CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Grid computing is a form of distributed computing, hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities as defined in Foster and Kesselman (1999). Grid is a system that

- Coordinates resources that are not subject to centralized control,
- Uses standard, open, general-purpose protocols and interfaces, and
- Delivers non-trivial Qualities of Service.

1.2 GRID COMPUTING

Grid computing facilitates solving computationally intensive or data intensive problems by sharing resources (CPU, storage, data, software, peripherals, etc) of disparate, high performance computing systems. The systems may be distributed geographically across multiple administrative domains where a domain may be any virtual or physical organization. Each domain will have its own security constraints. Security related rules of a domain are restricted only to that domain and should not override the security issues of the physical domain it belongs to.
There are three different kinds of entities present in a grid environment. The first kind is the consumers who request for specific resources, the second the Resource Providers (RP) who provide the resources. These two entities are present in the lowest level of the hierarchy of the architecture. The third entity is the broker which acts as an intermediary between the consumers and RP; this is present at the next higher level in the hierarchy. The broker is the one responsible for considering the consumer’s functional requirements and allotting them to an appropriate RP. Running a consumer’s job in a grid environment involves four main stages as discussed in Li and Backer (2005) are

- Resource discovery
- Resource selection
- Schedule generation
- Job execution.

**Resource discovery**

The goal of resource discovery is to identify a list of authenticated resources that are available for job submission. In order to cope with the dynamic nature of the grid, a grid environment needs to have some way of incorporating dynamic state information about the available resources into its decision-making process.

A grid environment typically uses a pull model, a push model or a push–pull model for resource discovery. The outcome of the resource discovery process is the identity of resources available (Ravailable) in a grid environment for job submission and execution.
The pull model

In this model, a single daemon associated with the scheduler can query grid resources and collect state information such as CPU loads or the available memory. The pull model for gathering resource information incurs relatively small communication overhead. Unless it requests resource information frequently, it tends to provide fairly stale information which is likely to be constantly out-of-date, and potentially misleading. In a centralized environment, the resource discovery/query process could be rather intrusive and begin to take significant amount of time as the environment being monitored gets larger and larger.

The push model

In this model, each resource in the environment has a daemon for gathering local state information, which will be sent to a centralized scheduler that maintains a database to record each resource’s activity. If the updates are frequent, an accurate view of the system state can be maintained over time. Obviously, frequent updates to the database are intrusive and consume network bandwidth.

The push–pull model

The push–pull model lies somewhat between the pull model and the push model. Each resource in the environment runs a daemon that collects state information. Instead of directly sending this information to a centralized scheduler, there exist some intermediate nodes running daemons that aggregate state information from different sub-resources and which in turn respond to queries from the scheduler. A challenge of this model is to find out
what information is most useful, how often it should be collected and how long this information should be kept around.

**Resource selection**

Once the list of possible target resources are known, the second phase is to select those resources that best suit the constraints and conditions imposed by the user such as CPU usage, availability of RAM or disk storage. The result of resource selection is to identify a resource list $R_{selected}$ of all resources which can meet the minimum requirements for a submitted job or a job list. In the proposed work, apart from the minimum functional requirements, trustworthiness of the resources has also been considered. The relationship between resources available ($R_{available}$) and resources selected ($R_{selected}$) is:

$$R_{selected} \subseteq R_{available}$$

**Schedule generation**

A scheduler in a grid should always know what resources it can access, how busy it is, how long it takes to communicate with them and how long it takes for them to communicate with each other. With this information, the scheduler optimizes the scheduling of jobs to make more efficient and effective use of the available resources.

The generation of schedules involves two steps - scheduling jobs and producing resource scheduling strategies.

Resource scheduling strategy: The resource scheduling process is used to choose resource(s) from the resource list ($R_{selected}$) for a given job.
Since all resources in the list \( R_{\text{selected}} \) could meet the minimum requirements imposed by the job, an algorithm is needed to choose the best resource(s) to execute the job. Although random selection is a choice, it is not an ideal resource scheduling policy. The resource scheduling algorithm should take into account the current state of resources and choose the best one based on a quantitative evaluation. A resource scheduling algorithm normally takes CPU, RAM and software environment, etc., into account.

