



Review of the Golm Architecture Discussion

Author(s):	Andre Merzky, Gabrielle Allen
Document Filename:	GridLab-00-TB-0001-GolmArchReview
Work package:	Technical Board
Partner(s):	ALL
Lead Partner:	ZIB
Config ID:	GridLab-00-TB-0001-1.0
Document classification:	INTERNAL

Abstract: This Documents provides a review of the discussion about the general GridLab architecture. This discussion took place during the joined GAT and TB meeting from March 11-12 in Golm, Germany.





Contents

1	Architecture Discussion	2
1.1	Constraints to the Global Architecture	2
1.2	The Main Contradiction	4
1.3	Major Opinions	5
1.3.1	The Poznan Interpretation	5
1.3.2	The Golm Interpretation	6
1.4	Proposed Compromise	7
A	Terminology	8
B	Yulos Comments	8
B.1	Extended Description	8
B.2	Advantages and motivation of our approach	9
B.3	Other comments	10
C	Toms Comments	10

1 Architecture Discussion¹

During the 2 days of the joint GAT and Technical Board meeting in Golm on March 11-12 2002, the discussion about a general architectural concept for the GridLab project emerged very frequently, and in various facets. Therefore, no detailed notes about that discussion are provided (these would be incomplete and probably VERY confusing), but we will try here to review the major arguments, and the way to the architecture proposal the meeting ended up with.

1.1 Constraints to the Global Architecture

First we want to review some constraints to the architecture, originating from various sources and assumptions:

- **Proposal Objective:**
The architecture must allow the implementation of a GAT which is usable in real world (i.e. today's) grid environments.
- **Proposal Objective:**
Services should be modular and swappable.
- **Utilization of Globus Services:**
The architecture must allow the incorporation of 3rd party services.
- **General Consensus:**
The architecture must be cleanly layered.

Further constraints come from different WGs and the end user groups²:

- **Security:**
The architecture must allow incorporation of security on global level.
- **Cactus/Triana:**
GAT must be usable in disconnected/firewalled environments.
- **GAT:**
Control and fail safety must be existent on all layers.

As known to everybody, basically we all agree about following points:

- the GridLab architecture is a layered architecture
- it consists of three main layers:
 - User Layer (Application + GAT)
 - Service Layer³ (GridLab Services)
 - System Layer (Core Services: Globus, 3rd party)

¹For clarification of terms, see Appendix A

²The proposal implies that “The development of the GAT API and GAT implementation should be driven by application requirements”.

³Please check the information about the term *Capability* in the Glossary, Appendix A!

- all network interactions are to be secured via a global security infrastructure (on the Service Layer)
- fail safety, control and intelligence are inherent to all layers, by possibly different means
- any interaction from the User Layer to the System Layer should pass through the Service Layer

The resulting general concept is depicted in Figure 1.1.

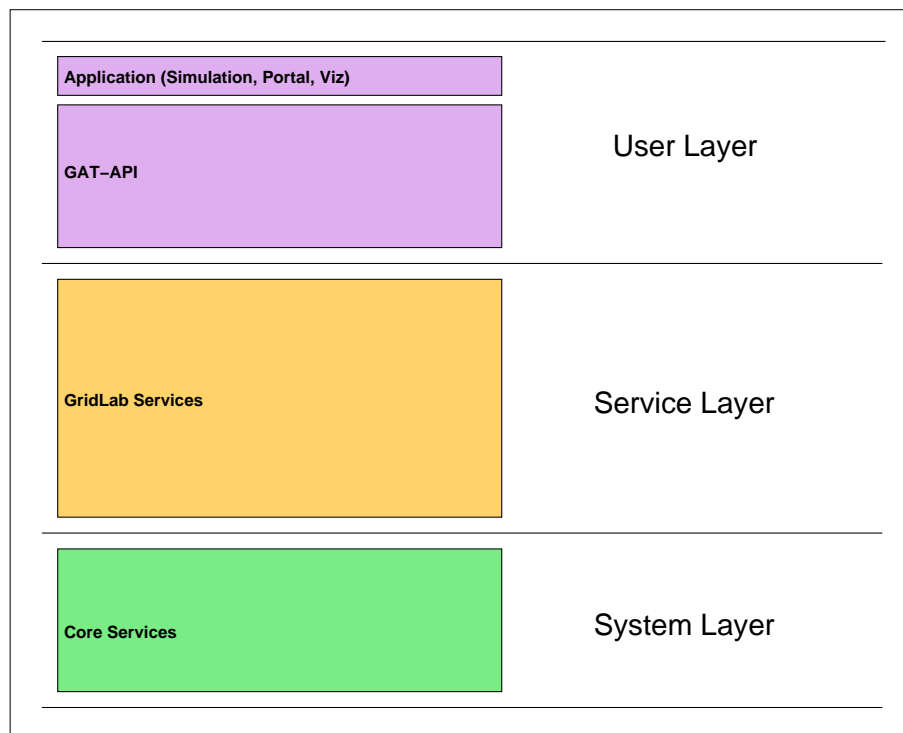


Figure 1: **The General GridLab Architecture:** This image shows the layered architecture of the GridLab project in very general terms.

