034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D3.5

Final Report on achievements and results of the first 18 months' period

Due date of deliverable: 2008-03-31
Actual submission date: 2008-03-31
Document code: Phosphorus-WP3-D3.5

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
Forschungszentrum Jülich GmbH

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

Table of Figures

0 Executive Summary

This report summarizes the achievements of WP3 during the first 18 month period, including in particular the second version of middleware and applications as defined in milestones M3.5 and M3.6.

1 Overview

This report summarizes the achievements of WP3 during the first 18 month period, including in particular the second version of middleware and applications as defined in milestones M3.5 and M3.6.

WP3 is separated into a number of tasks which separate different themes within the package. These tasks are:

1. Adaption of Applications
2. Middleware
3. Interfacing between G²MPLS and Resource Management System via G-OUNI
4. Integration and Evaluation of Middleware

The report builds on this separation of tasks, thus section 2 reports on the adaption of applications, section 3 on middleware, section 4 describes the work done so far in the interface between G²MPLS and RMS via G-OUNI, and section 5 describes the integration and evaluation of middleware. The task descriptions are complemented by a report about the deliverables, which were due during the first 18 month period. The entire report is summarized in section 6, which also draws some conclusions.

1.1 Work package objectives

The work-package objectives during the first 18 months timeframe were:

- Formulation of user requirements to contribute to the definition of the functions and services to be implemented by the overall project
- Development and provisioning of the middleware services to orchestrate requested for an application, i.e. network, compute resources, visualisation devices, etc.
- Integration with network layer developments (Control Plane and Grid-NPRS) with Grid middleware to enable high performance applications running efficiently in the test-bed

- Adaptation of selected Applications to showcase the benefit from the integrated test-bed environment developed in the project

- Definition of the properties that network resources (services) should have in order to integrate seamlessly into an environment with other grid services

- Enabling user and applications to automatically get access to the resources that match their requirements e.g. in terms of performance and QoS.

- Enabling users and application to plan resource usage in advance and to dynamically adapt the resource usage to the changing requirements of the application thus avoiding expensive waste of resources.

- Managing all resources required to run an application in an integrated manner with a single service based interface towards to user or the application respectively.

- Testing the provided networking functionality based on application use cases defined in the first project stage and deriving requirements for extensions and enhancements to be carried out in the second 12 month period from the test-bed experiments

2 Task 3.1 – Adaption of Applications

This task deals with the adaption of applications as a result of first evaluations in the Phosphorus global and local test beds. Also, applications needed to be extended due to the specific use cases layed out in Phosphorus.

Initially, use cases for all applications participating in Phosphorus were developed and documented. The use cases exploit the specific features that Phosphorus offers. WP3 also organized an internal face-to-face meeting and meetings with other work packages, namely WP1, WP2, and WP6, which took place in December 2006 in Amsterdam. These meetings primarily dealt with interfaces between work packages.

Early implementation work commenced shortly after the completion of the application use cases. Features and general enhancements to the existing applications were developed, which could be foreseen without the existence of the detailed use cases. At the same time, extensions and design changes of the applications and middleware were defined based on the application use cases.

Further development started following the initial implementation work and the specification of extensions and design changes of applications based on the application use cases. This development is ongoing and guided by the requirements and use cases defined initially as well as the test conducted during the course of the reporting period.

First tests were conducted when the full functionality of the global Phosphorus test bed wasn’t yet available. These tests were done over general internet connections. The reason for these preliminary tests was that it was expected that even though they weren’t conducted via reserved network links of a dedicated network, they could already unveal some issues in the applications that could be dealt with even before the availability of the full global test bed.

The global test bed, comprised of the local testbeds and the links between them, was available shortly after tests via internet connections had started. Testing over internet links first also created the opportunity to compare the performance of generally available and dedicated networks.

The following sections describe the achievements and results specific to each of the applications in detail.

2.1 KoDaVis

2.1.1 Introduction

KoDaVis enables users from various remote sites to collaborate in visualising scientific data sets. It is made up of several components:

- Data server
- Collaboration server
- Visualisation client
- UNICORE 6 Client

For the purpose of the developments and tests conducted with this application, the data server and collaboration server components have been installed at FZJ. Client components (visualization and UNICORE) were deployed at various other sites. Figure 2.1 shows an experimental setup of two visualization clients both located at PSNC for demonstrational purposes. The visualization clients could as well be located at geographically dispersed sites. Any action taken by the user of either of the clients will be displayed in the other clients as well and thus allows the collaborative visualization of data. KoDaVis is a demanding application both in terms of latency and bandwidth. For high latencies, the user experience quickly deteriorates and may render the system unusable. At the same time, large sets of data may have to be transmitted, which requires high bandwidth links between KoDaVis servers and clients (~700 Mbit/s).

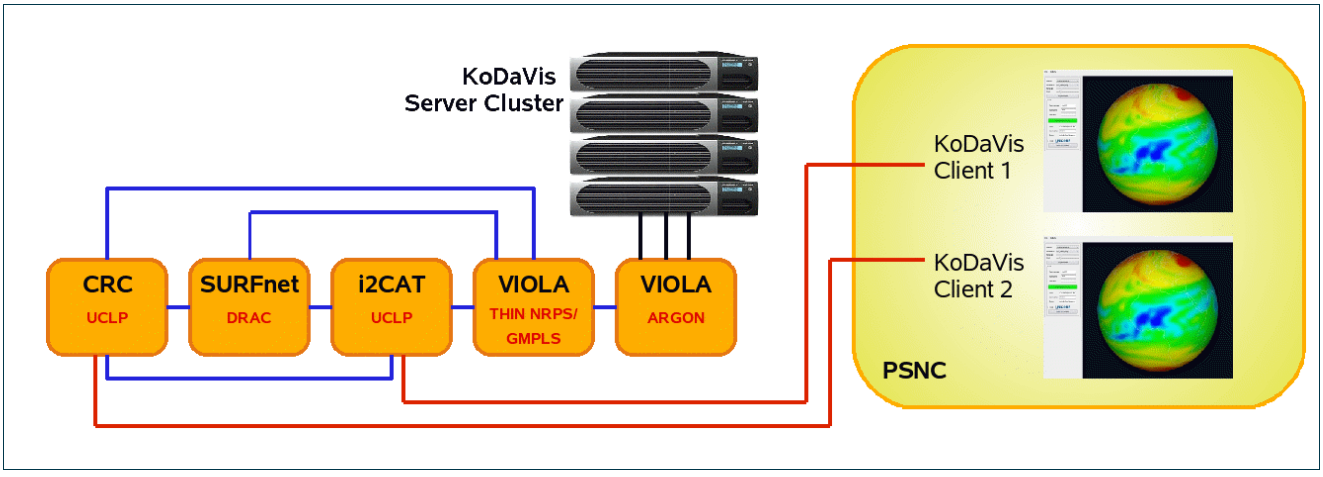


Figure 2.1: Setup of a KoDaVis experiment, connecting two visualization clients at PSNC over different routes to the KoDaVis Server cluster in Juelich.

2.1.2 Project development

For the KoDaVis application, a use case and call for participation in testbed experiments were written at the start of the project. The use case guided the design of changes and enhancements to the application as well as the extraction of requirements for the testbed experiments. Application requirements were communicated to WP6 partners to guide the planning of the testbed setup.

As for most applications in the Phosphorus scenario, KoDaVis needs application specific support in the middleware and its clients. In the Phosphorus testbed, UNICORE has been chosen as the middleware to be integrated with KoDaVis. To this end, the changes to both the KoDaVis application and the UNICORE middleware have been described and further refined in use cases in January 2007. The development work on the KoDaVis application itself commenced with the implementation of VTK visualisation classes. The first installation of the application in the local testbeds of VIOLA and PSNC was done as early as April 2007.

Following this, the design work for the UNICORE 6 client support of the KoDaVis application started. The first open source KoDaVis client based on VTK and QT was completed in May. The KoDaVis data-server and client were again deployed in the VIOLA and PSNC local testbeds. In order to unveil performance bottlenecks, the data-server was enhanced to provide network performance data.

KoDaVis was installed in the local testbed of the University of Essex in July 2007. Enhancements of the visualisation client were introduced, like the optimization of ssh-tunnel communication and the communication interface towards the UNICORE client. The data-server received an interface to enable the control by the UNICORE service.

Also, the UNICORE client and graphical KoDaVis client had to be coupled. Together with the integration of the data server with the UNICORE 6 server side, this made a fully workable solution, which was also tested via the Phosphorus testbed between PSNC and FZJ. In addition to the first tests, the solution has also been demonstrated at SC'07 in Reno. For this purpose, KoDaVis was installed on demo nodes in Poznan and the use cases and demonstration procedure has been documented for the booth personnel at SC'07.

