Abstract — Open Resource Control Architecture (ORCA) is a resource control and management testbed, which provides on-demand resource allocation service. This paper proposes to apply a dynamic, context-aware, computational trust model to ORCA, in which previous behaviors under different operating contexts are used to evaluate the trustworthiness of service manager and site authority actors. This trust model can be extended to support existing policy plug-ins to assist resource allocation. We also implement this trust model and demonstrate that our implemented trust model assigns relatively suited resources to user requests.

Keywords — Trust Model; ORCA; GENI; Slice; Broker; Service Manager; Site Authority; Trustworthiness

I. INTRODUCTION

ORCA (Open Resource Control Architecture) [1] is a distributed testbed developed by Duke University. The purpose of ORCA is to control and manage networked physical resources so that users can run various applications or services while sharing the same physical infrastructure without interference. In other words, it is a kind of “middleware” that aims to implement “virtual computing” [30]. ORCA is designed to allow users to negotiate to reserve and then access computing physical resources across different organizations. So far it has been employed by several researchers and organizations, like Duke University, IBM, Williams College, and RENCI.

To manage the whole platform, ORCA controls and monitors all interactions among nodes, researchers, services, applications and hardware resources, which have the following characteristics: nodes are distributed over a large geographic region; user behaviors and application status are dynamic and unpredictable; users and resources belong to different organizations, thus various trust and policy domains.

Given the distributed nature of ORCA, simple authentication, authorization, access control or even a combination of these mechanisms cannot manage the system well and maintain security and reliability. The effectiveness of these mechanisms are limited for various reasons, including: Researchers have to manage several certificates for resource access; When resource providers and users belong to different management domains, they tend to not trust each other; Even if researchers are trusted, it is not reasonable to assume that the applications they launch are also trustworthy; Given the variety and dynamic nature of guest applications, it is not easy to authorize application-specific and user-specific privileged system calls to control access rights; Fifth, compared to user demands, hardware resource provisions are not always sufficient. As one of ORCA’s main characteristics, resources are allocated, shared and adjusted automatically by detecting the status of services and resources, which is different from PlanetLab [21], because PlanetLab provides “best effort” services [1]. Therefore, a trust management scheme is proposed for negotiation, resource allocation and task distribution in ORCA and it is compatible with the implementation of the current system.

Our contributions are:
- We analyze trust relations among actors in ORCA system and quantify them.
- We apply a context-aware computational trust model (CAT) [32] and make it more suitable for ORCA system.
- We design a trust mechanism, implement the trust model and incorporate it into the policy module of ORCA.
- We demonstrate its effectiveness and argue that this trust model can better regulate the behavior of users and providers, moreover, assign relatively suitable resources to clients’ requests so that increase stability and reliability of the whole system.
- We also explain the scalability of our trust mechanism.

The remainder of the paper is organized as follows: Section 2 reviews related work in this area. Section 3 gives a brief summary of ORCA’s architecture. Section 4 describes the design of proposed trust model for ORCA. Section 5 details the design of the trust management mechanism which is employed to maintain and update this trust model. Section 6 presents the implementation of trust model and experimental results. Section 7 discusses some related problems. Section 8 presents the conclusion and future work.

II. RELATED WORK

Trust models are used to qualify or quantify trust relations among entities within a system. They have been applied in various contexts and served different purposes, for instance, distributed system [9], [13], peer-to-peer system [12], [14], pervasive computing [10], [16], and semantic network [11]. They can help clients to select the optimized...
service from several service providers [19]. Trust models can also be incorporated into a group of email servers for antispam [13], and it can assist agents to make decisions by evaluating the trustworthiness of entities. Additionally, tasks can be scheduled and then distributed according to Bayes trust model [17].

Trust models are categorized differently according to various standards. In this paper, trust models are summarized from the perspectives of trust relations, sources of trust, and trust measurement.