1.2 The Main Contradiction

A major difficulty remained, and gave reason to lengthy discussions on the mailing lists and during the meeting. The origin of the problem is a contradiction between two groups of requirements:

1.
 - the GAT must be usable in disconnected environments
 - control and fail safety exist on all layers

2.
 - is cleanly layered
 - allows incorporation of security on global level

To try to explain the contradiction:

IF
an application runs in a disconnected environment,
OR
a GAT fails to utilize some Service Layer service,
OR
a user *chooses* not to utilize some Service Layer service,
THEN
the need for service functionality must be fulfilled by GAT w/o Network Access.
FAILING IS NOT AN OPTION.

If service functionality is provided w/o network access, but by Local Services⁴, this has two implications:

- (1) GAT is not only talking to network services in the Service Layer, but also to non-network services (Local Services).
→ This is weakening the pureness of the architecture layers.
- (2) security can not always be guaranteed via a network service.
→ This seems to circumvent the global security policy.

⁴I want to make clear, that Local Services are **not** for prototyping purposes only! This impression seems to be widespread, but is not correct!

1.3 Major Opinions

There are two major opinions in the group about this problem (and many variants in between of course).

1.3.1 The Poznan Interpretation – Figure 1.3.1

Layer 2 (Service Layer) contains only Network Services, any Local Service is completely contained on User Layer. This keeps architecture clean, and enables global security policy.

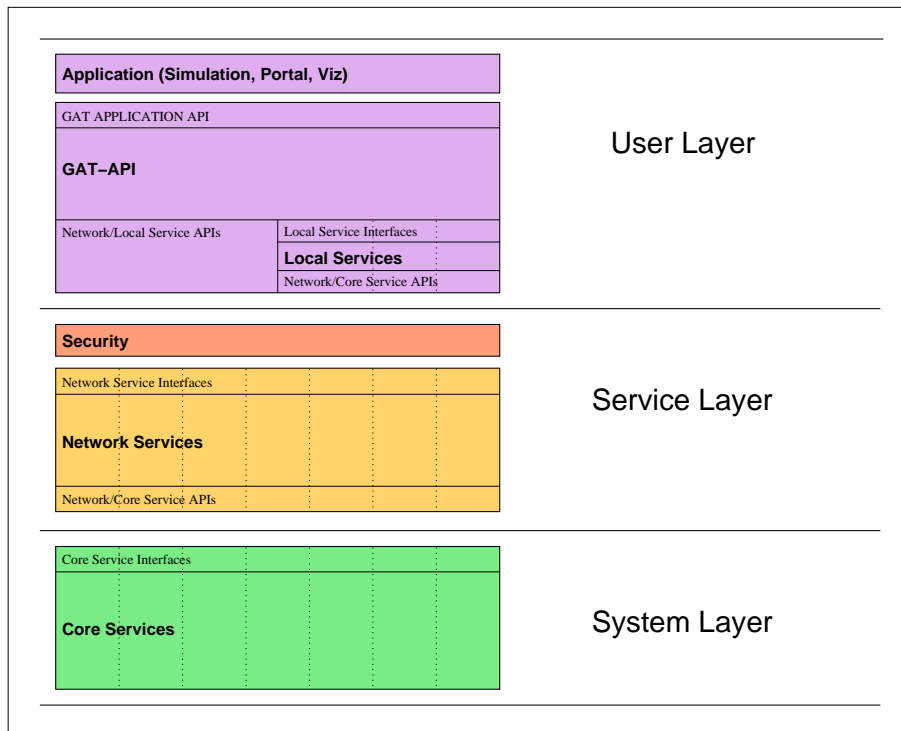


Figure 2: **Poznan Proposal for GridLab Architecture:** This image shows the layered architecture of the GridLab project as proposed by the Poznan group. Local Services are only of concern to the User Layer. This scheme preserves a very clean Service Layer with global security.

