Message-Based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services

Geoffrey Fox, Sang Lim, Shrideep Pallickara, Marlon Pierce
Community Grids Lab
Indiana University

Abstract

We examine the creation of peer-to-peer Grids in which autonomous Grid components ("Gridlets") may be federated into dynamic Grid collections. We examine two important aspects of these systems: federated security and autonomic considerations. As we discuss, both may be implemented using a messaging system as a substrate. Messaging systems provide the means of bridging Gridlet realms, organizing access control areas, and providing autonomic building blocks like discovery, reliability, resilience, and robustness of the peer-to-peer Grid..

Keywords: grid federation, publish/subscribe systems, reliable invocation and messaging, access control

1 Introduction

Significant effort has been devoted over the last decade to create the basic infrastructure needed for Grid computing [1], yet many research issues are still open and much development work remains to be done [2]. In this paper we focus particularly on the need to support collections of grid peers as a solution to problems related to fined grained access control, reliability, and fault tolerance. We propose a Cellular, or Granular, Grid in which one composes a Grid out of smaller Grid components, here called Gridlets, into a peer assemblage [3].

Gridlet components are not necessarily individual computing resources as in the peer-to-peer (P2P) case but are instead self-contained Grid systems that may have several resources and services. We define a "gridlet" as a self-contained Grid deployment built in the standard fashion: say, a particular version of the Globus Toolkit installed by a particular (real or virtual) organization on a collection of distributed computing resources. A Grid installation in a particular university department or government lab would be an example of a gridlet. A gridlet is defined not just on a technical level, but on an organizational and policy level as well: a typical gridlet will have a long-term, well-defined usage, security, and maintenance policies.

We seek to address the problems from assembling larger grids from these stable gridlet components. The Cellular Grid is intended to provide the infrastructure to make Gridlets, or subsets of their resources, available to highly dynamic virtual organizations: a particular grid installation may be long lived, but it may need to devote some of its resources to multiple virtual organizations through federation. We describe here the architecture for a system that will simplify this grid resource federation. To achieve this

federation, we are implementing a message-based substrate for grids, from which we may derive additional benefits such as reliable message delivery and robust services.

This cellular grid arrangement is shown in Fig. 1. The Gridlets form cells that are linked by a messaging system which can, as we describe later, enforce security, federation and mediation services at cell boundaries. We thus compose all Grids out of smaller Grid cells and then dynamically link these cells, plucking resources from different Grids into dynamic overlays. As we will discuss, this architecture naturally provides service management (service discovery, reliability, fault tolerance) that can for example provide one approach to Autonomic Grids.

We are building a prototype of these ideas --*InterGrids*--that will support lightweight, dynamic Grids that can be composed into larger distributed systems. If the cells are individual computers and the messaging is that used in systems like Gnutella, Cellular Grids become "just" P2P Networks. If a Grid has but one (large) cell, then one gets the "Classic Grid". In the discussion of this paper, we will use our internally developed messaging system, NaradaBrokering, but this is of course not required in general. We note that NaradaBrokering can support either centralized messaging (appropriate for classic Grids) like traditional JMS (Java Message Service) or P2P architectures like JXTA.

We are motivated to pursue Cellular Grids because we see it as inevitable that Grids will be established as islands of resource collections that will need to be federated at later dates in lightweight manners. These islands of resources may result from a number of reasons. First, a group may set up a Grid of resources that it owns and directly controls but must overcome sociological and /or organizational barriers for interoperating with other Grid installations, even when the same software is used. Second, we see the emergence of Grids built out of differing baseline technologies with perhaps incompatible communication systems. Third, we may need to dynamically create Grids out of selected resources ("bits and pieces") from other Grid systems.

Figure 2 shows an example of the third case, where a Grid needed for a particular course

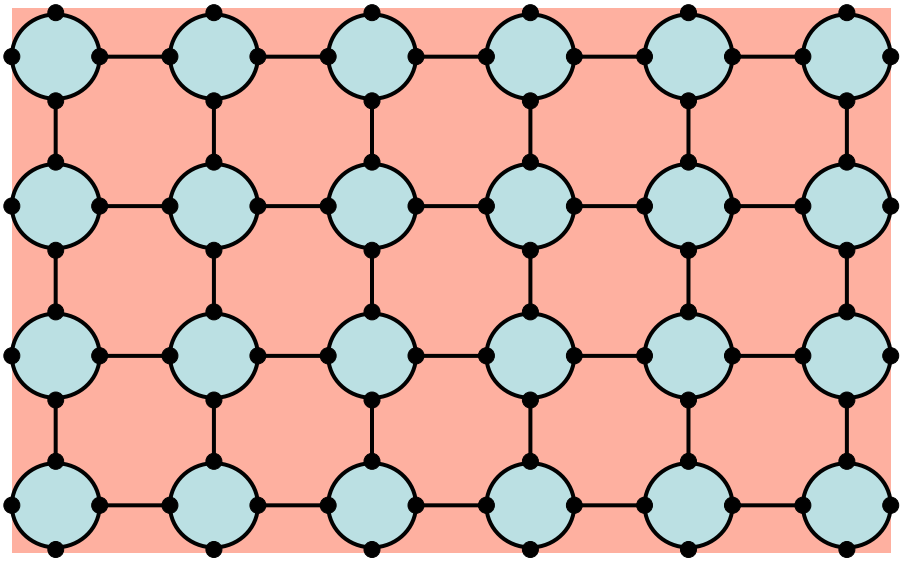
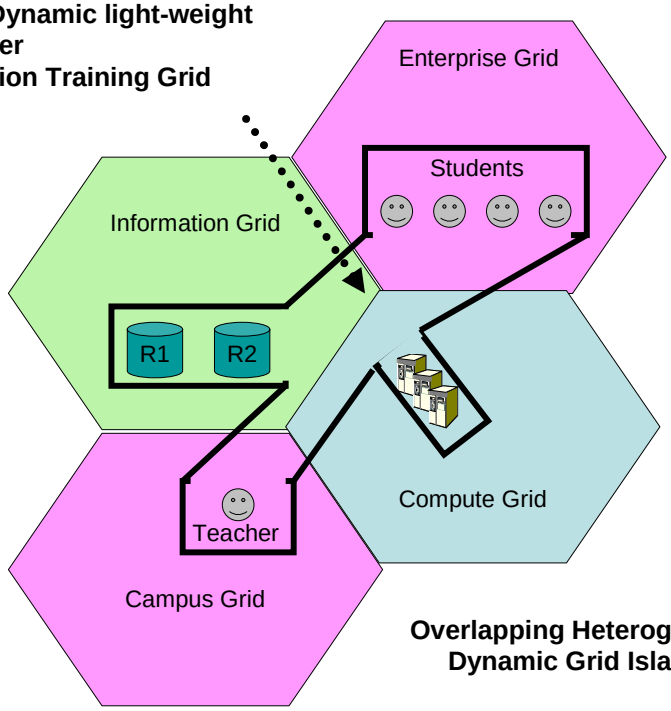