Trust relations can be represented as hierarchical relations [18], group [12], [13], social network [11], and Bayesian network [15]. Grandison and Sloman classified trust relations into four categories: a trustee has privileges to access a trustor’s resources ("Access to a Trustor’s Resources"), a trustor trusts services provided by a trustee ("Provision of Service by the Trustee"), a trustor trusts the certification of a trustee issued by a 3rd trusted party ("Certification of Trustees"), a trustor trusts a trustee to delegate a certain privileges ("Delegation"), a trustor trusts his infrastructure ("Infrastructure Trust") [24]. We use Grandison et al’s categories of trust relations to classify trust relations in ORCA.

Sources of trust provide evidences that can be used to qualify or quantify trust relations. Most trust models derive trust from two sources: direct interactions and recommendations, like the enhanced Trust Model proposed by Wu X. for P2P Networks [12]. These two sources are defined as “direct experience”, because they are personal direct interactions and direct interactions of others. Direct experiences are evaluated based on satisfaction of interactions, number of interactions and timestamp (trust decreases over time). If no previous interaction or recommendation exists, trust can also be quantified using a uniform distribution [17].

In the literature, trust has been measured by either qualitative properties [8] or quantitative metrics [9]. Representations of quantitative metrics are discrete or continuous, while continuous values of trust are more fine-grained. Additionally, trust can be measured and then represented as a fuzzy set [20]. Griffiths et al [26] use fuzzy logic to label trust metrics with “qualitative terms and linguistic labels”, and then quantify them. According to Ries et al [23], discrete representation is easy to understand while continuous values are well supported by mathematical theories. Fuzzy logic quantifies trust, at the same time, ensures its uncertainty.

Although a trust model for ORCA has not been proposed before, Peterson et al analyze trust in PlanetLab [8], which is another platform that controls and manages physical resources. However, Peterson et al do not quantify trust, but provide a trust management mechanism that implements trust by issuing credentials and monitoring network traffic in PlanetLab.

CAT [32] is a context-aware dynamic computational trust model, in which every entity is assigned a trust value that is derived from its previous behavior under different contexts and is updated automatically over time. We apply this trust model to ORCA and explain more detain in section 4.

III. ARCHITECTURE OF ORCA

ORCA is one of the five clusters employed in GENI [2] spiral [3]. It is open-source software that incorporates several existing projects: Shirako [4], Cluster-On-Demand (COD) [5], Automat [6] and SHARP [7]. SHARP’s main purpose is to cryptographically protect resource claims and redeem mechanism, while allowing different sites to “trade” their resources. Shirako and COD are employed to manage and allocate different types of resources while utilizing SHARP’s cryptographic operations [4]. Automat is a web portal to control and monitor controllers.

A. Entities in ORCA

ORCA has several basic elements including nodes, slices, tickets, leases and guest applications. A node is a server running on a physical machine to support applications or services. A site is composed of a bunch of nodes and always within a certain geographical or administrative boundary. A slice is a set of physical resources that have been allocated to a certain application or service. After creating a slice, specific nodes will be selected from a pool and then virtual machines will be launched on those chosen nodes to run the application or service. A ticket is “a soft claim” [7], compared with ticket, it is the actual resources that are allocated to the service manager for a certain application or service. Tickets differ from corresponding leases because requests cannot be satisfied all the time. A guest application is an application or service that researchers plan to execute within the ORCA environment.

In ORCA, three main actors interact with each other: service manager, site authority, and broker. A service manager actor starts and ends an application, requests a slice or a set of resources for that application, and monitors the status of the application and its respective resources in order to dynamically negotiate resources. A site authority actor manages hardware resources within a provider site or domain and maintains resource and slice status. It also binds applications or services with actual physical resources. Additionally, it isolates applications or services from each other on shared resources. A site authority actor has the highest privilege over nodes that it manages. A broker actor maintains a list of physical resources and maps them to a request of a guest application. Its main functionality is to coordinate resources from various provider sites or domains. It accepts resource information from site authority actors and assigns proper resources to requests from service manager actors. A site authority actor may delegate its resources to a broker actor within the same organization or trust domain, or to broker actors outside the boundary that aggregate resources from several organizations [30].

