

# GridAdmin: Decentralising Grid Administration using Trust Management.

Thomas B. Quillinan, Brian C. Clayton, and Simon N. Foley  
Department of Computer Science,  
University College, Cork, Ireland.  
{t.quillinan, b.clayton, s.foley}@cs.ucc.ie

**Abstract**—Administration of Grid resources is a time consuming and often tedious job. Most administrative requests are predictable, and in general, handling them requires knowledge of the local resources and the requester. In this paper we discuss a system to provide automated support for administrative requests, such as resource reservation and user account management. We propose using trust metrics to help judge the merits and suitability of each request. We outline how these metrics can be implemented using trust management techniques into a practical system we call GridAdmin.

KEYWORDS: Grid; Security; Trust Management; Decentralised Administration.

## I. INTRODUCTION

Grids [1], [2], [3] consist of numbers of sites cooperating to share resources. These resources are heterogeneous in nature and are maintained by a range of administrators, from full-time professionals to volunteers.

The purpose of these sites is to share their resources and knowledge throughout the Grid. Grid middleware such as Globus [1], [4] facilitates the sharing of these resources and provides an identity based security infrastructure using X.509 certificates. X.509 certificates provide authentication support for users submitting jobs to remote sites. However, this does not directly address the issue of administration across the Grid: It is difficult for an administrator to decide how to react to a request from a different site, or to know if an executable or configuration file from another site should be trusted. This is a hinderance to resource sharing.

Facilitating inter-site administration requires the definition of policies and some knowledge of each principal in the Virtual Organisation (VO). This is a major overhead for system administrators. In this paper we propose an automated system to support system administrators. In such a system, a request from a well-trusted administrator at a different site would be approved automatically, whereas a request from an under qualified user (for example, a student requesting a resource reservation) would require further investigation.

It is also important to consider how automation of user requests, software installation and upgrades, resource reservation, both within, and across sites should be achieved.

### Local Problems

At any site, administrators face several basic issues. Users need accounts to operate in that site. Each user may have a

different software requirement, and will need their specialised software installed on each resource they intend to use. Users may wish to request exclusive use of the resource, for example for critical timing, or simply to reserve access to the resource. The system administrator will handle these requests according to his/her knowledge of the user making the request, and within the constraints of the local policy.

*Example 1:* Site A has a policy which states that only postgraduate students and staff are allowed to make requests on their compute cluster (for accounts, software installation and for resource reservation and/or co-reservation). Priorities are assigned based on seniority of the requester and the urgency of the work. An undergraduate student could not be granted an account by their supervisor, but couldn't request a cluster booking: such a request would have to be brokered by their supervisor.  $\triangle$

### Cross-site Problems

When these local problems are translated to a Grid environment, they become more challenging. The same decisions must be made with less information available to the administrator.

For example, a user at site A wishes to use resources at site B, where both sites are in the same VO and have a functioning Grid. Currently, the procedure would be to send an email to site B's system administrator requesting that the attached executable file be installed. This would be accompanied by a configuration file used to set up the executable file. The administrator must now decide:

- does the user have a right to access resources on site B (covered under the terms of the VO agreement and handled by Globus);
- whether the user at site A is authorised to make such a request (perhaps local policy dictates that only senior staff members can make such requests);
- whether he should trust the executable file, or should the source code be consulted, and
- whether the configuration file is trustworthy.

It is easy to see that the ability to trust the user would greatly improve cross-site cooperation, and facilitate resource sharing. Requests from system administrators of different sites who have established relationships, and thus have some level of shared trust, are easier to accommodate. These decisions could be taken according to the established trust relationship and in terms of the local policy.

In the absence of a trust model, it is tempting to take a authorise all, or authorise none approach, to requests from outside the administrator's domain. These approaches may hinder the spirit of the agreement and reduce inter-site cooperation.

System administrators at different sites in a Grid tend to talk to one another, even if only to exchange the bare information required to set up a Grid (for example: machine names for firewalls; usernames for grid mapfiles etc.). For this reason, some level of trust (or history) is established between them. This trust can be leveraged, along with the constraints of the local policy, and the VO agreement to interpret requests. This is rarely the case with ordinary users. It is more likely that administrators will have little or no knowledge of individual users from other sites, but will have established relationships with their administrators. The user can normally only prove their identity and their right to access the resources under the VO agreement through the Globus X.509 security system.

This paper proposes to use Trust Management [5], [6] to manage the trust relationships between Grid administrators and Grid users. Trust Management is an approach to constructing and interpreting the trust relationships between public keys that are used to mediate security critical actions. Cryptographic credentials are used to specify delegation of authorisation among public keys.

Trust Management is used to help automate administrative decisions rather than replacing the existing Globus security infrastructure. The contribution of our approach is to provide a framework in which Grid administration becomes more practical. In this paper we explore two approaches —Explicit and “fuzzy”— to supporting Trust Management in Grid Administration. Explicit delegation of authorisation requires full authorisation details to be encoded within the Trust Management credentials. However, it does not capture the flexible nature of a real system. To this end, we propose alternative metrics to provide a means to make “fuzzy” delegations. These allow administrators to quantify the level of trust they apply to each of their users.

The rest of this paper is organised as follows: Section II introduces Trust Management, in particular the KeyNote trust management system. Utilising these techniques to specify trust metric supporting automated grid administration in decentralised manner is examined in Section III. Implementing these concepts in a practical system, WebCom, is discussed in Section IV. Finally, in Section V, we discuss the results and draw some conclusions.