1.3.2 The Golm Interpretation: – Figure 1.3.2

Layer 2 (Service Layer) incorporates a single concept (Services) and does not differentiate between Network Services and Local Services. Both concepts can be treated in a uniform way.

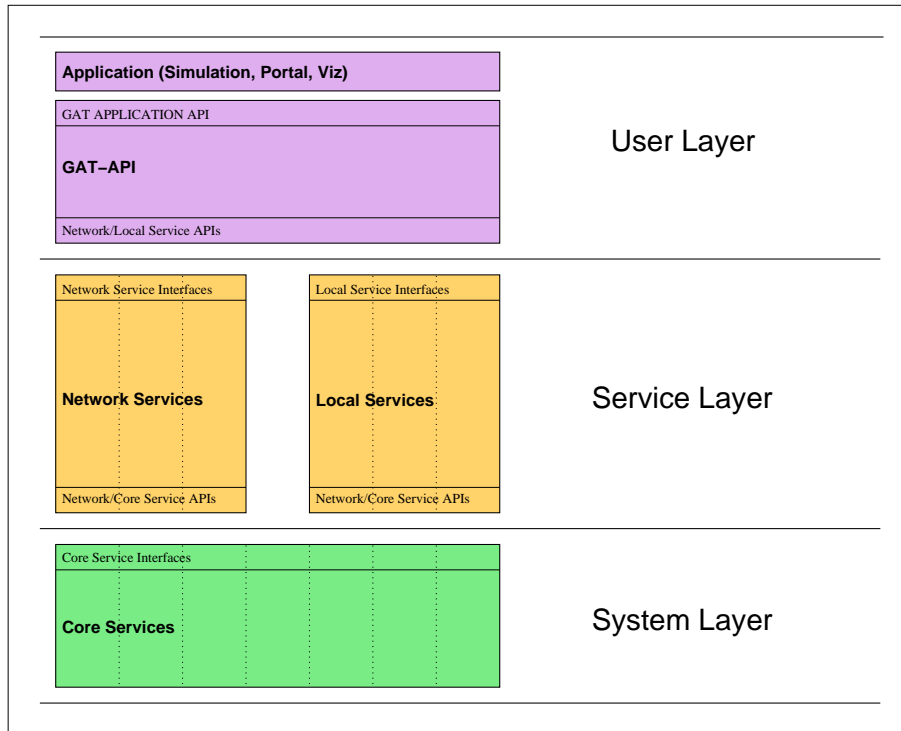


Figure 3: **Proposal for GridLab Architecture by Golm Group:** This image shows the layered architecture of the GridLab project as proposed by the GAT/Cactus group: Local Services are equated to Network Services (from an application’s view point, the main interest is that the functionality of the GAT API call is provided). This seems also to preserve clean layers, but the differences between the services imply difficulties in designing a global security infrastructure, and a consistent scheme of communication inside the Service Layer.

Obviously, both opinions have great merits, but they don’t go together!

1.4 Proposed Compromise

Omitting a great deal of following discussion, the following scheme was proposed as a compromise (Figure 1.4):

- The Service Layer contains Network Services and Local Services
- Local Services are exempted from the security policy, but only as long as their actions stay local.