B. Current Trust Mechanism

In ORCA, before service manager, site authority and broker interact with each other, they authenticate each other using assigned key pairs. Therefore, these actors establish trust relations by credential authentication [29]. For instance, a broker actor authorizes a service manager actor to request resources from it by registering its public key. Broker
authorizes service manager to access site authority’s resources by issuing a signed ticket.

IV. TRUST MODEL

We identify six trust relations that exist in ORCA. These relations specify the actions that must be trustworthy to ensure the proper allocation and utilization of resources by ORCA actors. In our trust mechanism, we use identity authentication and CAT to establish these trust relations.

A. Applying CAT to ORCA

Rather than designing a brand new trust model for ORCA, we leverage CAT [32] and make some necessary changes to it so that it better fits into the ORCA testbed. CAT is well suited for ORCA for the following reasons: First, both CAT and ORCA are interaction-based and open, where entities can join, leave at any time and each entity can interact with anyone else. Second, both CAT and ORCA are dynamic: resource allocations are not static and depend on the status of applications and nodes, and service manager and site authority actors may create on-demand relationships with multiple brokers. Third, both CAT and ORCA are context-aware: In CAT, the trustworthiness of an entity is related to the type of service it provides, while in ORCA, context refers to the category of a guest application or service. According to the work conducted by J. Albrecht, et al [27], applications running on ORCA are classified as either:

- “short-lived distributed applications” – require resources that will enable the timely execution of tasks and delivery of data. has high performance requirements;
- “continuously running internet service” – requires resources that will enable reliability and dynamic recovery from failure;
- “grid-style parallel applications” – require resource that will enable low-latency communication.

They argue that ORCA’s resource allocation approach is similar to that of PlanetLab. Therefore, we use Albrecht’s application classification for PlanetLab to define context in ORCA.

We also make the following changes to make CAT fit ORCA better:

- We extend CAT to support three types of entities: site authority, service manager and broker. The current version of CAT assumes that all entities function in the same manner. For example, all entities will evaluate the trust of all other entities. But, given our trust model for ORCA, entities within ORCA do not function in this manner. Broker actors evaluate service manager and site authority actors directly; service manager and site authority actors evaluate the trustworthiness of broker actors directly; and service manager and site authority actors evaluate trustworthiness of each other indirectly via the corresponding broker actor. The details of our trust model are presented in section 4.2.

- CAT is theoretically strong and reasonable; we modified the formula for interaction satisfaction evaluation based on the features of trust rules designed for ORCA.
- Mostly, it is the site authority actors that delegate broker actors to allocate their resources. However, broker actors can also delegate other broker actors to fulfill this goal. As the centralized evaluation engine calculates the trustworthiness of all service manager and site authority actors, we only consider direct interaction and direct recommendation in this trust model, not including indirect recommendation as CAT.
- As Yumerefendi et al [6] proposed a digitally signed action approach to maintain accountability of actors, we assume that any actions taken by service manager, site authority and broker are able to be audited based on action records. So we don’t consider accuracy of direct recommendation.
- We design trust rules specifically for ORCA system, by which trustworthiness of service manager and site authority actors are evaluated. Furthermore, those trust rules are quantified by flexible approaches specifically based on their characteristics. Besides, they are also scalable for future extension.

