Trust Management for SOA with Real-Time Data Validation

Kshitij D. Karki, Adeela Huma

Department of Computer Science, Virginia Tech

kdk560@vt.edu, ahuma@vt.edu

***Abstract*—In a community where there are multiple service providers that provide same or similar services, one of the biggest concerns for consumers is the time to wait to have the service available to them. While each service provider can provide service to multiple consumers at the same time, there is a limitation on number of parallel consumers due to different resource constraints. So when a service provider is providing service at the full capacity then any additional consumers will have to wait until the service provider is able to accommodate new consumers. Ideally, in such communities, the service providers can advertise their current wait time so that the consumers can decide which one to use assuming that the quality of service are relatively similar. However, the challenge is that a service provider may advertise a false wait time to lure or avoid more consumers. A service provider can collude with one or more service consumers in such false advertisements. A non-malicious consumer will not be able to verify the fact until it is able to directly interact with the service provider, which is a loss of time and resources. So in this paper, a centralized trust management design is proposed for systems where the consumers determine the best service provider based on the advertised time to wait. A consumer can determine the likelihood of the advertised "Time to Wait" of a service provider to be close to the actual time-to-wait based on 2 factors: a) the trustworthiness of the service provider from past experiences and b) from the verification from consumers currently receiving the service, the witnesses. Eventually the consumer will be able to validate both the advertised time to wait from the service provider as well as the verifications from the current consumers when it interacts directly with the service provider with the intention to use the service. Upon validation, the consumer will update its trust score towards the service provider and the witness consumers. The novelty of this proposed design is in the utilization of discounted validation information from witnesses in conjunction with the trust score of a SP to determine the likelihood of the SP advertising accurate information.**

***Index Terms*—SOA; trust management; services provider; service requester; beta reputation; collusion attack.**

I. INTRODUCTION

THERE are numerous examples in our day to day life where we find that consumers of a service are constantly trying to find the best service provider for a specific service. When there is a high number of service providers available to a consumer, the consumer will naturally try to determine the best one available. A common method of determining the best service provider is to compare them based on their advertised service. A service provider is expected to provide the service as advertised with some margin of error. However, a malicious service provider can advertise false claims. So several trust management designs have been proposed in the literature to mitigate such behaviors.

While a service provider can be recommended by other consumers, a direct experience is always preferred to determine the quality of service of the service provider. As the service providers compete for consumers by providing service that is on par with respect to one another, it eventually comes down to which service provider can provide the service earliest to the consumer when demanded. In other words, if multiple service providers are available with similar quality of service then the consumer will select the one with least time to wait. This paper specifically focuses on this aspect. As explained later in section III, it can be realized that this design can be extended to more than one criterion to determine the preferred service provider. However, for simplicity a single criterion "Time to Wait" is used to determine the best available service provider.

While this service model can be exemplified by a computer network design or resource sharing network design, a more interesting example would be that of restaurant services and their advertisement in a cloud service for expected wait time. In the world of eatery, the service providers compete with each other that are co-located and especially with the ones that serve the same or similar cuisines. There are several services available that allow customers to search for restaurants based on location, cuisine, and even ratings. A good example of this is Yelp. So, if a restaurant could provide the current "Wait Time" in such cloud services as Yelp then it would be really beneficial to the customer to decide to pick a restaurant. Of course, just like the ratings, the advertised wait time cannot be validated completely until the customer arrives at the restaurant and experiences the service quality. The trust management algorithm proposed in this paper will attempt to verify the advertised wait time and make best possible judgement.

The trust of a customer towards a service provider can be determined based on the past experience. However, the novelty of the trust management algorithm proposed in this paper is that the customer can verify the advertised wait time with other customers who are present in the restaurant at that moment, the witnesses. Every patron of the restaurant who are either enjoying the service or are in queue can verify the wait time based on their personal experience. In other words, they can act like a witness to verify the wait time advertised by the restaurant. While one or more patron can collude with the restaurant to advertise false wait time, the proposed trust management algorithm will minimize the effect of such malicious collusive attacks. Eventually, the service requester will validate the restaurant and each of the verifying witnesses by physically arriving at the restaurant.

The reason for centralizing the trust management is to avoid cold start issue as well as to avoid any limitations that are present in local storage of trust scores towards service providers and other customers. As discussed in [1], limited storage space is a big limitation. So different strategies are implemented to work around such limitations. Another advantage of centralizing trust management is because the services are advertised in the cloud service along with the quality of service. So as the customers access the cloud service to find the available service providers, having the trust score will add minimal overhead. While there are drawbacks of centralizing trust management that have been pointed out in literature, it works well for the proposed model.