## II. TRUST MANAGEMENT

Unlike identity based authorisation systems, such as those using X.509 [7] certificates, where authorisation is based on linking a detailed identity to a public key, Trust Management addresses the need to associate abilities to public keys. In other words, identity based certificates answer the question “Who is the holder of this public key?” whereas ability based certificates answer the question “What can I trust this public key to do?”.

Conventional identity-based secure applications verify that the certificates presented have not been revoked, and are signed by a recognised and trustworthy source. The names are then extracted from the certificates and a database is queried to determine if the requested action is authorised. This is cumbersome and aspects, such as the database lookup, are outside of the scope of the certificate system. Furthermore, there is the problem of determining the correct identity of an individual: there may be more than one John Smith in a particular organisation[8].

Trust management systems eliminate the extraction of names and database lookup. The certificates holding the abilities of the public key requesting the action are instead submitted to the trust management system, along with the action request. The trust management system verifies that the action is authorised by the certificates provided. For the purposes of authorisation, trust management systems are not concerned with verifying personal identity of a requester. These questions, while valid security questions, are not relevant to an application attempting to determine whether an action is authorised.

Trust Management systems have a number of advantages compared to the traditional systems created using X.509. Policies and certificates are created and maintained separately from the application in a very natural way. The attributes used within the policies/certificates are application defined, and they are represented in a free-form fashion, allowing the application designer to decide what characteristics are required. Changing the format of the attributes does not require changes to the trust management system used. By removing the traditional lookup of an identity's authority, and instead representing that authority within the certificate, applications no longer need to consider the security of where this authority is stored. An additional benefit of utilising a trust management system within applications is that designers and implementers of these applications are required to consider trust management applications explicitly. This in itself encourages good practices when considering the overall security of such applications. Trust management policies are easy to distribute across networks, helping to avoid reliance on centralised configuration of distributed applications.

KeyNote [9], [10] is an expressive and flexible trust management scheme that provides a simple credential notation for expressing both security policies and delegation. A standard KeyNote Application Programming Interface is used by an application to make queries about whether security critical requests (to the application) have authorisation or not. The formulation and management of security policies and credentials are separate from the application, making it straight forward to support trust management policies across different applications. KeyNote has been used to provide trust management for a number of applications including active networks [11] and to control access to Web pages [12].

Figure 1 illustrates how Trust Management can be integrated

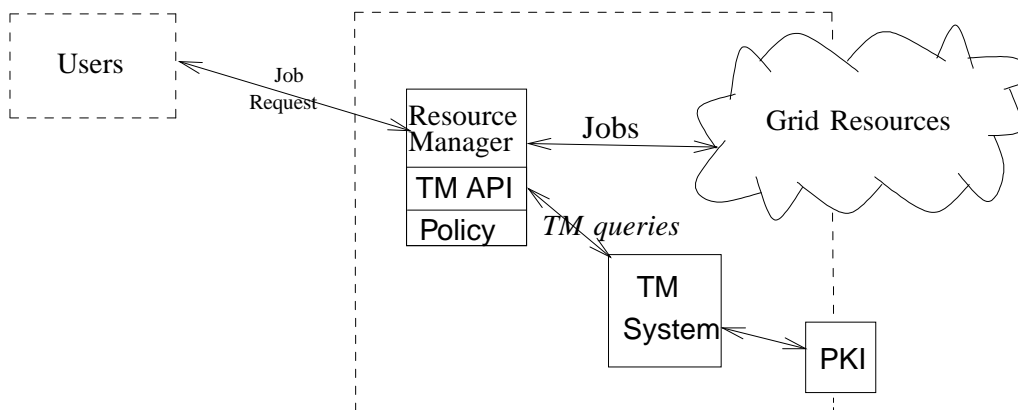


Fig. 1. A User-centric view of Trust Management on the Grid

into a Grid environment. Users send job requests to the resource management server (such as Globus). A job request is accompanied by the user's credentials. The resource manager uses the Trust Management system to determine whether the requester has sufficient authority; if this is the case, the resource manager executes the request on its grid resources.

When a request from an untrusted principal (key) is made to a networked application to execute a particular action, then, authentication notwithstanding, the application must determine whether the key(s) that made the request is authorised. Authorisation comes in the form of digitally signed public key credentials that bind public keys to the authorisation to perform various actions. In practice, authorisation is achieved by a collection of credentials that exhibit the necessary trust relationships between their keys. Given a policy (public keys, trusted in known ways), and a collection of credentials, a network application must determine whether a particular public key is authorised to request a particular operation.

*Example 2:* A Grid scheduling application conditionally accepts work from users. Operation `schedule` is used to request the scheduling of a job onto the Grid, while the request `cancel` is used to cancel a job. We expect that, in practice, a user will have the authority to schedule work to the Grid, and administrators will have the authority to both schedule and cancel jobs; this authority will be delegated by the operators of the individual resources.

Assume that the owner of the public key `KUCC-Admin` is trusted to administrate the Grid resource, `UCC-GRID`. This is specified by the following KeyNote credential.

```

Authorizer: POLICY
licensees: "KUCC-Admin"
Conditions: App_Domain == "GridAdmin" &&
             Resource == "UCC-GRID" &&
             JobManager =~ "Globus|MPI" &&
             (Operation == "schedule" ||
              Operation == "cancel");

```

Fig. 2. Policy Credential allowing Administrator UCC-Admin to schedule and cancel jobs

Figure 2 is a special *policy* credential that defines the

conditions under which requests from the licensee key `KUCC-Admin` may be trusted by the job managers `Globus` and `MPI`. These conditions are defined using a C like expression syntax in terms of the *action attributes*, in this example, `App_Domain`, `Resource` and `Operation` which are used characterise the circumstances of a request.