Job scheduling strategy: The goal of job scheduling is to select a job from a job queue for execution. Four strategies that can be used to select a job are given below.

- **First come first serve:** The scheduler selects jobs for execution in the order of their submissions. If there is no resource available for the selected job, the scheduler will wait until the job can be started. The other jobs in the job queue have to wait. There are two main drawbacks with this type of job selection. It may waste resources when, for example, the job selected needs more resources to be available before it can start, which results in a long waiting time. And jobs with high priorities cannot get dispatched immediately if a job with a low priority needs more time to complete.

- **Random selection:** The next job to be scheduled is randomly selected from the job queue. Apart from the two drawbacks with the first-come-first-serve strategy, job selection is not fair and job submitted earlier may not be scheduled until much later.

- **Priority-based selection:** Jobs submitted to the scheduler have different priorities. The next job to be scheduled is the job with the highest priority in the job queue. A job priority can be set
when the job is submitted. One drawback of this strategy is that it is hard to set an optimal criterion for a job priority. A job with the highest priority may need more resources than available and may also result in a long waiting time and inability to make good use of the available resources.

- Backfilling selection: The backfilling strategy requires knowledge of the expected execution time of a job to be scheduled. If the next job in the job queue cannot be started due to a lack of available resources, backfilling tries to find another job in the queue that can use the idle resources.

**Job execution**

Once a job and a resource are selected, the next step is to submit the job to the resource for execution. Job execution may be as easy as running a single command or as complicated as running a series of scripts that may, or may not include set up or staging.

### 1.3 MOTIVATION FOR TRUST AND POLICY-BASED RESOURCE SELECTION

Resource selection is an important issue in a grid environment where a consumer and an RP are distributed geographically across multiple administrative domains, as discussed in Azzedin and Maheswaran (2004). In such a scenario, it is essential that the selection of an RP is not only based on service functionality, but also on the process behavior such as trustworthiness, QoS of the RP and policy-constraints of both the communicating parties. In a broker-based trust management system, intermediaries known as brokers are
responsible for ensuring the trustworthiness of the RP and its QoS to the consumer based on two aspects.

One aspect for consideration is the trust-index of any entity involved in grid environment. The other aspect is the matched-policy constraints, discussed in Constandache et al (2005). Both these aspects help to encourage and establish trust between the two entities (consumer and RP) that are strangers and dissuade participation by those who are dishonest.

Need for Trust

Grid computing systems that have been the focus of much research activity in recent years provide a virtual framework for controlled sharing of resources across institutional boundaries. As part of such a geographically distributed environment, an entity will have the privilege of using pools of resources that would not be available to it otherwise. Unfortunately, the idea of having a virtual framework such as the grid is not appealing to some entities because of the risk of being associated with the notion of sharing resources. Because of the sensitivity and the vitality of data or information, such entities prefer to use their own “closed box” resources. This is not just costly for the individual entities but also an inefficient way to utilize resources. To make grid computing more appealing, trust must be addressed and trustworthy domains must exist where an entity can use resources safely.

Each grid node may act as a consumer or RP. The consumer and the RP for a specific transaction may be in different domains. A consumer, belonging to one domain may require executing the jobs over a remote resource belonging to another domain. In such a scenario, the consumer and the RP do not have complete control over each other. The consumer expects good QoS from a trustworthy RP. The RP expects its resources to be
protected and may allow its resources to be utilized by a ‘trustworthy’ consumer. To achieve this end, it is necessary to establish trust across the grid between the consumer and the RP.

Trust is a complex subject relating to an entity’s belief in honesty, truthfulness, competence and reliability of another entity. In most of the existing distributed heterogeneous networks, trust between a consumer and an RP is established based on identity and reputation. This identity-based trust model is concerned with verifying the authenticity of an entity and what it is authorized to do. This, however, does not ensure consistency, promptness of service and QoS, resulting in loss to the consumers.