The novelty of this work not only lies in the trust management algorithm but also in the way the service requesters participate in validating the advertised service quality of the service providers in real time. The contributions of this paper are as follows:

- In most cases when we think of service in cloud computing and IoT, we think of operations that require execution of few transactions that take under a second to few seconds. However, the kind of service that is considered in this paper are the ones that take a long time to complete during which time part of the resources of the service provider will be used by the consumer. So while a consumer is receiving service from the service provider, it can continue monitoring the service provider activities such that it can validate the quality of service they are experiencing and the service quality advertised by the service provider and judge the trustworthiness of the service provider.
- Since a service provider can provide service to multiple consumers simultaneously, it means there can be multiple service requesters receiving the service at the same time. However, this means when a service provider is providing service at full capacity then any additional service requesters who prefer to receive service from that particular service provider will have to wait for resources to be available in the service provider to serve it. When a service requester waits in the queue, it can validate the wait time from its personal experience to what is being advertised by the service provider and determine the trustworthiness of the service provider.
- While this paper does not consider recommendation trust to determine the trustworthiness of a service provider, it still considers utilization of other service requester to validate the quality of service of the service provider. When a service requester is able to receive service from a service provider, it will advertise its current experience in the cloud service. While receiving service from the service provider, the service requester becomes witness of the service provider. In fact, a service requester becomes witness of the service as soon as it goes in queue to receive service from the service provider. In order to downplay Sybil attack, the central system can verify that a single service requester can advertise validation data for only a single service provider at a time.
- Finally, a service requester will determine the trustworthiness of the service provider by directly interacting with it. After determining which service provider to use for a given service based on the advertised wait time, the trustworthiness of the service provider, validating with the witnesses, the service requester attempts to consume the service from the service provider. In doing so it will be able to find out the quality of service first hand and able to determine if the service provider advertised the service truthfully as well as if the witnesses validated truthfully.

This paper is organized as follows: In Section II, related works are analyzed such as trust models, centralized trust management system, and cloud services. In Section III, assumptions and system models are explained including threat models. In Section IV the trust management protocol is analyzed in detail. In Section V the proposed model is evaluated experimentally. In section VI we conclude the paper and outline future research direction.

II. Related Work

There are several important works in the literature of trust management in cloud services and service oriented architecture. The existing literatures cover concepts of trust propagation through both direct experience and through recommendation. A direct experience will always provide a consumer an accurate information about a service provider but it is limited because a consumer can only consume so many services. If a consumer wants to try something new or get into a new territory then a recommendation would be really helpful. However, identifying a reliable recommendation is quite a challenge. In either case, there is a common concept of using past experience from service providers to predict their future behavior. There are quite a few algorithms that predict with very high probability. So as a compliment to existing trust management protocols, a novel idea is presented in this paper which provides a basis for a service requester to receive a real-time information of the service provided by a service provider. Such real-time data can be combined with past experiences to predict the behavior of service providers more accurately.

Using real-time information may not be application in all types of services. However, in the cases where it is possible, it is obvious that it will add a great value and increase the predictability. One of the key concepts utilizing real-time information comes from [2] where Timpner et al. describe how to use direct assessment to build and manage trust score towards the constituents of a community to help each other find parking spaces. Trust assessment through recommendation is avoided not only to avoid any added complexity that comes with recommendation trust but also to provide a more accurate trust assessment that is possible through only direct interaction.

The concept of witness is derived from [3] where Erinaki et al. use the concept of witness to get opinion from a user regarding another user in a social community. The concept of witness is quite powerful due to the fact that it not only helps provide information based on direct experience, it also helps mitigate collusive attacks. In this paper, we further extended the concept of witness as described in [3] and utilize it to assess trust of service requesters too. Every service requester will have opportunity to be a witness but that comes with added responsibility of being truthful. If a witness falsify any information regarding a service provider then the reputation of the witness will go down which will make its claims in future less worth.

Finally, while there are many algorithms defined to aggregate and calculate trust scores, we decided to use Beta Reputation System as define by Josang et al. in [8]. It is a proven algorithm. The concept of discounting trust score of a service requester towards a service provider is an effective defensive mechanism to weed out collusive attacks. In this paper, the credibility and reputation rating of a service requester is used to determine the weight of it trust score towards a service provider. This helps maintain a reliable trust score regardless of any collusive attacks.

III. System Model and Assumptions

This section details the system model and assumptions including the attack model.

A. *System Model*

While there is a growing trend in distributed trust management system designs, the proposed trust management system is based on centralized system. While the proposed design can be easily adopted to a distributed system, albeit requiring more computational power for each device, this design can be included in any existing system as a complimentary feature. The reason for selecting a centralized reputation system is to foster sharing and give all involved consumers equal benefit by placing the load of processing on the central system as explained in [5]. So regardless of computational and storage limitation, the involved consumer devices will be able to participate with equal benefit as long as each of these devices are capable for processing and handling communications with central system.

