SeLeNe Report:
Service-based Architecture

Kyriakos Karenos, George Samaras
and Eleni Chistodoulou
cs98kk2@ucy.ac.cy, cssamara@ucy.ac.cy

May 2, 2003

Abstract

In this report we present the “Grid” architecture and propose a possible set of services that could be applicable in the case of SeLeNe. Generally, a service-based architecture is proposed that allows a user to submit requests to the SeLeNe Network (the “educational grid”) for the completion of some specific task. Although different services may reside at different physical sites, *“authority”* sites should be used for providing extended services (such as in a super-peer model). Service placement produces a number of possible architectural alternatives.

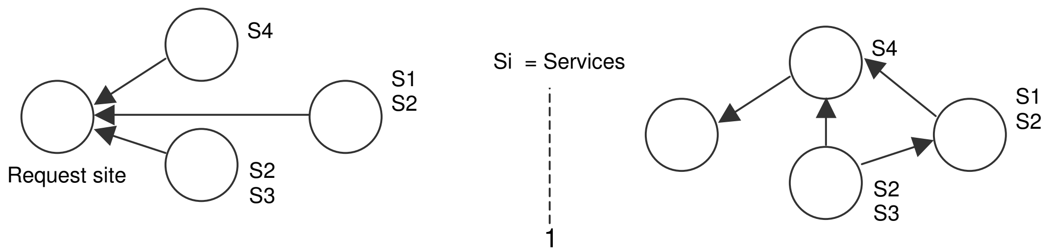
1 Introduction

Our proposed architectural model is characterized as *service-based*. Tasks that must be completed should be submitted to the educational grid (the SeLeNe) via a minimal interface that should simply support the submission operation. This interface’s implementation should call upon lower level basic services that every node will have to provide. An example of such an operation could be the submission of a query or a request for collaboration with another user. However, the various services – as described in the Open Grid Services Architecture Layers (OGSA) – need not be implemented within every physical site. (This is also proposed with the SWAP system [8]). Each node may create and offer different services to the system, included in a predefined set. Services may require the collaboration of many nodes, thus can be characterized as distributed (e.g. a search service).

In [4] an attempt is made to provide an infrastructure for future eScience, termed the “Semantic Grid”. Of our interest, is the adoption of a service-based perspective to meet the needs of a global and flexible collaborative system for educational – the paper limits it to eScience – purposes. Three conceptual layers are proposed: The *Data layer* that deals with the low level, computational and storage resources, the *Information layer* that is related to the data representation, access and maintenance and the *Knowledge layer* that addresses some specific high-level problem or objectives while supporting and monitoring the learning procedures. Note that these levels meet the SeLeNe *“three levels of abstraction”* defined in the User Requirements.

The simplest P2P services scenario is the following: A node ‘A’ contacts a centralized, known server ‘S’ and requests a service ‘s1’. ‘S’ replies with the location and API description of service ‘s1’ at a server ‘B’. Node ‘A’ communicates in a P2P manner with ‘B’. Later the reverse procedure may take place if for example ‘A’ is the owner of a service ‘s2’ requested by ‘B’. The difference in SeLeNe is that the Grid itself takes the role of ‘S’ (by the use of distributed service/lookup catalogs and searching indices.)

It is also possible that, after services are discovered, a task may be completed by combining (reserving) a number of available services by the requesting node/site (Fig. 1,a) or by the collaboration of several sites through a service composition procedure (Fig. 1,b).



Request site

F Figure 1 (a) that some vital services nee Figure 1 (b) (e.g. secure communication provided by a third party, registration etc) as well as the fact that we need to provide some method for information integration. Therefore we propose that *"authority" sites* should be present (such as in a super-peer model [3]) that will be more reliable and may acquire the role of *mediator* (e.g. to interconnect related sites by clustering) or *coordinator* (e.g. to support ontology mappings when and if necessary).

At least two leading platforms enable the implementation of service-based systems, namely JXTA (P2P) and Globus (Grid). JXTA offers a purely Java-based services core [10]. The Globus project also provides a Java version, the CoG Kit [5].

2 Grid Computing

One definition of the *Grid* (or *Grid Computing*) could be: "applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data". It can also be viewed from a software/hardware combination perspective comprising an infrastructure for resource access and sharing within and among *virtual organizations* [6]. The "infamous" among Grid computing cycles, term "virtual organization" (VO) refers to the individuals and/or institutions that participate in a *conditional* sharing system. (Rules that may change over time define what is shared and who may access it). VOs are not actually the participants of a Grid; rather a Grid comes into play to address the requirements of VOs: both at the higher level (i.e. solving the problem itself posed by the VOs) and at the lower level (i.e. the problems appearing due to the VOs need for interaction.) Therefore we can now spherically conceive the "Grid problem" definition being:"*the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations".*

At the conceptual core of the Grid architecture, the key requirement is *Interoperability*. However good the mechanisms may be, the benefit to the system comes to a naught due to the environmental, platform and language diversity within the Grid context.

