Evolving and Managing Trust in Grid Computing Systems[*]

Farag Azzedin and Muthucumaru Maheswaran
University of Manitoba and TR*Labs*
Winnipeg, Manitoba
Canada
E-mail: {fazzedin, mahes}@cs.umanitoba.ca

Abstract

A Grid computing system is a geographically distributed environment with autonomous domains that share resources amongst themselves. One primary goal of such a Grid environment is to encourage domain-to-domain interactions and increase the confidence of domains to use or share resources (a) without losing control over their own resources, and (b) ensuring confidentiality for others. To achieve this, the "trust" notion needs to be addressed so that trustworthiness makes such geographically distributed systems become more attractive and reliable for day-to-day use. In this paper, we view trust in two steps: (a) verifying the identity of an entity and what that identity is authorized to do, and (b) monitoring and managing the behavior of the entity and building a trust level based on that behavior. The identity trust has been the focus of many researchers, but unfortunately the behavior trust did not catch much attention. We present a formal definition of behavior trust and reputation and discuss a behavior trust management architecture that models the process of evolving and managing of behavior trust in Grid computing Systems.

Keywords: security, trust, Grid computing

1. Introduction

Grid computing systems [FoK99, FoK01] 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 or services. 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 *trust domains* must exist where an entity can use resources or deploy services safely. Trust is a complex concept that has been addressed at different levels by many researchers [MeO01, AdF99, AbH00, DaD01]. We classify trust into two categories: *identity trust* and *behavior trust*. Identity trust is concerned with verifying the authenticity of an entity and determining the authorizations that the entity is entitiled to access and is based on techniques including encryption, data hiding, digital signatures, authentication protocols, and access control methods. Whereas behavior trust deals with a wider notion of an entity's "trustworthiness." For example, a digitally signed certificate does not convey if the issuer is an industrial spy and a digitally signed code does not convey if the code is written by competent programmers [AbH00].

In this paper, we propose a trust model that deals with *behavior trust*, how it is evolved based on transactions between entities, and how it is managed in Grid computing systems. In the rest of the paper, when we state "trust," we mean "behavior trust" unless explicitly stated. Section 2 defines the notions of trust and reputation and outlines mechanisms for computing them. An overall trust model for Grid systems is presented in Section 3. The notion of *trusted domains* in a Grid environment is discussed in Section 4. An example of a trust transaction involving two domains and how the trust relationship is built and maintained is illustrated in Section 5. Related work is briefly discussed in Section 6.

2. Trust and Reputation

2.1. Definition of Trust and Reputation

The notion of trust is a complex subject relating to a *firm belief* in attributes such as reliability, honesty, and competence of the trusted entity. There is a lack of consensus in the literature on the definition of trust and on what constitutes trust management [Mis96, GrS00, AbH00]. The definition of trust that we will use in this paper is as follows:

> *Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.*

That is, the *firm belief* is a dynamic value and spans over a set of values ranging from *very trustworthy* to *very untrustworthy*. This *trust level* (TL) is built on past experiences

[*]Proceedings of the 2002 IEEE Canadian Conference on Electrical Computer Engineering 0-7802-xxxx-x/02/$10 ©2002 IEEE

and given for a specific context. For example, entity y might trust entity x to use its storage resources but not to execute programs using these resources. The TL is specified within a given time because the TL today between two entities is not necessarily the same TL a year ago.

When making trust-based decisions, entities can rely on others for information pertaining to a specific entity. For example, if entity x wants to make a decision of whether to use machine M_j, which is unknown to x, then x can rely on the reputation of M_j. The definition of reputation that we will use in this paper is as follows:

> *The reputation of an entity is an expectation of its behavior based on other entities' observations or information about the entity's past behavior within a specific context at a given time.*

2.2. Computing Trust and Reputation

In computing trust and reputation, several issues have to be considered. First, trust decays with time. For example, if x trusts y at level p based on past experience five years ago, the trust level today is very likely to be lower unless they have interacted since then. Similar time-based decay also applies for reputation. Second, entities may form alliances and as a result would tend to trust their allies and business partners more than they would trust others. Finally, the *trust level* that x holds about y is based on x's direct relationship with y as well as the reputation of y, i.e., the trust model should compute the eventual trust based on a combination of direct trust and reputation and should be able to weigh the two components differently.