B. Trust Relations

We use Grandison’s [24] trust categories to classify these relations. We examined ORCA and identified the following trust relations for site authorities, service managers and brokers:

- “Delegation”[24] - Guest applications delegate to service manager actors to request physical resources and node agents delegate to site authority actors to manage resources.
- “Access to a Trustor’s Resources”[24] (R1) - A site authority trusts a service manager to utilize their physical resources.
- “Infrastructure Trust”[24] (R2) - A service manager trusts a site authority to provide the requested physical resources, to isolate applications and provide confidentiality and integrity of their data, to ensure applications execute successfully, and to protect and maintain physical resources within nodes.
- “Provision of Service by the Trustee” [24] (R3) – A site authority delegates its physical resources to a broker and trusts the broker to allocate resources. The following trust relations are all defined as “Provision of Service by the Trustee” [24] as well:
  - (R4) - A broker trusts a site authority to collect status of resources and misbehaviors of applications.
  - (R5) – A service manager trusts a broker actor to map resources to applications and services based on priorities and demand, and to guarantee service quality.
C. Source of Trust

The trustworthiness of service manager and site authority actors accumulates over rounds of direct interaction evaluations by broker actors. A centralized evaluation engine collects those evaluations from all brokers and then calculates the trustworthiness of service managers and site authorities. Each direct interaction with a service manager is evaluated based on reports about the behavior of a guest application from the respective site authority, and resource status and slice status reports. Misbehavior reports will decrease the trustworthiness of the service manager directly. Examples of misbehavior reports include: the application runs outside of its memory bounds; the service manager requests more resources than is really needed; the clock of a service manager is manipulated [22].

Similarly, each direct interaction with a site authority is also evaluated based on reports from the corresponding service manager. The trustworthiness of a site authority will decrease if the service manager reports the following events: failure of resource availability in site authority, deficient provision compared with resource ticket, failure of connecting to the designated site authority, site authority employs low fidelity silvering mechanism, guest application is interfered with by applications running in other slices, guest application suffers uncovered interruption or reports fake reports about service manager. For both service manager and site authority, their trustworthiness measure determines how resource allocation requests are prioritized. In other words, after using one of existing resource allocation policies, a request will be considered if it has the highest trustworthiness in the context of application category. Similarly, resources will be first allocated first from the site authority with the highest trustworthiness within the context of application category.

D. Evaluation of Trustworthiness

We use CAT to formulate the equations that are presented in this section. We describe how we extended CAT in section 4.

1) Basic Representations and Trust Rules: Context is represented as C. The trust value of entity E, evaluated by entity E, in the context of C, at time t is represented as TV (E, E, C, t). The trust value of an entity is updated after each incoming interaction or recommendation. In this trust model, trust value is derived from “direct experiences” [31], including both of direct interaction (TV) and recommendations from other brokers (TV).

2) Satisfaction of Interactions: The satisfaction of every interaction is evaluated according to predefined trust rules. The trust rules for trust relation R1 are: the guest application behaves within boundary (tr11), the service manager doesn’t over-subscribe (tr12), the service manager doesn’t control its clock to cheat (tr13), and the service manager reports authentic information about site authority (tr14). For trust relation R2, the trust rules are: the site authority provides enough resources as it claims (tr21), the site authority isolates different slices well (tr22), the service manager successfully connects to the appointed site authority actor (tr23), the site authority does not cause an unrecovered service interruption (tr24), and the site authority reports authentic information of service manager (tr25). In the future, new trust rules can also be incorporated extend the current framework.

After every interaction, a satisfaction value will be assigned to each trust rule. Those satisfaction values are used to calculate the trust value for the specific actor. We quantify trust values for ORCA actors in a non-uniform manner. For example, we use Boolean values to quantify trust values for some trust rules, while for other trust rules, require more fine-grained representation. Therefore, we use two or more discrete values to calculate trust values for those trust rules. Take for an example, tr12, the service manager actor doesn’t control its clock to “cheat”. Two discrete values are sufficient to evaluate it. When considering tr12, the service manager actor doesn’t over-subscribe, we use more than two discrete values are required to quantify this rule. Currently we propose to quantify this trust rule according to amount of over-subscription.

