CHAPTER 4

Grid Reputation-Based Architecture

4.1 Introduction

This chapter introduces the Grid Reputation-Based Architecture. The chapter provides an overview of the service architecture, its scope within grid environments and the main characteristics which differ GridPeerTrust from existing Grid reputation based Trust Management System and elaborately discusses the workflow of Trust Management System provided by the GridPeerTrust architecture.

4.2 Grid Reputation-Based Architecture

4.2.1 Overview

The GridPeerTrust Architecture provides a solid reference implementation for the reputation-based trust model. GridPeerTrust is required to support Virtual Organizations (VO) during the formation and operation phases, that is - during the negotiation period with the VO candidates regarding their participation in the VO as well as during the main life-cycle execution period. During the VO formation phase GridPeerTrust is used to assist in screening the participating parties to ensure that they would be able to fulfill their service level agreements.

4.2.2 Layered Architecture:

Grid System consists of the following major groups of components:

1. Hardware resources like processors, storage, networks, and possibly sensors including the corresponding system software;
2. Domain-independent software to manage, for instance, virtual organizations, data, and access to the various resources;

3. Application software dedicated to the needs of the various virtual organizations within a virtual research environment.

4. Trust based Layer is introduced to deal with the reputation of grid entities.

This classification of grid components strongly suggests representing the organizational structure of the grid by using layers.

The basis of the Secured Grid is layered architecture as shown in Fig. 4.1

**Fabric Layer**

This layer represents all the physical infrastructure of the Grid, including computers and the communication networks. It is made up of the actual resources that are part of the Grid, such as computers, storage systems, electronic data catalogues and even sensors such as telescopes or other instruments, which can be connected directly to the network.

**Middleware Layer**

This layer refers to the grid middleware that incorporates necessary components for authentication, monitoring and discovery of grid resources, execution of job in grid resources, file transfer between grid resources.

**Trust Based Layer**

This layer evaluates the trust value of all the grid resource providers. It computes overall trust value using a proposed trust model and stores them in the database. This trust value is used to identify the most trusted resources for job execution. Suitable grid resources
that match the job requirements are discovered and they are ranked on the basis of their trust value. The resource that has most trusted value is selected for grid services.

**Application layer**

The highest layer of the structure is the application layer, which includes all different user applications (science, engineering, and business, financial), portals and development toolkits supporting the applications. This is the layer that users of the Grid will see and interact with using a Grid Portal;
4.2.3 Functional Requirements

With the context to Grid computing environments, GridPeerTrust is argued to be suited for providing advanced trust management service as part of Grid layered architecture.

It is expected to address three requirements:

(i) Satisfaction factor of a single or set of consumer’s requirements
(ii) Trust evaluation of a single or a set of resources providers
(iii) Inference of the trustworthiness of a single resource provider based on the global context

A typical Grid reputation-based scenario involves a grid resource broker acquiring information regarding the availability of computing resources utilising an information service such as the MDS [89]. Resource providers may advertise their capabilities and publish them to the information service so that they can be queried by the resource broker. Each Service Level Agreement contract contains one or more resource-guaranteed QoS parameters. It may specify the levels of availability, serviceability, performance, operation, or other attributes of the service like billing and even penalties in the case of violation of the Service Level Agreement.

Fig. 4.2 illustrates Grid layered Architecture utilising GridPeerTrust.

Before Transaction:


2. The Resource Broker contacts a Grid Information Service (GIS) in order to retrieve a list of available resources and their published resource-guaranteed QoS capabilities and calculate Satisfaction Score.
3. The Resource Broker contacts the Resource Database to retrieve the desired consumer’s requirement for the given job including client-expected QoS capabilities.

4. The Resource Broker decides on the final list of resources, constructs a reputation-policy query and submits the query to GridPeerTrust which evaluates the query and then returns a reputation policy report.

5. The Resource Broker evaluates the report and decides on the final list of resources to utilize for the job computation.

6. During the execution of the job, a Grid Monitoring Service (GMS) monitors the performance of the selected resources.

After Transaction:

1. Grid Consumer provides the feedback ratings and grid portal send them to the Distributed Database.

2. The Service Provider list and recommender list for grid resource provider is upgraded on the basis of feedback.

3. Send the Complete Report to the Consumer and help the consumer to choose the best suitable Resource Provider.

4. After transaction completion, the Grid Consumer compares the actual level of service provided with the one guaranteed by the SLA contract. Any noted differences are recorded and deducted from the total score. The Grid Consumer uses this report for selecting the resource to indicate which factors to provide ratings for.
Fig 4.2 GridPeerTrust Model
4.3 Proposed Trust Management System

The proposed Trust Management System does not use a centralized database for storing the trust information. Rather, the trust information is stored in distributed manner over the entire grid. Trust data that are needed to compute the trust measure for grid entity are stored across the grid in a distributed manner. Each grid entity has a trust manager that is responsible for feedback submission and trust evaluation, a small database that stores a portion of the global trust data, and a resource locator for placement and location of trust data over the network.