Figure 1: A Cellular Grid built from (24) separate Gridlets linked by a Messaging system. This need not be a two dimensional mesh as shown here.

must be formed as a dynamic overlay (network) Grid. This overlay Grid aggregates resources from multiple different Grids containing teacher, students, computing and information resources. The need to subdivide Grids is also seen in the fine-grain authorization needed within a Grid with individual resources often needing particular

Figure 2: Dynamic light-weight Peer-to-peer Collaboration Training Grid



authorization. Peer-to-peer networks which are composed from individual resources (computers) naturally support dynamic authorization and construction of custom Grids. We have previously [3] explained how one can discuss Grids and P2P networks in a common Service model with different implementation mechanisms; in each case messaging links services with themselves and with clients.

Of course these ideas solve some problems but lead to others. One is the performance overhead of mediating at cell boundaries and another is the problem of sharing cells between several Grids. We will not tackle these difficult issues in detail here. We note that we have extensively benchmarked NaradaBrokering and found overheads of less than a millisecond in traversing brokers; this is small compared to network delays which are often a hundred times larger and so we expect that building Grids whose “diameter” (number of cell boundaries crossed in typical data transfer) of 10 to 50 is reasonable. Thus we expect it practical to build Grids with thousands of cells without special attention to performance. Note that Fig. 1 shows that a Cellular Grid is rather like a classical massively parallel computer. The cells of the Grid are now small Gridlets instead of compute nodes, and for messaging one needs systems like WebSphereMQ, JMS or NaradaBrokering to replace Message Passing Interface (MPI) style parallel messaging [4] [5]. In particular the latencies of Grid messaging systems are many orders of magnitude larger than those for MPI and so can support sophisticated routing and management mechanisms that are impossible and generally unnecessary in parallel systems.

In looking at performance, one would need to carefully consider the treatment of real data streams as opposed to control messages containing perhaps data references. Performances issues are obviously less critical for the latter but we have shown that up to around 100 megabits/second of video (in 200 streams) can be successfully routed by NaradaBrokering on a low-end Linux server. Thus we expect to be able to use our approach for all types of messages with large data streams being routed to minimize overhead.

2 Federation, Interoperability, and Mediation

Suppose we have a collection of Gridlets which each consist of one or more services; within each Gridlet we assume there is a well defined architecture using defined but possibly Gridlet-idiosyncratic Grid architectures and interfaces. Some Gridlets could use OGSA [6] interfaces built on OGSI [7] as in the Globus Toolkit 3 (GT3); others could be built from previous Globus Toolkits; others "pure" Web services; others JXTA; some from different commercial vendors; etc. We assume that a given service has a defined interface internally in each Gridlet; we do not assume that this interface is the same in all Gridlets. Then we need to implement two capabilities

- *Interoperability* – "particular application Web Service in Gridlet X" can utilize "core service of Gridlet Y"
- *Federation*– "core service of Gridlet X" can be integrated with "core service of Gridlet Y" to provide an integration/amalgam that is also a realization of core service.

Federation would be needed for metadata and registry services. Interoperability would be needed for the job execution service on a computer in a particular Gridlet. In our approach GT3, Jini, JXTA, OGSI, GAF or pure Web service approaches can be Gridlets; the different architectures and interfaces are reconciled by the *mediation* services at the "edge" of each Gridlet. In this approach, OGSA (with possible extensions) is used not only as the service interface for GT3 style Grids but also as a common intermediate form for mapping between interfaces and architectures in the mediation service. We use this mediation approach in our Audio-video conferencing system CollabMMCS (http://www.collabmmcs.org) where media streams in one codec are posted to a NaradaBrokering-based publish/subscribe broker; agents subscribe to these streams to convert to other codecs which are then reposted to the server as separate topics.

Note that although Fig. 1 shows a mesh of brokers, one only needs a P2P style routing from node to node if one needs to search the Grid for a certain service. NaradaBrokering supports fully long distance routing skipping across Gridlets and Grids. This approach virtualizes many features of a Grid including service interfaces, destinations, routing etc. It allows WS-Addressing[8], GSR/GSH [7], and URI/URL/PURL naming systems to be used to specify service locations. We see this as very important for allowing the construction of large distributed systems at a time when many key architecture and interface specifications are still being discussed. We personally believe it likely that multiple architectures optimized on different criteria will emerge, rather than a single architecture. InterGrids is designed to support the integration of such diverse Grids. In

section 4, we describe how this design generalizes firewalls, proxy servers and VPN's (Virtual Private Network) and suggest a VPG (Virtual Private Grid) as one security model. The mediation service supports both interoperability and federation; further it has a fault tolerant distributed design described in section 3 and has no single point of failure.

There are many open issues on building registration services that will allow Gridlets to declare their architecture and architectures to declare how their interfaces are to be mapped. We currently intend to follow the simple mechanisms used in JXTA search to federate multiple services. We also need to support the correct operation of services or Gridlets shared among multiple Grids. This in general needs priority schemes and for single-use services, a token allowing them to be dynamically bound to a single Grid. Either prior use or lack of an appropriate interface mapping could lead the mediation service to reject an access request; this must be done gracefully. In the near term, we will focus on the virtualization of OGSI services allowing the interoperation of standard Web Service Grids (such as outlined in [9]) and OGSI Grids. The mediation service will virtualize key OGSI features such as factories, notification and Service Data Elements (SDE) [7]. For example, information port SDE requests could be automatically routed to a meta-data catalog specified in the registration of a Web Service based Gridlet. More straightforwardly, the publish-subscribe architecture of the mediation service will allow it to automatically provide a powerful OGSI compatible notification service.