Three basic constructs must be considered to support interoperability*: Protocols, Services and APIs / SDKs.* Protocols play a primary role to the Grid Architecture which, in fact, *is a protocol architecture.* Protocols aim at the externals (interactions) of VOs rather than internals (software, resource characteristics) allowing for different implementations to each VO with respect to its internal functional structuring and management. Standard services are also of fundamental importance towards interoperability and are defined within the contexts of their protocol and behavior. Finally, APIs and SDKs are required to provide abstractions for sophisticated application development.

One proposed Grid architecture, which is widely acceptable in term of being largely used, follows the "hourglass" principles. The narrow neck defines a limited number of core abstractions and protocols. This is the *Resource and Connectivity layer* on top of which the broad, *Collective layer* is built that defines the high level behaviors. The bottom layer, called *Fabric*, includes the underlying technologies and diverse resources. Figure 4 represents this design.

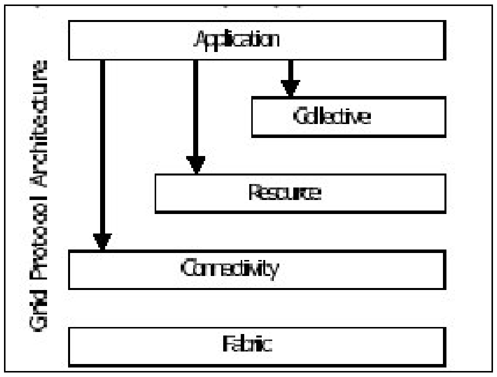


Figure 3: The Grid layers

- **Fabric:** The Grid Fabric layer provides the resources to which shared access is mediated by Grid protocols: for example, computational resources, storage systems, catalogs, network resources, and sensors. A “resource” may be a logical entity, such as a distributed file system, computer cluster, or distributed computer pool; in such cases, a resource implementation may involve internal protocols (e.g., the NFS storage access protocol or a cluster resource management system’s process management protocol), but these are not the concern of Grid architecture.
- **Connectivity:** The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.
- **Resource:** The Resource layer builds on Connectivity layer communication and authentication protocols to define protocols (and APIs and SDKs) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. Resource layer implementations of these protocols call Fabric layer functions to access and control local resources. Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and atomic actions across distributed collections; such issues are the concern of the Collective layer discussed next.
- **Collective:** While the Resource layer is focused on interactions with a single resource, the next layer in the architecture contains protocols and services (and APIs and SDKs) that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. Because Collective components build on the narrow Resource and Connectivity layer 'neck' in the protocol hourglass, they can implement a wide variety of sharing behaviors without placing new requirements on the resources being shared. Collective functions can be implemented as persistent services, with associated protocols, or as SDKs (with associated APIs) designed to be linked with applications. In both cases, *their implementation can build on Resource layer (or other Collective layer) protocols and APIs.*
- **Applications:** The final layer in the Grid architecture comprises the user applications that operate within a VO environment. Figure 5 illustrates an application programmer’s view of Grid architecture. Applications are constructed in terms of, and by calling upon, services defined at any layer. At each layer, well-defined protocols provide access to some useful service: resource management, data access, resource discovery, and so forth. At each layer, APIs may also be defined whose implementation (ideally provided by third-party SDKs) exchange protocol messages with the appropriate service(s) to perform desired actions.

