task, it sends back the true value to the auction engine.

4.4 Auction-based trust negotiation protocol

To enable this architecture to prevent inciting behavior from unfair raters, the steps of the VCG mechanism are embedded in the trust negotiation protocol, which guides the interactions between layers in this architecture. This protocol is realized by key components across three service layers. The protocol consists of four stages, namely interrogation, negotiation, interaction and termination. The invocations and messages of the auction-based calculation are labeled with the stage number as shown in Figure 4.

At stage one interrogation, a requester at the service layer find services that meet its functional requirements.

At stage two negotiation, the requester checks the trust level of the discovered services through trust negotiation.

To calculate the trust level, the trust engine first instructs the reputation engine to calculate the rated service’s reputation through the reasoning manager. Once receiving the requests from the trust engine, the reasoning manager then instructs the auction engine to perform auction-based calculation.

The calculation steps are as follows: (1) the auction engine initiates a new auction round (for each rated service) by setting an auction time and aggregating ratings from raters; (2) the auction engine terminates the auction; (3) the reasoning manager sends all values of ratings already excluded to the auction engine; (4) the auction engine finds the winning rater and instructs the controller to monitor the winner’s true value; (5) the controller then monitors the winning rater’s true value; (6) the auction engine instructs the VCG component to perform utility computation; (7) the auction engine then announces the winning rater and its utility through the reasoning manager; (8) the reasoning manager perform the reward and punishment process and update the newly raters’ reputation to the service registry; (9) the reasoning manager then calculates a rated service’s reputation (using Eq.2).

The trust engine then uses the reputation produced by the reasoning manager to calculate the rated service’s trust level (using Eq.1). Once the computation is completed, the trust level is then returned to the requester to determine whether each rated service’s trust level exceeds the requester’s minimal trust threshold to further negotiate.

At stage three interaction, the requester negotiates with chosen services to interact with. The requester and the chosen services use the trust framework as a mediator to establish the trust negotiation between them.

At stage four, termination, after the trust negotiation is established, the requester then locates the chosen services.

---

Figure 4. Auction-based trust negotiation protocol
4.5 Discussion

The architecture is designed to address the integration of the VCG mechanism with the trust framework. The VCG mechanism is encapsulated in the auction engine, which interacts with the rest of the architecture through the reputation engine. The main advantages of our architectural design are threefold.

First, the decoupling of the reputation engine and the trust engine enables an extensible architecture - modifying or updating the auction mechanism such as VCG in our case without affecting other components. Therefore, the architecture can scale to support the integration of auction mechanisms without degrading existing trust-based capabilities. The architecture further decouples the controller that captures events, the auction engine that computes the VCG logic, and the reasoning manager that maintains a presentation of reputation within the reputation engine. Thus the architecture helps to reduce the complexity of the architectural design, allowing maintenance of each component independently.

Second, the architecture delivers the desired property of the VCG mechanism through the VCG component, which interacts with other components according to the VCG logic embedded in the trust negotiation protocol.

The outcome of the VCG is communicated by the reasoning manager in the publish-subscribe model, which uses asynchronous message communication to send out the result to a number of subscribers. This asynchronous communication makes our architecture much more efficient when performing an auction amongst dynamic raters whose availability might not be known.

Finally, the architecture also makes it possible for the reasoning manager to capture varying bidding strategies so that the desired property of VCG is maintained.

Since the VCG’s impressive theoretical properties is incentive compatible; making it in the best interest of raters to truthfully reveal their private information, the behavior of the raters is much simple and consequently has only one dominant strategy (i.e., report truthful ratings) for all raters. However, there might be some case that this dominant strategy does not hold.

For example, an aggressive rater can perform his strategy by bidding his resource (e.g., reputation) more than he actually can afford. Such a rater would win the auction but he doesn’t have sufficient reputation to bid. As a result, the affected auction might be canceled due to his insufficient resources. Therefore, this bidding strategy might motivate some raters who want to hamper the calculation process for one service’s reputation based on their own benefits (e.g., hide his partner’s bad reputation or obstruct one service to discover the competitor’s good reputation). Consequently, the VCG desired property is violated since the overbidding strategy is much more preferable than reporting truthful ratings.