The Trust Manager performs two main functions:-

1. It submits feedback to the network through the resource locator, which will route the data to appropriate Resource Provider.

2. It is responsible for evaluating the trustworthiness of a particular Resource Provider.

Feedback Submission and Trust Evaluation by the Trust Manager is performed in two steps:

1. The Grid Entity submits the feedback to the grid portal, which is considered as the entry point to the Grid, and collects trust data about the target Grid Entity from the network through the Resource Locator.

2. The Resource Locator contains the information about trust data stored at different Grid Entities and uses this information to route the feedback to the appropriate Grid Entities for storage and then computes the trust value.

3. The trust evaluation is executed in a dynamic and decentralized fashion at each Resource Provider. Instead of having a central server that computes each
Resource Provider’s trust value, Trust Manager acts Grid Services invoked at institution level obtains resource provider’s trust data and computes the trust value of this Resource Provider.

Resource Locator gathers information using information providers like Ganglia using XML mapping of the GLUE schema and reports it to WS-GRAM service which publishes it as resource properties.

This information includes:-

- Basic host data (name, id)
- Memory size
- OS name and version
- File System data
- Processor Load Data
- Other basic cluster data

Trust Manager is also responsible for evaluating the trust worthiness of a particular grid entity. This is done by collecting trust data about target grid entity from the grid through the resource locator and then computes the trust values using the trust function defined for the system. The trusting entity is required to contact its trust manager in order to evaluate the trustworthiness of one or more trusted entities.