The owner of public key `KUCC-Admin` (The resource administrator) has the authority to delegate this trust to other keys and does so by signing the credential shown in Figure 3 for a user who owns public key `KALice`.

```

Authorizer: "KUCC-Admin"
licensees: "KALice"
Conditions: App_Domain == "GridAdmin" &&
             Resource == "UCC-GRID" &&
             JobManager == "Globus" &&
             Operation == "schedule";
Signature: ...

```

Fig. 3. Credential allowing User Alice to schedule operations on the GRID

In signing the credential in Figure 3, authoriser `KUCC-Admin` delegates authority for scheduling jobs to the key `KALice`. When Alice attempts to schedule a job (sending a request signed by `KALice`), she presents this credential as proof of authorisation. We can confirm that this key is indeed authorised since, by default (policy), we trust `KUCC-Admin` to schedule and delete jobs and `KUCC-Admin` has delegated some of this trust to `KALice`, by virtue of signing the credential.  $\triangle$

The delegator of a credential can also attach additional constraints, such as a prohibition on further delegation. An application may use a Trust Management (TM) scheme such as KeyNote [10] to determine whether requests to it are authorised, without the application having to know how that determination is made.

*Example 3:* When the `GridAdmin` application (Example 2) queries the KeyNote TM system to determine whether it is safe to execute a particular request, it must specify the circumstances of the query. These circumstances include: *action authorizers*, corresponding to the key(s) that made the request; *action attribute set*, which is a set of action attribute

name and value pairs that characterise the request; *policy* credentials, representing the keys that are trusted, and other *credentials* as provided by the requester and/or PKI.

For example, when KALice requests to schedule a job then the Grid Scheduler (Globus) queries KeyNote with action authorizer KALice, action attribute set:

```
{App_Domain ← "GridAdmin",
Resource    ← "UCC-GRID",
JobManager  ← "Globus",
Operation   ← "schedule"},
```

the policy credential for KBob above, and a set of signed credentials provided by Alice. KeyNote must determine if the given request is authorised based on the evidence provided. The application interacts with KeyNote via calls to the KeyNote API.  $\triangle$

The KeyNote architecture provides a level of separation between the provision of security policy authorisation and application functionality. As a software engineering paradigm, techniques that support separation of concerns for security [10], [13], synchronisation [14], and so forth are desirable since they lead to applications that are easier to develop, understand and maintain.

### III. TRUST METRICS FOR GRID ADMINISTRATION

The previous section outlined how a Trust Management system can provide a basis for a decentralised security administrative infrastructure in the Grid. Such a system has the ability to make authorisation decisions about user requests. Considering the problem of how to administrate islands of resources on the Grid, we can readily recognise the advantages of using Trust Management credentials to drive administrative actions.

Analysing the administrative problem further we identify three common administrative transactions:

- 1) Adding a remote user to a local Grid resource. Users in remote systems often request access to local systems. The local administrators may have no personal knowledge of the remote user, and are forced to make blind decisions regarding the user's eligibility and access level;
- 2) Providing the ability to book exclusive resource allocations. Users often require exclusive access to resources for diverse reasons. The administrator makes decisions regarding these requests based on such considerations such as past behaviour. For example, when the user last had exclusive access, did they use it properly?, and
- 3) Providing an infrastructure for users to request custom software installation. Administrators make decisions on new software installations again based on the requesting user's past performance and the skill level of that user; An experienced user will often receive a more positive response than a novice!

There are several potential approaches to solving these problems with a Trust Management framework.

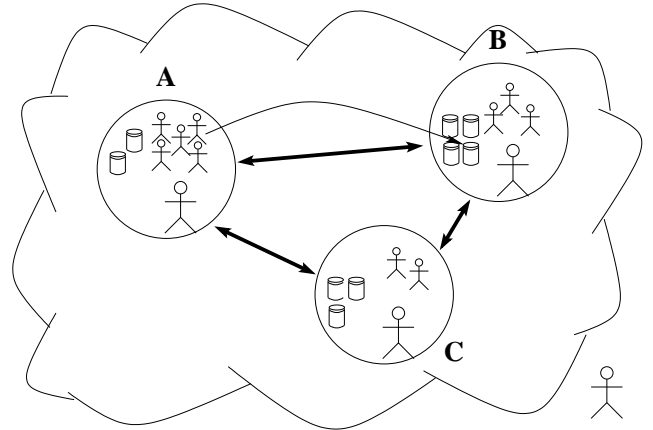


Fig. 4. A Virtual Organisation, with three organisations sharing resources.

#### A. Reputation Based Metric

Administration of different domains, such as the VO shown in Figure 4, rely on informal relationships between the administrators of those domains. Formalising these relationships into a model would provide a more consistent outcome for user requests. An analogy can be seen in the relationships between nightclubs in a locality. In general they are competing businesses. However if a person misbehaves in one club, their reputation often proceeds them to the other clubs in the area, through the “network” of doormen. This is a model well suited to the administration of Grid resources.