Once the integration of the KoDaVis components, data server, collaboration server, and graphical client, with the UNICORE 6 environment had been completed, tests were conducted to improve or fine-tune the utilization of the available bandwidth. KoDaVis was the application, which was demonstrated at the first annual project review in Poznan on December 13th, 2007.

A report about these initial tests can be found in [D6.4].

Even though all components were integrated quite well, the collaboration server needed some enhancements, which were introduced at the time of testing. The server side of the UNICORE KoDaVis integration has been shown to be sufficiently flexible, when it had to be installed on a different machine at FZJ due to a hardware failure in January 2008.

The original intention for providing performance data was the discovery of performance bottlenecks in the connections between the clients and the data server. In KoDaVis, the limiting factor of a collaborative visualisation session is the slowest of these connections, thus the information about this is important to improve

the user's experience. The initial implementation of the provision of performance data created a severe performance bottleneck itself, as there was only a limited means of influencing the granularity of the performance information. This is currently bein enhanced by the second implementation, which removes this restriction. The display of this information is also being implemented in the UNICORE client.

The performance data gained by this extension was used to identify the component of the distributed application that limits the overall performance. It turned out that the serialization of data delivery by the data-server to the visualization clients was the main bottleneck. Therefore, we have started to develop a parallel version of the data server. While it already provides the general functionality, the current prototype lacks the required stability to use it for general deployment.

2.2 WISDOM

2.2.1 Introduction

The WISDOM use-case consists of the virtual screening techniques AutoDock and FlexX, computing compounds of large-scale molecular dockings on targets implicated in diseases like malaria.

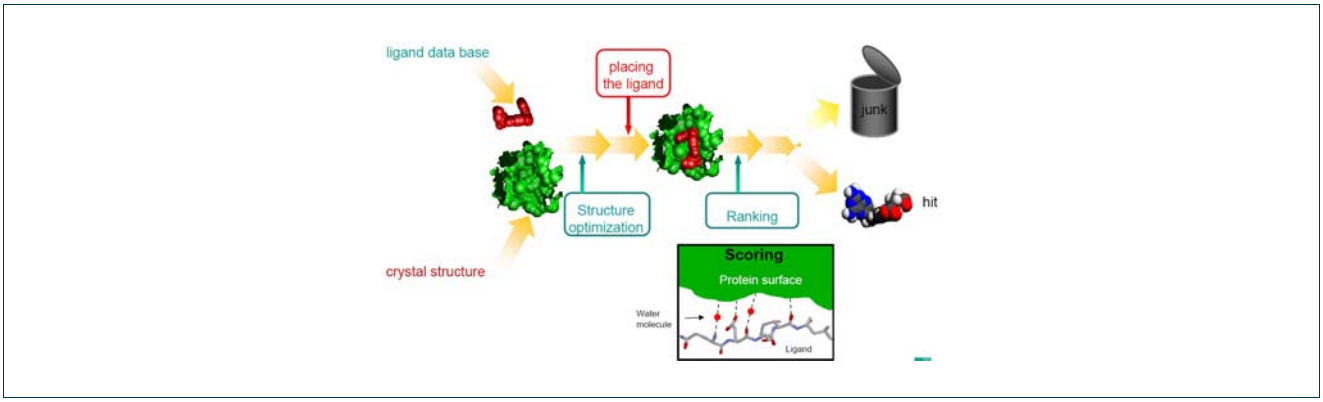


Figure 2.2: WISDOM Overview

AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of a known 3D structure.

FlexX is an extremely fast, robust, and highly configurable (FlexX-able) computer program for predicting protein-ligand interactions.

More details about the PHOSPHORUS use case WISDOM and its applications are described in deliverable D3.3.

2.2.2 Planned actions

Based on the experiences made during the EGEE WISDOM data challenge the goal in Phosphorus is to implement the WISDOM workflow with UNICORE in order to improve correctness, completeness and reliability of results. Thus both the distribution of input data and especially the transfer of result data of the millions of docking processes from the participating sites back to the user's site were complex, cumbersome and therefore resulting in significant data losses. As a consequence of this, in PHOSPHORUS we will concentrate on the implementation of a perfect workflow for WISDOM and the test of network and middleware functionalities.

After analysis of the WISDOM workflow, first tests are planned on the PHOSPHORUS testbed to identify application specific requirements for middleware (MSS) and UNICORE 6 workflow support. After the availability of needed features and support, MSS/UNICORE 6 enabled executions of such workflows will be performed.

More details about the WISDOM codes' workflows and requirements for execution are described in deliverable [D3.4].

2.2.3 Achievements and results of the first 18 months' period

The work on the PHOSPHORUS use case WISDOM started with a call for participation. Finally, four sites (Fraunhofer SCAI, JSC Juelich, PSNC, UESSEX) offered to provide compute resources and support to execute the WISDOM use case and its applications AutoDock and FlexX.

The next step of preparations was to inspect the first EGEE Data Challenge and their WISDOM workflows and identify necessary changes for the PHOSPHORUS environment. As a result of this analysis different WISDOM workflow phases were identified, which must be enabled based on MSS and UNICORE 6: Stage-in, execution and stage-out phases for each workflow. So, we are concentrating on automation of the WISDOM workflow.

- The WISDOM stage-in phase consists of licensing (for FlexX only) and distribution of input data from a specified input server. In the case of FlexX, these are RDF and Mol2 files. AutoDock uses Grid maps and DPF files for execution (see below).

- The WISDOM execution phase needs a good job control / monitoring and in cases of errors job resubmission functionality.

- The WISDOM stage-out phase includes the transfer of the local output data (FlexX Mol2 files, AutoDock DLG and Mol2 files) after termination from each site to a specified output server into a directory hierarchy or a database.

Additionally there are pre- and postprocessing actions, e.g. input data format conversion, output data analysis, output data filtering.

For this first test phase of WISDOM a test data suite from BioSolveIT was selected. So, in the beginning local installations and tests are done on all four PHOSPHORUS sites (see above) to test the docking software modules. For these runs the BioSolveit flexx-200 Testdata suite was used for both the applications AutoDock

and FlexX. The BioSolveit flexx-200 Testdata suite is a subset from the PDB, where each ligand was separated from the protein-ligand-complex and hand-fixed concerning protonation, aromaticity, delocalisation, and formal charges. After these first modifications both WISDOM applications use the same input data set with their specific data formats. Additionally start scripts and data paths were modified to realise a good basis for UNICORE 6 executions. Finally code installations and test data sets were available on all participating PHOSPHORUS sites.

AutoDock was installed without any problems on all participating sites: UESSEX (Wisdom node), Fraunhofer SCAI (PACK cluster), SC Poznan, JSC Juelich (Cray-XD1). After the Cray-XD1 system at JSC Jülich was becoming unavailable due to an unrecoverable hardware failure, a new installation on the replacement system (Juggle) will be used for further work.

FlexX was installed on three participating sites: UESSEX (Wisdom node), JSC Juelich (Cray-XD1), Fraunhofer (PACK cluster). Special port configurations are done on all sites to realize the contact to the Fraunhofer FlexX License Server system from all sites. At JSC Jülich the replacement system (Juggle PC Cluster) will be used for further work. Unfortunately no FlexX installation was possible for the platform of SC Poznan (Itanium cluster), because this hardware is up to now not supported by code owner BioSolveIT.

Currently, the workflow engine of UNICORE 6 is only available in a prototype status. Thus the execution of both codes is actually done script-based. These scripts can be executed based on reservations of the available local resource management system. Different scripts are produced for the execution of AutoDock and FlexX jobs, which use the schedule file of the RMS for the execution. The jobs are running by ‘ssh calls’ on the reserved computing nodes. This need of alignment between node reservations and script files will be changed soon with a better UNICORE integration.

On all sites the file system is mounted to all computing nodes and therefore is identically on all nodes. To avoid writing to the same output files, the naming of the output files uses the hostname of the executing node. Thus output-data files are produced based on the schedule file in a node-specific way. This naming convention is also needed to transfer all results back to one single node, the so called output server node.