Boolean values are usually 0 and 1: 0 represents high disbelief, and 1 represents high belief. v(tr) is assigned trust value for a trust rule tr. Other discrete values, such as those defined in CAT, are incorporated when needed. The total satisfaction of an interaction S (E1, E2, Ci, t) is calculated by equation (1).

\[ S = \sum_{i=1}^{N_{tr}} w_{tr} \times v(tr) \]

where \( N_{tr} \) is the number of all trust rules. \( w_{tr} \) is the weight of trust rule tr and the sum of all weights is 1. For our implementation, all \( w_{tr} \) are set as the same default value, which is \( 1/N_{tr} \). This formula is a revised version of quantified satisfaction towards an interaction in CAT. In CAT, each interaction is also evaluated based on trust rules. Each rule is
assigned one of seven discrete values: 0, 0.2, 0.4, 0.5, 0.6, 0.8, 1. The total satisfaction in weighted sum of trust values for each rules.

\[ \sum_{i=1}^{N_{tr}} \omega_x p \circ (\tau_{p_x}) \left( \sum_{i=1}^{N_{tr}} \omega_{x_i} = 1 \right) \]  \hspace{1cm} (1)

3) **Direct Interaction Trust and Recommendation Trust:**
Direct interaction trust from \( E_1 \) to \( E_2 \) in the context of \( C_i \) is calculated by equation (2), which is the weighted sum of the old trust value and the satisfaction value of the latest interaction. \( \delta \) is the weight of the old trust value, with ranges [0, 1]. Its default value is 0.5. \( R(C_i, C_j) \) is the similarity of context \( C_i \) and \( C_j \). Equation (3) specifies that when no direct interaction occurs under context \( C_i \), the trustworthiness is derived from context \( C_j \) according to the similarity of \( C_i \) and \( C_j \). Equation (4) calculates the direct trust value from \( E_3 \) to \( E_2 \) and the confidence that \( E_1 \) has about \( E_3 \)’s opinion. As in ORCA, all brokers are currently assumed to be trusted after identity authentication, so it is set to be 1. The recommendation trust value of entity \( E_2 \) from \( E_3 \) is the sum of all direct recommendations from other entities, see equation (5).

\[ T_{xi} (E_1, E_2, X_i, \tau) = \delta TV_i (E_1, E_2, X_i, \tau) + (1-\delta) S (E_1, E_2, X_i, \tau) \]  \hspace{1cm} (2)

\[ T_{xi} (E_1, E_2, X_i, \tau) = T_{xi} (E_1, E_2, X_i, \tau) \ast p (X_i, X_p) \]  \hspace{1cm} (3)

\[ T_{sp} (E_3, E_1, E_2, X_i, \tau) = \eta T_{xi} (E_3, E_2, X_i, \tau) \]  \hspace{1cm} (4)

\[ T_{sp} (E_1, E_2, X_i, \tau) = 1/N \sum_{i=1}^{N} T_{sp} (E_1, E_1, E_2, X_i, \tau) \]  \hspace{1cm} (5)

V. **TRUST MECHANISM**

We use existing policies to deploy our proposed trust mechanism. The architecture of this mechanism is illustrated in Figure 2 and Figure 3.

VI. **EXPERIMENTAL EVALUATION**

A. **Experiment Setup and Results**

In our experiment, one broker holds an inventory, which includes an equal amount of CPU, memory, bandwidth and storage from four site authorities (assigned IDs are SA1, SA2, SA3 and SA4). Three service managers (assigned IDs are SM1, SM2 and SM3) request resources for application category 1, application category 2 and application category 3 respectively.

We generate behavior profiles for service managers and site authorities prior to starting the experiment. We intentionally make some resources have more failure events
for type-1 applications, some for type-2 and others for type-3. Similarly, we use a uniform distribution to generate misbehaviors of service managers for three application categories. After incorporating the trust model, the broker actor first derives behavior records of site authority and service manager actors; these values are shown in Table 1 and Table 2.