Using reputation based metrics for measuring trust is a well established technique [15], [16], [17]. Analysing the Grid architecture in order to use reputation to promote data integrity has previously been explored [18]. Knowing the reputation of a user can provide an insight into what access you give that user. Maintaining a measure of each user's reputation allows an administrator to make decisions about allocating the resources of the system to those users. We call this measure a user's “Karma”. Karma is a numerical value representing the level of trust that a user has attained in the local system. This value represents the user's previous behaviour in the system. The higher the value, the greater the user's potential access. Applying a numerical weight to users allows creation of more user-understandable policies. For example, depending on the karma level of a user, automatic decisions regarding access to resources may be made.

We envision karma to be represented by a value between 0 and 1. Users could potentially receive their initial karma level ( $K_{d_i}^u$ ) depending on their introducer's karma in the local domain ( $K_{d_i}^i$ ) and the user's karma in the introducer's domain ( $K_{d_i}^u$ ). An introducer is an authority in an associated domain, trusted to some level by the authorities in the local domain. For example:

$$K_{d_i}^u = K_{d_i}^i * K_{d_i}^u$$

Over time, depending on the user's behaviour, karma will rise and fall. Good behaviour, such as properly using reserved resources, is rewarded with increased karma, and therefore

```

Authorizer: "KUCC-Admin"
licensees: "KBob"
Conditions: App_Domain == "GridAdmin" &&
           Resource == "UCC-GRID" &&
           karma = 0.52;
Signature: ...

```

Fig. 5. Karma Credential for User kBob.

```

Authorizer: POLICY
licensees: "KUCC-Admin"
Conditions: App_Domain == "GridAdmin" &&
           Resource == "UCC-GRID" &&
           (node_request > 5) -> (karma > 0.6);

```

Fig. 6. Karma Policy, allowing conditional access to Compute nodes.

access. Consequently bad behaviour, such as requesting new software, but not using it, will result in reduced karma.

Karma could be encoded into trust management credentials, such as the KeyNote credentials such as those shown in Section II. For example, Figure 5 shows a karma credential for user KBob. This credential, signed by an administrator of the UCC-GRID domain, sets his karma level as 0.52. The flexibility of this system is apparent when examining the sample policy shown in Figure 6. This policy indicates that if a user wishes to reserve more than 5 compute nodes in the local domain, their karma must be over 0.6. Additional conditions could be placed in this policy such as, if a user was below 0.6, they would need to provide a request co-signed by another user.

Other usage of karma could include assigning a karma level to machines, based on their setup. This would allow the creation of policies where machines that are very stable (high karma) would be reserved for users who also have high karma. When machines have problems, their karma drops. Consequently a high availability increases the machine's karma level.

Difficulties with a karma-based metric are in the administration of updates to the user karma values. How are changes to the user's karma level stored and enforced? Users will probably be willing to throw-away an old credential, where the replacement has a higher karma level. However getting them to use a new credential with lower karma is more difficult. This issue can be addressed in a number of ways. Expiry dates could form part of the user karma credential, forcing the user to obtain a new credential periodically to continue using the system. Unlike Certificate Revocation Lists (CRLs), this places the burden of proof of authorisation on the user [19]. Another solution would be to store changes to each user's karma in a central "Karma Server". This, however, introduces a single point of failure into the system, and does not have the advantages of a decentralised approach.

Furthermore we must consider how to handle multiple introducer credentials for a given user. How do we aggregate different karma levels from different introducers? Also should changes to the introducer's karma reflect on the karma of the user? An example of such a system can be seen in [20]. Finding adequate solutions to these problems is important in

order to create a useable system, and is a topic for future research.

### B. Monetary based metric

Using money as a trust metric has been growing in popularity in recent years. Applying monetary or insurance [21], [22], [23] terminology to trust decisions is appealing as the stakes involved in a system are readily understandable. Unlike the reputation based metric, a monetary based metric requires no storage of the changes in each user's fortunes: users take care of their own money.

In a monetary based system, money is exchanged between principals. To use a resource, an agreed sum must be paid to the owner of the resource. It is important that the trust mechanism has a low computational and administrative cost, and also that contracts between users must be both verifiable and subject to conflict resolution. Such a system can be implemented using a trust management system [24], [25], [26]. These systems work on the basis that either the payments act as electronic cheques, that are reimbursed later, or are used as a closed currency. In a closed currency system, payments take the form of coupons, traded for resources. Ideally principals must either "save up", or several principals must combine, to request an "expensive" resource. Such a system discourages bad behaviour, as the abuser will lose money in the transaction.

A difficulty with such a metric becomes apparent when considering problems experienced in economics. For example in [27], Krugman introduces the problems with Babysitter clubs, that are common in the US. In these clubs, each set of parents are initially issued a fixed number of seed coupons. These coupons are used to pay other parents when a babysitter is required. When a parent wants a night out, they spend a coupon and another parent babysits for them. However over time, the system collapses due to hoarding of coupons by parents "saving up" for a special occasion. Other parents noticing the lack of babysitting jobs also stay in, preferring to save their coupons for emergencies. Applying this behavioural result to the proposed metric, leads to the conclusion that similar problems may well be experienced.

Instead of a coupon based metric, consider instead a deposit based system. In such a system, a "promissory note" is signed by the principal requesting the resource. If they behave properly, the contract is returned after some period. However if abuse of the resource takes place, the owner of that resource cashes in the contract, reducing the future purchasing power of the principal. This is analogous to an insurance policy.