Let D_i and D_j denote two domains of entities. The trust relationship based on a specific context c at a given time t between the two domains, expressed as $\Gamma(D_i, D_j, t, c)$, is computed based on the direct relationship for the context c at time t between D_i and D_j, expressed as $\Theta(D_i, D_j, t, c)$, as well as the reputation of D_j for the context c at time t expressed as $\Omega(D_j, t, c)$. The weights given to direct and reputation relationships are α and β, respectively. As far as D_i is concerned the "trustworthiness" of D_j is based more on direct relationship with D_i rather than the reputation of D_j. Therefore, α is larger than β. Direct relationship is computed as a product of the *trust level* in the *direct-trust table* (DTT) and the *decay function* ($\Upsilon(t-t_{ij}, c)$), where c is the specific context for the trust relationship, t is the current time, and t_{ij} is the time of the last update or the last transaction between D_i and D_j. The time factor t as explained earlier is very critical because information well-received from an entity five years ago might be ill-received today based on the validity of the information as well as how trustworthy is the entity today. The reputation of D_j is computed as the average of the product of the *trust level* in the reputation-trust table (RTT), the *decay function* ($\Upsilon(t-t_{kj}, c)$), and the recommender trust factor $(R(D_k, D_j))$ for all domains k. In practical systems, entities will use the same information to evaluate direct relationships and give recommendations, i.e., RTT and DTT will be the same. Because reputation is based primarily on what other domains say about a particular domain, we introduced the recommender trust factor R to prevent cheating via collusions among a group of domains. Hence, R is a value between 0 and 1 and will have a higher value if D_k and D_j are unknown or have no prior relationship among each other and a lower value if D_k and D_j are allies or business partners.

$$\begin{aligned}\Gamma(D_i, D_j, t, c) &= \alpha \times \Theta(D_i, D_j, t, c) + \beta \times \Omega(D_j, t, c)\\ \Theta(D_i, D_j, t, c) &= DTT(D_i, D_j, c) \times \Upsilon(t - t_{ij}, c)\end{aligned}$$

$$\Omega(D_j, t, c) = \frac{\sum_{k=1}^{n} RTT(D_k, D_j, c) \times R(D_k, D_j) \times \Upsilon(t - t_{kj}, c)}{\sum_{k=1}^{n} (D_k)}$$

3. Trust Model

3.1. Overview

Figure 1 shows the overall trust model in which the Grid is divided into *Grid domains* (GDs). We associate two virtual domains with each GD, namely a *resource domain* (RC) to signify the resources within the GD and a *client domain* (CD) to signify the clients within the GD. Trust agents exist in each GD with mechanisms to: (a) update the GDs' trust tables, (b) allow entities to join GDs and inherit their trust attributes, and (c) apply a decay function to reflect the decay of trust between domains.

A straight forward approach to creating and maintaining the trust level table can result in an inefficient process in a very large-scale system such as the Grid. This process is made efficient in our model by various methods. First, as mentioned previously, we divide the Grid system into GDs. The resources and clients within a GD inherit the parameters of the RD and CD that they are associated with. This increases the scalability of the overall approach. Second, trust is a slow varying attribute, therefore, the update overhead associated with the trust level table is not significant. A value in the trust level table is modified by a new trust level value that is computed based on a *significant* amount of transactional data. Third, by limiting the number of contexts, we can reduce the fragmentation of the trust management space. In our study, the contexts are limited to primary service types such as printing, storage, and computing.