This can be prevented by the tight coupling between the reasoning manager and the service registry. The reasoning manager can dynamically detect bidding strategies and perform checking on each rater’s resource by the service registry at runtime to exclude ratings from raters who have insufficient resources to bid in an auction.

5 Case study and evaluation

In this section, a scenario derived from a travel agent is used to demonstrate our architecture. We first evaluate how effective our mechanism can prevent any benefits when raters lie and can also scale to handle the majority of unfair raters. We second observe the computation overhead incurred when integrating VCG in the architecture.

5.1 Travel agent scenario

Consider the travel agent scenario in section 2 (depicted in Figure 5), where a travel agent service sends the trust framework a request asking for the reputation of all potential hotels that have similar function in providing complete vacation packages. The reputation produced can help the travel agent to form a trust belief on a hotel and to compare the trustworthiness of a set of hotels that are likely to provide their guaranteed quality of services.

To acquire ratings to calculate each hotel’s reputation, the trust framework allows any services (e.g., car rentals, other hotels, and car rentals) to give their QoS ratings (i.e., the amount of satisfaction \( \varepsilon [0..1] \), delivery time \( \varepsilon [0..1] \) and availability \( \varepsilon [0..1] \)) of hotels through auctions.

Figure 5. Travel agent scenario

We build a simulation operating with our framework. The simulation involves 50 raters and up to 500 hotels. These raters are grouped into two groups: fair and unfair rater. In our setting, we initially set the reputation of unfair raters to any random numbers between 5 to 15 percents while for the fair raters is between 10 to 20 percents based on 100 transactions previously conducted.

A total of auction rounds (equal to the number of the hotels) have been executed with 1 minute per auction. A fair rater offers QoS ratings to a hotel exactly the same as it perceived while an unfair rater randomly offers QoS ratings above or below it perceived. To simplify our problem, we consider the case where all raters have past experience with a certain hotel with 70% probability. Each of them has to rate each hotel with 50% probability.
At the end of each auction, a rater’s reputation is updated based on the utility gain/loss using Eq.6. For example, if a rater, whose reputation is 20% for 100 transactions conducted, get +0.6 utility gain, the newly updated reputation is (0.2 x 100 + 0.6 x 1) / (100 + 1) %.

To make a trust decision whether to interact with a hotel, the travel agent chooses one or a subset of potential hotels whose reputation exceeds its trust level’s threshold (trust level ≥ 0.7) to interact with.

The architecture and the key components presented in section 4 are now applied to process these ratings to ensure the effectiveness of VCG to prevent the benefits gained by unfair raters when they simply lie, and the appropriateness of the VCG approach in terms of the computation overhead (measured by performance metrics) it incurs. To demonstrate our approach, two tests are devised as follows:

- **VCG property test** evaluates the practical usage of the architecture in the case that it can efficiently avoid untruthfully incentives. Unfair raters should gain nothing or even get a penalty, especially when the majority of raters lie about their ratings,
- **Computation overhead test** evaluates the performance of the architecture when using VCG. The plan is to implement two prototypes with and without VCG deployed, and compare both performance metrics.

### 5.2 Testbed setup

The architecture in Figure 3 has been implemented as a set of Web services (depicted in Figure 6). All services including the travel agent, all raters and all hotel services are hosted by the Apache AXIS 1.0 Web Server. The travel agent service and all the raters’ services interact with the trust-based service application via the Web service deployed in the IIS5.0 Web Server that receives service requests and sends responses from the application.

The reputation and the trust engine are developed as Java EJBs and deployed as the single trust-based application hosted by the Tomcat Application Server. This Application Server processes requests from the Web Server and sends responses back to the Web Server.

The Application Server implements the auction-based trust negotiation protocol. It communicates with services and the service registry using SOAP messages and connects to the SQL Database Server using the JDBC driver. The service registry implements LDAP components to support service registration.

The test environment includes two identical Windows XP machines with 3GHz Core 2 Duo processors. One is used for hosting the trust-based service application, and the other hosts the rest of the server components.