Seeding the system requires that a trusted source, for example a bank, must set limits on all the principals using the system. This can be achieved using a trust management system. Initially authorities in a Virtual Organisation are delegated a certain amount of credit. These authorities can then pass on portions of this credit to their local users.

*Example 4:* In the Virtual Organisation shown in Figure 4, there are three component domains, A, B and C. The KeyNote policy shown in Figure 7 assigns a credit of 1000 to KAngela, the Administrator of domain A's key.

```

Authorizer: POLICY
licensees: "KAngela"
Conditions: App_Domain == "GridAdmin" &&
           Resource == "UCC-GRID" &&
           Credit = 1000 &&
           Validity <= 200404312359;

```

Fig. 7. Administrator Angela is delegated a credit of 1000.

These credentials have a validity date, up to when the credentials are valid. These validity dates allow the legitimate reuse of the credit that a user holds, without requiring the administrator to explicitly return the deposits. KAngela can then delegate parts of this total to users in her domain. Such a delegation is shown in Figure 8. This credential

```

Authorizer: "KAngela"
licensees: "KBob"
Conditions: App_Domain == "GridAdmin" &&
           Resource == "UCC-GRID" &&
           Credit = 100 &&
           Validity <= 200404142359;
Signature: ...

```

Fig. 8. Administrator Angela delegates a credit of 100 to user KBob

delegates a credit of 100 to KBob. KBob could now use this credential to generate a contract, guaranteeing good behaviour when requesting a resource in domain B. As each of the domains would trust administrators in the other domains, such a contract would be honoured in domain B.  $\triangle$

The metric outlined is essentially the opposite of a reputation based metric. Good behaviour simply guarantees continual access to resources. Bad behaviour would result in default of the contract, reducing the amount of money available in the future. If a principal misbehaves, a conflict resolution process would be enacted. Using this process, the complainant would furnish the contract credential, and some proof of the bad behaviour. If the complaint is upheld, at the start of the next renewal period for the user credit credentials, the credit of the misbehaving principal would be reduced, and the credit instead issued to the complainant. Over time, if principals can show good behaviour, in terms of contracts successfully completed, their issuing authority could choose to raise their credit limit. This is comparable to a credit card company increasing the credit limit of a good customer.

There is a potential problem with such a metric. Due to the decentralised nature of the proposed system, double spending, or promising the same deposit to more than one principal, becomes possible. A principal could make guarantees in domains B and C using the same collateral. However, we propose that this is in fact a desirable characteristic. If a principal acts properly in both domains, the double spending will never become apparent. However if a default occurs in both domains, the digital signatures will prove the guilty principal, and a conflict resolution process would take over. Such a system will reward a principal who takes more risks, yet who's behaviour is good. Good behaviour is likely to be increased, as principals are risking potential disaster if discovered.

#### IV. TOWARDS TRUST MANAGEMENT SUPPORT FOR GRID ADMINISTRATION

We will now further examine the concepts discussed in Sections II and III, and apply them to the WebCom system. The WebCom system [28], [29], [30] is a distributed secure and fault-tolerant architecture that can be used to coordinate the distributed execution of functional components across a network. The system uses implicit program parallelism, separates the application program from the underlying computation engine and is efficient yet compact. The heart of the WebCom system is the Condensed Graphs computational model that it employs [31], [32]. Applications are coded as hierarchical graphs which provide a simple notation in which lazy, eager and imperative computation can be naturally expressed. A Secured WebCom environment [28] uses KeyNote to help manage the trust relationship with other Secured WebCom environments.

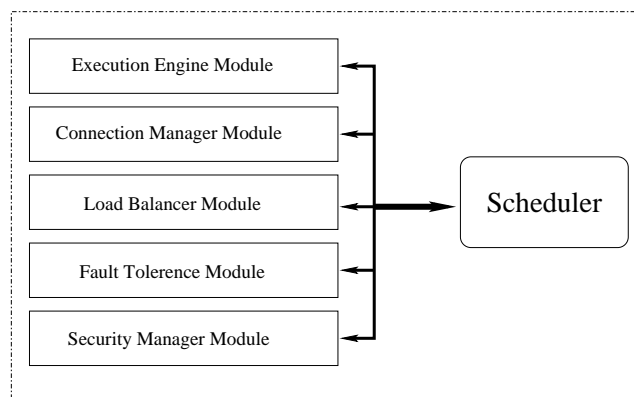


Fig. 9. WebCom Architecture Diagram

A WebCom environment [33], [34], [35] (Figure 9) is made up of several modular components: Execution Engine Modules, Connection Manger Modules, Load Balancing Modules and Security Manager Modules. The Scheduler initialises the required modules and handles communication between them. Execution Engine Modules take functional components and runs them; Connection Manager Modules handle communication between WebCom clients and servers; Load Balancing and Fault Tolerance Modules handle faults and balances the load over the clients of the server; Security Manager Modules check each executable component and ensure adherence to the local system security policy. Each WebCom server can have as many of each type of module as required; each is consulted where appropriate. For example a WebCom system might have two Execution Engine Modules, one handling Condensed Graph Applications and the other acting as a gateway to a Globus grid. Thus when a Globus job is uncovered, it is targeted to the Globus Engine Module.