This problem is overcome in reputation-based trust management. Reputation of an entity is a measure derived from direct or indirect knowledge of the entity’s earlier transactions. In this model, a certification process verifies the consistency of resources offered by an RP. The consumers who have had transactions with the RPs give feedback on various aspects of the resources offered by the RPs. The feedback received for an RP from various consumers is aggregated over a period of time. This forms the reputation of the specific RP. The consumer first confirms whether the behavior of the RP is trustworthy or not, before proceeding to use the RP. This ensures QoS for the consumer. This scheme is appropriate in a grid environment where entities are distributed geographically and are unknown to each other.

**Need for matched policy-constraints**

Policy is a subject relating to third party accreditations that an entity has acquired. In this policy-based system, a consumer concludes on the worthiness of an RP purely based on its credentials (policies) acquired from standard organizations such as signed certificates from Certification
Authorities. As an example, the consumer might be interested in knowing whether an RP has the information security practices certified by international standards authorities such as BITS, ISO 270001, etc. Similarly, an RP decides on the worthiness of consumers based on the credentials (policies) acquired from standard organizations. Different weights are assigned to each of the policies depending on the third party that provided the accreditations.

The consumer requests include certain policy requirements that the RP should possess. Considering the RP’s point of view, when an RP associates itself with a broker, it specifies a set of policy requirements that the consumer should possess to utilize its resources. Policies are well suited to specify access control conditions that are eventually meant to yield a Boolean decision. The broker processing the consumer request tries to find an RP that not only matches the consumer's constraints with good trust-index for providing trustworthy resource but also the one with matched policy requirements of both consumer and RP. Thus, runtime failure due to policy mismatch is avoided.

1.4 RESEARCH OBJECTIVES

In this thesis, the proposed broker architecture supports the choice of RP based on their trust-indices along with matched policy constraints maintained at intermediaries called brokers.

In the Single Broker Architecture proposed initially, there is only a single broker present in each domain. All entities (RPs and consumers) in that domain are associated with this broker. All resource requests and consumers’ feedbacks in that domain are processed only through the single broker of that domain. There is a possibility of heavy traffic at the broker’s site, and the
broker is heavily loaded. Also, the presence of a single broker may lead to the problem of single point failure.

In order to resolve these problems, we propose a Multi-Broker Architecture. In this architecture, multiple brokers are present in a domain; each broker is associated with multiple entities (consumers and RPs), and each of the entities is associated with more than one broker. The broker is responsible for maintaining the information about the trustworthiness of the entities and for the selection of a suitable RP for a consumer’s request. This is a delegation model, since the consumer’s request is not only processed by the associated broker, but also delegated to other brokers in the same domain as well as other brokers in other domains, in order to select the best trustworthy RP for the consumer’s request. However, the Multi-Broker Architecture suffers from biasness of brokers, where a broker gains brokerage charges for every transaction that is transacted through it. Therefore, it tries to choose an RP under its fold for the consumer’s request to avoid sharing of brokerage charges, though a more trustworthy resource is available with another broker.

To counter the issues of Multi–Broker Architecture, the consumer’s request is processed by the associated broker whenever possible, delegation of consumer’s requests happens whenever necessary. So the consumer’s request simply will not be delegated to other brokers. If there is no RP that satisfies the request constraints of the consumer’s request, the consumer is free to approach another broker to process the request. To further reduce the broker biasness and avoid the approach of choosing an RP through another broker by itself (consumer) in case of the failure of finding an RP under a chosen broker, we propose a Hierarchical Broker Architecture with three tiers for the simulation purpose, but it can be easily extended to any number of levels. It is implemented with Regional Resource Administrators (RRA) at the top level, Resource Brokers at the middle level and the RP at the bottom level. The
objective of forming RRAs that serves to provide mechanisms to objectively manage the operations in a dynamic, competitive open grid environment, prone to disruptive and malicious behavior. The RRA does not derive any compensation from any of the entities (consumers as well as associated brokers) for the transactions undergone. It obtains compensation only from the registration, renewal and audit charges of the associated brokers. Such an arrangement of entities in the Hierarchical Broker Architecture releases the RRAs from being considered biased in the choice of a suitable RP for the consumers’ request. However, this architecture has certain pitfalls.