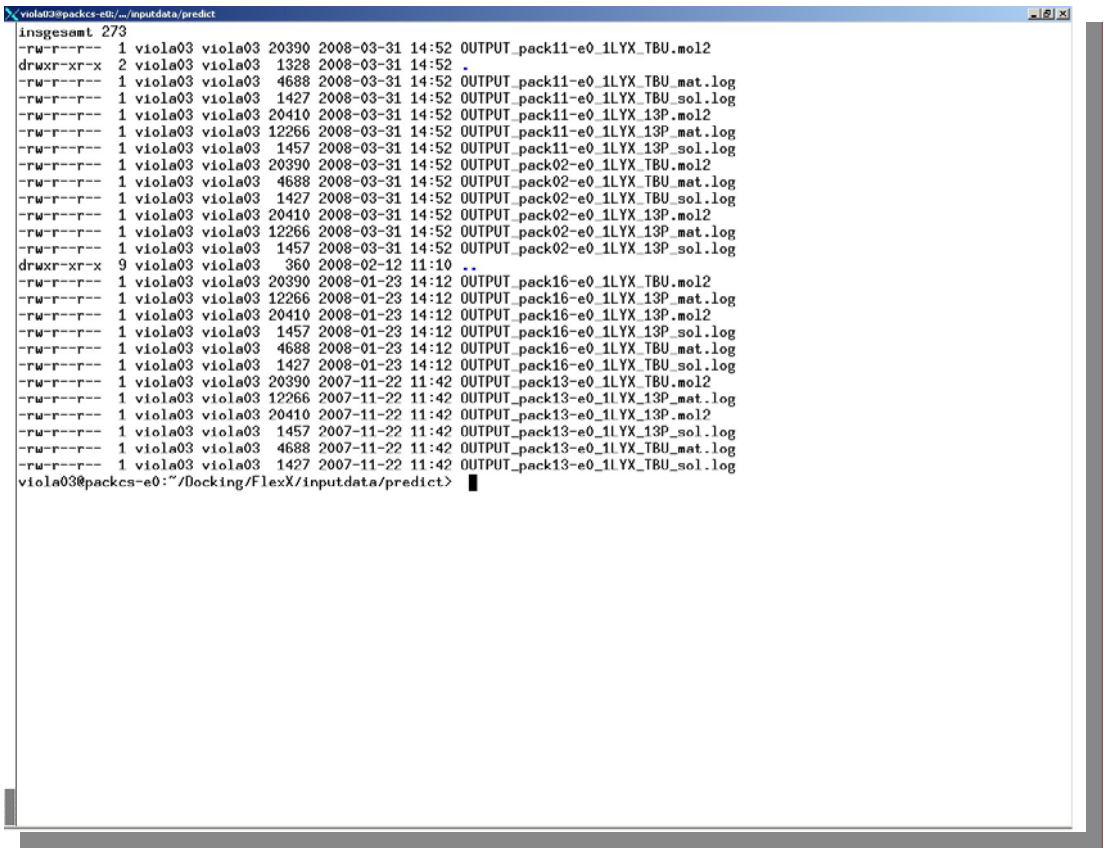


Figure 2.3: Screenshot after FlexX jobs with local output files on PACK cluster

Further post processing functionalities should be supported after additional tests in the next project phase.

So, after the phase of local tests and adaptations, first GRID-executions were performed on the PHOPHORUS Grid testbed.

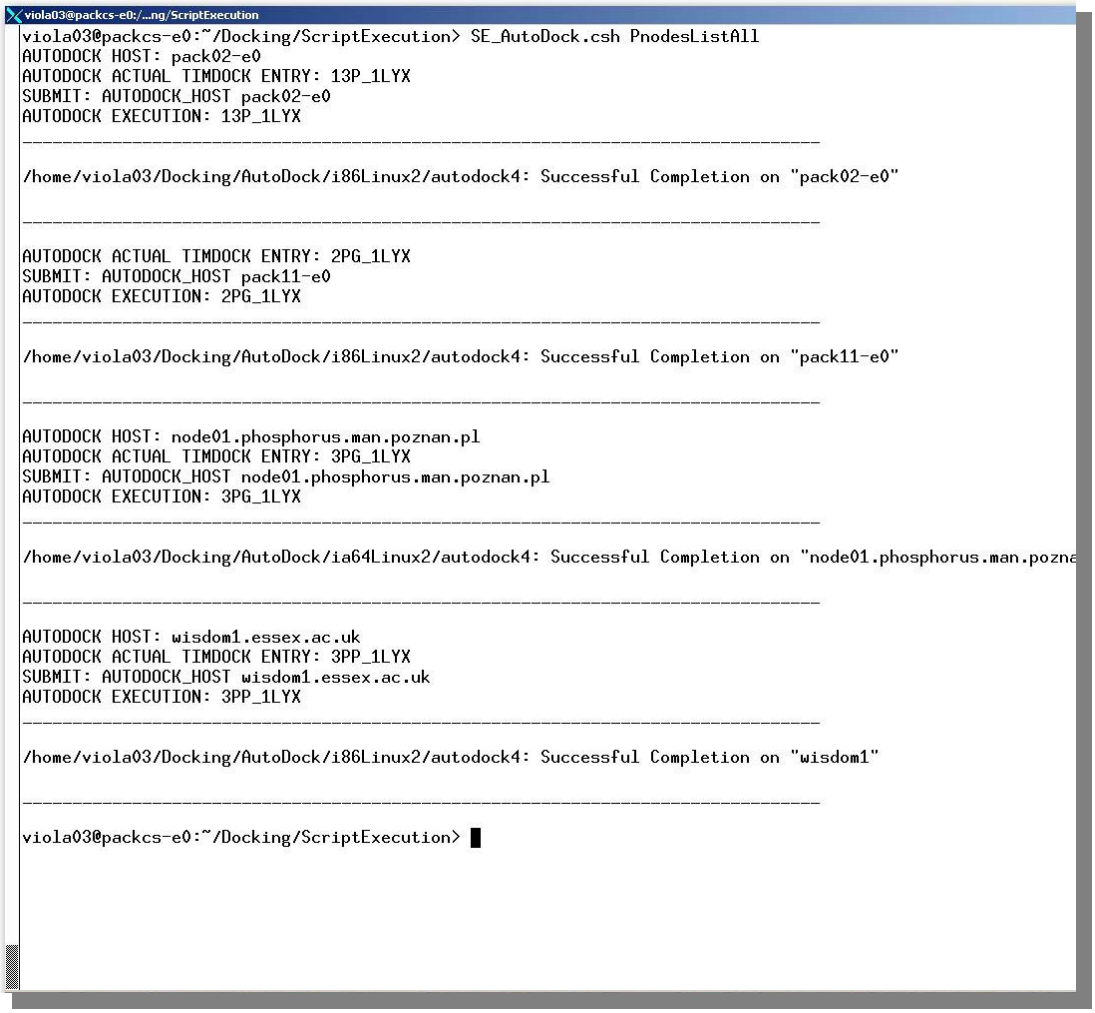


Figure 2.4: Screenshot after script-based AutoDock run on PHOSPHORUS testbed

In this second phase deliverable D3.4 was produced and requirements for automatic WISDOM workflow executions were described. Additionally, application parameters were specified, which are needed for execution and data distribution, and which would improve user friendliness too (e.g. changeable standard parameter settings). Most of these settings will be realised as parameters of the script for WISDOM executions.

These different scripts for executions and data distributions are tested on the PHOSPHORUS Grid testbed, and will be needed for the automatic execution of WISDOM workflows with MSS and UNICORE 6. Thus further work and tests are needed in the following project phase to enable WISDOM workflows to be executed by MSS and UNICORE 6.

2.3 TOPS

The TOPS application has been chosen as application because it is ideally suited for testing and demonstration of a real-time, high bandwidth application within the Phosphorus testbed. It is a point-to-point image streaming application for ultra-high resolution images (image sizes up to 100,000x100,000 pixels), intended to be used by researchers working on very large datasets at computing centers like SARA. Because these large datasets are impossible to mirror locally, methods have been developed to browse the data remotely using image generation techniques. The resulting images are still stored locally at the computing center, and only the part that is being viewed is transferred to the researcher's site. At his site, the researcher has access to a large resolution tiled panel that is capable of displaying this image stream.

In contrast to distributed applications in grid computing the TOPS application is running under interactive control of the user. Besides computing and networking facilities the application renders pictures that are viewed on a remote, large size display by a user.

The TOPS application can take care of all the necessary steps to transport and display the image data. Adaptations were necessary to support various modes of visualization (synchronization, stereo display, animation & video), as well as to support the Phosphorus test bed infrastructure. Finally, steps were undertaken to support a network provisioning scheme, so as to be able to demonstrate all the Phosphorus accomplishments within one application.

During the whole project period, effort has been invested into keeping up to date with network provisioning methods and image streaming applications. Where possible and useful, TOPS and the Phosphorus projects have been presented at relevant venues, be they national, European and global.