##### A. Trust Management in WebCom

The default security architecture in WebCom utilises the KeyNote trust management system to ensure only authorised clients are scheduled secured components [28], [29]. Providing

support for either reputation based or monetary based metrics becomes a matter of performing additional authorisation checks. For example, when an attempt is made to reserve resources in a grid, the request takes the form of an authorisation check and should be accompanied by credentials supporting the request. However, these authorisation credentials can only provide an approximation of the metrics described. There are limitations, such as the need for a centralised karma server, and the conflict resolution protocols required for a monetary based system.

### B. Reputation based metric in WebCom

Supporting a reputation based metric in WebCom requires checking the requesting user's karma level, compared to the level required by the system policy for the requested administration action. If the user's karma is high enough, the request is accepted. If not, either a request for confirmation is made to an administrator, or, if the policy so dictates, the request is automatically denied. These administrative actions are specified as Condensed Graph Workflow applications.

*Example 5:* Reservation of Grid resources is specified in a Condensed Graph workflow application shown in Figure 10. When a user wishes to reserve such resources, this workflow is launched and the components executed on the relevant resources. This security policy of the environment, in which

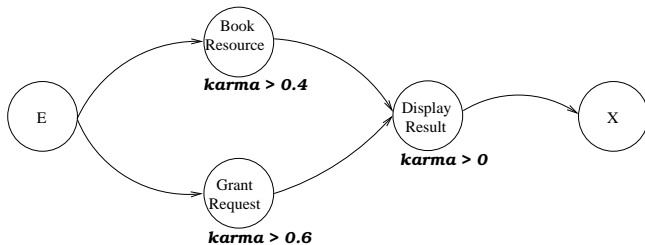


Fig. 10. Condensed Graph Workflow application to reserve a resource

the workflow application is running, defines that the Book Resource operation should be scheduled to a principal whose karma is greater than 0.4; The Grant Request operation is only allowed to be scheduled to a principal with karma greater than 0.6; The Display Result operation can be scheduled to any valid user. This policy is defined in a policy credential, such as in Figure 6.

KBob's credential (shown in Figure 5) indicates that he has sufficient karma to make a request to book a resource, however he would need someone else to approve that request, as he does not have enough karma to do so by himself. The result of the request would be displayed on KBob's machine, as he is a valid user of the system (i.e. his karma is greater than zero).  $\triangle$

Using this combination of the workflow abilities of Condensed Graphs, and the Security Managers of WebCom we can construct a flexible automated security administrator. However, this does not address the accounting problems with such a reputation metric.

### C. Monetary based metric in WebCom

Alternatively, supporting a monetary based metric in WebCom requires a different type of credential infrastructure. When a user wishes to make an administrative request, they create and sign a contract credential. This credential is then sent to the Administrator of the resource requested. If the Administrator accepts the contract, the request is granted. These decisions are taken based on the local policy of the resources requested. For example, if the policy stated the cost per minute of reserving a node, the user would have to offer at least this amount for the request to succeed. Even though this metric is in practice the opposite of the reputation metric: principals must prove their worth, the system is not required to maintain state; These administrative requests can be specified in the same form as those used with the reputation metric.

*Example 6:* Principal KClare wants to reserve 15 compute nodes, for 10 hours in order to generate some accurate results. In order to achieve this, she creates a contract credential, shown in Figure 11.

```

Authorizer: "KClare"
licensees: "KUCC-Admin"
Conditions: App_Domain == "GridAdmin" &&
            Resource == "UCC-GRID" &&
            Request == "BookResource" &&
            Nodes = 15 &&
            Time = 600 &&
            Deposit = 100 &&
            Validity <= 200404142359 &&
            [...];
Signature: ...
  
```

Fig. 11. KClare contract for reserving 15 compute nodes for 10 hours.

This contract credential allocates a deposit of value 100 to KUCC-Admin to guarantee KClare's good behaviour while using the requested compute nodes. For this request to be successful, KClare would have to provide a credential from a source trusted by KUCC-Admin, giving her the right to create such a contract credential. Figure 12 shows a credential fulfilling these requirements.

```

Authorizer: "KUCC-Finance"
licensees: "KClare"
Conditions: App_Domain == "GridAdmin" &&
            Deposit <= 250 &&
            Validity <= 200404312359;
Signature: ...
  
```

Fig. 12. Credit Credential from UCC's Finance Department, giving KClare's Credit limit.

This credential, signed by a key belonging to the Finance department in UCC, gives KClare the right to sign contracts up to value 250, in the GridAdmin application. Finally, KUCC-Admin's local policy must declare what price the Administrator is willing to accept for reservation of nodes. The policy must also trust the KUCC-Finance key for this request to be successful.

```

Authorizer: POLICY
licensees: "KUCC-Finance" ||\
           "KNUIG-Finance" ||\
           "KTCD-Finance"
Conditions: App_Domain == "GridAdmin" &&
           ((Request == "BookResource" &&
             (Deposit >= Time * Nodes * 0.01)) ||
            (Request == "InstallSoftware" &&
             (Deposit >= Nodes * 100)));

```

Fig. 13. KUCC-Admin’s policy, trusting the keys of several Finance departments to assign credit limits. It also dictates the terms acceptable to the Administrator.

Figure 13 shows such a policy. In this policy credential the administrator has defined the conditions under which certain administrative requests are acceptable. Specifically, in order to reserve nodes, principals must provide a deposit based on the number of nodes required and the length of time (in minutes), they are required for.  $\triangle$

This system can be extended to encompass all the administrative actions concerning the administrator. Placing a monetary value on the actions allows the administrator to discourage certain actions, without outright refusal. For example in Figure 13, the administrator has defined the value 100 as the price to install a new piece of software on each node. These conditions can be as fine-grained as the administrator requires. For example, reserving an SMP machine could be much more expensive than a uni-processor node.