The Hierarchical Broker Architecture does not address the issue of ensuring security. Attacks on RRA, the backbone of the architecture, can paralyze the underlying architecture from functioning. Next, a broker or consumer registered under an RRA may oscillate in its behavior. It may behave honestly initially to build its trust only to degrade or enhance other entity’s trust later. Next, the dishonest feedback given by the entities (both the consumers as well as the brokers) remain unfiltered which affect the trust indices of the associated entities (RPs). The proposed Enhanced Hierarchical Broker Architecture which addresses the above issues, providing cost-effective services to the consumers by the RPs who best satisfy the requests constraints and policy constraints, is presented. Apart from this, an improved scheme that supports the verification of computations is also added to the Enhanced Hierarchical Broker Architecture.

1.4.1 Attacks

Grid applications involve large scale resource sharing (data and computing resources) that must be secured. The grid technologies currently available support authentication, authorization, resource access, resource discovery, etc. However, it is necessary to ensure the availability of high
performance Grid resources by preventing spamming and Distributed Denial of Service (DDoS) attack. We propose solutions for securing Hierarchical Broker Architecture with DDoS defense mechanism and Spam filtering mechanism that aim at minimizing the wastage of resources and maximizing the resource availability.

1.4.1.1 DDoS problem

In a Grid environment, consumers requesting for resources are dispersed over various domains. Therefore, it is difficult to predict the behavior of consumers and verify their requests for legitimacy. An attacker may mimic a genuine consumer to hack the grid infrastructure. A typical example is the DDoS attack, where the attacker acts as a consumer and targets a particular RP’s site to exhaust the network and system resources. The DDoS attack attempts to make a system’s resources (e.g. Network bandwidth, CPU time, etc) unavailable to the genuine consumers. The DDoS consumers may flood the infrastructure with the illegitimate DDoS traffic.

Thus, the consequence of not detecting these illegitimate requests leads to wasteful exhaustion of system resources in the entire architecture. Therefore, the system has to be protected from such attacks posed by the consumer attackers. This led to the proposal of the enhanced Hierarchical Broker Architecture with the DDoS Defeat Engine comprising one or more Authenticators. An Authenticator authenticates the consumers and helps in detecting the DDoS attackers.

A DDoS attack can be characterized as a large-scale, coordinated attack that is launched indirectly through multiple compromised hosts, called zombies, on victim network resources, with the purpose of preventing legitimate users from using those resources. DDoS attacks hog resources
rendering them unavailable to legitimate users thus reducing the usefulness of the grid. A DDoS attack on commercial SUN grid in 2006 brought down the entire grid to its knees within hours of its launch necessitating a login procedure change.

Presence of an online mechanism that does not require the server to hibernate or to be shut down is a must for Grid. Also the defense mechanism must be able to distinguish attack packets from legitimate ones with high accuracy and minimal resource consumption. The complexity of DDoS problem suggests that the solution will require a collaborative defense mechanism.

For this purpose, a three level Hierarchical Cognitive Memory Model proposed by Atkinson-Shiffrin in Artificial Network is proposed to build a collaborative DDoS defense mechanism in our Hierarchical Broker Architecture. At the base level, registers capture and analyze the traffic characteristics and pass it on to Short-term memory at the second level which stores these patterns and tries to detect a DDoS attack locally. The results of the analysis are communicated to the Long-term memory present at the highest level which is used for global detection of an attack using collaboration.

1.4.1.2 Spam problem

Spam refers to unsolicited commercial bulk electronic messages. Spam is one of the major problems of the mail system. According to the global survey conducted by Radicati group (2007), 75% of all mail traffic is spam. An individual on an average receives 93 mail messages per day and as high as 18% of that is spam. Even as spam filtering techniques adapt and improve, spammers are continuously trying to outsmart the filters. As a result,
spam manages to creep into the mailbox. In a grid system, spam can deplete resources (CPU as well as storage) which can otherwise be put to use for problem solving. A spam filtering mechanism aims to minimize this grid resource depletion and maximize the resource availability.

