

# TRANSACT

(Tool for Rational Automated Negotiation of Secure Authorisation Contracts)

Russell Lock (PA9)  
Computing Department, Lancaster University  
Email: r.lock@comp.lancs.ac.uk

*This position paper explores the issues surrounding authorisation in modern grids, focusing on the development and use of Globus middleware. In particular this paper focuses on the future needs of inter-organisational business relationships in comparison with the currently available infrastructure. Finally an approach to resolving these issues is provided building upon existing security mechanisms.*

## 1 Introduction

The ongoing adoption and commercialisation of grid technologies has increased the pressure to develop new service based grid architectures. As the applications for grids have increased in scope, grid research has evolved away from its distributed batch processing roots to provide a more complete extendable and modifiable environment. This evolving design can be seen clearly in the works of the Global Grid Forum (GGF<sup>(1)</sup>) and from individual middleware platforms such as the Globus toolkit. One of the most influential papers has been “The Anatomy of the Grid”<sup>(2)</sup> which laid out a service based model for grids. This in turn became the Open Grid Service Architecture (OGSA), upon which this project has been designed to build.

There is a growing realisation in the grid community that although research in the area is advancing quickly, there are some areas that have been neglected in the last few years, the most notable being that of security. From a Globus users perspective it has not changed since the move to the OGSA compliant v3 in early 2003. The security model v3 utilizes was designed for relatively small isolated grids working independently, preferably in an environment where its user’s were trusted. This situation is changing though, as the number of grids is growing substantially with increasing need for interoperability; and a user base that has grown beyond the limits of a simple trust relationship.

Many of the higher level issues relating to security are only now being addressed. For example the Globus v3 security model has been carried over from v2 with only minor alterations to proxy layout. Given that v2 was designed for isolated research settings it is clear that either extensions to the existing, or the implementation of alternate authorisation mechanisms is required to keep in line with VO (Virtual Organisation) grid development needs. TRANSACT concentrates on the area of negotiation with grid services, which has been overlooked mainly due to its reliance on the future development of grid services, to the point where economic considerations take over from the development effort. Automated negotiation becomes preferable when the complexity of the agreements between user and service reach the point where assistance is required to make a meaningful decision within a given timeframe.

### 1.1 Security in Globus

The security model within Globus v2 and v3 (OGSA) relies on X.509 certificates for authentication and authorisation purposes. X.509 certificates represent a public key infrastructure (PKI) which has proven very useful in grid and specifically Globus installations. The standard Globus toolkit installation allows authentication based on the use of X.509 keys, and authorisation based on an Access Control List (ACL) entry at a given service. This entry must then match a unique identifier encoded in the X.509 certificate itself. This provides rudimentary access control at a very coarse grained level: that of allowed access or

denied access. This security model was designed with research environments in mind, where a limited number of people, perhaps widely geographically spread, could utilize a given installation. Though most installations manually enter information into the lists this does not represent a huge problem. The automation of this part of the system would be relatively straightforward allowing scalability through good design; indeed there has been some work completed in this area including the development of a Community Access Service (CAS<sup>(3)</sup>) by the original Globus team. The more interesting issues however relate to the access/denial authorisation policy. This authorisation constraint leads to two prevalent design decisions common to Globus installations:

- Where no further constraints can be placed on a users' access to a grid once permission has been granted. This policy is unsuitable for a service based access paradigm where multiple services could run from the same grid node, possibly with an economic element of cost involved.

Or

- The use of high level behind the scenes access control lists detailing what can and cannot be accessed for each individual user. This method allows fine grained access models but requires considerably more information be stored to check for authorisation. It has issues relating to single points of failure, replication needs, dependability and reliability issues.

There are however other ways around this issue, many of which are covered in the next section. The grid development community itself has also not been blind to this situation, and more advanced systems have been designed, including CAS which addresses this problem partly through the use of Role based access controls (RBAC).

## 2 Alternate authorisation mechanisms

A number of alternate mechanisms exist for allowing access to a given system; many of which have been adapted for use in grids. The following section contains a brief overview of the main types.

### Token based (limited time span)

The most popular form of token based authorisation mechanism used by Globus users is Kerberos<sup>(4)</sup>. It represents the only freely available authentication / authorisation alternative to the existing supported X.509 mechanism. Current support is limited by versioning in that only certain existing Kerberos installations can be used, but there is considerable development effort going into increasing support for the interoperability of X.509 to Kerberos interaction and vice versa.

### RBAC (Role Based Access Control)

RBAC systems have never gained more than a niche in the market due to a variety of issues including:

- Revocation of Access to a system
- Creation of suitable "Roles" for a given situation
- Overlap between Roles with regards to access of parts of the system

They are however ideally suited to situations where the roles of users are clearly defined. For example many of the most successful utilisations of RBAC have been in hospitals, where jobs have carefully demarcated roles.