Additionally, using the architecture of the WebCom system, we can implement the “Nightclub” model previously discussed. Using the communication capabilities of WebCom, advisory credentials, written by administrators, could be distributed throughout the system and integrated into the trust management decision. These credentials could specify that a higher “Entrance fee” is required from users who have misbehaved on other systems. This provides a means to instantly reduce the purchasing power of individual users, without waiting for the renewal of credit credentials. This concept is similar to the idea of Certificate Cancellation Notices (CCN) in SPKI [36]. CCNs are an informal version of Certificate Revocation Lists (CRLs) with most of the benefits, but at reduced cost.

These decisions take place in a fully decentralised manner. Different administrators have different priorities, and so the policies will vary from domain to domain. Another advantage of this decentralised architecture is the ability to “sub-contract” work. It would be possible that KUCC-Admin decides to sub-contract some work to another domain. This is achieved by the creation of a new contract credential by KUCC-Admin to another administrator, delegating the deposit from the received contract credential. The original contract credential would be passed along to preserve the delegation chain.

#### D. GridAdmin Architecture

Supporting these automated administrative requests for any metric, requires an administrative helper “daemon” running on the Grid resources. These agents take administrative requests,

such as node reservation, and perform the low-level changes to the resources. For example, a successful request to reserve compute nodes would call the agent in charge of those nodes, and modify the system to only allow that user to log in during the reservation period.

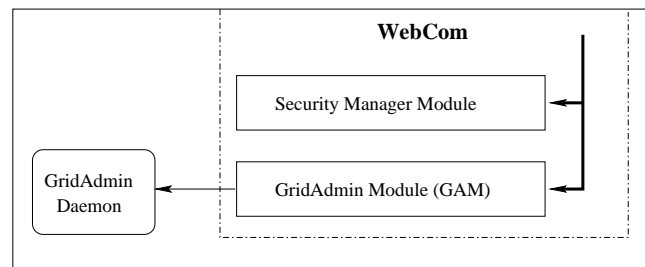


Fig. 14. GridAdmin Module and Administrative Daemon

Figure 14 shows a representation of the GridAdmin architecture. It is made up of two parts; a GridAdmin Module within the WebCom architecture and the trusted GridAdmin Daemon executing as the administrator on the machine. The daemon acts on requests from the module, making the required low-level changes to the system setup.

## V. DISCUSSION AND CONCLUSION

In this paper we have introduced GridAdmin, an automated administrator, empowered to handle the tedious administrative requests, such as the reservation of compute nodes, common in grids today. This system provides a “value-added” service to Grid administration, sitting on top of the existing Grid architecture rather than replacing the existing security architecture. For example, the cryptographic keys used to sign credentials are the same keys used by the principals to authenticate themselves to the Grid management software, such as Globus.

Explicit delegation of authorisation as described in Section II requires full authorisation details to be encoded within the Trust Management credentials. However, it does not capture the flexible needs of a real users. To this end, Section III proposes alternative trust metrics that provide a “fuzzier” notion of trust. Each approach was found to have advantages and disadvantages, such as problems of aggregation in the karma system and conflict resolution in the monetary system. How these schemes might be implemented within a practical Grid environment was considered in Section IV.

The metrics proposed encompass alternative ends of the possible design of such a system. However we believe that the monetary metric provides an interesting, yet useful simulation of the real-life situations administrators find themselves in. Often we ask ourselves: “Whats in this for me?; What guarantees do I have that this will not break our system?” These questions are addressed using a monetary system, with the cost/benefit analysis being readily understandable.

We are in the process of deploying such an automated system in regards to the Cosmogrid [37] project. This will reduce the time required to administrate, and increase both

flexibility and sharing of resources between the component sites.

In the future, we intend analysing both the usability of the GridAdmin architecture, and the suitability of the proposed metrics over time. The flexibility of a trust management based approach allow each site to alter their policies to suit local conditions, while providing a consistent infrastructure throughout the sites. Integrating some of the features of both proposed metrics into a combined metric also may provide some interesting results. More research into these metrics is required.

#### ACKNOWLEDGEMENTS

The support of both the Informatics Research Initiative of Enterprise Ireland and the Cosmogrid project are gratefully acknowledged.