3.2. Direct and Reputation Trust

To evaluate the trust relationship at a given time t between two domains D_i and D_j for a specific context c,

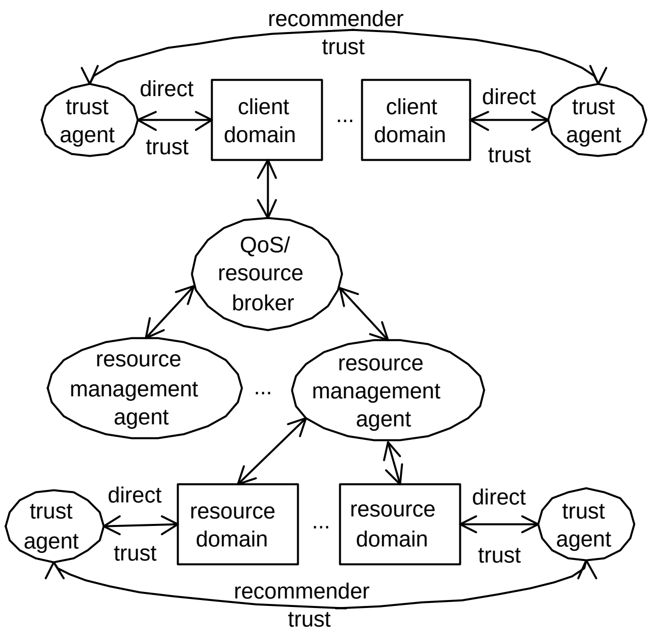

Figure 1. Components of a Grid resource management trust model.

Table 1. Description of the required trust levels.

Trust Level (TL)	Description
A	very low trust level
B	low trust level
C	medium trust level
D	high trust level
E	very high trust level
F	extremely high trust level

two components have to be considered: (a) direct relationship (direct trust), and (b) the reputation relationship (indirect trust based on recommendations). Each domain's trust agent will maintain a DTT as shown in Table 2. From this table we see that for a specific context c_i, D_k can utilize resources or deploy services using D_j's resources and hence a direct relationship will exist between these two domains. Since a direct trust relationship is asymmetric, each of these two domains involved in this direct relationship will have its own interpretation (see Table 1) of how well or how bad this direct trust relationship is. When D_i wants to have an interaction with D_j, in addition to the direct trust relationship, D_i can rely on recommendations from other domains about D_j (i.e., asking for the reputation of D_j). Therefore, each domain's trust agent will evaluate the direct as well as the recommender trust as illustrated in Sections 3.5 and 3.6.

Table 2. Direct trust table maintained by D_k.

Context	Domains		
	D_1	$\ldots$	D_j
c_1	$TL_{k1}^{c_1}$	$\ldots$	$TL_{kj}^{c_1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_i	$TL_{k1}^{c_i}$	$\ldots$	$TL_{kj}^{c_i}$

3.3. Direct and Reputation Trust Weights

Weights α and β are given to direct and reputation trust relationships, respectively and they range between 0 and 1. Assigning values to these weights is up to the individual domain, for example: (a) D_i might trust business partners or allies more than other domains. Therefore, D_i will give more weight to its business-partners and allies as recommenders or as domains to directly interact with, (b) D_i might have a policy stating that D_i will only accept recommendations from domains whom D_i has a direct trust relationship with. That is, D_i will assign a value of zero ($\beta = 0$) to recommenders whom it does not have direct interaction with.

3.4. Decay Function

As any other relationship, trust decays with time. For instance, if D_i has not interacted with D_j for five years, then the TL between them today is likely to be weaker unless they have interacted since. In our trust model, we introduced a *decay function* to reflect this drop when modeling trust between domains. To compute the decay function $\Upsilon(t - t_{ij}, c)$, we look at how old (in terms of time) is the TL that resulted from the last transaction between D_i and D_j. Each domain might have different decay function and might be looking at other factors that accelerate or decelerate the TL decay. For example, the acquaintance that D_i has with D_j such that both domains have the same legal obligations (i.e. from the same union, country, etc.). Therefore, D_i might decide to decay TLs for those domains from unfamiliar environments faster than domains acquainted with.