This includes obtaining the reputation-based trust evaluation requirements and contacting one or more recommendation entities in order to collect their feedbacks regarding their historical experiences with the evaluated trusted entities.
Fig 4.3 illustrates an example of information gathered by Resource Locator.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<grid>
    <host id="1.1.1">
        <Processor>4</Processor>
        <OperatingSystem>Linux</OperatingSystem>
        <MainMemory>4</MainMemory>
        <FileSystem>4</FileSystem>
    </host>
    <host id="1.1.2">
        <Processor>3</Processor>
        <OperatingSystem>WindowsXP</OperatingSystem>
        <MainMemory>4</MainMemory>
        <FileSystem>4</FileSystem>
    </host>
    <host id="1.1.3">
        <Processor>4</Processor>
        <OperatingSystem>Windows</OperatingSystem>
        <MainMemory>3</MainMemory>
        <FileSystem>3</FileSystem>
    </host>
    <host id="1.1.4">
        <Processor>4</Processor>
        <OperatingSystem>Linux</OperatingSystem>
        <MainMemory>2</MainMemory>
        <FileSystem>3</FileSystem>
    </host>
    <host id="1.1.5">
        <Processor>5</Processor>
        <OperatingSystem>Mac OS</OperatingSystem>
        <MainMemory>5</MainMemory>
    </host>
</grid>
```

Fig 4.3 Information gathered by Resource Locator
There are minimal requirements set of requirement which any reputation-based trust model must fulfil:

a) Simplicity

b) Robust

c) Scalability

d) Dynamic

Fig. 4.4 gives a sketch of the system architecture of GridPeerTrust. There is no central database. Trust data that are needed to compute the trust measure for entities are stored across the network in a distributed manner. If an individual database fails due to negligence or intentional corruption by some entities, it will be corrected by valid information in the rest of the community.

In addition, the trust computation is executed in a dynamic and decentralized fashion at each entity. Instead of having a central server that computes each entity’s trust value, an entity obtains another entity’s trust data from the rest of the entities and computes the trust value of this entity on the fly. This allows entities to get an up-to-date evaluation of the entity by other entities. The callout of the entity shows that each entity maintains a small database that stores a portion of the global trust data such as complaints filed.

Each entity has a trust manager for submitting feedbacks, collecting feedbacks and computing trust measures. Each entity also has a data locator, a data placement and data location component for placement of trust data over multiple entities and managing the access to the trust data.

The resource locator provides a grid data location scheme for accessing and updating data in the network.
Different applications may use different data placement and location schemes, which determine how and where the data can be inserted, updated, and accessed.

Fig. 4.4 Trust Manager in Secured Grid Resource Management
4.4 Calculating the Trust of Entities:

There are few assumptions with respect to Grid Environment:

- All resource providers must be part of any institution of Virtual object of global grid.

- The basic information of resource provider is stored in trust data of VO.

- Resource Consumer may or may not be part of grid

- Feedback is sent in encrypted form.

If there are many resources fulfilling both features then resources with maximum given requirement can be fetched. The trust can be calculated for trust fulfilling Grid Resource Providers. The trust evaluation component is responsible for computing the trust measure based on the reputation data that are collected about a grid resource.

Data Structures used in the algorithm:

1. **Local Index Table**

   It keeps tracks of the Grid entities, which are the members of the Grid and the transaction is performed among the Grid entities which are the members of the Grid environment. The Grid entities are required to register to be part of the Grid.

2. **Global Index Table**

   It keeps tracks of the entities, which are not members of the Grid and the transaction is performed among the Grid entities which are the members of the Grid environment and non member of the Grid.
3. Local Feedback Table

It stores the feedback of the grid entities, which are the members of the grid and the transaction is performed among the grid entities which are the members of the grid environment. This feedback can be stored in the form of cryptography as a part of the security.

4. Global Feedback Table

It stores the feedback of the grid entities, which are the members of the grid and the transaction is performed among the grid entities which are the members of the grid environment. This feedback can be stored in the form of cryptography as a part of the security.

The proposed algorithm is designed on the basis of Peer Trust algorithm [7]. The PeerTrust model has certain limitations with context to Grid Computing

First, a minimum number of interactions are required, which is a disadvantage for newcomers and reentry nodes, which are common in P2P systems.

Second, the balance factor used is a peer’s trust value; the system assumes that a peer with a higher trust value always gives more reliable feedback than a peer with a lower trust value, which might not be true.

Third and important limitation is that peer’s behavior changes over time. More recent feedback is closer to a peer’s current behavior than older feedback. In this model, all previous feedback has the same weight in evaluating a peer’s trust value.

The major drawbacks of PeerTrust Model are handled in GridPeerTrust Algorithm by changing definition of Satisfaction Criteria and adding a Decay function, in given algorithm.
Algorithm:

Trust Data contains the following information about any resource provider.

Features; Number of Transaction; Current Feedback \((u,i)\); Previous Feedback

Inputs: Client’s requirement, Local Index Table, Global Index Table, Local Feedback Table, Global FeedBack Table

Steps:
1. The Trust Manager receives Grid Resource Provider List; \(J \[n\]\) created in the Grid, where \(n\) is the number of resource Providers.
2. Sort the resources as per client’s (Resource Consumer) requirement.
3. Calculate Satisfaction value for these sorted resources.
4. While trust calculated for selected resource providers from Resource Database.
5. Get the Number of transaction from Resource Database
6. Get the feedback (local & global) for the resource provider using 
   \[
   F_{(u,v)} = \frac{\text{Current} - F_{(u,v,i)} + \sum_{i=n-1}^{i} \text{Previous} - F_{(u,v)}}{N \_Transaction(u,v)}
   \]
7. Calculate the Decay function
8. Calculate Trust
   \[
   \text{Trust} \_\text{final}(u) = sat_r + \alpha \times F_{(u)} - \gamma_{(u)} + \beta \times TCF_{(u)}
   \]
9. End While

Output: Sorted Trust based Resource Provider List, Service Provider List and Recommender’s List

Fig 4.5 Grid Peer Trust Algorithm