In the first month of the projects, the hardware and software requirements for TOPS were documented. A call for participation was written and published, aiming to gather more researchers and/or participating sites for testing and demonstrations. Also, an overview of other image streaming tiled panel environments was made, to be able to compare them with TOPS. At the SC|06 supercomputing conference (Nov. 2006, Tampa, Florida), these were presented alongside, together with a general presentation of the Phosphorus projects, its goals and aspirations. Finally, input was supplied to deliverable D3.1, defining necessary application interface changes.

In the following months, test machines at SARA and FHG have been set up for the TOPS application and documentation was supplied for partners and researchers to learn more about the application and its possibilities and limitations. As distribution of a graphics visualization application is not state of the art, decisions at FhG had to be made to allow external access to the FhG-IAIS display facilities. Due to internal (i.e. firewall) policies the test machines have been set up and put outside the IAIS institute network. As for using the display facilities a human operator (viewing the pictures) must be on site, it has been decided that the assignment of the test machine cluster to the display (i.e. switch video sources between machines and projectors, switch on/off projectors) is done by the user, using the available local procedures. It is not intended to make it an automated process as the situation always is different for different sites. For the application it is assumed that a high resolution picture stream is available at the network interface of the machine acting as a video source for a display.

Content was generated in different formats. Cross-Atlantic tests were performed between SARA and CalIT2 (California, USA), yielding valuable performance data on light paths over long distances. Network tests were set up and performed with the i-CONE® display facility at FHG. All these activities resulted in a first design document on necessary adaptations of the TOPS software.

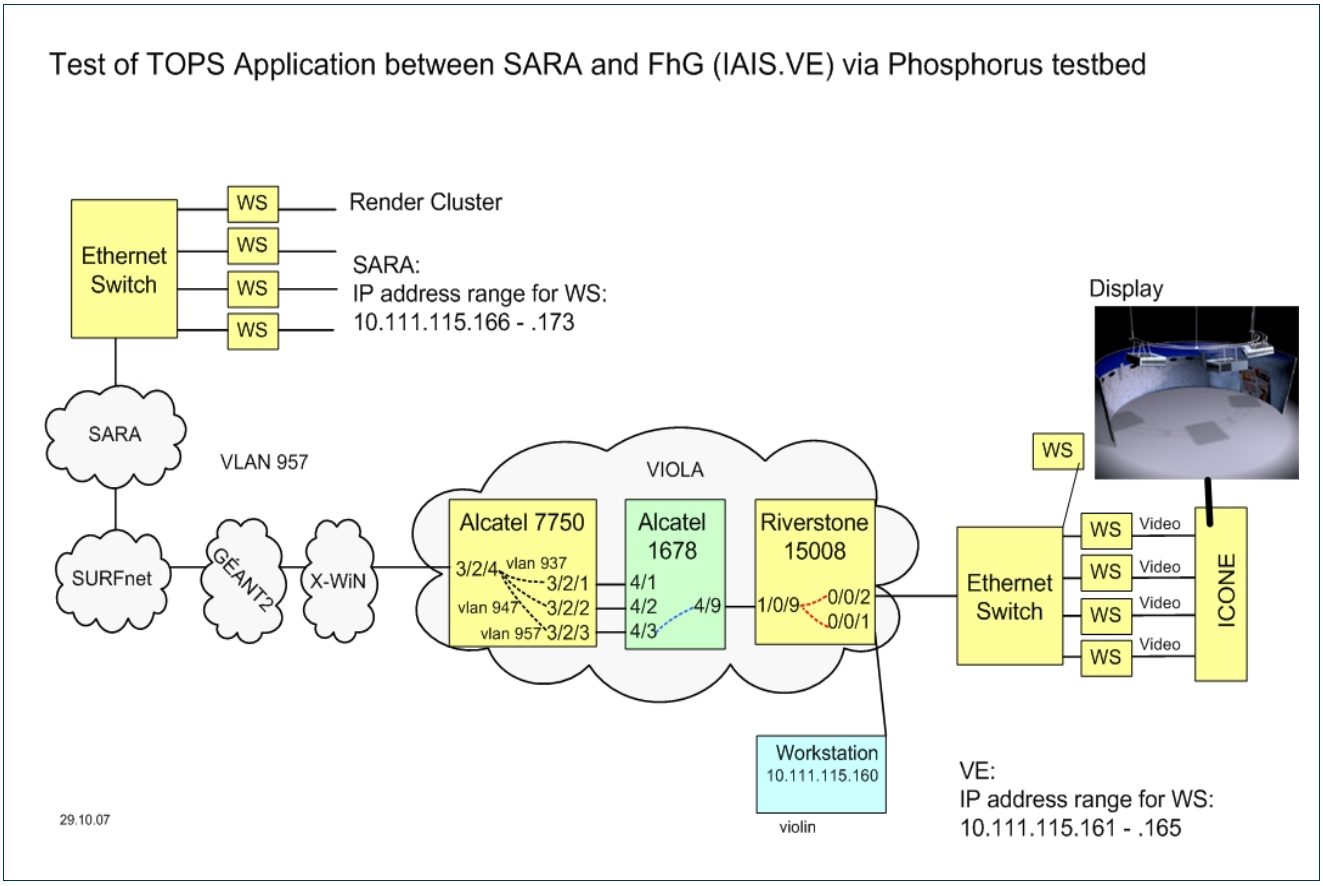


Figure 2.5: Test of TOPS Application between SARA and FhG (IAIS.VE) via Phosphorus testbed

In March 2007, connectivity with the Viola test bed was established, remote access was provided and TOPS was installed at the FHG i-CONE® display facility. Parts of TOPS were rewritten as per the design document. Some storage hardware was upgraded in the SARA infrastructure to accommodate the large datasets that were planned in the Phosphorus test bed. A DRAC test bed was set up together with SURFnet in the Netherlands, and tests were initiated in this local test bed.

In the next few months, further tests between SARA en FHG were performed. Networking issues with firewalls, access control lists and MTU settings were found and tackled. The test bed machines have been tuned for optimal performance for image streaming using TOPS. The display facility at FHG was supplied with the necessary hardware to steer the TOPS application.

After the summer break, tests were performed with high resolution (4k) video streams. A multicast demonstration of 4k video streaming was given at the GLIF meeting in Prague, as well as at the eChallenges 2007 conference in The Hague. The SARA and FHG test machines were moved from the Viola test bed and

connected through the Phosphorus infrastructure. A dedicated 1 Gbps light path was used to connect the SARA rendering facility with the i-CONE® display facility at FHG, and various test sessions were held. A number of networking issues (MTU mismatch) had to be resolved, and further tests between SARA and FHG were performed. A number of issues with the application were identified, resulting in a second design document for TOPS. At the SC|07 supercomputing conference, the TOPS application use case was demonstrated in co-operation with a researcher that is currently using the TOPS software and infrastructure at his own lab.

Figure 2.6: View of high resolution pixel stream sent from SARA to the FhG-IAIS i-CONE® display through the PHOSPHORUS testbed

From January 2008, the requirements and recommendations as set forth in the second design document have been implemented in a final version of the TOPS application. Mostly, the changes are necessary to allow sites to use the software without rewriting configurations or implementing code changes, and make use of a grid infrastructure in line with the Phosphorus project. During this process, it has become clear that a direct connection to the Phosphorus network provisioning system(s) would be invaluable to the use case, and it has been decided to extend the development of the User Interface of TOPS beyond the 18 month boundary in order to implement an interface to these systems. To that end, contact has been made with SURFnet and Nortel for access to DRAC resources.

The final version of TOPS has the following improvements compared to the original 2006 version:

- resource allocation through central scheduling and configuration user interface
- complete separation of render and display code
- implementation as daemon invoking the actual display or render code
- parameterization of:
    - display size & bits per pixel
    - display configuration
    - MTU size
    - IP configuration
    - data set location
    - user interface location
- hardcoded values removed and put in configuration files
- better support for other file types

2.4 DDSS

Distributed Data Storage Systems (DDSS) are widely used to store, share, exchange, transport as well as backup, archive and restore the data in scientific and commercial applications. Two DDSS applications were chosen for PHOSPHORUS test-bed: GridFTP application [GridFTP] and a commercial backup/archive/restore application from IBM Tivoli [TSMBA].

GridFTP server exchanges data with single (one-to-one setup) or multiple clients (many-to-one setup). The single client can communicate with single (one-to-one setup) or many servers at the same time (one-to-many setup, 'striped' transmission mode). Data transmission between a single client-server pair can be done over one TCP/IP stream or in parallel using up to 128 parallel streams; this allows to effectively exploit the bandwidth of the network link even in the presence of a big transmission delay. The setup used in PHOSPHORUS is mainly the one-to-one scenario, run between the test nodes located in PSNC, FZJ, FhG and UEssex.