### 5.3 The experimental results

#### 5.3.1 VCG property test

The purpose of this experiment is to examine how our mechanism can effectively prevent benefits gained by unfair raters when they lie about their ratings. The gain can be measured by the changes in the reputation of unfair raters when they constantly provide unfairly high or low ratings to targeted hotel services. We then observe and compare the results of changing in all raters’ reputation when the number of the targeted hotel services increases.

We performed a series of experiments by varying the number of unfair raters from 10% to 90%, with 10% incremental per experiment. Each experiment involved 50 raters, each of which had to rate QoS ratings of hotels (i.e. the amount of satisfaction, delivery time and availability) based on given probabilities in section 5.1. A total of 10, 300, and 500 auction rounds (equal to the number of the targeted hotel services) have been executed to observe the reputation changes incurred by raters’ inciting behavior.

Figure 7 shows that the unfair raters’ reputation decreases when the number of raters lying increases. This is because when the majority of raters are unfair, the unfair raters have more chance to win the auction, however their gains of reputation are impaired by the punishment made to these unfair winning raters by VCG. In contrast, we can see that the reputation of fair raters increases as the result of the reward granted by the VCG process.

![Figure 7. The changes in the reputation of raters](image-url)
unfair. These VCG-based rating results are in clear contrast to existing preventive mechanisms that fail to motivate raters to report truthful ratings when the majority of raters lie. Also, our approach promotes a direct incentive for fair raters participating in an auction due to the increasing of reputation when they give fair ratings.

5.3.2 Computation overhead test

In this experiment, we intend to observe the computation overhead of the trust framework when integrating VCG. We measured the computation overhead in terms of CPU usage, memory usage, and application response time (second) with and without VCG by varying a number of raters from 50 to 1,000. Experiments were performed with 10 concurrent auctions (i.e., 10 hotel services), each of which is conducted for one particular hotel service.

The results show that no significant overhead in terms of memory usage for 1,000 raters (see Figure 8). We can notice the slightly CPU overhead (48% vs. 53%) that can be linked to the computation of VCG to search for the winning rater and the second highest bid among a large number of 1,000 raters. However, the response time is initially identical regardless of using VCG (see Figure 9). This is due to the fact that the computation of each auction can be performed in parallel. The performance overheads are approximately 3.9% higher than those without VCG, as the number of raters is up to 1,000.

The plot demonstrates that the VCG mechanism is lightweight and does not produce any significant computation overhead to the original trust framework.

<table>
<thead>
<tr>
<th>For 1,000 raters</th>
<th>Without VCG</th>
<th>With VCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage</td>
<td>48%</td>
<td>53%</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>17.42%</td>
<td>19.35%</td>
</tr>
</tbody>
</table>

Figure 8. CPU and Memory Usage with and without VCG (For 1,000 Raters)

Figure 9. The performance variation when using VCG

6 Related work

Much previous research has applied statistical methods to handle the problems of unfair ratings. These approaches detect and exclude unfair ratings based on the recognized patterns automatically learned from statistical data. Dellarocas [2] proposes two schemes for detecting unfair ratings. The author uses controlled anonymity to avoid unfairly low ratings and cluster filtering approach to separate unfair and fair ratings by grouping the members in the nearest neighbor set according to the values of their ratings. Whitby et al. [7] proposes iterated filtering approach extending with beta distribution to filter out the ratings that are not in the majority of fair rating, the ratings between the boundaries of the distribution.

Although these detective techniques provide a promising approach to predict the trend of unfair ratings, they still suffer from one major drawback – lack of sufficient ratings. The main reason is that the raters might not have a direct incentive to provide ratings to rate others. This is because providing ratings for others requires some effort which might end up with losing business profits due to wasting time or decreasing the bandwidth of raters’ running services. As a result, the trends of untruthful behavior cannot correctly be used to detect unfair ratings or even can misrepresent one service’s reputation due to insufficient ratings captured.