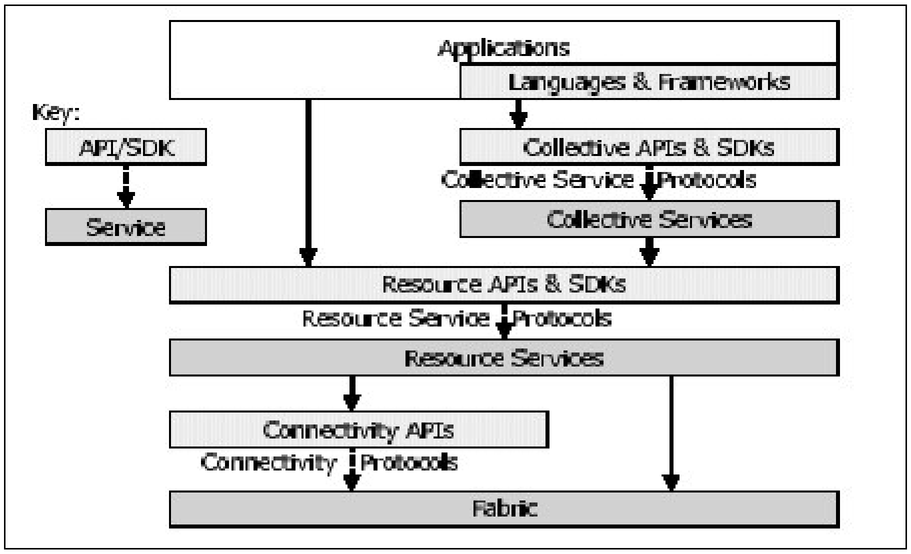


Figure 4

3 An Early Proposal for SeLeNe Services

3.1 Service Classification

As mentioned earlier, not all services are required to reside at each knowledge site. From a specific site's point of view, some services may be local (i.e. implemented at the local machine) and/or distributed (i.e. implemented at more than one site.) For example, an "integration service" may be local for one site and remote/distributed for another. The service's API, however, should be *identical* for both sites regarding the same service.

Additionally, a small set of services is characterized as mandatory or *core* and should be implemented at every site. Note that core services may be fully implemented locally or in a distributed fashion (i.e. the completion of a request for such a service needs to be processed beyond the local site.)

We need to have in mind that each service is found at some layer of the Open Grid Architecture and therefore may call upon other, lower level services.

Table 1 presents all proposed services from the system's point of view. To clarify their target functionality, the OGSA and 'Semantic Grid' corresponding layers are shown. A general description will be provided in the following subsections.

Service Name	**Core**	**OGSA Layer**	**'Semantic Grid' Layer**
Trail Management		*Application*	*Knowledge*
Authoring			
Collaboration			
Caching		*Collective*	*Information*
Search			
Replication			
Access		*Resource*	
Information	**X**		
Integration			*Data*
Registration		*Connectivity*	
Security			
Communication	**X**		
Storage	**X**	*Fabric*	

Table 1: SeLeNe Services

3.2 Core Services

Core services need to be present at each site of the system (Figure 2). They present the minimum set of operational entities that are necessary for the functionality of other services. This does not mean that a service must use the core services to function. It is only meant that a site may, at any time, assume the existence of the core services at every site of the system. We briefly describe the functionality of each core service below:

- **Information Service:**
  The information service's purpose is to provide access to information available in the system regarding: (i) the *metadata* about LOs stored at some node as well as any relations between them (ii) the *available services* at that site and in the system (e.g. a distributed service discovery mechanism over UDDI [12]), since some services may be persistent (e.g. at super nodes) or transient.
- **Storage Service:**
  This service is located at the lower layer of the Grid Architecture (Fabric) and is related to the actual stored LOs. It should include access methods for local requests as well as appropriate manipulation of locally stored LOs (e.g. local updates). Physical data that are placed at the Fabric layer are accessed by this service. For each site we require: the *site's content* (LOs), the *metadata* and information *indices.* Indices contain information about neighboring sites. *Manipulation and usage of storage (including indices) is usually done by the Information service.*
- **Communication Service:**

This service provides the basic communication mechanisms for exchanging data. Current protocols may be used but we should consider creating a simple “SeLeNe specific” communication service (i.e. for the exchange of specific type messages e.g. task submission.)