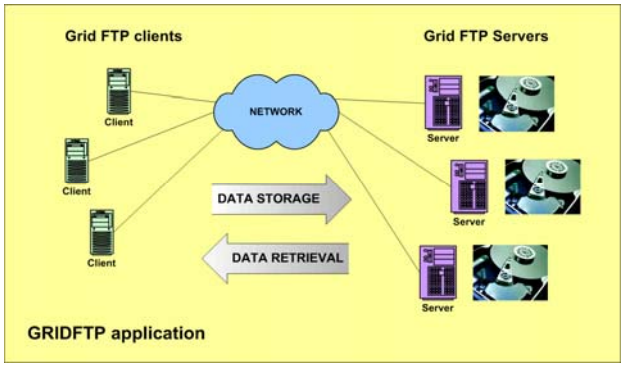


Figure 2.7: DDSS GridFTP application servers and clients

Backup/Archive clients collect the end-user data stored on the client machines and send them through TCP/IP streams to the Backup/Archive server. In case of failure or corruption of the user data, they can be restored from the server using the Backup/Archive client. B/A server itself manages the storage pools that can include disks, tapes and other storage media. User data are stored in these pools, according to a policy defined for a user. The setup that is used in the PHOSPHORUS test-bed contains two B/A servers: in PSNC and in FZJ. B/A clients are located both in PSNC and FZJ.

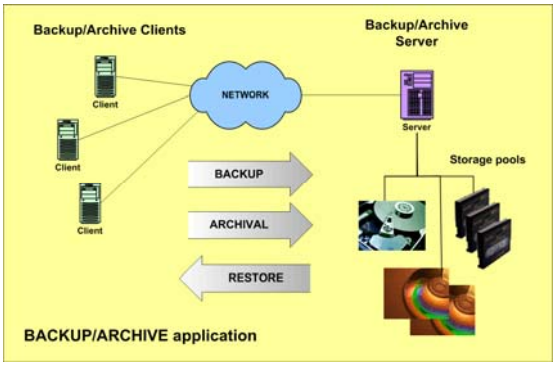


Figure 2.8: DDSS Backup/Archive application servers and clients

DDSS applications can generate various types of workloads. This makes possible evaluating various features of the PHOSPHORUS optical network and its link reservation capabilities. Link bandwidth can be examined by moving huge data volumes over the network links using multiple parallel transmission streams – such workloads are typical for moving large files between computers. Latency-related features of optical links can be examined using these DDSS scenarios in which multiple (hundreds to thousands) of small files are exchanged by DDSS clients and servers; this happens e.g. during making the backup copy of the users' home directories stored in file servers. Specific features of the protocols used in both DDSS applications used in PHOSPHORUS test-bed vary significantly. GridFTP protocol extends the well-known FTP protocol, by adding security,

parallelism, commands pipelining and other features [GridFTP2]. Backup/Archive application in turn, uses proprietary vendor protocol that includes compression, encryption and parallelism.

Adaptation of the DDSS applications to the PHOSPHORUS required some implementation activities as well as configuration and testing work.

First, the use cases that fit into the idea of testing PHOSPHORUS optical network features were defined. They were described using use case description template provided by the WP3 coordinator.

Second, the applications were deployed in the local test-beds and the application setup was adapted to the capacities of these test-beds. Three institutions responded to the 'DDSS Call for Participation', which was prepared in the November of 2006 by PSNC. The applications use-cases were tuned up in order to fit to the capabilities of the local test-beds and the connectivity between them. This activity took place between December 2006 and January 2007. It was done by PSNC in collaboration with FZJ, FhG and UEssex. In the same period, the local test-bed setup started. It has been continued during February till March 2007 period. DDSS test-bed was set-up in PSNC firstly, then in the local test-beds of the rest of the partners. From March 2007 to June 2007, local DDSS application testing was performed in PSNC. In the same time, PSNC and partners have set up the local test-beds and started preliminary, remote testing using Internet links. In this period a lot of configuration fixes and test-bed debugging actions were performed, including opening the firewall passes and tuning another settings of test nodes' operating systems to the application characteristics.

Third, in parallel to test-bed configuration and preliminary test actions, we investigated the possible changes of DDSS GridFTP and DDSS Backup/ Archive applications (February 2007) and implemented them (March 2007 to June 2007). In case of DDSS GridFTP, we introduced the functions which contain the NRPS function calls used to assign the optical network links to the application source code. We have also added timers and instrumentation functions to the GridFTP client code. They allow us to evaluate performance of the GridFTP data transfers done in the optical network. Opposite to DDSS GridFTP, DDSS Backup/Archive application source code was not available for us (as this is a commercial application from IBM Tivoli). Therefore, we decided to wrap the B/A application binary with the binary code prepared by us. The wrapper contains calls of the network resource reservation functions and instrumentation functions (timing, performance measurements). At that stage of the project, the NRPS function calls, added to DDSS applications, were in fact 'dummy' calls – they have not been implementing the actual reservation, as the reservation facility was planned to be available in the later phases of the project.

Fourth, since July 2007 to August 2007, we developed the tools for automating the execution of the DDSS use-cases and scenarios in the PHOSPHORUS. We also developed the mechanism for collecting the results of the tests. Automation tools run the DDSS tests in the test-bed periodically. Thus, they allow us to acquire a lot of measurements concerning the application and network performance. Results collection mechanism collects and registers the results in the relational database and generates graphs that depict the results of particular rounds of tests (graphs are put to the web-site). This allows us to perform result evaluation and analysis just after the test rounds finished and beyond.

Fifth, we have run DDSS tests periodically, using above-mentioned automation tools. In August 2007, we started periodic testing the DDSS transfers between PSNC and FZJ. After completing the configuration of the rest of the test-beds – configuration of FhG and UEssex test-beds finished between August and October 2007

– we have also performed a periodic runs of DDSS tests between PSNC and FhG as well as between PSNC and UEssex.

Sixth, starting from October 2007, we attempted to switch from Internet links to optical network links. In October, the link between PSNC and Viola (FZJ, FhG) was configured. Switching to optical links required reconfiguration of network interfaces in test nodes and some changes in the configuration of test automating tools. There were some troubles and bugs in optical network configuration, that were partially solved till February 2008. There was also a major failure of one of the test-bed sites, that required reconfiguration of DDSS part of the test-bed. We were dealing with these issues during January 2008 and February 2008. March 2008, we worked on establishing the reserved optical link between PSNC and FZJ for DDSS purposes and worked on solving optical connectivity issues between PSNC and UEssex.

At the end of the first 18 months' period of the PHOSPHORUS project, we have made the following improvements and modifications of the DDSS applications. First, the applications are ready to perform the optical network reservations, as this functionality will be ready. Second, the applications contain the instrumentation functions that allow measuring performance of the applications operation. Third, automation tools allow us to perform multiple rounds of application tests automatically. In that way, we are able to acquire multiple measurements captured for various application use-cases and various network links configurations (they can be changed in a meantime). Fourth, thanks to collecting the test results in the relational database, we are able to analyse the results any time, having access to both recent and historic performance data.

All these improvements and extensions of DDSS applications enable using these applications as effective benchmarks for evaluating the optical network links reservation features developed in PHOSPHORUS. From the other hand, thanks to extensions, these applications can benefit from NRPS features developed in the project.

3 Task 3.2 – Middleware

This task is responsible for the adaptation, development and implementation of middleware to seamlessly integrate with new network capabilities and to allow the co-allocation of multiple, heterogeneous resources to the application. Key activities in this task are:

Activity 3.2.1 Definition of middleware-NRPS interface

- Check existing and evolving standards, protocols and implementations
- Design and implement required changes and extensions of the middleware and applications to use the new services
- Define requirements of applications and middleware towards the resource provisioning system and the common framework under development on WP1.
- Design and define functionality, protocols and interfaces to resource provisioning system. The work under this and the previous bullet will be carried out jointly with WP1 (Activity 1.3.1). This activity will finish at M6, however the results will be published in the report that is part of D1.6 (Definition and development of the Network Service Plans and northbound interfaces development), to be issued on M14, which includes report and prototypes.

Activity 3.2.2 Implementation of Middleware extensions

Extend the middleware to allow changes in resource utilisation, either by agreements made in advance or dynamically. We will also set up the collaboration with EGEE as described in Section 11.2 during this activity.