The only property of service used to determine the quality of service is the wait time. However, this model is flexible enough to utilize a different property of service. In fact, this model is scalable to use more than one service property. The proposed design can be extended to analyze the quality of service by validating

The system consists for 4 main actors that interact with one another. The 4 actors of the system are:

- Central hub (CH): The central storage and processing unit of the system. CH allows service providers and consumers to register in the system so that they can be authenticated. CH manages reputation scores. CH provides APIs to allow service providers to advertise one or more services offered by them. CH also provides APIs to query for service providers based on one or more criteria.
- Service Provider (SP): Every service provider who wants to advertise their service must register their service in CH so that they can be categorized appropriately. SP advertise their services along with wait time for each service during operating hours.
- Service Requester (SR): Every service consumer who wants to use the services advertised in CH must register with CH to be able to search and view service details. A SR is able to query for services in CH based on one or more search criteria. A SR provides trust score as feedback to SP after receiving the service. A SR also provides feedback to service witnesses as binary value.
- Service Witness (WT): A service witness is a registered service consumer who is consuming service from a service provider at any given instance of time. A witness is also a user who is waiting in a queue of a service at any given instance of time. A witness is able to determine the service advertised by SP by direct interaction with the SP. A WT can validate if the wait time advertised in CH by the SP based on its personal experience. A WT can advertise its current experienced wait time in CH for the SP.

One of the primary concerns of reputation based trust management is preservation of user privacy. As users get involved in rating by providing feedback for a service, if their private information is exposed then it will not only jeopardize the user but also tear down the basis on which the trust management it built on . So the trust management system must ensure user privacy. There are several literatures covering user privacy but the one described in [6] is specifically targeted for trust management systems. As described in [6], *Zero-Knowledge Credibility Proof Protocol* can be implemented to prove users' feedback without including the sensitive user information.

A SP registers its services in CH so that the services can be advertised. The service providers are responsible to update any parameters associated with service(s) for which the CH will have necessary APIs available. When a SR executes a search query to find services that satisfy the search criteria, the search filters out any service that does not fully qualify at the moment of search execution. So the onus is on the SPs to update as frequently as possible to ensure that the most up to date data is available in the central hub. The wait time advertised by a service provider, $T_{advertised}$, is one of the properties that is expected to be updated by the SP on regular basis. This parameter tells a SR how long the SR has to wait before being able to use the service of the SP if the SR were to arrive at the SP at this moment. There is an error of margin for wait time, T_{margin}, which is predetermined to provide cushion to the service wait time based on the type of the service. As long as the advertised wait time of the service is within the error margin, the service is considered satisfactory by a SR. Since each SP will have its own limitations based on the available resources and current active consumers, each SP will have different rate of service time. The service time is not counted towards wait time as the SR is not considered waiting as soon as the service is available to the SR. However, while a SP is consuming the service, that resource is not available to any other consumer such any consumer in queue will experience wait time.

A SR searches for a set of SP based on certain preferences. The preferences are formalized into search criteria and are executed against CH database by the means of available APIs. The API provides the SRs to search based on different parameters, such as, in case of restaurant service, location, cuisine, ratings, and even wait time. This search criteria is executed in CH and based on the available information at that moment for all the services in the system, the query returns a set of service providers that satisfy the criteria. Each instance of the result of the search query has detailed information about a unique SP. Each result consists of trust score of the SP in the system, $T_{advertised}$, and a list of witnesses (WTs) that claim to be currently in direct contact with SP. Each WT record in the list of witnesses consists of the wait time experienced by the WT, the total feedbacks the WT has received, and the positive feedbacks received by the WT. The margin of error for wait time, T_{margin}, is a system-wide value and it may be returned in each response to a service or could be requested separately through available API. The set of SPs returned in the search result is expected to contain only the ones that presumably satisfy the requirements of the SR. Once the result is available to the SR, the SR will implement the best judgement based on the current $T_{advertised}$ of each SP and the likelihood of the advertised wait time being within T_{margin} based on the SP trust score and the validation wait time, $T^{i}_{validate}$, provided by each WT. Details on the trust calculation and trust management protocol is explained in section IV.