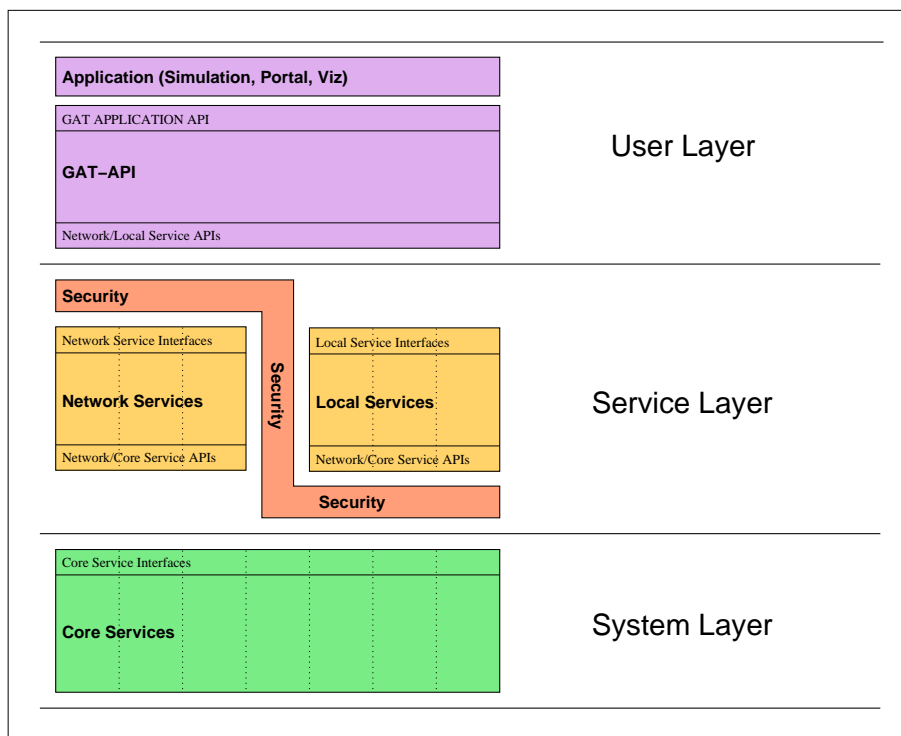


Figure 4: **Compromise Proposal for GridLab Architecture:** As a compromise, the shown architecture was proposed. While preserving a uniform notion of Services (including Local Services and Network Services), a global security infrastructure embeds all non local entities.

This seems to fulfill the requirements of the Golm group completely, but incorporates the security principles of the Poznan Interpretation.

There are still major restraints from this compromise due to the fact that the Clean Layers Principle seems to be violated.

A Terminology

In order to be able to talk about the architecture problem, I will define some terms here. These are not necessarily definitions shared by the majority of the group, but are (in my opinion) best fitted for this very problem.

Capabilities: This term is not used in the text. The term *capability* or *capability provider* came up during the struggle for a definition of *Services*. Throughout the discussion, the Service Layer was temporarily renamed **Capability Layer** to allow to include local capability providers (now Local Services) into that layer. After the agreement on the terms *Network Services* and *Local Services* the layer was renamed back to **Service Layer**.

Service: An entity providing some capability/functionality (John Shalf)

Network Service: A network enabled entity, providing some capability/functionality (Julo Pukacki)

Local Service: A non-network enabled entity, providing some capability/functionality (Tom Goodale)

Secure Service: A Service or Network Service obeying some global security policy (following the Security WP, for Grid Security this implicates network interaction!)

B Yulos Comments

This Appendix contains the specific comments of the PSNC team regarding the GridLab architecture and the document preceding this Appendix. We would like these comments to be discussed again by the Technical Board during the TB meeting in Berlin.

B.1 Extended Description

At this point we would like to explain more precisely our original point of view on the GridLab architecture model. By "original" we mean the very beginning of our discussions, independently on what we have agreed upon during the meeting in Golm. By the "architecture model" we mean the general framework for the architecture of the system. This general framework will be then used to put into it the components of the GridLab architecture.

Main problem concerns the middleware layer in the GridLab architecture – the Services Layer. In our opinion there is no need to introduce the term "Local Services" in the architecture model. Our idea of the architecture model is illustrated by Figure 1.1 not Figure 1.3.1!

So called "Local Services" should be treated as a part of implementation of GAT-API which is needed to serve "unnatural" situation of work on disconnected computer (or in "behind-firewall environment" without access to Services) – which we find difficult to call grid computing. From a general architecture model's point of view Local Services can be ignored.

The construction of architecture model should be driven by essential idea of GridLab project, not by need of serving special situations.

The way we would like to define the services is:

- Service is a network-enabled entity that provides some capability.

The term "network-enabled" means that a service is available for any other entity through network. This is one of the most important features of a service: it provides its capability for any other party in grid environment. This is the main idea of service-oriented approach for building distributed systems.

- Application can exploit grid resources only by using services

This assumption means that a service hides from an application the complexity of methods for accessing the resources. This is done by providing a layer of capabilities which can be used by application in some uniform way (one technology of service implementation)

All what is needed for application is a knowledge how to access the Services Layer.

- Functionality of services is forced by application's requirements.

This obvious assumption means that creation of services should be driven by needs of the application scenarios which are to be made possible by the GridLab project.

- System created with set of services should be as much reliable as possible

This assumption means that a system (built with services) should be implemented in a way that minimizes the possibility of occurring the situation when a service cannot be accessed.

So an application should trust reliability of services, similar way as using IP network applications trust reliability of DNS service, and not trying to implement its own copy of DNS in case of failure in access to it. (Obviously, lack of DNS is only a limitation of functionality and you can always use IP addresses. Yet, we can go further that way: what about lack of appropriately configured routing?).