3.5. Evaluating Direct Trust

TL resulting from a direct trust relationship means that D_i is directly involved in a transaction with D_j. There are two *required trust levels* (RTLs), one from the client side and the other from the resource side. For example, if two domains D_i and D_j are directly engaging in a transaction, D_i may not want its applications mapped onto resources that are owned and/or managed by a domain it does not trust. Similar concerns apply from the resource producer side as well, i.e., D_j may not want its resources being utilized by applications that are owned by a domain it does not trust. Hence, each of D_i and D_j will specify a RTL that should not be violated. D_i evaluates the direct trust relationship with D_j based on the behavior of D_j and how well D_j abides by and respects D_i's RTL. The same can be said about D_j when evaluating its direct trust relationship with D_i.

Violating a RTL can be a result of many abuses to a system such as: (a) consuming more resources than requested, (b) leaving behind data and not doing "garbage collection" after using the resources, (c) going to places out of the allocated boundary, and (d) instantiating tasks they are not supposed to instantiate. Such intrusions can be detected by *audit data* [HaC92] generated by the operating system or post-mortem analysis tool such as *Intrusion Detection systems* (IDSs) [Lun93, SmW94]. Determining to what degree and what violates a TL depends on each domain's local security policies and practices. For example, leaving behind data on a storage media might have different affect on different domains because of the storage variability each domain owns. A domain that owns a huge disk space might not be affected as a domain who has a smaller disk space. Hence, it is up to the individual domain to come up with *trust penalty levels* (TPLs) to assess the experience of a direct trust relationship with another domain. Furthermore, the TPL should be more severe if a violation is found during a post-mortem IDS analysis as opposed to real time IDS analysis. The post-mortem IDS analysis is more costly as it leaves a leeway to the intruder to go undetected for a while and hence has the potential of causing more harm to other domains. Depending on how well or how bad D_j abided by D_i's RTL, D_i forms its TL at time t based on its direct trust relationship with D_j expressed as $TL(t_{ij}, c)$. In a similar fashion D_j will form its $TL(t_{ji}, c)$.

3.6. Evaluating Recommender Trust

When D_i wants to have a transaction with D_j for a specific context c at a given time t, D_i can rely on recommendations from other domains regarding D_j' s trustworthiness pertaining to c. Let us assume that D_i receives a recommendation from D_k. D_i can not evaluate D_k's recommendation until D_i directly interacts with D_j within the same context for which the recommendation was made. After the direct trust relationship is evaluated and the resulted $TL(t_{ij}, c)$ is obtained, see Section 3.5, D_i will be in a position to update the recommender trust factor (R) for its recommenders.

3.7. Required and Existing Trust Levels

Table 1 shows the description of the RTL values used in our model. An existing trust relationship can have TLs ranging from A to E. Therefore, RTL F is not provided by any existing trust relationship. This is supported in our model so that domains can enforce enhanced security by increasing their RTL value to F.

3.8. Updating Direct and Reputation Trust Tables

To update the DTT, a simple formula such as $DTT(D_i, D_j, c) = (1 - \delta) \times DTT(D_i, D_j, c) + (\delta) \times TL(t_{ij}, c)$ can be used, where $TL(t_{ij}, c)$ is the trust level for context c resulted from the direct trust relationship between D_i and D_j at time t, and $DTT(D_i, D_j, c)$ is the trust level in the DTT for context c resulted from the last direct transaction between D_i and D_j, and δ is a value between 0 and 1. If $\delta > 0.5$, more preference is given to TL resulting from the current direct trust relationship between the two domains. In a similar fashion, the RTT can be updated.

3.9. Trust Inheritance

In such a distributed environment, entities can join or leave a domain D_i at anytime. Hence, a trust model suitable for such an environment should have mechanisms for managing trust for such entities. Our trust model accommodates for this as follows. When an entity x joins a domain, it inherits the TLs in the domain' s DTT as well as in the domain's RTT. However, the other domains might not trust x to be as trustworthy as another entity who has been with D_i for a longer period of time. Therefore, there is a *member weight* associated with every entity to indicate if the entity is a new, recent, or an old member with its domain and it is up to the individual domain to decide what constitutes an entity to fall in one of these *member weights*.