Core API
Information
Communication
Storage

Figure 2: Minimal SeLeNe site

3.3 Appended (System Required) Services

- **Registration Service:**
  Each site should be able to register to the system mainly in order to advertise its content and services. Also it should be able to make its presence known to other sites. Registration allows for the update of the indices of neighbors as well as the directly connected authority site(s). Site registration (connecting the system) and content advertisement (using the LO metadata) are two distinct tasks since the latter is an evolving process.
- **Search Service:**
  Searching the system is an important feature, although not a core service, since minimal search capabilities are provided by the Information Service. The Searching Service is distributed and should allow for intelligent search message routing in order to forward queries to sites most possible to provide answers. A good, super-peer based technique is provided in [2] where a clustering technique is used to mediate heterogeneous schemas. Authority sites are responsible for keeping semantically meaningful indices about other neighbouring sites.
- **Access Service:**
  The Access Service should provide the API for accessing remote data. We may decide on a common API for data accessing but we may also allow for each (heterogeneous) LO set to be accessed using pre-defined methods such as handling defined granularity level accessing. For example, a “presentation” LO may have an access service method for retrieving a single “slide.” Access services also allow for a high level “browsing” among remote LOs available in the system.
- **Caching Service:**
  The caching service provides an API for temporarily storing LOs for future use. Local cache implementation also defines a replacement policy. LRU is one very possible and functional suggestion. Caching is also used for temporarily holding locations and descriptions of services for efficient lookup, since it is very probable that a service may be repeatedly used within a short time interval (e.g. repeated searches/lookups.) We need to make clearer that caching is indeed a service: Assume that a site does not have a caching service locally but wishes to store a request message to a remote site for asynchronous replies. Another example would be adopting functionality similar to web-proxies; a site sets a neighbouring site with caching capabilities as its “content proxy.” This would be particularly efficient if both sites belong to the same cluster. Summarizing, we identify three caching types: *content caching (LOs)*, *services metadata caching* and *message caching.*
- **Replication Service:**
  Replication services enable underlying distribution of data for optimization purposes (e.g. replicate popular LOs or copy relevant LOs closer to their users.) In addition, a site should be able to request an LO to be replicated locally and later on receive any related updates. A replication service should be available for these purposes, implemented primarily over the communication and information services. The replication service should include transferring of LOs as well as update propagation. It should also make extensive use of the Access Service.
  We make the assumption that only a single site owns a specific LO and is the only one allowed to modify it. However, replicas can be created from an owner as well as from other replicas. Under this model, only the originator needs to be known for keeping a copy consistent such as in the master-slave model. In case of user-requested replica creation, pull-based asynchronous update propagation may suffice. However, when LOs are replicated for

network traffic reduction and faster file copying, updates need to be automatically propagated. In such cases the Replication Service maintains the *replica catalog*; a distributed catalog or index of *logical-to-physical object id* (one-to-many) mappings. Since sites may enter and leave independently, the replica catalog cannot be purely hierarchical. A functional practice is to keep additional (redundant) catalog information on each node (such as in Chord [8]) in order to increase reliability.

- **Security Service:**
  Each site may adopt a security policy concerning the access rights on its data. This service's main purpose it to allow the local site to determine which information should be shared and which should not and who should be allowed to access them. Additionally a security service allows for trusted (third-party) authentication.
- **Collaboration Service:**
  A collaboration service should allow the communication between users and groups of users and it is proposed that this is mediated by a central authority site. At least two sites should request the creation of a collaboration session and others may be added later. Collaboration services may include already available systems such as Blackboards, Message Boards, CVS (for collaborative code writing) or e-mail and instant messaging services. The SeLeNe Collaboration Service lies above these services in order to provide connections to other SeLeNe services.
- **Integration Service:**
  The Integration service should be basically responsible for enabling the coordination of combining different ontologies. This may be mediated or have each site provide necessary mappings. The Integration service provides the semantic and structured connections among LOs. This structure can be provided to the Authoring service to be visually presented to (and edited by) the user.
- **Authoring Service:**
  The Authoring Service should provide an environment for creating new LOs. This could also be an authoring tool. It may also make use of the Collaboration service. It should also make use of Integration services and enable the high level creation of user-defined views of LOs.
- **Trail Management Service:**
  This service should provide the methods for creating and maintaining trails that represent the knowledge of a user. This is obviously located at the Application layer of OGSA (or more correctly, at the "Knowledge Layer" [4]) and should make use of Storage, Communication and Searching services while receiving input from the Authoring service. Trail Management is responsible for the creation of user profiles and using underlying services for the monitoring of LO access for profile maintenance. An appropriate model should be decided for trail representation combined with the LO interconnection itself.

3.4 Authorities

Until now, a general understanding of the functionality of "authorities" must have been perceived. In this subsection we solidify previous dispersed references to "authorities".

"Authorities" are similar to super-peers in a hybrid P2P model or brokers in a Grid model (although they provide more than just brokering services in the SeLeNe context). They are considered more reliable, offer persistent services and are characterized by their static nature regarding their participation to the system. As such they can be used to provide persistent services including Collaboration, Caching, Search, Registration, Access, Integration and Security. It is not obligatory that some authority site should offer all these services, although it is obligatory that it should ensure the service's persistency.

Authorities may take care of search message routing as well as integration/mediation of metadata [2]. They enable the creation of clusters of relevant sites via evolving indices based on the semantic description of the sites' content.

Authorities can also serve as the third party in an authentication process between two sites as well as allow for the collaboration of multiple sites for the creation of collaborative trails.