3 NaradaBrokering

NaradaBrokering [10][11][12] is a distributed messaging infrastructure, which provides support for centralized, P2P and distributed interactions. NaradaBrokering efficiently routes any given message between the originators and registered consumers of the message in question. Messages could be used to encapsulate information pertaining to transactions, data interchange, system conditions and finally the search, discovery and subsequent sharing of resources.

Messages encapsulate expressive power at multiple levels. Where, when and how these messages reveal their expressive power is what constitutes information flow. NaradaBrokering manages this information flow. NaradaBrokering places no constraints either on the number, size or rate of these interactions. Scaling, availability and fault tolerance requirements entail that the messaging infrastructure managing this information flow be based on a distributed network of cooperating nodes.

The smallest unit of the messaging infrastructure that would provide a back bone for routing these messages needs to be able to intelligently process and route messages while working with multiple underlying network communication protocols. We refer to this unit as a *broker*. We avoid the use of the term *server* to distinguish it clearly from the application servers that would be among the sources/sinks to messages processed within the system. This distinction is illustrated in Figure 3, in which brokers (circles) are connected for messaging routing. Service providers (rectangles) register with brokers as either publishers, subscribers, or both. Entities within the system utilize the broker network to effectively communicate and exchange data with each other. These interacting entities could be any combination of users, resources, services and proxies thereto.

System clients (shown as hexagons) also register with brokers to publish and/or subscribe. The distinction between clients and services is artificial in this configuration. Publishers are message sources and subscribers are message sinks. Endpoint entities can register interest in multiple topics and can have different publishing and subscription privileges.

NaradaBrokering incorporates an extensible transport framework [13] and virtualizes the channels over which entities interact with each other. Entities are thus free to communicate across firewalls, proxies and NAT boundaries which can prevent interactions from taking place. Furthermore, NaradaBrokering provides support for multiple transport protocols such as TCP (blocking and non-blocking), UDP, SSL, HTTP and RTP. NaradaBrokering incorporates a monitoring service [14] at individual broker nodes which monitor the state of the links originating from a node. The performance metric measured include loss rates, communication delays and jitters among others.

NaradaBrokering is JMS compliant [15] and also provides support for routing JXTA interactions [16]. Work is currently underway to provide support for routing Gnutella interactions. NaradaBrokering has also been used to support audio-video conferencing [17] applications.

To address the issues [18] of scaling, load balancing and failure resiliency, NaradaBrokering is implemented on a network of cooperating brokers. In NaradaBrokering we impose a hierarchical structure on the broker network, where a broker is part of a cluster that is part of a super-cluster, which in turn is part of a super-super-cluster and so on. Figure 3 depicts a sub-system comprising of a super-super-cluster **SSC-A** with 3 super-clusters **SC-1**, **SC-2** and **SC-3** each of which have clusters that in turn are comprised of broker nodes. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes during failures. This organization scheme results in “small world networks” where the average communication pathlengths between brokers increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings.

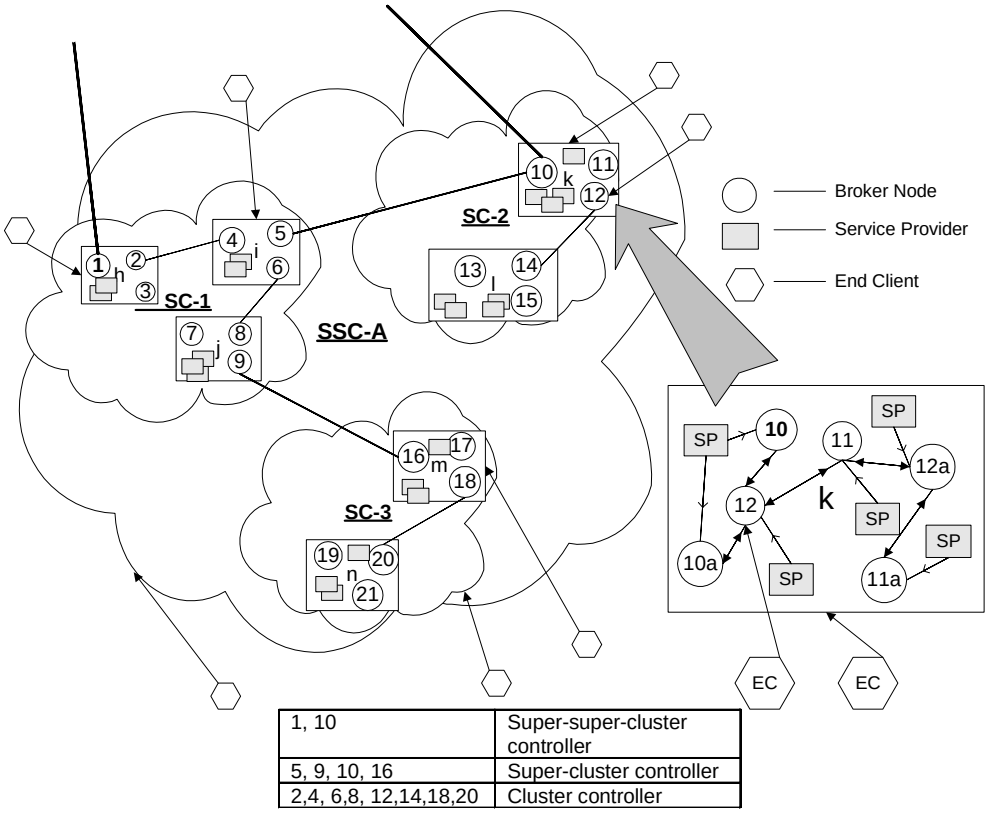


Figure 3:An example of a NaradaBrokering broker network sub-section managing gridlet realms.

This distributed cluster architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to a very large size. Within every unit (cluster, super-cluster and so on), there is at least one unit-controller, which provides a gateway to nodes in other units. For example in figure 1, cluster controller node **20** provides a gateway to nodes in cluster **m**. Creation of broker network maps and the detection of network partitions are easily achieved in this topology.