A witness, WT, is a valid registered service requester who has decided to go with a SP after judging the best available SP based on the available data. Once a SR decides to use the service of a SP, it may or may not have to wait in queue before receiving the service from the SP as this depends on the availability of necessary resources of the SP at the moment. A SR because a witness whether it is in queue or is consuming the service. When a SR arrives at the SP and agrees to wait in queue then it will be provided a wait time by the SP which is based on how soon the SP can provide the service to the SR. So the SR can then validate the actual wait time, T_{actual}, it experiences at the SP and the one the SP is advertising, $T_{advertised}$, in CH. Then the WT can provide feedback to CH regarding trustworthiness of the SP as well as to other WTs that currently advertising their personal experienced wait time. Since a WT will have a personal experience at the SP regarding how long it had to wait, it can advertise that in CH through API provided in CH. The personal experienced time by each WT will be associated with the SP in CH such that when a SR runs a search query that happens to include the SP, any witness information associated with the SP will also be included in the response to the SR. In order to prevent Sybil attack, CH will verify that a SR is advertising experienced wait time of only a single SP at a time. Eventually it's up to the SR to stop advertising the wait time it experienced at the SP once it completes receiving the service from SP.

B. *Attack Models and Assumptions*

In this paper we consider malicious actors as the only form of attack. Other forms of attack that are known in this kind of system setup, such as Distributed Denial of Service (DDoS), system intrusion and data leaks are beyond the scope of this project. There are several independent and related researches to address such active attacks such as [7]. In this paper, we focus specifically on malicious SPs that self-promote or try to collude with one or more SRs to either promote itself or to bad-mouth other non-malicious SPs. The possible attacks due to a malicious SP or SR or both are as follows:

- *Self-promoting attack*: A malicious SP will attempt to advertise false wait time to attract more consumers. This will not require any collusion with any other SPs or SRs. A SP will simply advertise wait time which is may not be true and a consumer will experience a much longer wait time than advertised, well outside the error margin.
- *Collusion attacks*: A collusive attack occurs when a SP colludes with one or more SRs to advertise false wait time or to bad-mouth a non-malicious SP.
    - *Ballot-stuffing attack*: In order to gain profit by attracting more consumers, a SP will advertise false wait time and will have one or more SRs backup the false wait time. A non-malicious SR will be more likely to trust the SP if malicious SRs act as witnesses.
    - *Bad-mouthing attack*: 1 or more malicious SRs advertise false wait time as witnesses of a non-malicious SP. This will in turn benefit a malicious SP as it will make the non-malicious SP less likely to be selected.
- *Opportunistic attack*: In such attack, a malicious SP or a malicious set of SRs colluding with a malicious SP will act malicious until they realize that their trust score is dropping. Then they will not perform any malicious behavior until the trust score is good enough.

IV. TRUST MANAGEMENT PROTOCOL

In this section, we will first explain how a SR determines its trust score towards a SP. Then we will explain how a SR provides feedback for WT after selecting a service. Finally we explain how a SR uses the trust score of a SP and reputation of WTs to determine the SP with most likely the least wait time.

A. *Trust Score from a Service Requester towards a Service Provider*

The centralized trust management protocol proposed in this paper is dependent on *beta reputation system* proposed by

Josang et al. as explained in [8]. The reputation engine is based on beta probability density function. Beta reputation system is used to calculate the trust score of a SR towards a SP based on the historical transactions between the two entities. However, the novelty of this proposed paper is in the utilization of discounted validation information from witnesses in conjunction with the trust score of a SP to determine the likelihood of the SP advertising actual trust score. In addition to the individual trust score from a SR towards a SP, CH will also calculate overall trust score of each SP based on weighted average of all the trust scores towards the SP from SRs. This trust score is advertised in the CH.

To circumvent "cold start", every new SP joining CH will receive an initial trust score of 0.5 which represents neutral trust score. SR queries CH with wait time criteria such that CH will then return list of SPs matching the input wait time criteria. The SR will then pick the SP with highest trust score with lowest wait time. When a SR receives service from the SP, then it will compare the wait time advertised by the SP with the actual wait time observed. If the actual wait time is within the error of margin of advertised wait time then a score of 1 is provided towards the SP for that transaction. If the actual wait time is not within the error of margin of the advertised wait time then a score of 0 is given for the transaction towards the SP. In [8], Josang et al. calls this score of 1 or 0 as feedback. This feedback record for the current transaction will be combined with historical transactions between the SR and SP and a new trust score is determined. The trust score of a SR, *sr*, towards a SP, *sp*, is calculated as

$$TS_{sr}^{sp} = \frac{\alpha}{\alpha+\beta} \tag{1}$$

where $\alpha = r + 1,\ \beta = s + 1,\ r$ is total number of positive feedbacks, and s is total number of negative feedbacks from *sr* towards *sp*. We consider only α in numerator to normalize the trust score in the range of [0, 1].