3.5 Service Availability

An important aspect of the Grid *not* extensively addressed in the various Grid-based systems is service availability. The question is: “Can we be sure that a service that will be requested will in fact be available?” The primary goal is to have multiple instance of the same service running at different sites, therefore having more than one “starting points” (such as in DNS).

A functional framework, with respect to SeLeNe, for supporting service availability is provided in [1]. The proposed framework is based on the assumption that a set of services exists (such as our proposed SeLeNe services) on servers able to communicate with each other (also possible with SeLeNe). Each service is *statefull* and allows access in user *sessions.* During a session, other servers are notified on the current state of the user. Therefore, in case of failure another server can continue offering the service. Not surprisingly, the authors in this paper selected a “search service” and an “educational service” (where a user/student accesses LOs) as two of the basic example scenarios.

Some additional ideas on how we can maximize the possibility of some service being available are proposed below:

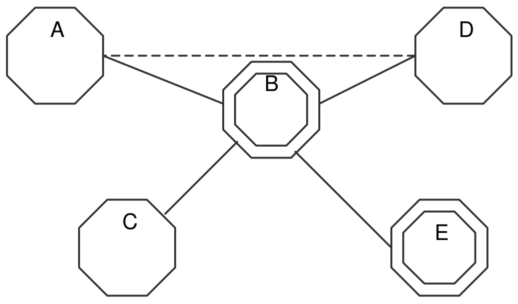
- Basic services are made core.
- Reliable sites (authorities) provide popular/demanding services.
- Allow for appended services to be added (installed) onto sites. (*Service replication.*)
- Break services to smaller components (or services).
- A site may include some (appended) service but may not have it “running” (e.g. to preserve resources). Remotely starting a service may be done via any one RPC implementation.
- Handling of predictable service shutdown (i.e. when a site voluntarily disconnects from the Grid).

4 Some Scenarios

4.1 Set up

Consider the following network of SeLeNe sites along with their offered services:

Site	Services
A	Core, Search, Access,
B	Core, Collaboration, Registration, Integration, Search, Replication
C	Core, Search, Authoring, Cache
D	Core, Search



4.2 Scenario 1

→ an individual author registering a new base LO and its associated metadata
Suppose that site ‘A’ has used the some Authoring service and created an LO. Firstly, the LO needs to be registered locally and its metadata and relation to other local (user) LOs need to be defined via the Information Service. Then the user looks up a Registration Service. The Registration service is available in at least one Authority site (‘B’), which updates its index to include site ‘A’. The LO description remains at the authority site until it is searched for.

4.3 Scenario 2

→ an individual author creating a new derived LO and its associated metadata from other LOs
Using the Search Service the user at site 'C' identifies sites 'A' and 'D' to contain relevant LOs. The Integration service at site 'B' is used to return a joint structure of connections between LOs in 'A' and 'D' which is used as input to the user's Authoring service to create a user-understandable view. The user may use the Access service at site 'A' to access the relevant LOs (saved at the local cache) and finally decides to replicate locally a part of the LO at 'D' and the LO from 'A' (via the Replication Service at 'B.') The Information service is used to produce the newly created interconnections/relationships between local LOs.

4.4 Scenario 3

→ a group of learners collaborating in retrieving and using LOs and in maintaining a shared trail space
A Collaboration service is used at 'B' to create a session between 'A' and 'D.' The role of this service is to keep the state of the current procedure and to allow for the creation of common views on accessed LOs (similar to the views created locally at scenario 1.) The Collaboration service will only create the structure of the views. Each user will visualize the view in a personalized manned, based on local user profile (e.g. local ontology). Any new LOs created are registered with both Information services but only one user becomes the owner.

5 The Service-based "Consumer-Broker-Producer" Model

5.1 General and Alternatives

We still maintain the service-oriented focus but we somewhat shift the architectural paradigm towards the Consumer-Broker-Producer (C-B-P) model (or "roles"). As the name implies, this model categorizes its participants into three distinct roles: *consumer services carriers*, intermediary *brokering servers* and content *producers*.