NaradaBrokering also provides support for routing real-time XML events to consumers based on the specified XPath queries [19]. NaradaBrokering incorporates a message based security scheme to facilitate end-to-end message security.

4 Security in Cellular Grids

4.1 Introduction

As we have described in the proceeding section, NaradaBrokering allows us to create scalable, distributed messaging systems that can be used to relay events to multiple parties, span network impediments such as firewalls and poor performance, and define hierarchical topologies. We now examine how these features can be applied to Grid security. Before proceeding, we first note that there are other aspects of federated Grids that are described elsewhere. Principally, Virtual Organizations require federated roles in such systems provide the scalability and loose coupling needed for federation. This is discussed in detail by Chivers [20]. We note here that user identity in current Grid

security implementations has two major shortcomings: it does not scale and it undermines traditional Web security by effectively creating a super-Virtual Private Network that includes users outside the direct control of the constituent (real) member organizations of a VO. An expanded discussion of Grid security issues is also available from [21].

4.2 Grid Authentication and Authorization

The Grid Security Infrastructure (GSI) [22] defines the standard for Grid authentication. We note the resemblance of standard Grid security to Virtual Private Networks, but as has been pointed out, this actually subverts known best practices from commercial Web security [23]—essentially a grid administrator must trust users outside of his or her authority to securely maintain their private keys. This risk can be mitigated when authentication is coupled with authorization. We focus here on what we believe is the middle ground between the secure deployment scenarios (with lessons learned from the e-commerce world) discussed in [23] and the finer-grained policy issues addressed by such systems as Akenti [24] and CAS [25]. We are interested in using a distributed publish/subscribe metaphor to enable simple but powerful distributed authorization systems for federating Grids. Such a system may be used to create a "layered virtual network," in which we are able to replicate in application software some of the best practices from the commercial sector (VPNs, Firewalls, DMZs). The express purpose of such as system is to develop rings of security both within Grids and around Grids in a lightweight fashion, without requiring, for example firewall and/or network router modifications for every VO membership or policy change.

Web service solutions for conveying security policies will inevitably have an impact on Grid services as current Grids evolve into OGSI implementations. Web services communicating with SOAP have an open-ended ability to encapsulate other XML messages through one or more SOAP headers and the SOAP body, so the main XML payload (in the body) may be supplemented with several different XML messages describing authentication and authorization assertions. WS-Security, WS-Policy, and SAML are all examples of these. Thus there will need to be mechanisms that will enable messages to cross boundaries between grids with different security mechanisms.

4.3 Virtual Private Networks for Cellular Grids

Broadly speaking, a VPN is a network tunneling mechanism that connects distributed resources securely into a private network by using encryption technologies rather than actual physical network lines. In practice we may think of these as virtual private intranets—the VPN provides access to restricted networks to computers outside the restricted intranet. Secure connections are made at the message level rather than through secure transport links, so the message can securely transverse an untrusted network. Trust here is implicitly all or nothing. The source and destination have decided to trust each other completely. Nothing else is trusted, including the VPN service provider. As we shall see, VPNs are very similar to "single-hop" Grids.

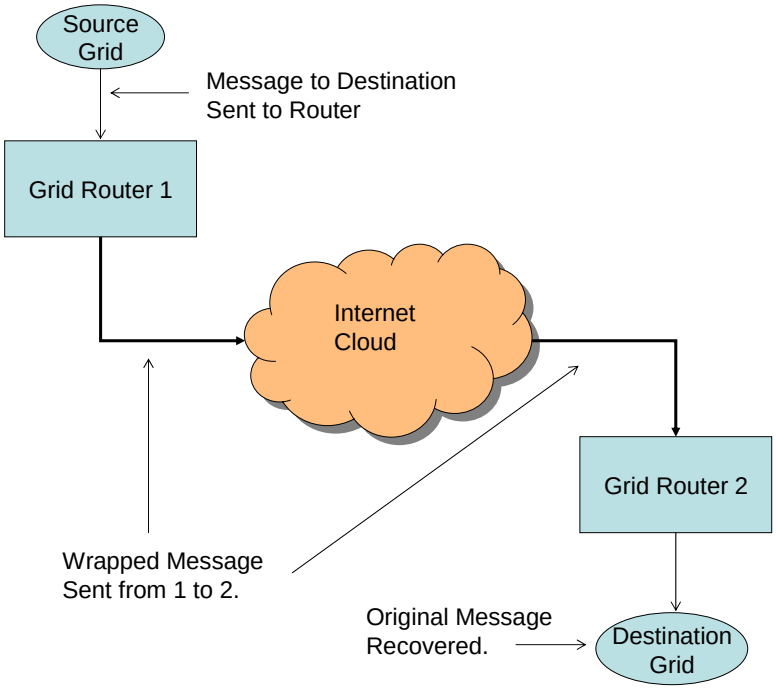


Figure 4 Grid authentication resembles a VPN.

Figure 4 depicts a GridVPN. We may easily change Figure 4 so that the Source and Destination Grids are instead Source and Destination Networks of a traditional VPN, and the Grid Routers (GR) are just routers. The GRs as specialized peer brokers that route messages between the source and destination Grids. The separate Grids may be set up to ignore all outside traffic, so that the GR acts as a proxy server. That is, users outside of the Destination Grid may not talk directly to the Globus Gatekeeper on port 2119 of some server. The users instead have to talk to the GR on some other port on some other host, which routes the request if all goes well. This would allow Grid administrators to maintain "standard operating Grids" of their own resources and users, but by configuring the GR, they may quickly turn on (and turn off) access to their Grid resources from other Grids.

As described, GridVPNs incorporate authentication and secure transmission. This has three limitations. First, we must trust the authentication mechanism. Users (typically by possessing access to the right private key) can prove their identity to the GridVPN and are thus allowed to send messages. The GridVPN then makes sure the message goes to the right place securely. This is fine as long as the organization that maintains the Grid has strict control over the private keys as well. However, Grids imply collaboration between separate organizations, thus in affect requiring trust in the maintenance of private key security in users from other organizations. Thus, the destination Grid in Figure 4 must trust that all the users of the Source Grid have properly secured their private keys. Second, the GridVPN itself does not make any access control decisions. Such decisions are usually made by firewalls that are coupled to the routers. Firewalls are discussed below, although the line between the GridVPN and firewalls for Grids is blurred. Third and finally, there is one single, homogenous security realm, when in

practice we know that we must instead have rings of security that can quickly seal off and isolate compromised areas. To address this issue, Grids actually will need to be built up out of several hierarchical GridVPNs. We examine this issue in the next section.