However, just determining the trust score towards a SP based on personal experience will underutilize the added benefit of centralized trust management protocol. If a SR can incorporate (to some extent) the overall trust score of a SP in the system to determine its trust score towards the SP, then it will provide each SR to adapt to the trend of the system towards the SP. So a possible way to enhance the trust score from a SR towards a SP can be calculated as weighted sum of its personal experience and the overall score.

$$TS_{sr}^{sp} = \delta\left(\frac{\alpha}{\alpha+\beta}\right) + (1-\delta)TS^{sp} \tag{2}$$

where δ is weight factor with value between [0,1] and TS^{sp} is overall trust score of SP *sp* in the system. This could possibly provide robustness to the trust score calculation. However, we decided to use equation 1 for our simulation and use this approach as a future enhancement.

B. *Feedback towards a Witness from a Service Requester*

As explained in section III, a SR will also provide its feedbacks towards all the WTs who provided their personal experienced wait time of the SP. Once the SR is able find out the actual wait time of the SP then besides updating the trust score towards the SP, it will also provide feedback to all the WTs.

When a SR determines the actual wait time of a SP, then for each WT the SR will determine if the wait time they are claiming is within the margin of error or not. If the wait time claimed by a WT is within the margin of error then SR provides feedback of 1 towards the WT. If the wait time claimed by a WT is outside the margin of error then SR provides feedback of 0 towards the WT.

CH will persist all the historical feedback towards a SR from all other SRs. There are 2 advantages of this feedback. The first is the calculation of reputation rating of the SR. The second is the credibility of the SR. By default, a new SR registered with system will have a reputation rating of 0.5 meaning a neutral user. However, its credibility will be 0 as there will be no feedbacks from other SRs in the system for a newly registered SR. As SR gets involved in more transactions and gets feedback from other SRs, its credibility will rise.

The reputation rating is also based on [8] and is calculated as follows

$$R_{sr} = \frac{a}{a+b} \tag{3}$$

where $a = r + 1,\ b = s + 1,\ r$ is total number of positive feedbacks, and s is total number of negative feedbacks. We consider only a in numerator to normalize the reputation rating in the range of [0, 1]. After each new feedback, CH will update the reputation rating of a SR in the system.

The credibility is directly proportional to the number of feedbacks it receives. However, it must be noted that credibility is only a relative term. For example, SR_1 with 100 feedbacks is 100 time more credible than SR_2 with 1 feedback but it is half as credible as SR_3 with 200 feedbacks. When the reputation rating of a SR needs to be determined, then it is weighted by credibility of the SR with respect to other SRs. The details is explained in subsection C of this section.

C. *Overall Trust Score of a Service Provider*

When a SR updates its score towards a SP in CH, then CH needs to update the overall score of the SP. When determining the overall trust score of each SP, CH must consider the reputation of each SR. The trust score towards a SP provided by SR with less reputation rating must have less weight than that with higher reputation rating. The final trust score, overall trust score, of a SP is average weighted sum of all the trust scores from SRs.

$$TS^{sp} = \frac{\sum_{i=1}^{n} \omega_i TS_{sr_i}^{sp}}{n} \tag{4}$$

where,
$TS_{sr_i}^{sp}$ is trust score from service requester sr_i towards *sp* as defined in equation 1 which utilizes beta reputation system
n is total number of SRs that provided trust score towards service provider *sp*
ω_i is weight of trust score of sr_i

In [9], Jiang et al. calculate trust score towards a service provider by aggregating recommendation trust and discounting reliability and familiarity. While we don't have any concept of recommendation trust in this model, we use similar concept of discounting reputation rating of SRs to compute trust score of SP. So the weight of trust score of sr_itowards service provider *sp* is calculated as follows:

$$\omega_i = \frac{\gamma_i R_{sr_i}}{\sum_{j=n}^{n} \gamma_j R_{sr_j}} \tag{5}$$

where γ is the credibility of service requester sr and is the total count of the feedback received from other SRs. So a trust score from SR with higher reputation rating will have higher weight towards the calculation of overall trust score of SP. The inclusion of credibility will add additional weight to SRs with more feedback. Since a SR will not have any feedback initially, it will be counted same as having 1 feedback.

D. *Implementation of Trust Score and Witness Reputation*

When a SR queries for service providers in CH that satisfy a predetermined condition, it may be provided with 2 or more SPs in which case the SR will have to use the best judgement to determine the SP with most likely least wait time and highest trust score as CH also returns the trust score of each SP. So for each SP returned by CH, the SR will determine SP how likely the SP actual wait time to deviate from advertised wait time is. Both the overall trust score of the SP, TS_{sp}, and the wait time claimed by WTs will be used by SR. The claims provided by WT is only complementary information as it may not be the case that there will be a WT all the time for all the SPs so main basis of the determination of trustworthy SP is the trust score of SP. WT claims is used only as secondary information if it is available but it will add a great value nonetheless. The deviation is calculated as the probability the advertised wait time will be further away from actual wait time. Since the deviation could result in calculated wait time being either more or less than advertised wait time, we only consider the worst case scenario such that we will only assume the deviation would mean the calculated wait time is more that the advertised wait time by the deviation amount.