Activity 3.2.3 Implementation and integration of ontology for service abstractions

Design and implement an integrated CIM-based ontology of the service abstractions of resources available in the test-bed including the work done in WP1 for network services. Provide the reasoning interface to allow automatic selection of services (semantic mediation) according to the users' request or the requirements exposed by the applications.

Task 3.2 interacts with WP1, WP2, WP4, and WP6 to reach a common understanding of requirements and to define common interfaces based on this.

During the reporting period, work package participants started by generating use-cases, requiremtns and the design of changes and extensions of the applications and middleware. Based on the requirements following from the use-cases, extensions and design changes of applications and middleware were specified.

The participating middlware solutions were deployed in local testbeds, where their changes and overall funcitonality and applicability for the use-cases would be verified. Essentiall, this involved the setting up of machines and the installation and configuration of software.

In July 2007, several applications have prepared for inter-test-bed tests over the internet. For that purpose, middleware and firewall configurations at several sites had to be modified. Furthermore the implementation work has been continued as planned. The overall focus is to complete a first functional version of middleware and applications that is ready for using the global test-bed until end of September. The preparations for the tests planned for October and November were continued. Implementation work and deployment in the test-beds have been the focus of the work of the applications and the middleware. Preliminary tests using the Internet were performed before the final Phosphorus testbed becama available.

Connections between local test-beds VIOLA, PSNC and SARA became available in October 2007. The attached systems, the middleware and the applications have been configured to use these connections and have performed promising initial tests. Furthermore the implementation work for the test-bed tests has been continued.

Joint experiments with WP1 were defined around November 2007. This war particularly useful for the demonstrations at SC07 and the Phosphorus one year review meeting.

The test-bed relapsed, when the Cray XD1 cluster at FZJ suffered a fatal hardware failure. A new machine had to be provisioned and connected to the Phosphorus network. This problem had implications on all installed middleware and applications on the original machine.

3.1 UNICORE

UNICORE has been chosen as the middleware to be linked with both the WISDOM and KoDaVis applications. UNICORE supports a plugin architecture at the client side that allows for application specific plugins to be included. The suitability of the plugin concept has been evaluated for both applications. The work on middleware design started in March 2007.

The first component of the UNICORE middleware to be adapted to the KoDaVis use-case was the client. An application specific plugin was written that can be added to any UNICORE client via the plugin architecture described above. In the following, a specific service for the UNICORE server side was developed, wich allowed the creation and management of KoDaVis data server instances as well as collaborative sessions. This service was then coupled with the gaphical KoDaVis client through the UNICORE client.

The meta-scheduler (MSS), which is another component of the middleware landscape employed in Phosphorus, had to be integrated with UNICORE. Thus, it would be possible to co-allocate network and Grid resources by means of the MSS and make advance reservations of UNICORE resources through the

middleware. For UNICORE, this meant to extend the interface of the target systems to expose the advance reservation capabilities of the underlying batch system. Also, as UNICORE supports a host of batch systems, the Target System Interface (TSI) needed to be adapted for the particular batch system running on the Grid resource. The TSI was first developed for the Cray XD1, which was the Grid resource dedicated to Phosphorus at FZJ. The TSI had to be implemented for the combination of the Torque batch system and Maui scheduler, when the XD1 was replaced with another machine due to a hardware failure.

The requirements from the PHOSPHORUS application use-cases concerning work-flow support were communicated to the core UNICORE development team and were implemented in the work-flow support of the 6.1 version of UNICORE. This new features will be used in particular by the WISDOM application.

Ths use of UNICORE for multiple applications within the project required all developers to gain further insight into the middleware. Therefor, a UNICORE workshop was held at FZJ on July 25th and 26th, 2007. Further information about features like clients, job submission, data distribution, and data stagin was provided.

3.2 MSS

The MetaScheduling Service (MSS) was initially developed in the German project VIOLA. Its main purpose was the co-allocation and reservation of compute and network resources for demanding Grid applications that used more than one compute infrastructure.

Since the VIOLA testbed was a part of the PHOSPHORUS tesbed from the very beginning MSS was selected for the PHOSPHORUS project for the co-allocation and reservation of compute and network resources as well. To satisfy the requirements of the applications, the different network service and control planes the MSS has to interact with in the PHOSPHORUS testbed MSS has been extended. The different enhancements of the MSS as part of the middleware environment in the PHOSPHORUS testbed are described in chapter 5.2.

- The MSS is composed of a set of web-services that allow to
- find the next available slot of resources, both compute and network
- negotiate the usage of an available slot and reserve it for a user
- create a service level agreement (SLA) with the responsible software entity of the resources to reserve such a negotiated slot (local resource management systems (RMS) in case of compute resources, network resource provisioning systems (NRPS) in case of network resources
- monitor the SLAs
- display the state of a reservation
- cancel a reservation

In addition to the MSS a resource selection service (RSS) has designed that will allow a user to only specify the name of the application she is intending to use while all other information like the requirement of this application with respect to the capabilities of the compute resources or the network connection is automatically retrieved from an ontology. This ontology of the application requirements has been created together with a second one that stores the capabilities of the compute resources of the PHOSPHORUS testbed. These two ontologies are used by the RSS to match application demand and resource capabilities to identify one or more resources suitable for the execution of application. The MSS uses this list to negotiate the reservation and – based on the network requirements, e.g. for the stage-in or stage-out phases of the job – to also reserve the required network connection with the necessary QoS.

For the MetaScheduling in PHOSPHORUS two different scenarios for co-allocation and reservation are provided:

1. The classical approach like in the VIOLA project where the MSS is aware of potential resources based on information coming from the middleware. The MSS is negotiating with all resources to be available for a job, talking to the local RMS and the NRPS.

2. The new approach of PHOSPHORUS where the G²MPLS is aware of the resources connected to the network and queries the MSS for availability information, reservations etc, using the web-services of the middleware layer.

To allow an easy integration also in other web-serviice based middleware environments and to ease the interoperability across different middleware systems the MSS implementation is standards-based:

- Web Service Resource Framework (WSRF from OASIS)
- Web-services Agreement (WS-Agreement from the OGF)
- Web Services Security (WS-Security from OASIS)
- Resource selection Service (OGSA-RSS from the OGF)
- Job Submission Description Language (JSDL from the OGF)

3.3 INCA

INCA is an intelligent storage network that provides data-transfers with high performance. As we depict in the following figure 3.1 the functionalities of INCA are divided in two layers. The first layer is called **Data Management Layer** and its purpose is to infuse in the system totally distributed and automated management functionalities. In more detail we test and evaluate three major features that each storage network must have:

The first one is scalability. In order to achieve this goal we have implemented a distributed hash table (DHT) that is used for the distributed metadata maintenance and the distributed location of them.

The second is the balanced use of storage network resources in terms of storage space and network bandwidth. In order to achieve this goal any data chunk is hashed and according to the output of a uniform hash function is stored in the proper storage node.

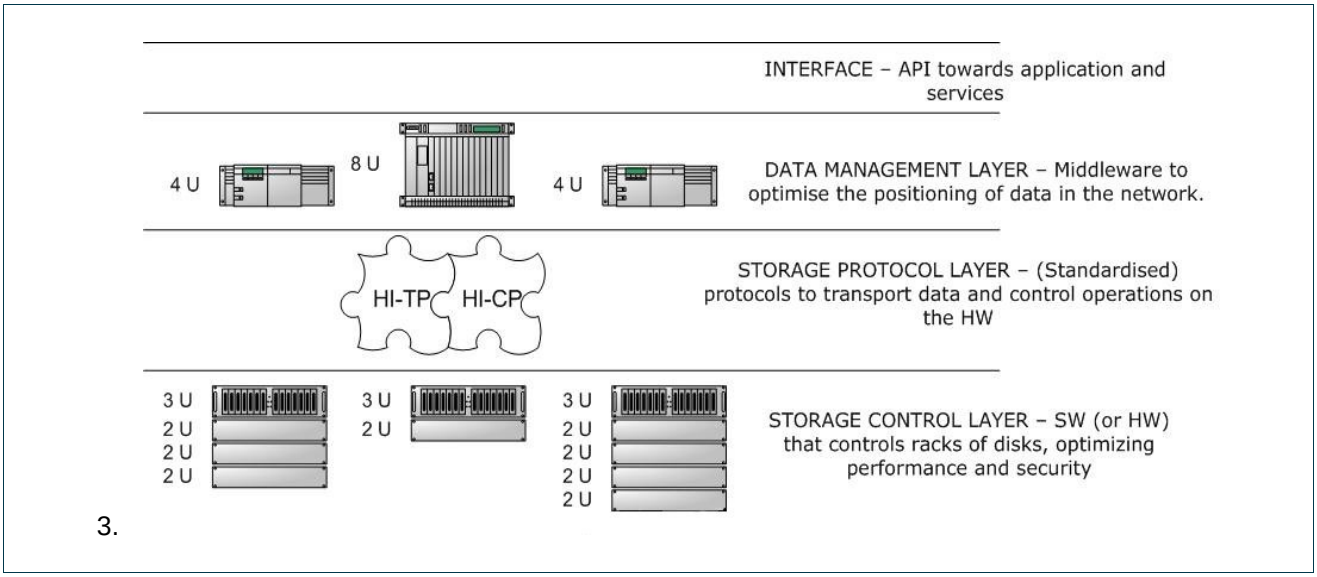
The third is the fault-tolerance of a storage network. A stable routing algorithm used for the data and metadata location has been developed and has been evaluated.