### Hybrid approaches

Some of the more popular solutions have been hybrids like Akenti<sup>(5)</sup>, which relies on the interpretation of a number of distinct certificates. Each of the three types is concerned with a different aspect of the negotiation. Policy certificates which outline stakeholders; use condition certificates which specify the user requirements for access and Attribute Certificates which specify attributes the users can prove they have.

### Current Usage of contracts

Issues relating to authorisation and use of services are generally dealt with in the outside world through contracts and Service level agreements (SLAs). Previous attempts to translate these heavily worded documents to allow Quality of Service (QoS) specifications within computer systems have met with limited success. A considerable amount of research into the dynamics and phases of electronic contract negotiation exist. Some of the more notable works include Research on B2B contracts by Goodchild<sup>(6)</sup> which deals with many of the issues relating to storage of contracts in XML format. Also work by both Angelov<sup>(7)</sup> and Daskalopulu<sup>(8)</sup> which examine the use of computers as tools (both automated and passive) to create contracts as well as highlighting the projects presently underway to develop tools for the future.

The SNAP<sup>(9)</sup> protocol designed at Argonne Laboratories alongside Globus relies on SLAs to allow QoS specifications to be addressed. However it takes a hierarchical view with a series of interconnected SLAs built up between resources. Currently the model recognises the need for negotiation and lays out a language for specification, but does not address the specifics of negotiation at this time. Interestingly it also assumes relative trust between parties and currently does not as yet deal with the economic cost element of contracts.

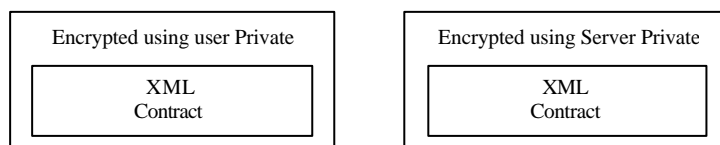
The area of grid computing only encompasses a small subset of the area general contracts serve. This avoids many of the problems that expert systems and automated agent based contract negotiators have had in the past due to lack of domain knowledge. Therefore it is possible to conceive that by simplifying the contracts themselves beyond that of Goodchild's model whilst retaining the use of XML as a storage medium; coupled with a rule based language similar to that laid out in SNAP it would be possible to automate the process of grid contract creation. Such a system would rely on negotiation between contract negotiating software on both sides, with input from users only at the beginning and end of the process, removing the need for slower manual asynchronous negotiation techniques. The mechanisms needed to allow a user to utilize this type agreement for authorisation are shown in the next section, starting with the securing of a given contract for authorisation.

## 3 Contracts for grid authorisation

### PKI Contracting

Globus authenticates using public key X.509 technology (PKIs). PKIs secure information based on mathematical formula that allow easy conversion into a form which can only be converted back using a different key, and vice versa. This allows a person to encrypt something using a person's public key knowing it can only be decrypted using that person's private key. Reverse this approach and you can prove a person encrypted a file using the private key by decrypting using the public one. Presuming the private key is kept private this can be used to prove persons credentials. Globus uses this technology for authentication, in TRANSACT it is also used for encryption of the contracts themselves. The basic structure of this is illustrated below.

Fig 1:



If both contracts match when decoded, they both agreed to the same contract. These can be authenticated by anyone with the public keys of both parties.

This model of authorisation requires a considerably more lightweight server end authorisation mechanism to process incoming requests due to its reliance on the actual information received (the contract) rather than a database of individual permissions stored locally. This allows for easier replication of authorizing mechanisms without the need to control multiple copies of databases.

### **Automating the Negotiation process**

The negotiation process is based around these XML encrypted contract instances. The contract model itself works on a request / reply basis with the user making demands and the service counter demands. From each of these contract iterations decisions are made by the user side automation on how to increase the acceptability of a given offer. The request / reply circle then continues.

The contract iterations are split into two sections, a header and main section. The header contains compulsory information regarding the basic needs of a contract. For example, the names of the parties, the validity and expiration date of the agreement etc. The main section contains all the field instances that are being negotiated upon, for example Auto\_Resubmission (on failure). These fields can contain either a numerical or textual description of the values being negotiated. The user can extend the main section to include any fields they wish, presuming they are supported by the service they wish to negotiate with.

Each field has a name, a value, a description and definition. The latter two are used to ensure the field can be linked to a textual description of its meaning and a definition of where that description was obtained respectively. The description helps reduce the ambiguity factor in human understanding whereas the definition helps avoid clashes of field names between institutions. An example of this is shown below:

Field:

Name:	Payment
Value:	50
Description:	In £. "Payment" means the price for the goods excluding carriage, packing, insurance and VAT.
Definition:	Standard_Contracting_A1.txt

### **User Controls**

The user controls operate under the premise that users should be removed from the negotiation itself as far as possible, with input only at the beginning to indicate acceptable bounds and needs, and to accept or decline a contract at the end. This way a truly autonomous system can be achieved without user related bottlenecks once the process has commenced.