The deviation only from trust score of the SP is as follows

$$D_{TS_m} = 1 - TS_{sp_m} \tag{6}$$

where *m* is the index of the SP returned in the search query result by CH. If there are no witnesses then the final deviation is only based on the trust score of the SP. The likelihood maximum value of actual wait time based on the advertised wait time and the trust score of the SP is determined as follows:

$$T_{TS}^{m} = \left(1 + \left|D_{TS_m}\right|\right) T_{advertised} \tag{7}$$

If a SP returned in the search result has any WTs, then the claims from the witnesses should also be considered determining the reliability of the advertised wait time. Since each WT has a reputation rating, the claim by each WT should be weighted based on its reputation rating. The weighted average wait time of a SP based on WT claims is calculated as follows:

$$T_{WT}^{m} = \frac{\sum_{i=1}^{n} (T_{WT_i}\left(\frac{\gamma_i R_{sr_i}}{\sum_{p=1}^{n} \gamma_p R_{sr_p}}\right))}{n} \tag{8}$$

where *n* is total number of WTs providing the claim information. Based on T_{TS}^{m} and T_{WT}^{m} the SR should be able to determine maximum wait time for the SP as weighted average mean. The weight can be based on average reputation rating of the WTs. If there are no WT claims then only the trust score of the SP will be used but if there are WT then in our experiments we considered up to 50% of the weight from WTs. The average reputation rating is

$$R_{WT_{avg}}^{m} = \frac{\sum_{i=1}^{n} \gamma_i R_{WT_i}}{n \sum_{j=1}^{n} \gamma_j} \tag{9}$$

where n is total number of witnesses. Then the weight for SP wait time calculation is determined as:

$$\rho = \begin{cases} 0 & , if\ n = 0 \\ 0.5\ x\ R_{WT_{avg}}^{m}, & if\ n > 0 \end{cases} \tag{10}$$

The wait time of a SP is calculated as

$$T_{calculated}^{m} = (1 - \rho) T_{TS}^{m} + \rho\, T_{WT}^{m} \tag{11}$$

Once a SR is able to determine the calculated wait time of all SP then it will select the SP with least wait time and proceed to request the service from the SP.

V. Implementation and Experimental Evaluation

For experimental evaluation, 10 Service Providers (SP) have registered with Central Hub (CH). Initially there are 30 Service Requestors (SR) that will try to find SP with least wait time. If 2 SPs have same wait time then one is picked randomly. Each SP can provide service up to 3 users simultaneously. Any additional user trying to receive service from the SP will have to wait in queue. A service time of 5 units is defined for all SP. A malicious SP advertises fake wait time by reducing the actual wait time by 25% once it reaches full capacity. For our trust protocol following evaluation metrics are used:

1. Convergence of trust score
2. Service Improvement
3. Percentage of SRs served by SPs

A. *Convergence of trust score*

To deal with cold start problem initially all service providers have a trust score of 0.5. Initially all Service Providers (SP) advertise wait time of ‘0’. Since we have 30 SR and 10 SPs so each SP will be serving up to 3 SR initially and in this case both malicious and non-malicious SPs will have the same trust score. As the more SRs start requesting service the score of the malicious will go down based on the actual wait time observed by SRs (that’s is more than the advertised wait time) and feedback registered by SR’s about SP in the CH – negative in case of malicious SP.
In figure 1, x-axis represents the number of transactions in the system and y-axis represent the trust score. Trust score value is between 0 and 1. Figure 1 shows the trust score convergence between a malicious and non-malicious service provider.