3.10. Evolving Trust

Our model allows domains to build up their TLs from scratch without any prior experience nor trusted recommenders. One might argue that as a newcomer, there is always the chance that a rogue domain may take advantage of the unwitting newcomer by pretending to offer "assistance" for malicious hidden motives. It is true that a newcomer is faced with a high degree of uncertainty about other domains. However, our trust model is designed in such a way that newcomers are protected from such malicious motives. Let D_i be a newcomer that wants to interact with D_j. Each of these two domains will have a RTL. So as a newcomer, D_i can set its RTL value to F, which is no existing trust relationship has, and thus enhanced security is enforced to guard D_i's resources or applications. As D_i interacts with other domains, it can build its own trust values.

4. Trusted Domains

The integration of "trust" into network computing systems introduces trust awareness that enables total isolation of different resource pools as well as client pools

Table 3. Recommendations received by D_i

Context	Domains	
	D_1	D_2
printing service	D	C

into "trusted domains". For a distributed computing environment such as the Grid, trusted domains increase and encourage more business-to-business or organization-to-organization applications to be engaged which will in turn: (a) create more applications to services, and (b) can create new forms of service models. Furthermore, the efficiency of running these applications as well as the utilization of resources will improve due to the minimization of security overhead.

5. Trust Transaction Example

To illustrate the use of our model, we provide an application example of evaluating direct as well as reputation trust relationship in the context of a "printing service" where D_i is providing a "printing service" to other domains. Let us assume that D_i is a newcomer and hence its DTT as well as RTT are empty. Another domain D_j is looking for a "printing service" to print its annual report. Although both D_i and D_j have "trust" concerns, we focus on how D_i evolves and builds its "trust" regarding this experience with D_j. A resource management agent, as illustrated in Figure 1, contacts D_i as a candidate RD since it provides the service sought. Having no direct trust relationship with D_j and being a newcomer, D_i sets its RTL to F. D_i also can rely on recommendations and say that it receives the two recommendations about D_j as shown in Table 3.

Once the transaction between D_i and D_j starts, D_i evaluates the direct trust relationship with D_j (i.e., D_i updates its DTT) by examining whether D_j abides by its RTL. D_i does this evaluation by two mechanisms: (a) using an audit trail analysis [Lun93] to determine if D_j is an abusive domain by detecting failed commands issued by D_j, and (b) monitoring sequences of system calls to detect an abnormal behavior of D_j [HoF98]. Assume that D_i has a classification system to classify the behavior of other domains as shown in Table 4. Furthermore, let us assume that D_i indeed detects an abnormal behavior of D_j and assigns a trust value of three, corresponding to a TL of D. Having $TL(t_{ij}, c) = 3$, D_i's DTT can be updated, as explained in Section 3.8. Initially $DTT(D_i, D_j, c)$ was 0 and assuming that the value of δ is 1, the updated value in D_i' s DTT will be: $DTT(D_i, D_j, c) = \delta \times TL(t_{ij}, c)$. Therefore, D_i is able to build its direct trust relationship from scratch (i.e., update its DTT) and the new TL for $DTT(D_i, D_j, c)$ is set to D.

Second, after evaluating and updating its DTT, D_i can update its RTT in a similar fashion as explained in Section 3.8. In practical systems, entities will use the same information to evaluate direct relationships and give recommendations, i.e., RTT and DTT will be the same. Hence, $RTT(D_i, D_j, c)$ is set to D.

Table 4. D_i' s classification system

Classification range	Classification description	Trust level assigned
$0-2$	very little harm	E
$2-4$	little harm	D
$4-6$	medium harm	C
$6-8$	high harm	B
$8-10$	very high harm	A

Third, to evaluate the recommender trust, D_i has to update its *recommender trust factor table* (i.e., update R) as explained in Section 3.6. Two recommenders, D_1 and D_2, recommended D_j and gave a recommender trust levels of D and C, respectively. After interacting with D_j, D_i found that D_i's TL is D. Therefore, values of 1 and 0.6 are given to D_1 and D_2, respectively as their R factors. These factors indicate that D_1 was on target in recommending D_j while D_2 was off by a margin.