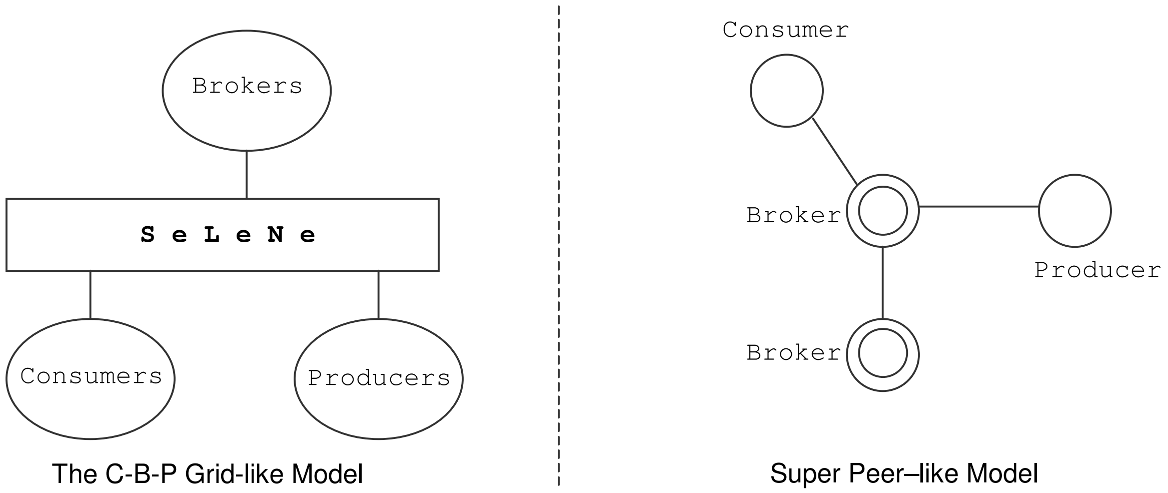
We have noted that a service-based design remains our target concept where, in the C-B-P model, node/site roles are more strictly specified. Comparing this approach to the previous one we note that service provision and placement is predefined in the sense that, depending on the type of each site, specific services must be supported. In other words we view each type being built up as a set of pre-specified services. In the following subsections we provide a proposal as to which services could be supported by consumers, brokers or producers, extensively based on the previous sections' discussion.

Regarding this model, two alternatives are proposed. Placing service sets at different sites, produces different setups from which we may choose. In the first setup Producers can be viewed as LO repositories which are accessed by consumers with Brokers' mediation. Each site must be either of the three. In the second setup the separation of Producer/Consumer functionality (by the services included in each set) is purely made for organizational purposes meaning that a single site may support both functional sets. Also the role of the Broker differs slightly in each case.

In our first alternative consumer sites represent the end-users. Their capabilities are limited to looking up learning material, browsing and retrieving LOs. This is done via the mediation of brokers which keep metadata on available resources and manage the LO "inventory". Brokers can also be similar (but not identical) to Authorities as proposed in subsection 2.4 regarding their possible roles. Brokers, however, must always support defined mediation services but are not required to ensure their availability. The brokers' metadata "inventory" is built up as producers communicate with them subscribing offered LOs. LO subscription is assigned *only* to Producers. LOs are located and maintained physically at Producers. (This can also be called a *"client-broker-repository"* model). On discovering and selecting the requested LOs brokers should be viewed as producers, transparently to consumers.

The second alternative allows a single site to act both as a consumer and a producer. In particular, services for the creation of base and derived LOs are mainly local from the user's viewpoint. Of course a site can support just the consumer or just the producer services in which case collaboration of at least one consumer site and one producer site will be necessary. This will become clearer in the following sections.

Closing this subsection we present two views pertaining to the structure of the C-B-P architecture. As shown in Figure 3, Consumers, Brokers and Producers are linked together over the E-learning Network, which – reminding the reader – is actually composed and constructed by the service protocols and interfaces gluing together the Grid’s sites. Another view to this model can be related to a Super Peer model where super peers (brokers) are mediators in a localized setting (communicate with neighboring nodes – either in a hierarchical or flat structure). The slight difference is rather conceptual since implementers may find a number of common aspects in the two models. In the C-B-P Grid-like model (and Grid computing in general) the Grid infrastructure is used to allow communication between all participating sites. For example, a consumer may contact any broker it wishes, whereas the super peer concept usually considers super peers servicing a number of neighboring nodes and extending their functionality by contacting other neighboring super peers.



The C-B-P Grid-like Model

Super Peer–like Model

Figure 3: Views to the C-B-P Model

5.2 Alternatives

The first alternative considers a “client-broker-repository” setup for placement of services at consumers, brokers and producers, respectively.

Consumers represent the end-users. They contain and provide services for the actual users of the LOs, which may look up, browse and retrieve LOs. Consumers communicate with brokers in order to search and retrieve interesting LOs. A consumer may communicate with more than a single broker, although it is preferable that consumers simply view brokers as a unified middleware. Consumers can create derived LOs but *cannot make any changes to them* and thus produce new LOs. This role is assigned only to producers (see subsection 4.3). Consider consumers as the analogue of database users who may define views. Continuing this analogy, Producers may execute database insertions, deletions and updates whereas consumers will need to receive updates on their “views” and may, at some point, “materialize” them. In addition consumers may also collaborate to create a joint definition on their derived LOs but must eventually subscribe them with some producer.