4.5 Hierarchical GridVPNs

A Virtual Organization may itself be composed of smaller Virtual Organizations. That is, in the GridVPN of Figure 4 the Source and Destination may themselves be federated networks. Also, of course, we may wish (even within a Grid of our own resources) to subdivide resources into smaller subunits for the purposes of access control and firewall-style security. That is, in our Grid we may have some resources that are for internal use only (private), some that are for trusted partners (shared), and some that are more openly accessible from other Grids in a test bed.

Thus, our single Grid installation (or a VO that is part of a larger VO) must enforce multiple levels of access and protect different access partitions from compromises in the other partitions. For example, the testbed may include some resources that we own that we are not overly concerned with. We want to prevent them from being hacked, but if they do get hacked it is not a disaster. Such test beds would need to be thoroughly "sandboxed" and we should be able to quickly pull the plug on them if they are compromised without bringing down the rest of our Grid.

We may represent this situation by magnifying the Source or Destination Grid bubbles in Figure 4, as shown below. The layered Grid described above may be configured as in Figure 5. The Grid installation itself is accessed at the perimeter through a Grid Router, GR1, which forwards all incoming messages to the internal Grid Routers. Messages without proper destinations and credentials are rejected here. Within the Grid itself, we have Resources R1-R5 that are partitioned as shown. Access to these resources is controlled by one or more internal Grid Routers. For example, the Shared and Restricted resources are separated from each other by GR3 and GR4 and both separated by an additional security wall, GR2, from the Test Bed. Thus if the Test Bed's security is compromised, it can be isolated from both the internal and external resources by shutting down its Grid Router, GR5. Such changes need to be propagated to the other Grid Routers.

This figure is drawn with NaradaBrokering (Section 3) in mind. Here, incoming requests for resources are divided logically into topics and "physically" into different brokers running on different machines. A single publish/subscribe realm is distributed across several brokers for load balancing and fault tolerance, as well as security. Incoming requests are matched to publication privileges. For example, one may have publication

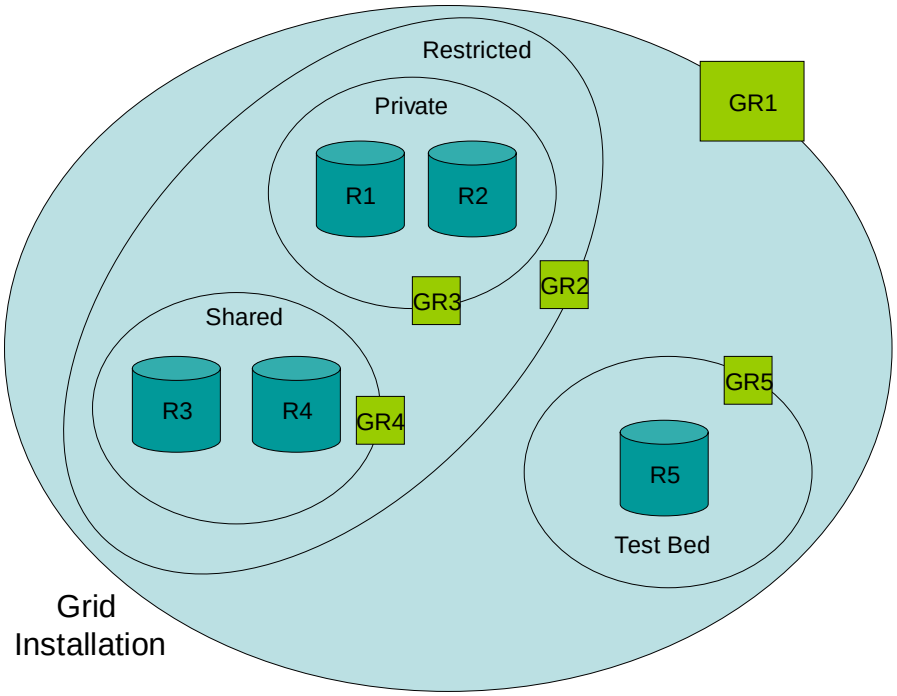


Figure 5 Brokers allow us to subdivide grids into hierarchical VPNs.

privileges to the topics “Test Bed” and to “Restricted/Shared” but not to the topic “Restricted/Private”, so Resources R3-R5 may be accessed within the Grid for executing services.

4.6 Brokers for Firewalls and DMZs

The brokers shown in Figure 4 for GridVPNs may be extended to act as firewalls. Grids and broker systems often consider firewalls to be impediments and devote a lot of effort to getting around and through them, but the use of firewalls is an important security practice that limits the effects of security breaches. Our hierarchical security arrangement can be used to create/mimic firewall behavior. First, firewalls may act as proxies and gatekeepers, single entry points for restricted resources. Thus protected resources do not need to be accessed directly from the outside and are instead only accessed by a trusted gatekeeper. This provides the very practical advantage that in cases of security emergencies, the restricted resources can be protected immediately by shutting down the gatekeeper. For endemic security problems, the gatekeeper can remain up while denying access to known untrusted sources or perhaps denying access to all except a few privileged outside sources.

As shown in Figure 5, firewall protection can be inverted, surrounding and cutting off access to compromised resources, preventing the compromised resources from attacking other resources. In analogy with common enterprise practice, important Grid resources should be surrounded by buffer regions. These buffer regions are defined by firewall boundaries and are traversed through a trusted proxy server. This configuration is usually referred to as a “demilitarized zone”, or DMZ. The broker topology and messaging system we advocate can be used to construct this kind of protective buffering.

5 Secure Mediation Services in Federated Grids

Section 4 primarily addressed the security implications (and advantages) of federation. The larger issue remains communication across Grid boundaries, especially when the Grids are built from different underlying technologies and use different protocols. Technologies such as JXTA, JINI, RMI, Avaki/Legion and others may be used to build OGSI grids or simply Grid-like systems. We may see this even within Globus Grid installations, where older (pre-OGSI) installations use legacy protocols while newer installations move to SOAP. We may also imagine a Grid composed of several different available Grid software pieces. WSDL is protocol independent, so it becomes unclear in the OGSI picture how such protocol negotiation will occur when one Grid component discovers and wishes to access a resource in another Grid in a federation.