Through our experiments we have evaluated our architectural decisions and the performance of the algorithms that we have used in order to enhance the storage network with these features. The results show that a DHT is capable to manage well a storage system but special attention has to be paid in order to maintain all the necessary data structures during dynamic conditions.

As for the second layer it is called **Storage protocol layer**. It's purpose is to provide:

1. Point-to-point buffer transfer in order to store chunks in the chosen (by the upper layer) node;

2. The glue between **Data management layer** and the storage control layer.



3.

Figure 3.1: The INCA Architecture

In the first stage we have observed that Phosphorus is an environment where a storage network with a totally distributed management will bring out its advantages. In more detail due to the high performance of the underlying network a single point of management that a centralized architecture would have introduced it would act as a bottleneck for the whole system. Additionally the automated and distributed nature of the data placement and retrieval will exploit the resources of such an environment. Due to these reasons we have selected a DHT in order to be used as the basis for the Data management layer. This DHT is called CAN (Content Addressable Network).

CAN is a distributed lookup and routing protocol. All nodes participating in CAN are equal and no hierarchy exists. ID space is organized as a virtual d-torus (a 2-d torus is like the surface a sphere). Each node is responsible for the keys in a zone of the space (in a 2-d torus zone is a rectangular parallel to the axes of the coordinates). Each value (item name) corresponds to a key. Given a key system has the ability to route the message to the node responsible for that key. CAN implements three basic functions item insertion, lookup and delete.

Nodes in CAN are self-organized into an overlay network that represents this virtual coordinate space. Each node has to discover and maintain a routing table with the IP addresses of the nodes adjoining its own zone. These nodes are called its neighbors. All values are mapped into the space using a uniform hash function. This function creates the coordinates in the space (key) for each data item (value). A query for a key is routed through the overlay network to the node responsible for the key. Fast and reliable routing is the most important service of CAN.

When a node enters the network an existing node's zone is divided in two halves and the new node is responsible for the one of them the old node for the other. Each node has a Virtual Identifier (VID). When a zone is divided to two each child's VID is created by adding abinary digit in its parent VID. To the node with the lower coordinates in the dimension of the division is added a 0 and to the other is added a 1 (Figure 3.2 represents a 2-D CAN with 5 nodes).

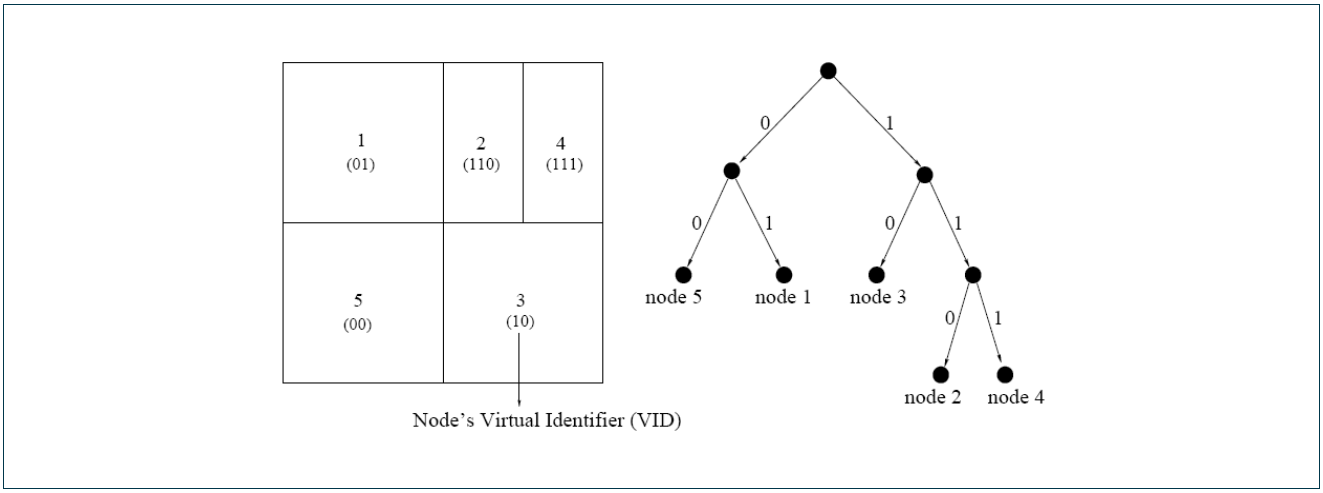


Figure 3.2: Representation of a two dimensional CAN (left figure) and the mechanism of the creation of the virtual identifiers (right figure)

For a new entrance a node already in the network must be discovered. After its discovery by using CAN's routing mechanisms it could be found an appropriate node to split its zone. Some keys are more popular than others and zones are not uniformly divided. For having a load balanced network an algorithm which finds an appropriate node for split its zone must be implemented. At last neighbors of the new and old node must be informed for this change so both nodes sending an update message to all of their current neighbors. It is also important to be mentioned that periodic messages are sent to ensure network's stability. A new entrance affects only O(d) nodes regardless of the total number of nodes in the system.

To route between two points in the space a node in CAN forwards a message with the destination coordinates to its neighbor with coordinates closest to the destination. The number of the average hops with this routing mechanism is (d/4)(n1/d) where n is the total number of nodes in the network.

When a node leaves the system its zone and its database (key-value pairs) is taken by a neighbor which is called takeover node. The two zones are merged into one if is possible or the takeover handles two zones. CAN needs to be tolerant to node failures so database can be replicated in a number of nodes to increase network reliability, to reconstruct the database by asking the participating nodes in the system requires much time and an enormous amount of bandwidth.

We have implemented CAN and with the functionalities that it provides we are able to manage INCA. In more detail when a node of INCA enters the network takes a zone in CAN. Metadata are hashed and assigned to the responsible node. Data are free to move in any node according to the policy of the management function but have to inform the responsible node for their location. For their retrieval the responsible node is asked through routing in CAN and it returns the address of their presence.

As for the storage protocol layer due to the high capabilities of the network TCP has been characterized as inefficient so it was necessary the implementation of a new protocol that will support the architecture above. So we have implemented a new protocol that will be responsible for point-to-point chunk transfers that requested from the layer above. This is parted from two layers:

- HITP: High-volume INCA Transport Protocol that aims to transport fast huge volumes (Terabytes) of data in an efficient way and dealing with lambda networks.

- HICP: High-volume INCA Control Protocol, it aims to control HITP and making the glue between middleware and network levels.

4 Task 3.3 – Interfacing between G²MPLS and Resource Management System via G-OUNI

G²MPLS is a Network Control Plane (NCP) architecture used in PHOSPHORUS that implements the concept of Grid Network Services (GNS). The GNS allows the provisioning of network and Grid resources in a single-step. The goal of the GNS is to make network resources in available in a similar way as other grid resources, like CPUs, memory, contents or OS processes.

G²MPLS is aimed to provide functionalities related to the selection, co-allocation and maintenance of both Grid and network resources. This goal translates in:

- Discovery and advertisement of Grid capabilities and resources of the participating Grid sites (Vsites);
- Grid and Network Service setup including:
    - Coordination with the Grid local job scheduler in the middleware responsible for the local configuration and management of the Grid job;
    - Configuration of the network connections among the Vsites participating to the Grid job;
    - Management of resiliency for the installed network services and possible escalation to the Grid middleware components that could be responsible for check-pointing and recovering the whole job;
    - Advanced reservations of Grid and network resources;
- Service monitoring both for the Grid job and the related network connections.

4.1 G²MPLS overlay model

In G2MPLS Overlay model, the Grid layer has Grid and network routing knowledge in order to provide Grid resource configuration and monitoring (as in its standard behaviour) plus network resource configuration and

monitoring. G2MPLS acts as an information bearer of network and Grid resources and as a configuration “arm” just for the network service part.