To guarantee secrecy and integrity of the trust data obtained from other entities and the accountability of the entities providing such trust data. The unauthorized manipulation of data can happen either in storage or during transmission. PKI-based scheme and data replication techniques are used to increase the security and reliability of the trust data management. For PKI-based scheme each entity requires to have a public and private key pair. Therefore, an entity ID will be either a digest of its public key, obtained using a secure hash function, or the public key itself. For feedback submission, an entity $v$ submits the feedback about entity $u$, signed with its private key $SK(v)$, along with its public key $PK(v)$. The fact that each piece of feedback is signed with the feedback source’s private key guarantees the integrity of the feedback and the authenticity of the feedback origin. Even though entities may tamper with the data that are stored in its local database and provide false or random data when processing a search request later, other peers are able to detect whether the data is corrupted by verifying the signature and discard the data if it is corrupted. When an entity $w$ wishes to evaluate the trustworthiness of entity $u$, it issues a search request for entity $u$’s trust data, including in the request its public key $PK(w)$. The entity responsible for the data, encrypts its response with $w$’s public key $PK(w)$, signs it with its own private key, and sends the signed encrypted response, together with its public key, to the polling peer.

Upon receiving the response, entity $w$ verifies the signature using the received public key and decrypts the message using its own private key $Sk(w)$. It then verifies the signature of each piece of feedback by the public key of the feedback source. The fact that the data are encrypted with the public key of the polling entity $w$ protects their confidentiality. The fact that data are signed with the responding entity’s private key allows the detection of integrity violations of the data and the authenticity of their origin.

With the above scheme, entities can still cause data loss by corrupting the data or selectively discard data that are stored in its local database. To combat this and data loss caused by other issues such as routing anomaly, data replication can be used at the second layer to improve the data availability and reliability. A secure trust computation algorithm proceeds as in Algorithm shown in Fig.4.6.
Secure Algorithm:

Trust Data contains the following information about any resource provider.
Features; Number of Transaction; Current Feedback (u,i); Previous Feedback
Inputs: Client’s requirement, Local Index Table, Global Index Table, Local Feedback Table, Global FeedBack Table

Steps
1. The Trust Manager receives Grid Resource Provider List; J [n] created in the Grid, where n is the number of resource Providers.
2. Sort the resources as per client’s (Resource Consumer) requirement.
3. Calculate Satisfaction value for these sorted resources.
4. While trust calculated for ten resource provider from Resource Database.
5. Get the feedback (local & global) for the resource provider using

\[
F_{(u,v)} = \frac{\text{Current Feedback}_{(u,v,i)} + \sum_{i=1}^{n-1} \text{Previous Feedback}_{(u,v)}}{N_{Transaction_{(u,v)}}}(PK(w))
\]

6. Verify the signature of Feedback (PK (w)) using the attached public key
7. Get the Number of transaction from Resource Database
8. Calculate the Decay function
   a) Decrypt the data with its private key SK (w)
   b) Calculate Feedback Decay = 0 -if last successful transaction is within set time
      .1 –if there is no transaction is within set time
      .5 –if last unsuccessful transaction is within set time
9. Calculate Trust

\[
\text{Trust}_{final_{(u)}} = sat_r + \alpha \cdot F_{(u)} - \gamma_{(u)} + \beta \cdot TCF_{(u)}
\]

9. End While

Output: Sorted Trust based Resource Provider List, Service Provider List and Recommender’s List

Fig 4.6 Secured Grid Peer Trust Algorithm
4.5 Workflow of Trust Management System

Fig 4.7 shows the event view of the life cycle of a Trust Management System.

Fig. 4.7 Life cycle of a Trust Management System
The life cycle is composed of mainly three different phases:

1. Trust Creation Phase

   The trust creation phase generally occurs before any trusted group is formed, and it includes mechanisms to develop trust functions and assumptions. This phase actually takes place before any transactions and is responsible for setting up the trust functions and the assumptions that will be used by the trust management system. The first step in the trust initialization phase is to determine the type of the trust management system, whether it would be policy-based or it would be reputation-based; in this case it would be reputation-based. Once the type of TMS system is decided; the assumptions are defined and created. The next important step in this phase is to determine the trust function. As mentioned earlier, trust functions can be of several categories: objective or subjective, transaction based or opinion-based, complete or localized, and threshold-based or rank-based. The choice of a trust function depends on the type of application the TMS is catering to and per the requirement; we choose transaction-based trust functions.

2. Trust Negotiation phase

   Trust negotiation, is activated when a new untrusted node joins the current grid environment. The second phase in the life cycle of the TMS is the trust negotiation phase which is started when a new entity or node joins the grid environment. Figure 4.8 shows a very high level overview of any trust negotiation phase. The phase essentially consists of three different steps: request, follow guidelines, and accept.

   The different steps in the Trust Negotiation Phase are:

   a. Request: This step identifies the client and the type of service the client wants from the system. This step can succeed a key establishment phase, where the
session key can be established by the two parties.

b. Follow Guidelines: This step requires the new entity to follow certain guidelines which have been set to be part of grid environment in the form of few assumptions. At this step, the trust computation of the new node can also be evaluated based on the system’s trust function.

c. Accept: In this phase, the new node is accepted or rejected to be the part of the grid environment.

3. Trust Management phase

The third phase, or the trust management phase, is responsible for recalculating the trust values based on the transaction information, distribution or exchange of trust related information, updating and storing the trust information in a centralized or in a distributed manner. The different activities that are part of this phase are:

a. Trust Computation: In this step, the trust value is computed based on the decided trust function.

b. Trust Distribution: This step includes the secure distribution of trust information to other nodes in the distributed system. Since a secure distribution is a necessity, all the principles of security, viz. confidentiality, authentication, integrity, and non-repudiation need to be maintained. This step also requires keeping in mind the type of trust function in use and the number of nodes where the information needs to be broadcasted.

c. Trust Storage: The trust information needs to be securely stored. The credential repositories like the MyProxy can be used for this purpose.
d. Trust Update: Updating the trust needs to be carried out either in an event by event basis or in a timely manner. Event based trust update can happen after a set of transactions or when the trust value or opinion crosses a threshold.

Fig. 4.8 Typical trust negotiation phase
4.6 Conclusion

This chapter introduced the Grid Reputation-Based TMS Architecture, including its functional requirements within the context of Virtual Organizations and the workflow of Trust Management System. The next chapter presents case study, which is highly applicable to the commercial Taxes implemented using Grid Reputation-Based TMS Architecture.