Let's examine the requirements of the mediation service from the point of view of Figure 5. Presume that a second, mirror image Grid has been federated with the Grid depicted in Figure 5. Messages between the Grids are exchanged through the Grid analogy to Figure 4. We presume the security mappings needed for cross authentication and account creation have already occurred, and the prospective client has obtained proxy certificates. We shall also presume that Grids A and B are both Globus grids but built with different versions. Grid A uses Globus 2.x and so has several pre-Web service protocols. Grid B is an OGSI implementation that uses SOAP. The client is in Grid B (and so is OGSA enabled).

Federation in this case implies several things besides sharing information and single sign-on. First, information discovery should be made forward-compatible on the legacy Grid through a mediator/translator running at the Grid boundary. This may be an additional function of the Grid Router GR1. Here GR1 serves as a translator

Second, Grid A may wish to limit its available resources to Grid B clients, allowing them only to access the test bed resources. Here GR1 has the additional task of managing subscriptions to Grid services. All resources in the Grid may be running information services, but it is up to GR1 to check incoming requests to see what subscriptions the outside client possesses. In this case, the client only has a "Test Bed" subscription and so only may interact information servers associated with the test bed resources. GR1 itself does not need to maintain the entire subscription list. It may instead distribute this responsibility to the inner Grid Routers.

Third, assuming the client finds what it seeks, it attempts to invoke a process to run an application on the test bed. The problem is that the client is a Web service/OGSA style client that wishes to remotely invoke methods through stubs, while the actual service is accessed through the GRAM protocol and uses the Gatekeeper/Job Manager. The GRs must then serve as translators, exposing an OGSI-style interface for the legacy grid, mapping that interface to, say, a Java CoG client, and translating the incoming SOAP request into a GRAM request. One can generally see a scaling problem in this approach with other "Grid" implementations if there were no OGSI. That is, one would have to have N(N-1) translations to connect N different Grid implementations (a JINI grid to a Globus Grid, a JINI grid again to a JXTA grid, and so on). Here OGSI, because it is

language independent, acts as the neutral expression of the interface. If the underlying Grid has implemented its services in OGSI-style WSDL, no mapping is required. If not, the GRs act as translators. Finally, the test bed application on Grid A executes and sends back information to Grid B. The GR intercepts the response, translates it to the appropriate protocol, and forwards it on to the destination.

6 Towards Autonomic Services in Cellular Grids

Here we discuss how the network of message brokers in a cellular Grid allows one to build a general management and a robust, reliable, and resilient framework for Grid Services [26]. We believe these capabilities are the foundation on which adaptable, autonomic Grids may be built. We illustrate the general discussion with a discussion of an autonomic GridFTP implementation that can be used for any Grid messaging and data transport.

Service reliability is an important problem for simple point-to-point services. The problem is even more important and difficult in the case of composed services, where a service may be comprised of interactions involving two or more services. However when a service is composed of multiple services it needs to manage and deal with the unpredictability concomitant in every one of the constituent services. Furthermore, a given service may be accessed and utilized by entities with myriad device profiles.

6.1 Autonomic Substrates

There are three specific aspects of fault tolerance that are relevant within the context of entities and services within the system:

(a) Service implementations should be internally fault tolerant.
(b) Entities and services must be generically fault tolerant.
(c) Inter-service communications and service-entity communications must be fault tolerant.

Item (a) refers to the quality of a particular service implementation. These implementation issues are out of scope for the current discussion, although reliable communication of publicly reported errors and exceptions is in scope.

To address issues (b) and (c), we propose in this paper an *autonomic substrate*. The autonomic substrate provides a service for managing and coordinating access involving entities and services within the system. The substrate should satisfy three core properties: robustness, reliability and resiliency. “Robust” corresponds to its construction with minimal assumptions regarding the realms in which it would operate. “Reliable” corresponds to the ability of the substrate to provide consistent, predictable, correct and dependable behavior. “Resilient” corresponds to the ability of the substrate to recover quickly and accurately from failures.

The autonomic substrate is based on building a robust, reliable and resilient distributed messaging infrastructure and facilitating the efficient dissemination of messages within the system. These messages encapsulate information pertaining to transactions, data interchange, system conditions and finally the search, discovery and subsequent sharing of resources. Messages also contain policy information associated with them. The policy

information encapsulates reliability, retransmission, ordering, causality, security, compression and other related information. It is the messaging infrastructure's responsibility to ensure that the policy requirements within these messages are satisfied.

For the most part the messaging infrastructure and the autonomic substrate that builds upon it are indistinguishable from each other. We now proceed to enumerate the functionalities that need to be provided by the autonomic substrate. These are considered in more detail in [26].

1. Failures: There should be no single point of failure within the system.
2. Discovery of Services: Services should be able to advertise themselves, and provide correct, up-to-date information.
3. Replication Management: The substrate should facilitate the replication of services and manage access to these services in a way which exploits the high availability provided by the aforementioned replications.
4. End-to-End Security: The substrate should facilitate secure interactions between communicating entities using message-based security.
5. Robust messaging: The messaging framework should ensure delivery in the presence of failure, recoveries and prolonged disconnects. The robust messaging should also manage causal (and source) ordering relationships between messages/interaction issued by entities
6. Notifications: The substrate should issue alerts while performing error and diagnostic reporting upon detection of failures, recovering services and service management related notifications. The notification service should also facilitate the instantiation of replicas to circumvent existing/future failures in services.

6.2 Advantages of this approach

There are several advantages to this approach of delegating complexity management to the autonomic substrate. Entities interact with each other through a single access point into the system. There could conceivably be millions of these access points for the substrate, thus providing a great degree of resilience to failures.

Services can acquire reliability and robustness simply by their placement within the substrate. The autonomic substrate infuses these properties into the service without a need for the service implementers to change either the interfaces (service advertisements) or their underlying implementations.