6. Related Work

Trust models such as the PGP [MeO01] and the X.509 [AdF99] as well as trust management applications such as PolicyMaker [BlF96] and KeyNote [Bla99] are concerned *identity trust*. These trust mechanisms do not consider the *behavior trust* which changes over time and thus these approaches have no mechanisms to monitor trust relationships. In addition, these trust models and trust management applications do not recognize the need for entities to learn from past experiences in order to dynamically update their trust levels [GrS00].

A model for supporting *behavior trust* based on experience and reputation is proposed in [AbH00]. This trust-based model allows entities to decide which other entities are trustworthy and also allows entities to tune their understanding of another entity's recommendations.

A survey of trust in Internet applications is presented in [GrS00] and as part of this work a policy specification language called Ponder [DaD01] supporting *behavior trust* was developed. Ponder can be used to define authorization and security management policies. Ponder is being extended to allow for more abstract and potentially complex trust relationships between entities across organizational domains.

Our model expands the work done in [AbH00, DaD01] in many ways: (a) trust decays with time, (b) an entity may trust its allies and partners more than it trusts others, (c) our trust model uses a mechanism such that trust values resulting from direct relationships weigh more than those resulting from reputation of an entity, and (d) our trust model accommodates for inheritance.

7. Conclusions

The Grid computing systems are being positioned as a computing infrastructure that will enable pools of resources to be shared across institutional boundaries. Unfortunately, the notion of "sharing" poses some concerns such as privacy, confidentiality, and autonomy. Hence, "trust" should be addressed in such a distributed environment. We view trust in two steps: (a) *identity trust* which is verifying the identity of an entity and what that identity is authorized to do, and (b) *behavior trust* which is monitoring and managing the behavior of the entity. Identity trust has been addressed by techniques such as encryption, data hiding, digital signatures, and access control. In this paper we proposed a trust management architecture that can evolve and maintain the behavior trust based on direct as well as reputation trust relationships. An example application is presented to illustrate how our model evolves and manages trust between two domains.

References

[AbH00] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," *Hawaii Int'l Conference on System Sciences*, 2000.

[AdF99] C. Adams and S. Farral, "RFC2510 - Internet X.509 public key infrastructure certificate management protocols," 1999.

[BlF96] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," *IEEE Conference on Security and Privacy*, 1996.

[Bla99] M. Blaze, "Using the KeyNote trust management system," *AT&T Research Labs*, 1999.

[DaD01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder policy specification language," *Workshop on Policies for Distributed Systems and Networks*, 2001.

[FoK01] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *Int'l Journal on Supercomputer Applications*, 2001.

[FoK99] I. Foster and C. Kesselman (eds.), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Fransisco, CA, 1999.

[GrS00] T. Grandison and M. Sloman, "A survey of trust in Internet applications," *IEEE Communications Surveys & Tutorials*, Vol. 3, No. 4, 2000.

[HaC92] N. Habra, B. L. Chalier, A. Mounji, and I. Mathieu, "ASAX: Software architecture and rule-based language for universal audit trail analysis," *European Symposium on Research in Computer Security (ESORIC'92)*, 1992.

[HoF98] S. A. Hofmeyr, A. Somayaji, and S. Forrest, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, Vol. 6, 1998, pp. 151–180.

[Lun93] T. F. Lunt, "Detecting intruders in computer systems," *Conference on auditing and computer technology*, 1993.

[MeO01] A. J. Menezes, P. C. Oorshot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Fifth Edition, CRC Press, New York, 2001.

[Mis96] B. Misztal, "Trust in modern societies," *Polity Press, Cambridge MA*, Polity Press, Cambridge MA, 1996.

[SmW94] S. E. Smaha and J. Winslow, "Misuse detection tools," *Journal of Computer Security*, Vol. 10, No. 1, 1994, pp. 39–49.