**TABLE I.** A log file records behavior of service manager actors

<table>
<thead>
<tr>
<th>Time</th>
<th>ID</th>
<th>Trust Rule</th>
<th>Rating</th>
<th>App Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1252254</td>
<td>sm1</td>
<td>SMNoCheat</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1252254</td>
<td>sm1</td>
<td>SMOOutOfBoundar y</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1252254</td>
<td>sm1</td>
<td>SMNoOverSub</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

...  

**TABLE II.** A log file records behavior of site authority actors

<table>
<thead>
<tr>
<th>Time</th>
<th>ID</th>
<th>Trust Rule</th>
<th>Rating</th>
<th>App Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1252254</td>
<td>SA1</td>
<td>SAConn-Success</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1252254</td>
<td>SA1</td>
<td>SANoInfere nce</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1252254</td>
<td>SA1</td>
<td>SANoInter-ruption</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1252254</td>
<td>SA1</td>
<td>SASufficien tProvision</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

...  

After considering trustworthiness of service manager and site authority actors, under the context of application category, we calculate the trust values of service managers. These results are shown in Table III. The requests will be processed according to the descending order of trust values. Similarly, as shown in Table IV, resources in the inventory are also sorted under three contexts.

**TABLE III.** Context-aware trust value of service manager actors

<table>
<thead>
<tr>
<th>ID</th>
<th>Trust Value</th>
<th>App category</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SM2</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>SM3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE IV.** Sorted resource table under the context of application category 1, 2 and 3

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 2</th>
<th>Category 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Trust Value</td>
<td>ID</td>
</tr>
<tr>
<td>SA2</td>
<td>1.0</td>
<td>SA1</td>
</tr>
<tr>
<td>SA3</td>
<td>0.5</td>
<td>SA4</td>
</tr>
<tr>
<td>SA4</td>
<td>0.5</td>
<td>SA3</td>
</tr>
<tr>
<td>SA1</td>
<td>0.25</td>
<td>SA2</td>
</tr>
</tbody>
</table>

**Figure 4.** Trust value evolution over several rounds. X-axis represents rounds of resource allocation. Y-axis denotes trust value of service manager SM1 and SM2.

**VII. DISCUSSIONS**

The proposed trust mechanism is able to regulate service manager and site authority actors, by translating trustworthiness into request and resource allocation priority. It provides incentives for users to behave in a good way.

As Irwin made the assumption that communications or requests may be delayed, but messages will finally be delivered unless either communicating node side of is down [22], so we suppose that feedback reports will also be transmitted successfully as long as both communicating parties are active. In addition, broker actors store and maintain the trustworthiness of service manager and site authority actors. Each actor employs LDAP and MySQL [22] to record lease states over time. Currently, the records that are maintained by information collectors and the trust value table that is maintained by evaluation engine are in the form of log file. In the future, databases can also be used for information logging.

In our experimental setup, all contexts have the same importance, and all trust rules are weighted equally. Additionally, we weight direct interaction trust and recommendation trust equally when calculating trust values. If it is possible to test a huge amount of concurrent requests, we can adjust those metrics accordingly. The incorporation
of recommendation makes it easier to move to a distributed trust mechanism in the future.

The central evaluation engine considers recommendations from all entities. These recommendations are digitally signed and recorded to allow auditing by a third party. Therefore reporting fake information about other actors will decrease an entity’s trustworthiness.

We assume that the centralized evaluation engine is trusted, which may introduce a single point of failure and performance bottleneck.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a trust model for ORCA that can be used to allocate resources for entities that behave in a trustworthy manner, thereby limiting the effects of misbehaving entities. We argue that when we allocate resources based on behavior, we increase reliability and robustness of this platform. The whole design of the trust model and implementation of a prototype is presented. As future work, we plan to evaluate various aspects of our model and propose extensions that will help to improve the overall performance.

ACKNOWLEDGMENT

This research was funded in part by the Center for Applied Cybersecurity Research (CACR), Indiana University. We would like to thank Dr. Jeff Chase and his student, Varun Marupadi for their invaluable assistance with ORCA.

REFERENCES