Services clearly benefit from the elimination of networking complexity. It is the substrate's responsibility to traverse firewalls, NAT, proxies and VPN boundaries. Moreover, since the substrate supplies messaging efficiency, the service's resources are freed to deal only with the functionality they provide, instead of managing networking and computational overheads related to communications.

Services also do not need to manage the complexities involved in robust messaging with multiple services. Managing the robust, messaging-related complexities within individual services would make even the simplest of applications, composed of a small set of

services, complex. The problems are compounded when one needs to deal with entity/service disconnections.

Since management of service versioning, replication and load balancing is now managed by the substrate, entities benefit from the high availability of services and invocations on the correct service instances. Service instances, on the other hand, are not overwhelmed by requests from entities since the invocations are now load-balanced by the substrate across the replicated service instances.

6.3 Message-Based Monitoring Infrastructure

Previously, we listed several capabilities that an autonomic computing grid substrate must provide, ranging from the mundane to the sophisticated. Simple monitoring capabilities fall perhaps in the mundane category but provide a crucial initial service: we have to know when a node on our distributed service network fails.

We are building service monitoring systems as straightforward implementations of NaradaBrokering. Service based grids consist of a collection of Web servers running on different hosts. Each server maintains a list of deployed services that may change over time. Information systems based on these servers cannot be static: information about deployed services must be harvested directly and frequently (perhaps on demand or else by polling) from the services.

More critically, we need to know when servers themselves are unavailable. Such events need to be distributed to multiple interested parties (subscribers). For example, client applications need to be notified so that they do not attempt to connect to unreachable servers, but also the monitoring system should notify system administrators (via email, for example) and should also publish these events in persistent storage locations (distributed logging). Publish/subscribe systems are designed for these kinds of applications.

The architecture for such systems is as depicted in Figure 3. NaradaBrokering clients are embedded into Tomcat web servers, where they register themselves with a broker controller. One or more listening clients may also register with the broker.

Monitoring topics must be maintained in pairs: “topic request” and “topic response”. Perhaps counter-intuitively, the embedded broker clients subscribe (instead of publish) to the monitoring topic. Monitoring messages are published on demand from interested clients and are distributed to all subscribed service endpoints. If these service endpoints are reachable, the “on message” event signals them to publish a response message to a separate topic. Inaccessible services never receive the request message, and so never publish their response.

Listening client applications can then take any appropriate actions upon receipt of the response message. For example, a listening client may maintain a list of servers that should be available and can compare this to the list that actually responds. Administrators of unavailable servers can then be notified.

More sophisticated information systems can be built around this simple “ping” service. For example, we have also implemented a “request info” topic that prompts registered servers to respond with their available services. The results of these services can then be compiled into an up-to-date service registry using WSIL, UDDI, or other systems. The publish/subscribe system obviously enable us to provide multiple views (via multiple subscribers) of the same information.

6.4 GridFTP on an Autonomic Substrate

We now examine the role that the reliable substrate can play in file/data transfer. In the Web (Grid) Service architecture, the state of any service is defined by its initial condition and all the messages (including ordering) that it receives. This is how shared event model of collaboration works. This is a “Finite State Change” model analogous to saving file and “undo” command in many editors. NaradaBrokering (NB) plus a robust store can guarantee to save all these messages for (all) services. This allows one to build both "autonomic data transport" and "autonomic services" since these services can sustain packet losses in transport and can also sustain failures of applications/brokers, since eventually archived messages (previous invocations, published events, etc) can be retransmitted to reconstruct state at the service or to correct a transport error. Further anomalies in message traffic (such as a publisher or subscriber are silent) can be detected by the brokers and signal problems.

We are building examples of both scenarios using GridFTP as our data transport example of these ideas. GridFTP already incorporates a number of reliability features. It is our goal to show that these reliability features can be decoupled from the implementation of the service and protocol, and instead placed into the messaging substrate. This will allow us to provide file transfer quality of service comparable to GridFTP in other file transfer tools (such as normal FTP, SCP, HTTP uploads, and similar mechanisms).

Figure 7 is present the basic architecture of integration between GridFTP and NaradaBrokering. We discussed two possible approaches:

- Proxy approach: the remote GridFTP server is simulated by the NB Agent A
- Router approach: using NB Agent A as simple router to transfer requests to the remote server.

The proxy approach is the preferred method since the GridFTP client code does not have to change: it simply contacts NB Agent A and uses it as if it is a regular GridFTP. NB Agent A then forwards all requests through the broker cloud to an available GridFTP server (which may be chosen based on topic subscription-style access control as described in section 4).

Our eventual goal is to create a proxy GridFTP server on NB Agent A, but for initial testing we chose the “router” approach as simpler. This requires some minor extensions to FTP/GridFTP client codes so that the client can tell NB Agent A that it want to use the GridFTP server. This requires disabling automatic hostname checking of messages by the client and some extensions to allow hostnames and ports to be changed by the agent

to match the remote server. The client and server communicate solely with the agents on the edge of the broker cloud.

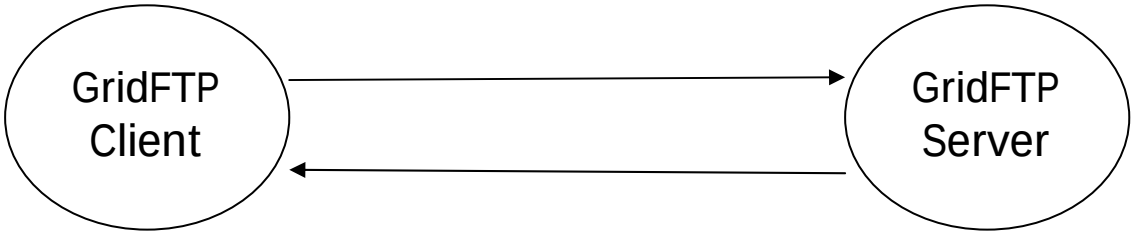


Figure 7a. Traditional GridFTP

Figure 7b GridFTP with NaradaBrokering

GridFTP requires two socket connections: a control channel that transfers commands ("put", "get") and a data channel that is used for data transfer. Currently, we have completed development of the uploading functionality of GridFTP with NaradBrokering using simple router approach. Connection between the GridFTP client and NB Agent A and NB Agent B and GridFTP server are connected with a high-speed, reliable, possibly local, connection. This connection is needed because if connection between GridFTP client and NB Agent A is lost, we cannot recover from this failure. Recovering from this failure is out of scope (GridFTP designed in this way). All the data will be first transferred and stored into the temporary local space of NB Agent A. This temporary data will be used when any failure is occurred inside of NB. Once all the data is stored locally in the NB Agent A, even if connection between GridFTP client and NB Agent A is lost, transferring to the server is guarantied by NB. This feature is not on the current GridFTP system. In the current GridFTP system, if a client fails, the client has to begin uploading again from the start. NB Agent B also store data into the temporary local space. This temporary data will be used when any failure is occurred to the GridFTP server.