The limitations of detective mechanisms have thus drawn intensive research activities on developing incentive techniques to either eliminate or prevent incentives to lie. The aim is to encourage raters to give faithful ratings, otherwise they will either lose their own benefits. However, no incentive mechanisms can claim their victory. They still have some limitations.

Payment mechanisms offer side payment to raters that fairly rate others. These mechanisms guarantee that lying is not in the best interest of the raters. Dellarocas [14] proposes “Goodwill Hunting” mechanism that encourages sellers to truthfully reveal their qualities of product by rebating some payment to sellers based on the similar quality of transactions among the whole communities.

Jurca et al. [13] describes incentive compatible payment scheme organized through a set of broker agents. These agents buy feedback and sell reputation information aggregated from the feedback. The author makes faithful reporting an optimal strategy by devising a payment scheme that pays a submitted report if it has the same value of randomly chosen report. Miller et al. [10] also propose an incentive mechanism using a proper scoring rule that is pretty similar to Jurca et.al except that the payment is made to raters by buyers after the next buyers instead of broker agents.

However, the side payment scheme does not work well when the majority of raters lie. This is because these approaches offer a side payment to raters depending on the majority of ratings provided by other raters. Therefore, if the majority of ratings are unfair, this opens up the possibility for dishonest raters to gain benefits from the payment given to similar ratings as many others.

In addition to the limitations of the mechanisms itself, very little attention has been given to the costs of trust management incurred during integration process with the proposed mechanisms. Without a proper integration, the mechanism might increase the costs of trust management in terms of computation overheads, making the trust frameworks inefficient or even failure.

There is some body of work that tries to diminish this
concern. Braynov et al. [11] describes a trading mechanism in which sellers always truthfully declare their trustworthiness to buyers. This mechanism chooses the quantity of exchange products that maximize both utility functions of buyers and sellers. The works aims to reduce the costs of trust management by simplifying the mechanisms. However, this work is applicable to only small individual trades in which only the quantity of the exchange products can affect the utility function of both buyers and sellers. Moreover, no evaluation has been conducted to prove his claim of eliminating the costs of trust management. Hence, it is not clear to what extent the trust framework can correctly function when integrating with the mechanism.

Our work contrasts with this method as we are not only confine our attention to simplify the mechanisms, we also address the issue of integrating the lightweight mechanism with the original trust framework to reduce the costs of trust management (e.g., computation costs).

In our approach, we utilize the VCG auction mechanism, which has been widely accepted to help eliciting truthful information from participants, to achieve protection against inciting behavior. Besides such impressive truth-telling property, the simple VCG calculation logic does not depend on how other participants will behave, making it possible for the mechanism to scale handling the majority of unfair raters.

To integrate VCG with the original trust framework [5] without degrading its existing trust-based capabilities, the loosely-coupled component-based approach [8] can decompose the trust-based application into functional components, making the architecture scalable when replacing the VCG component without reengineering other trust components. The separation of concerns with well-defined interface used for communication between components also provides benefits to capture the VCG auction mechanisms in highly dynamic environments due to various inciting behavior captured.

7 Conclusion

In this paper, we envision the architecture of trust-based service applications integrating with the VCG auction mechanism. The VCG mechanism is encapsulated with the auction-based trust negotiation protocol and realized in relevant components. These components interact with trust components in our devised architecture to provide an additional capability to prevent inciting behavior. An example based on a travel agent scenario is implemented to observe the correctness of our approach when raters give unfair ratings or simply lie.

Our contribution is a complementary alternative to existing trust frameworks with the extra capability to prevent inciting behavior. The notion of the lightweight VCG induces an effective trust negotiation by preventing the trust-based service application form being exploited by unfair raters, especially when the majority of raters lie about their ratings. Our approach also encourages a direct incentive to raters to provide fair ratings.

One drawback, however, is that computation is all centralized in the trust framework, with a central application taking all the decisions. This imposes the research questions in the architectural design that leverages the VCG’s benefit of preventing inciting behavior with strengthened reliability and optimized computing overhead. Our ongoing work involves optimizing architectural solution to support decentralized trust framework with the VCG capability. The resulting architecture will be optimized to support performance and reliability qualities of attribute.

Acknowledgement

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References