It is proposed that consumers should support a modified core services set generally similar to the one described in 2.2. These include the **storage, communication and information** service with the addition of the **security** service. In this model the communication service acts as an “external interaction” module and as a r*equest managing* mechanism. The **caching** service must be available to all consumers. Since consumers represent users who will regularly interact, it is expected that they will also need to exchange stored content directly. In order for this to be possible the **access** service must also be available. As mentioned in 2.3 the access service will allow browsing remote LOs

Finally we need to include an additional capability for the consumers, that is, **visualization.** This will take care of presenting the available contents based on the information service personalization configuration.

Producers include a set of services that allows the creation of new LOs and communicate with brokers to report availability of these LOs. Additionally they may receive requests for LO integration and creation of derived LOs. Reporting new LOs is done by the use of the **registration** service provided by brokers.

Producers are primarily equipped with an **authoring** service, which will allow for the creation of new LOs. Note that since this is a service it can be also provided to consumers. Producers can also combine LOs, therefore producing new, derived LOs as specified in the User Requirements document. This will require the **integration** service – which must support some sort of an integration mechanism at the lower layer.

Although consumers may use authoring and integration services they cannot register new LOs. If an LO needs to be registered this is done by a producer. This way we maintain the architectural semantics: Consumers will always request LOs with broker mediation, since LOs can only be provided by producers. Additionally producers are the only ones allowed to change/update an LO. Therefore any updates or changes that consumers will need to make to LOs must be reported to providers. Producers can be seen as LO repositories to which access is provided to consumers through the mediators at a higher level (i.e. at the application layer through the authoring service.)

The brokers, beside core services, provide two other basic services: **searching** and **registration.**

Searching services will be extensively used by consumers. This is actually a distributed service which will require message exchanging by a number of brokers which will look up their LO metadata catalogs to return possible results to consumers. The searching technique can be based on the information service as proposed previously in this document. In any case, brokers will need to build up some kind of topology in order to efficiently process searching requests.

Registration service is provided for producers. During this process base and derived LOs metadata are reported and included to the broker's catalog. It is also supposed that a parallel catalog can be maintained by brokers that will map ontological information to metadata if this mechanism is not integrated within the metadata itself. This should allow for the same LOs to be viewed in multiple ways regarding different consumers.

Brokers should also support services as described in the case of Authorities presented in subsection 2.4. These include trail management, collaboration services as well as user logging and charging (since they mediate LO access to and from Producers.)

In the second service setup, a site may support both consumer and producer services. In this way LOs can be stored at the site at which they were created. Having a site support both sets of services makes the creation procedure faster, although in service-based models provision of services is transparent to the users. Consumers may make changes directly to their stored LOs. A site may only support consumer or only producer services. For example a teacher – the producer – that provides the base LOs, working with a group of students – the consumers – and together built up new LOs, which the producer stores and registers.

Proposed services for a consumer and producer site are similar to the first alternative except for the fact that they may coexist at the same site.

Replication

Another basic service (that will be provided by *Producers*) is **replication**. For the first alternative, with broker mediation (brokers mediate all requests) producers will be instructed to copy and maintain LOs located at another, remote producer site. Since producers are equipped with an updating and access mechanism this should be possible. In this way the role of the producers does not change (after copying an LO they immediately suppose that the replicated LO is simply part of their collection. We may also completely remove this responsibility from producers and have brokers copy LOs to producers and register them automatically – i.e. *producers:* copy by themselves and then register OR *brokers:* copy, add to producer (update) and register automatically.)

For second alternative we propose that replication is done at the broker sites. LOs can be cached at consumer sites but more permanent and updatable copies should be stored at brokers that can directly serve requesting users. Changes can be made at any consumer site (instead of producer sites) and updates are forwarded to the brokers currently holding the updated LO's replica.

5.3 Comparison

A brief comparison of three models is presented focusing on the distribution of services, dynamicity and managerial ease / independence aspects. The architectures are: the Pure Grid, the Peer-to-Peer /Grid hybrid and the C-B-P Grid.

The pure Grid model concentrates its services at a relatively small number of static locations (server sites) and thus is described as rather “heavy”. Much weight is put on the server side nodes whereas the Grid/P2P hybrid and C-B-P architectures distribute the services to all sites thus making them more flexible. In addition, the hybrid model allows for sites of various capabilities (e.g. low storage capability or processing power) to be included in the system since they may contribute any services they can support. On the other hand, the C-B-P model encounters some difficulties, since it requires a number of specific services to run on each site.