The following sub sections are split into the different types of contract manipulation mechanisms, complete with explanations on use and screenshots where appropriate.

#### Bound Specification for numerical fields

A numerical type field must range between an upper and a lower bound. If for example the field were "cost" the number would doubtless be in pounds but the sign would be left out. The denomination type would be indicated in the description field. An example can be seen below.



A screenshot of a user interface element for specifying a numerical field. It consists of a grey rectangular box containing the text "Cost" followed by a small input field containing the number "0", the word "to", and another small input field containing the number "100".

Option Specification for Textual Fields

The textual fields operate on a simple pop up list from which a user can select a given option. Currently the negotiation engine will work off this one value, but a simple future extension to the system could be to allow users to rank options to indicate a preference between them. The lists are designed to hold different values dependant on the needs of the individual field. An example of this field type is shown below:



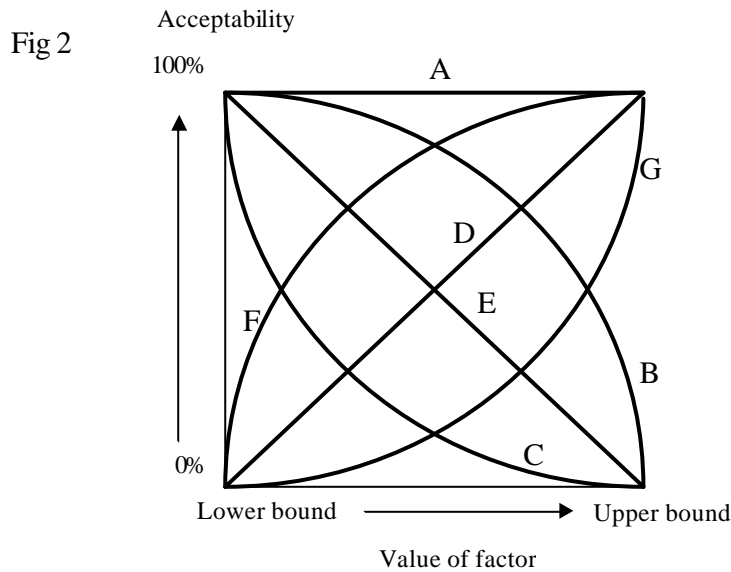
Bias

Bias can be entered into the system by a series of interlinked slider bars attached to each field. The values on the sliders individually add up to 100%, giving each of the sliders a percentage of importance in calculations. The sliders can be locked in place to avoid movement allowing a user to easily manipulate the levels of importance ascribed to the individual fields. An example can be seen below.



Pictorial references of acceptability

The final set of user options are a set of pictorial representations, of a series of preset formulas. These represent how acceptable a user would find a given value within the range of a given numerical field. There are 7 of these in total representing 7 different scenarios. These cannot cover all expected user needs but they provide a rough approximation of any given numerical scenario. Fig 2 shows a graph representing the different pictorial types with explanations as to their specific properties.



Key:

- A This straight line would represent a situation where all acceptable bounded values elicit a 100% acceptability rating. In layman's terms this means every value between the bounds is fine by the user.
- B This curve represents a situation where the user finds a low value more acceptable than a high one. They also become disproportionately dissatisfied as the value increases.
- C This curve is the opposite pair of B, where the user finds a high value most acceptable and becomes disproportionately unsatisfied the closer to 100% it is. This curve at first appears an unlikely scenario. It is designed to make the system keep trying for a higher value if it is nearly at the 100% mark.
- D A direct correlation between value and acceptability where a higher value is better
- E The inverse of D with lower values eliciting more acceptability
- F This curve represents the situation where a high value is better and the user finds the solution disproportionately more acceptable with small increases at the bottom of the range
- G This curved line represents the opposite of F where small increases at the bottom of the range have less effect than similar ones as the acceptability rating



### Logical Rule Input

The logical rule input is used to enter any specific interdependencies into the system prior to negotiation. The format uses keywords, values and numbers grouped into appropriate bracketed expressions to avoid ambiguity in the processing of user instructions. The list of currently supported keywords is listed below.