#### REFERENCES

- [1] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997.
- [2] B. S. White, M. Walker, M. Humphrey, and A. S. Grimshaw, "LegionFS: A secure and scalable file system supporting cross-domain high-performance applications," in *SC2001: High Performance Networking and Computing*, Denver, Colorado, November 10–16 2001.
- [3] *The Message Passing Interface (MPI) standard*, <http://www-unix.mcs.anl.gov/mpi/>.
- [4] I. Foster *et al.*, "A security architecture for computational grids," in *5th ACM Conference on Computer and Communications Security*, 1998.
- [5] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of the Symposium on Security and Privacy*. IEEE Computer Society Press, 1996.
- [6] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, "The role of trust management in distributed systems security," in *Security Issues for Mobile and Distributed Objects*, J. Vitek and C. Jensen, Eds. Fourth International Workshop, MOS'98, Brussels, Belgium: Springer-Verlag Inc., July 1998.
- [7] CCITT Draft Recommendation, *The Directory Authentication Framework, Version 7*, Nov. 1987.
- [8] C. M. Ellison, "The nature of a useable PKI," *Computer Networks*, no. 31, pp. 823–830, 1999.
- [9] M. Blaze, "Using the KeyNote trust management system," <http://www.crypto.com/trustmgt>, December 1999.
- [10] M. Blaze *et al.*, "The keynote trust-management system version 2," Sept. 1999, internet Request For Comments 2704.
- [11] M. Blaze, J. Ioannidis, and A. Keromytis, "Trust management and network layer security protocols," in *Security Protocols International Workshop*. Springer Verlag LNCS, 1999.
- [12] "Apache-ssl release version 1.3.6/1.3.6," Open source software distribution. <http://www.apache.org>.
- [13] S. Foley, "A kernelized architecture for multilevel secure application policies," in *European Symposium on Research in Security and Privacy*. Springer Verlag LNCS 1485, 1998.
- [14] C. Lopes and K. Lieberherr, "Abstracting process-to-process relations in concurrent object-oriented applications," in *European Conference on Object-Oriented Programming (ECOOP)*. Springer Verlag LNCS 821, 1994.
- [15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigen-trust algorithm for reputation management in p2p networks," in *In Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*. Budapest, Hungary: ACM Press, May 20–24 2003.
- [16] J. Sabater and C. Sierra, "Social regret, a reputation model based on social relations," *SIGecom Exch.*, vol. 3, no. 1, pp. 44–56, 2002.
- [17] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust in peer-to-peer communities," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2004, to appear in Special Issue on Peer-to-Peer Based Data Management.
- [18] A. Gilbert, A. Abraham, and M. Paprzycki, "A system for ensuring data integrity in grid environments," 2004, to Appear.
- [19] R. L. Rivest, "Can we eliminate certificate revocation lists?" in *Proceedings of Financial Cryptography '98*, R. Hirschfeld, Ed., no. 1465. Springer Lecture Notes in Computer Science, February 1998, pp. 178–183.
- [20] *Advogato's trust metric*, <http://www.advogato.org/trust-metric.html>.
- [21] B. Schneier, *Managed Security Monitoring: Closing the Window of Exposure*, 2000, <http://www.counterpane.com/window.html>.
- [22] M. K. Reiter and S. G. Stubblebine, "Path independence for authentication in large-scale systems," in *Proceedings of the 4th ACM conference on Computer and communications security (CCS97)*. ACM Press, 1997, pp. 57–66.
- [23] J. K. Millen and R. N. Wright, "Reasoning about trust and insurance in a public key infrastructure," in *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, Cambridge, England, July 03–05 2000, pp. 16–23.
- [24] S. N. Foley and T. B. Quillinan, "Using trust management to support micropayments," in *Proceedings of the Second Information Technology and Telecommunications Conference*. Waterford Institute of Technology, Waterford, Ireland: TecNet, October 2002, pp. 219–223.
- [25] S. N. Foley, "Using trust management to support transferable hash-based micropayments," in *Proceedings of the 7th International Financial Cryptography Conference*, Gosier, Guadeloupe, FWI, January 2003.
- [26] M. Blaze, J. Ioannidis, and A. D. Keromytis, "Offline micropayments without trusted hardware," in *Financial Cryptography*, Grand Cayman, February 2001.
- [27] P. Krugman, *The return of Depression Economics*. WW Norton & Co, 1999, 176 pages.
- [28] S. N. Foley, T. B. Quillinan, and J. P. Morrison, "Secure component distribution using webcom," in *Proceeding of the 17th International Conference on Information Security (IFIP/SEC 2002)*, Cairo, Egypt, May 2002.
- [29] S. N. Foley, T. B. Quillinan, M. O'Connor, B. P. Mulcahy, and J. P. Morrison, "A framework for heterogeneous middleware security," in *Proceedings of the 13th International Heterogeneous Computing Workshop*. Santa Fe, New Mexico, USA: IPDPS, April 2004.
- [30] J. Morrison, D. Power, and J. Kennedy, "A Condensed Graphs Engine to Drive Metacomputing," Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA '99), Las Vegas, Nevada, June 28 - July 1, 1999.
- [31] J. Morrison, "Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and Control-Driven Computing," Ph.D. dissertation, Eindhoven, 1996.
- [32] J. Morrison and M. Rem, "Speculative computing in the condensed graphs machine," proceedings of IWPC'99: University of Aizu, Japan, 21–24 Sept 1999.
- [33] J. P. Morrison, B. Clayton, D. A. Power, and A. Patil, "Webcom-g: Grid enabled metacomputing," *Neural, Scientific and Parallel Computations Journal.*, 2004, to Appear.
- [34] D. A. Power, A. Patil, S. John, and J. P. Morrison, "WebCom G," in *Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*. Las Vegas, Nevada: CSREA Press, June 2003.
- [35] J. P. Morrison, B. Clayton, A. Patil, and S. John, "The information gathering module of the WebCom-G operating system," in *Proceedings of the Second International Symposium on Parallel and Distributed Computing (ISPDC03)*, Ljubljana, Slovenia, October 2003.
- [36] Internet Engineering Task Force, "Simple public key infrastructure [SPKI]," <http://www.ietf.org/html.charters/spki-charter.html>.
- [37] *The Cosmogrid Project*, 2004, <http://www.cosmogrid.ie/>.