Summary

This paper has introduced a peer-to-peer based system, InterGrids, for federating Grid installations and resources. We have described how messaging systems may be used as a bridging system for connecting different Grid peers. Message systems may be further used to provide access control between the peers and may be used to mimic known best practices from Web commerce, including firewalls, DMZs, and VPNs. P2P grids also

potentially offer a path to autonomic computing, in which the entire grid is more robust, reliable, and resilient than the sum of its parts. We have examined the use of message brokers to enable this functionality. Finally, we have presented two case studies currently under development: peer-based monitoring services and substrate mediated file transfer with GridFTP. In the latter case, the messaging substrate, and not the service implementation, provides the reliable file transfer. Thus, any file transfer service (not only GridFTP) can inherit reliability assurances from its messaging substrate.

References

- [1] *Grid Computing: Making the Global Infrastructure a Reality* edited by Fran Berman, Geoffrey Fox and Tony Hey, John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0, March 2003. http://www.grid2002.org
- [2] Geoffrey Fox, David Walker, *e-Science Gap Analysis*, June 30 2003. Report UKeS-2003-01, http://www.nesc.ac.uk/technical_papers/UKeS-2003-01/index.html.
- [3] G. Fox, D. Gannon, S.-H. Ko, S. Lee, S. Pallickara, M. , X. Qiu, X. Rao, A.Uyar, M. Wang, and W. Wu, *Peer-to-Peer Grids*, Chapter 18 of *Grid Computing: Making the Global Infrastructure a Reality* edited by Fran Berman, Geoffrey Fox and Tony Hey, John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0, March 2003.
- [4] G. Fox, Messaging Systems: Parallel Computing the Internet and the Grid EuroPVM/MPI 2003 Invited Talk September 30 2003. Available from http://grids.ucs.indiana.edu/ptliupages/publications/gridmp_fox.pdf.
- [5] G. Fox, Message Passing: From Parallel Computing to the Grid, Computers in Science and Engineering, Vol. 4, No 5. (September/October 2002).
- [6] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
- [7] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, and D. Snelling, Open Grid Services Infrastructure (OGSI) Version 1.0 Global Grid Forum Draft Recommendation, 6/27/2003. Available from http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-26_2003-03-13.pdf
- [8] A. Bosworth, et al, Specification: Web Services Addressing (WS-Addressing). Available from http://www-106.ibm.com/developerworks/webservices/library/ws-add/.
- [9] S. Parastatidis, P. Watson, J. Webber, and T. Rischbeck, A Grid Application Framework based on Web Services Specifications and Practices. Available from http://www.neresc.ac.uk/projects/gaf/.
- [10] The NaradaBrokering Project at the Community Grid Labs: http://www.naradabrokering.org
- [11] S. Pallickara and G. Fox, NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003.
- [12] G. Fox and S. Pallickara. An Event Service to Support Grid Computational Environments Journal of Concurrency and Computation: Practice & Experience. Special Issue on Grid Computing Environments. Volume 14(13-15) pp 1097-1129.

- [13] S. Pallickara, G. Fox, J.Yin, G. Gunduz, H. Liu, A. Uyar, and M. Varank, A Transport Framework for Distributed Brokering Systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. (PDPTA'03).
- [14] G. Gunduz, S. Pallickara and G. Fox, A Framework for Aggregating Network Performance in Distributed Brokering Systems. (To appear) Proceedings of the 9th International Conference on Computer, Communication and Control Technologies.
- [15] G. Fox and S. Pallickara, JMS Compliance in the Narada Event Brokering System. Proceedings of the 2002 International Conference on Internet Computing (IC-02). Volume 2 pp 391-397.
- [16] G. Fox, S. Pallickara and X. Rao, A Scaleable Event Infrastructure for Peer to Peer Grids. Proceedings of the ACM Java Grande ISCOPE Conference 2002. pp 66-75. Seattle, WA.
- [17] A. Uyar, S. Pallickara and G. Fox, Towards an Architecture for Audio Video Conferencing in Distributed Brokering Systems. (To appear) Proceedings of the 2003 International Conference on Communications in Computing.
- [18] G. Fox and S. Pallickara, An Approach to High Performance Distributed Web Brokering ACM Ubiquity Volume2 Issue 38. November 2001.
- [19] S. Pallickara, G. Fox, and M. Pierce. Incorporating an XML Matching Engine into Distributed Brokering Systems. . Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. (PDPTA'03).
- [20] H. Chivers *Grid Security: Problems and Potential Solutions*. University of York, UK, Department of Computer Science, Yellow Report YCS-2003-354 available at http://www.cs.york.ac.uk/ftpdir/reports/
- [21] M. Pierce and G. Fox, Federated Grids and Their Security. White paper available from http://grids.ucs.indiana.edu/ptliupages/publications/FedGrid_Short.pdf.
- [22] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.
- [23] M. Surridge A Rough Guide to Grid Security version 1.1a, 11 September 2002, http://umbriel.dcs.gla.ac.uk/Nesc/general/technical_papers/RoughGuideToGridSecurityV1_1a.pdf.
- [24] S. Mudumbai, W. Johnston, M. R. Thompson, A. Essiari, G. Hoo, and K. Jackson, "Akenti- A Distributed Access Control System." *Proc. Supercomputing 1998* (1998). Avaliable from http://www-itg.lbl.gov/Akenti/sc98/akenti.pdf.
- [25] A Community Authorization Service for Group Collaboration**.** L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002
- [26] G. Fox, Shrideep Pallickara, Marlon Pierce and David Walker Towards Dependable Grid and Web Services ACM Ubiquity Volume 4, Issue 25. August, 2003.