IF	Test whether something has occurred
THEN	Do something specific based on IF
AND	Link between two arguments
OR	Link between two arguments
MUST	System must make this happen
MUSTNOT	System must not allow this to happen
SHOULD	System should comply with the command, if it is feasible
SHOULDNOT	System should not let this happen, but could if deemed necessary
WEIGHT	Used to raise and lower the importance of the given field in calculations

The system also understands the concepts of  $>$ ,  $<$ , and  $=$  for purposes of IF statement tests. For simplicity the system does not support  $\geq$  or  $\leq$  as these can easily be simulated using the  $>$  and  $<$  signs. The system uses brackets to separate arguments to avoid ambiguity which other wise would have been caused by statements like:

IF  $A > B$  OR  $B < C$  AND  $E > F$  THEN ...



```

If(mirroring issues have arisen)
{
    Check if situation can be avoided by not implementing logic rules using the “SHOULD”
    & “SHOULDNOT” keywords.

    If it can
        Don’t implement these specific rules (to try to break deadlock)
    Else
        Randomise field values. (Normally a change of roughly 20% in figures within
        the bounds acceptable)
}

If (outliers exist)
    Correct outliers (An outlier is merely a value outside acceptable bounds)
Else
    Detect 4 most unacceptable field values and alter in accordance with user requirements**

Recalculate the cost the user is prepared to pay for the service based on the aggregate acceptability
of the new offer. ***
}

```

Reply with new offer

\* Mirroring issues occur when the data on either side of the negotiation is an exact mirror image. For example if one side ranged 0100 on a diagonal upwards curve and the other 0100 on a diagonal downwards curve. This is a somewhat simplified explanation as more factors are taken into account when making offers but the situation leads to one side making an alteration and the other changing it back. This is recognised and corrected first through checking of the logic rules then, if all else fails through the randomising of some field values by a proprietary 20%.

\*\* The amount the values are changed dynamically based on the acceptability rating. For example if something was right near the bottom range for acceptability it would receive a bigger increase in change than one that was more acceptable. These changes range from a 1-10% increase on the current value.

\*\*\* The service will ignore the cost field to a certain extent. It is not counted as an active negotiation field but is calculated based on the circumstances and values of the fields in that iteration of the process. The value chosen is important in that a given service is programmed to reduce costs based on reasonable demands by the user. By implementing this approach users and services get better deals on both sides if their field dynamics and values are in harmony. i.e. they both think along the same lines.

On the service side negotiation strategy is represented by two options. A fixed option under which the service will provide the user with what they want (provided it is within the services bounds) or a variable bartering mode in which both sides use the algorithms shown above. In the fixed mode the only way the service can persuade a user to accept a contract offer the service prefers is to raise prices accordingly on other offers. Using the variable bartering mode the results are considerably more erratic (as the user will

never seem to get exactly what they asked for) but is representative of a more advanced model of negotiation. The bartering mode still requires substantial testing and tweaking to ensure both sides hone in on their Nash equilibrium quickly and efficiently.

## 5 External factors

In order to concentrate upon the core negotiation process shown above some parts of the architecture required to allow this type of system to operate have been left as external issues. These include the need to provide for revocation of contracts, mediation procedures for failed contracts, brokers to locate services and repositories for public keys.

## 6 Conclusions & Future work

The model shown above is currently at the prototype stage. Future improvements could include the ability for services to take into account system load when making offers; and more advanced negotiating algorithms which rely on the analysis of previous iterations to deduce trends rather than purely on an incremental improvement model. There is also scope for improvement in the use of textual field values by allowing ranked preferences.

## 7 Acknowledgements

I would like to thank the UK Engineering and Physical Sciences Research Council, grant number GR/M52786 and the Dependability Interdisciplinary Research Collaboration (DIRC).

## 8 References

- [1] Global Grid Forum (GGF)  
<http://www.gridforum.org>
- [2] Anatomy of the Grid  
Foster I, Kesselman C, Tuecke S. Anatomy of the grid. [www.globus.org/research/papers/anatomy.pdf](http://www.globus.org/research/papers/anatomy.pdf)
- [3] CAS  
<http://www.globus.org/security/CAS/GT3/>
- [4] Kerberos  
<http://web.mit.edu/kerberos/www/>
- [5] Akenti  
<http://www-itg.lbl.gov/Akenti/>
- [6] Goodchild  
Goodchild, A. Herring, C. Milosevic, Z. Business contracts for B2B  
<http://titanium.dstc.edu.au/papers/ISDO00.pdf>
- [7] Angelov  
Angelov, S. Grefen, P. B2B eContract handling.  
<http://www.ub.utwente.nl/webdocs/ctit/1/0000005e.pdf>
- [8] Daskalopulu  
Daskalopulu, A. Sergot M. The Representation of Legal Contracts  
<http://arxiv.org/ftp/cs/papers/0106/0106005.pdf>
- [9] SNAP  
Czajkowski, K. Foster, I. Kesselmann, C. Sander, V. Tuecke, S. SNAP.  
<http://citeseer.nj.nec.com/538954.html>