Spam filtering techniques include origin-based filtering, content-based filtering, context-based filtering, heuristics-based filtering and statistical filtering. Origin-based filters such as blacklist and white list use network information such as IP addresses and e-mail domains to classify messages. Content-based filters examine the message contents and try to “understand” them to detect spam. Several statistical techniques are merged with content-based filtering for accurate classification.

We propose a layered spam filtering mechanism consisting of origin-based filters (blacklist and white list) followed by statistical content-based filters (Bayesian filters) since these filters are able to detect most possible spam adaptively according to the requirement of the entities with minimal false positives.

1.4.2 Malicious Behavior of the Entities

The oscillatory behaviors of the resource brokers and consumers in the architecture have adverse effects on the trust-index computations of the other entities. One of the detrimental vulnerabilities of Hierarchical Broker Architecture is that a malicious entity alters its behavior in such a way that benefits itself such as starting to behave maliciously after it attains a high trust-index. In other words, a malicious entity acts genuine till it attains a high trust-index that can be used to hide its maliciousness thereafter. The attained trust-index can be used by the broker to boost its own ratings. The malicious entity may use the built-up trust value to badmouth genuine entities or to
enhance the trust-index of other malicious entities. Such malicious behavior of the entities disturbs the normal working environment.

We propose an Enhanced Hierarchical Broker Architecture with reference set that should be able to penalize such malicious entities for their oscillating and fluctuating behavior. The trust-index computations and its updating need to be effective. In the proposed architecture, the malicious behavior exhibited by an entity is reduced by ensuring that the trust-index raise up only at a slower rate for the genuineness of the entity but drops quickly as soon as it behaves maliciously.

This approach makes it difficult for a malicious entity as it has to act genuine for a higher number of transactions to attain a high trust-index. Similarly, the quicker fall of the trust-index prevents the entity from behaving maliciously for a number of transactions. The behavior of every entity is kept on track by the RRA by maintaining a reference set.

1.4.3 Dishonest Feedback Filtering

In the proposed Hierarchical Broker Architecture the trust-indices are updated based on the feedbacks provided by the associated entities after each transaction by the RRA. Sometimes the feedbacks received by the RRA may be dishonest which is difficult to identify immediately. Consideration of these dishonest feedbacks for the evaluation of trust-indices will lead to misleading results. Hence, it is necessary to identify and filter the dishonest feedbacks to ensure accurate evaluation of the trust-indices.

We propose a reference set in the Hierarchical Broker Architecture to pre-evaluate and rate the entities’ feedback behaviors. This rating is obtained by allowing each entity to transact with a reference set which is built
dynamically by the RRA. The deviation of an entity’s feedback from the RRA’s reference set forms its rating. The reference set is constructed dynamically once for every checking period $T$. During the checking period, the feedbacks obtained from the consumer and broker about an RP’s resource in the reference set is verified for honesty by the RRA for every transaction.

1.4.4 An Improved Computation Verification Scheme

The existing scheme for computation verification in a distributed grid system is replication. This scheme suffers from collusion and redundant execution of the jobs leading to high resource consumption.

In order to combat these issues, a scheme called an improved quizzing is proposed in the Hierarchical Broker Architecture. Some tasks of given jobs itself are given as quiz queries instead of separate set of quiz queries in order to avoid the differentiation of jobs from quiz queries at the RP’s site. This leads to the better resource utilization compared to the simple quiz-based result verification scheme.

1.5 ORGANIZATION OF THE THESIS

This thesis is organized into six chapters. The first chapter provides the background and motivation for the present research. Literature survey work for the general grid applications, various grid broker architectures, trust-based and policy-based resource selection, and possibilities of various attacks defense mechanisms like DDoS and spam filtering is discussed in chapter 2. The various proposed broker architectures and their advantages and drawbacks are explained in chapter 3. Computation of the reputation-based trust values of various entities involved and the simulation of the proposed architectures and their obtained results are presented in chapter 4. An
improved result verification mechanism using the proposed improved quizzing scheme is also discussed. Chapter 5 deals with the identity-based trust with the proposed counter mechanisms for the possible attacks like DDoS attacks, spam attacks over the proposed architectures along with the secured authentication mechanism. The conclusion is given in chapter 6.