In all three models dynamic addition and removal of services is possible. This is directly linked with connecting and disconnecting a node to and from the SeLeNe. It is thus obvious that there are higher chances that the system can continue functioning normally with the Grid/P2P hybrid since services can be accessed from other supporting sites. On the other hand, locating a service should take more time with this strategy. In a pure Grid model, losing a server can cause problems to the system since a number of services will not be available. Still mechanisms for ensuring service persistence are possible. Concerning the “dynamicity versus service availability” trade-offs the C-B-P model seems to be the most promising.

Finally, regarding managerial ease the pure Grid and C-B-P models are highly manageable since service placement and sets of services are more strictly specified with the pure Grid being the stricter of the two. The Grid/P2P model is difficult to manage since each node is rather independent with respect to the services provided, but highly dependent on the services provided by other sites. The C-B-P is less dependent on other sites’ service provision.
In conclusion, it is still an open question to decide which is the most befitting hybrid architecture for SeLeNe. We venture to suggest C-B-P with Grd-like services. (?)

6 Acknowledgements

We would like to thank Konstantinos Spyrou and Elena Charalambous for their valuable assistance.

7 References

[1] Alan Fekete, Idit Keidar: A Framework for Highly Available Services Based on Group Communication

[2] Alexander Löser, Walfgang Nejdl, Martin Wolpers, Wolf Siberski: Information Integration in Schema-Based Peer-to-Peer Networks

[3] Beverly Yang, Hector Garcia-Molina, Comparing Hybrid Peer-to-Peer Systems

[4] Beverly Yang, Hector Garcia-Molina, Designing a Super-Peer Network

[5] DataGrid Data Management (WP2) – Architecture Report

[6] David De Roure, Nicholas R. Jennings, Nigel R. Shadbolt: The Semantic Grid: A Future eScience Infrastructure.

[7] George Papamarkos, Projects Related to SeLeNe

[8] Globus CoG Kit: Available at www-unix.globus.org/cog/java/

[9] Gribble, S., Halcry, A., Ives, Z., Rodrig, M., Suciu, D., "What can Peer-to-Peer do for databases, and vica versa?”

[10] Houda Lamehamedi, Boleslaw Szymanski, and Zujun Shentu, Ewa Deelman: Data Replication Strategies in Grid Environments

[11] Ian Foster, Carl Kesselman, Steven Tuecke, The Anatomy of the Grid Enabling Scalable Virtual Organizations

[12] Ian Foster, Carl Kesselman, Steven Tuecke: The Physiology of the Grid (OGSA)

[13] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

[14] Jim Gray, Pat Helland, Patrick O'Neil, Dennis Shasha, The Dangers of Replication and a Solution

[15] Karl Aberer, P-Grid: A self-organizing access structure for P2P information systems

[16] Kevin Keenoy, George Papamarkos, SeLeNe Report: Existing Learning Management Systems and Learning Object Reporitories

[17] Kevin Keenoy, SeLeNe - Preliminary Report: Learning Objects, Meta-Data and Standards

[18] Leonidas Galanis, Yuan Wang Shawn, R. Jeffery David J. DeWitt: Processing XML Containment Queries in a Large Peer-to-Peer System

[19] Luciano Serafini, Fausto Giunchiglia, John Mylopoulos, Philip A. Bernstein, The Local Relational Model: Model and Proof Theory (2001)

[20] Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu, Data Management for Peer-to-Peer Computing: A Vision

[21] Project JXTA. Available at www.jxta.org and Edutella at edutella.jxta.org

[22] SeLeNe Kick-off Meeting Presentations (Summary of the SeLeNe Kick-Off Meeting)

[23] SeLeNe Project User Requirements - Draft 1.0

[24] The Garlic project available at http://www.almaden.ibm.com/cs/garlic/homepage.html

[25] The SWAP System available at http://swap.semanticweb.org/public/index.htm

[26] The TSIMMIS project available at http://www-db.stanford.edu/tsimmis/tsimmis.html

[27] The Universal Description, Discovery and Integration Protocol available at www.uddi.org

[28] Wee Siong Ng, Beng Chin Ooi Kian-Lee Tan, BestPeer: A Self-Configurable Peer-to-Peer System

[29] Wee Siong,Ng, Beng Chin Ooi Kian-Lee Tan Aoying Zhou, PeerDB: A P2P-based System for Distributed Data Sharing

[30] Wolfgang Hoschek, Peer-to-Peer Grid Databases for Web Service Discovery