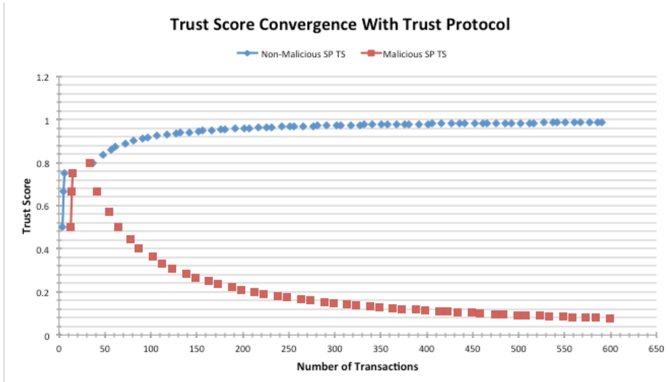

Figure 1 Trust Score Convergence with Trust Protocol

When the system starts, initially trust score for both malicious and non-malicious goes up but as number of transactions increase and more SR's get service from SPs (malicious/non-malicious) then they start logging feedback based on actual wait time observed. In case of malicious SPs, since its advertising fake wait time, the actual wait time observed by SR is more. When the advertised wait time is more than the actual wait time by wait time threshold, T_{margin}, then the SR logs negative feedback about SP and based on that trust score goes down. For non-malicious SP, the trust score goes up because SRs log positive feedback based on difference between observed wait time and advertised wait time less than T_{margin}. As it can be seen from figure 1 that as number of transactions increases the trust score for malicious SP converges to 0 and for non-malicious it converges to 1. The proposed trust protocol is able to improve the performance of system by correctly assigning the trust score of SP based on observed wait time from SRs.

B. *Service Improvement*

The proposed trust protocol is able to improve the system performance by recommending the non-malicious Service Providers to Service Requesters by utilizing the trust score metric.

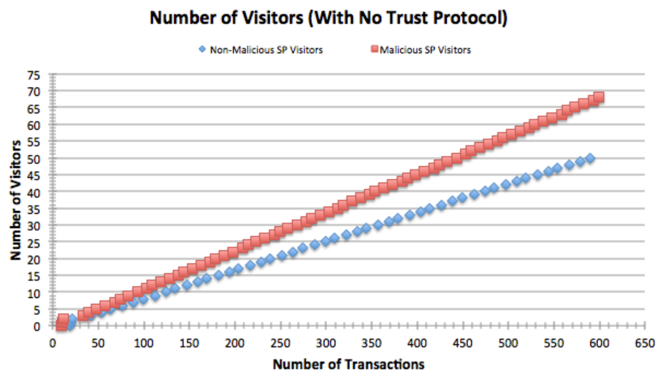

Figure 2 Numbers of Visitors (With No Trust Protocol)

As it can be seen from figure 2, in case of no trust protocol in place, malicious SPs are able to attract more SRs by advertising fake wait time. As number of transactions in system increases more SR's goes to malicious SPs because it's advertised wait time is lower.

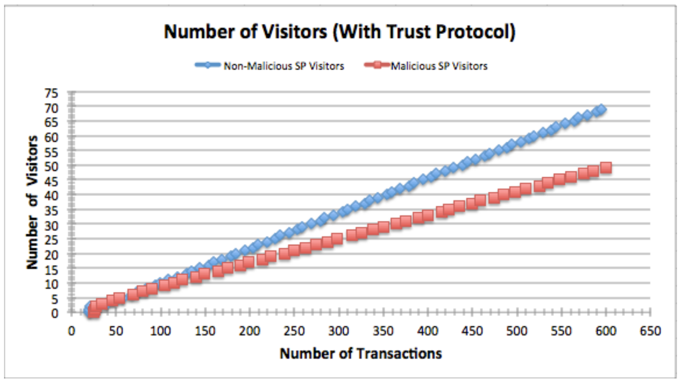

Figure 3 Numbers of Visitors (With Trust Protocol)

With proposed trust protocol (figure 3), system is able to identify malicious service providers and as number of transactions increases the trust score of malicious SP goes down and Central Hub is able to direct less number of SR to malicious SP. Non malicious SPs get more SR as the number of transactions increases.

C. *Percentage of SR's served by SPs*

Figure 4 and 5 show the percentage of SR's served by SPs with no trust and with trust protocol respectively. The wait time threshold for Service provider for simulation is 20% that means when the advertised wait time is over the actual wait time by 20% then SP is malicious.

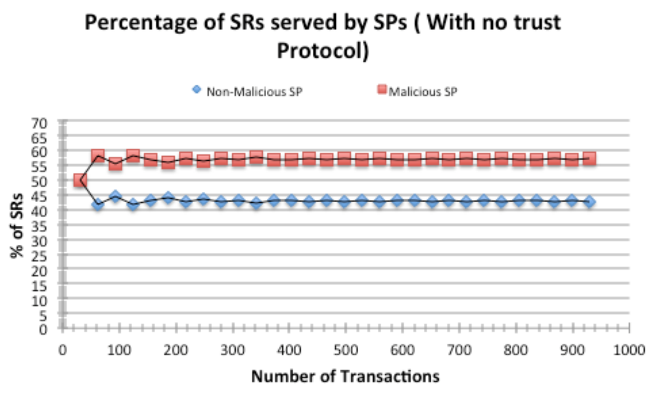

Figure 4 Percentage of SRs severed by SPs (With no trust protocol)

In figure 4 when there is no trust protocol in the system the percentage of SRs served by malicious SP increases as the number of transactions increases.