The conclusion is that in some cases failing is an option, because it is not good idea to fulfill functionality of all services, for application running on disconnected machine. Fail-safe mechanisms should be implemented in different way, for example through efficient replications of services, instead of duplicating their functionality.

If application (not on disconnected machine) wants to use some remote resources, and fails to access appropriate service from Service Layer than operation fails, or mechanisms dedicated for local run are used.

On the other hand if application runs on disconnected machine operation does not fail but some mechanisms of supporting local run are used by default.

- Access to services is controlled by global security policy.

The overall security level of grid environment relies mainly on consistency of security infrastructure. The main requirement for creating such infrastructure is clear and unambiguous architecture. For some further comments, please view the updated version of security requirements document, which should be available next week.

B.2 Advantages and motivation of our approach

- Basic, and the most important feature of grid is that it consists of distributed resources.
GridLab is a grid project so it should be focused on grid as a set of distributed resources. That is why in our opinion general architecture model of GridLab should be driven by network aspect of grid;
- Idea of service-oriented approach (with services as intelligent components placed in network) is the way that modern distributed systems are developed.
- Scalability – there is no problem to add new service
- Interoperability – it is easy for any other system to use GridLab Service;

- This approach does not prevent from satisfying the "slow startup" requirement or the requirement of running an application locally or remotely without recompilation which both seem to very important for application users.
- This model architecture can be easily mapped to the Grid Model Architecture given by Ian Foster et. al.

B.3 Other comments

Meaning of fault tolerance mechanisms on different layers.

User Layer: if application fails to conntact some service a fault tolerance mechanisms provides ability for application to run locally or pause execution until services are available;

Services Layer: fault tolerance is a set of mechanisms needed to make service infrastructure reliable (e.g. service replication);

System Layer: generally out of scope of the project;

- It is not good idea to put security block to general architecture model! Security stuff should be treated as some kind of property of individual layers, used for describing them, and interactions between them.
- Introducing term "Local Services" can cause duplication of functionality (the same functionality implemented in Local and Networked Services).

Yulo.

C Toms Comments

On page 4 you have:

```
...  
OR  
  a GAT fails to utilize some Service Layer service,  
OR  
  a user chooses not to utilize some Service Layer service,  
THEN  
...  
  FAILING IS NOT AN OPTION.
```

The first two statements are only correct if local services are not Service Layer components.

Failing is always an option if a service doesn't exist or exists but cannot do what the user asks of it. What we want is the flexibility of not always having to go over a network link to contact a service. All services could be implemented as network services, but that is an overhead both in development time and at run time, if the service always returns "localhost" as a resource name, for instance.

On page 4 you have:

(1) ... → This is weakening the pureness of the architecture layers.

This statement would only be true if local services are not part of the Service Layer. See above.

(2) security...

This statement is false. A local service is implicitly authenticated because it is running in the same process space. For local service access to external things, it is just as well or badly as authenticated as the equivalent service implemented as a network service.

My preferred diagram is actually figure 1. I do not see why we need to make a distinction at this level between network and local services.

As I see it we have three entities 'user', 'service' and 'system' which communicate via defined interfaces.

'User' asks 'service' to perform 'operation 1'
'Service' asks 'system' to perform 'operation 2'

All the user does is:

```
result = GAT_Foo (arguments);
```

'service' then does what it needs to do to return 'result' to 'user'. I can implement GAT_Foo either as

```
call do_something  
return result
```

or as

```
open_secure_network_connection_to_external_process  
send request to external_process  
  
external_process does  
  call do_something  
external_process returns result  
  
close_secure_network_connection  
return result
```

Hence I see no distinction at an architecture level between the two cases, only at the implementation level.

If you want to make such an arbitrary distinction, then figure 3 is ok, as it preserves the fact that network and local services are functionally equivalent.



Figure 4 is a bit of a mess, but if it makes people happy to put in the diagram explicit reference to the fact that all network communication is done via secure links, I can live with it. I also think we could leave the line out of the diagram and say it in words.

I do think it is very important that we state somewhere prominent that all network communication is authenticated and over secure channels, but I don't think that an architecture layer diagram is necessarily the best place for it.

In the comments to the proposed compromise, you need to clarify what you mean by 'clean layers principle seems to be violated'. The layers in the diagram don't violate the principle, but of course the security blob does.

In the Terminology appendix, the definition of local service is wrong. It should be:

Local Service: An entity providing some capability/functionality which is not connected to the application by a network connection.

I think the network service definition should also be similarly changed to

Network Service: An entity providing some capability/functionality which the is connected to the application by a network connection.

I hope these comments are clear enough.

Tom