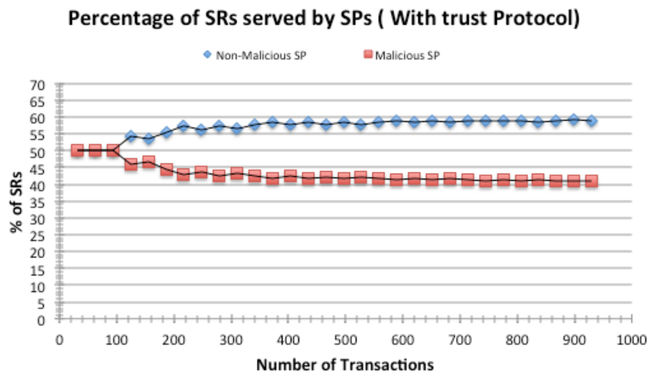

Figure 5 Percentage of SRs served by SPs (with trust protocol)

In figure 5, with trust protocol, the system is able to identify malicious SPs. Initially all SP have trust score of 0.5 (to deal with cold start problem) so CH will still recommend non malicious SP to SRs. As the number of transactions increase non-malicious SP gets to serve more SRs because its trust score is increasing as the number of transactions increase due to positive feedback from SRs. The proposed trust protocol is able to reduce the visitors served by malicious SP by 25%.

VI. Conclusion

With the popularity of service oriented architecture (SOA), the trust issue gains more attention from service requester's perspective. Centralized Trust based service selection can improve the system performance, overall customer satisfaction and can also find out colluding service providers in the system in addition to providing a central system for advertising and searching services.

The proposed trust protocol uses "Advertised Wait Time" metric exposed by service providers and actual wait time observed by service requesters to calculate the trust score of each service provider. It also takes into account the feedback given by witnesses in the trust evaluation. The most important contribution of the proposed design is the ability to collaborate the past actions of service providers with the real-time data from witnesses to predict the future behavior of service providers. Beta Reputation System was used to determine the trustworthiness of service providers and the concept of witness claims was implemented to validate information in real-time. It is concluded from the experiments that as the number of transactions increase in the system the proposed approach is able to recommend non-malicious service provider to service requesters and also the trust score converges to its proper value i.e. 1 for non-malicious service providers and 0 for malicious service providers. The robustness of this design has its roots in adoption of Beta Reputation System and the approach taken to discount the feedback from malicious actors during trust aggregation and calculation.

As future work, there are few enhancements we can think of to improve the trust calculation and improve the scalability for broader adoption. As mentioned in section IV A, the trust score calculation may be enhanced by incorporating overall trust score into individual trust score. This may provide a better defense against opportunistic attacks. Another enhancement is in terms of number of parameters used for validation. The proposed design considers only a single parameter for validation, the wait time. This design can be enhanced to 1 or more attributes pertaining to the quality of service to analyze the trustworthiness of the service provider. It may add complexity of different contexts for different sets of parameters but it will certainly open a new avenue for this design. Finally, the proposed design considers a centralized trust management system and tries to take the best advantage of it. As a future enhancement, this design can be modified to adopt a distributed design or a hybrid version such that the load of trust aggregation and calculation can be distributed to individual constituents of the system so that it may be suitable for broader adoption.

References

[1] I.R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-based IoT and Its Application to Service Composition," IEEE Transactions on Services Computing, vol. 9, no. 3, 2016, pp. 482-495.

[2] J. Timpner, D. Schurmann, and L. Wolf, "Trustworthy Parking Communities: Helping Your Neighbor to Find a Space," IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 1, 2016, pp. 120-132.

[3] M. Eirinaki, M.D. Louta, and I. Varlamis, "A Trust-Aware System for Personalized User Recommendations in Social Networks," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 4, 2014, pp. 409-421.

[4] P. Mendes, "Social-driven Internet of connected objects," in Proc. Interconn. Smart Objects with the Internet Workshop, Lisbon, Portugal, 2011.

[5] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," Commun. ACM, vol. 43, no. 12, pp. 45–48, Dec. 2000

[6] T.H. Noor, Q.Z. Sheng, L. Yao, S. Dustdar, and A.H.H. Ngu, "CloudArmor: Supporting Reputation-Based Trust Management for Cloud Services," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, 2016, pp. 367-380.

[7] R. Mitchell and I. R. Chen, "Effect of intrusion detection and response on reliability of cyber physical systems," IEEE Trans. Rel., vol. 62, no. 1, pp. 199–210, Mar. 2013.

[8] A. Josang, and R. Ismail, "The Beta Reputation System," 15th Bled Electronic Commerce Conference, Bled, Slovenia, June 2002, pp. 1-14.

[9] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An Efficient Distributed Trust Model for Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 5, 2015, pp. 1228-1237.