This model is intended to be mainly deployed when most of the computational and service intelligence need to be maintained in the Grid layer for specific middleware design and functional behaviours. The leading role in this case is played by the Grid scheduler, which is the overall responsible for initiation and coordination of the reservation process through the participating Grid sites and the network in between.

4.2 G²MPLS integrated model

In the G2MPLS Integrated model, most of the functionalities for resource advance reservation and commit are moved to the G2MPLS Network Control Plane. G2MPLS is responsible for scheduling and configuring all the job parts, those related to the Grid sites and those related to the network.

Grid sites are modelled as special network nodes with specific additional Grid resource information (ref. green and dark grey shapes in Figure 4.1. The resulting topology is flat and integrated with respect to the positioning of the Grid layer against the network layer.

4.3 The G-OUNI interface

The Grid Optical User Network Interface (G.OUNI) comprises a number of procedures to facilitate on demand as well as in-advance access to Grid services/resources by interfacing Grid end points and any Grid middleware with any type of network resource provisioning system.

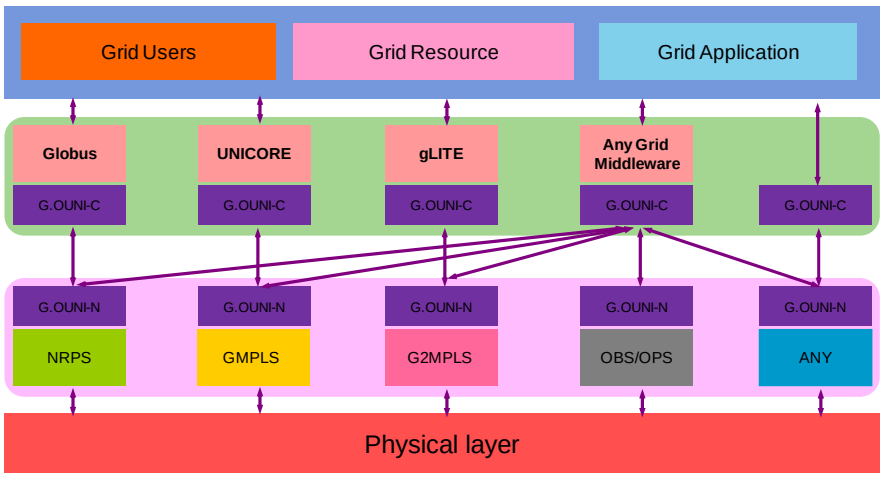


Figure 4.1: Grid User Network Interface with grid endpoints as well as Grid middleware with Network Provisioning Systems

The G.OUNI reference point acts as the interface between Grid End Points and the Grid Network Service Provisioning Systems as shown in Figure 4.1.

4.4 Communication with the G²MPLS layer

In order to use network functionality from the grid layer, MSS needs to communicate with the G²MPLS layer. This communication channel is provided by the G.OUNI gateway. The interface between the G.OUNI gateway and MSS was originally as intended to be based WS-Agreement and was described in deliverable D2.7. WS-Agreement is used by MSS to submit, monitor and manage the lifecycle of a job. Additionally interfaces are required to advertise new sides and to discover sides. These interfaces are intended to be based on the GLUE schema.

In order to enable the MSS to work with UNICORE 6, a new adapter needs to be developed to make UNICORE 6 functionality accessible via WS-Agreement protocol. Since UNICORE 6 poses high requirements in terms of security, a number of extensions have to be made to the MSS environment. These extensions include:

Enabling WS-Security for the MSS in order to digitally sign the SOAP messages send by the MSS client and server components, and to validate the digital signatures of the messages received by the MSS client and server components.

Enable the MSS to support UNICORE 6 trust delegations. UNICORE 6 uses SAML 2.0 assertions to delegate trust from the issuer of a job (e.g. a user) to the entity that actually submits a job to UNICORE (e.g. a scheduler). Therefore, additional functionality to generate UNICORE 6 trust delegations (TD) by the MSS client, to validate TD objects via the digital message signature on the server side, and to generate and validate of trust delegation chains is required for the MSS.

Integration with UNICORE 6. The MSS needs to be extended in order to submit jobs to UNICORE 6, to monitor the progress of these jobs, and to retrieve the results. Additionally, file transfers for staging in input data and staging out output data must be initialized.

During the first 18 month period the MSS environment was extended to support WS-Security and UNICORE 6 trust delegations. The integration for integration of UNICORE 6 is still ongoing. Additionally, MSS needs capabilities to reserve resources in order co-allocate and coordinate multiple resources, which are potentially of different type. Therefore, mechanisms were needed to reserve resources with UNICORE. The interfaces required to accomplish this task were designed and implemented in UNICORE.

For the integration of the G.OUNI gateway and MSS, the wsag4j software package was provided to WP2. This software serves as hosting environment for WS-Agreement based services. It implements mechanisms to create, monitor, and terminate agreements. Additionally, it provides means to digitally sign and validate the SOAP message during the communication process and to create the SAML assertion required for the UNICORE 6 interaction. WP2 evaluated the software for the implementation of the G.OUNI gateway. During the evaluation process it turned out that due to different technologies used for the implementation of the WS-Agreement parts and the G.OUNI gateway (e.g. different programming languages: Java, C), the client of the WS-Agreement engine could not easily be used. Therefore, the decision was made to use the gSOAP engine

to implement a web service based communication between G.OUNI gateway and MSS. Since gSOAP engine does not provide support for stateful web services, currently the OGSA BES (basic execution service) protocol is foreseen as the basic protocol for the MSS-G.OUNI interaction. The OGSA BES interface is implemented as a stateless web service. This makes the implementation on the G.OUNI gateway site much easier. Furthermore, UNICORE 6 provides an OGSA BES interface since version 6.1. Therefore, this functionality can be accessed directly from the GOUNI gateway.

5 Task 3.4 – Integration and Evaluation of Middleware

5.1 UNICORE

UNICORE 6 servers were initially intalled on the Cray XD1 at FZJ. Clients were available on those sites participating in the KoDaVis and WISDOM experiments:

- FZJ
- PSNC
- UoESSEX

The UNICORE 6 server landscape was later extended. FZJ provided a central UNICORE registry that allows all sites to be registered. Clients can thus discover all available UNICORE sites through this one central registry. The UNICORE installation, including the additional KoDaVis service, has also been used for demonstrations at SC07 and at the Phosphorus review on December 13th, 2007 in Poznan.

UNICORE needed to be installed a second time at FZJ due to a hardware failure. This involved a number of steps. For one, as the new resource was equipped with a different batch system, the TSI wrapper had to be adapted to this to support the advance reservation requirements in Phosphorus, see section 3.1. Secondly, this new installation was taken as an opportunity to change from the demonstration certificates used previously to real Grid certificates issued by the German DFN network's DFN Grid CA.

5.2 MSS

The initial version of the MetaScheduling Service (MSS) introduced in the PHOSPHORUS project was based on developments of the German VIOLA project.

During the first phase of the middlware adaptations WP3 started gathering requirements from the PHOSPHORUS applications (contributed by partners of WP3) and from the other WPs, especially WP1, WP2,

WP4 and WP6. In the same time a use-case document was produced that further contributed to the specification of requirements for the MSS. Based on this work a series of inter-workpackage meetings was organised in Amsterdam and Utrecht, which resulted in the final version of the specification of requirements and a first draft of the interfaces to be implemented towards the NSP, the G2MPLS and the AAI of the network layer.

During the first months of the project the integration of MSS into the UNICORE system was enhanced, making the overall architecture of the PHOSPHORUS middlware more consistent. Figure 1 presents the resulting architecture. In contrast to the VIOLA approach where the MSS was using own adapters for the communication with the local resource management systems (RMS), the MSS in this version of the integration is already using the target system interfaces (TSI) of UNICORE to communicate with the local RMS. In order to allow also reservations of resources and the negotiation of available time-slots for the execution of an application the TSIs have been extended. For the communication with the UNICORE systems components the MSS uses the UNICORE protocol language (UPL). To avoid extensive changes in the UNICORE internal communication it was decided to use an adapter that maps the WS-Agreement interface of the MSS to the UPL interface of UNICORE. To that end, we achieved interoperability while the internal protocol of both UNICORE and MSS could remain unchanged. With this new architecture we additionally prepared the ground for the integration with other middeware systems like the Globus Toolkit 4 as shown in Figure 5.1. Moreover, the ARGON NRPS is also connected using the same adapter mechanism: the MSS WS-Agreement protocol is mapped to the ARGON web service interface.

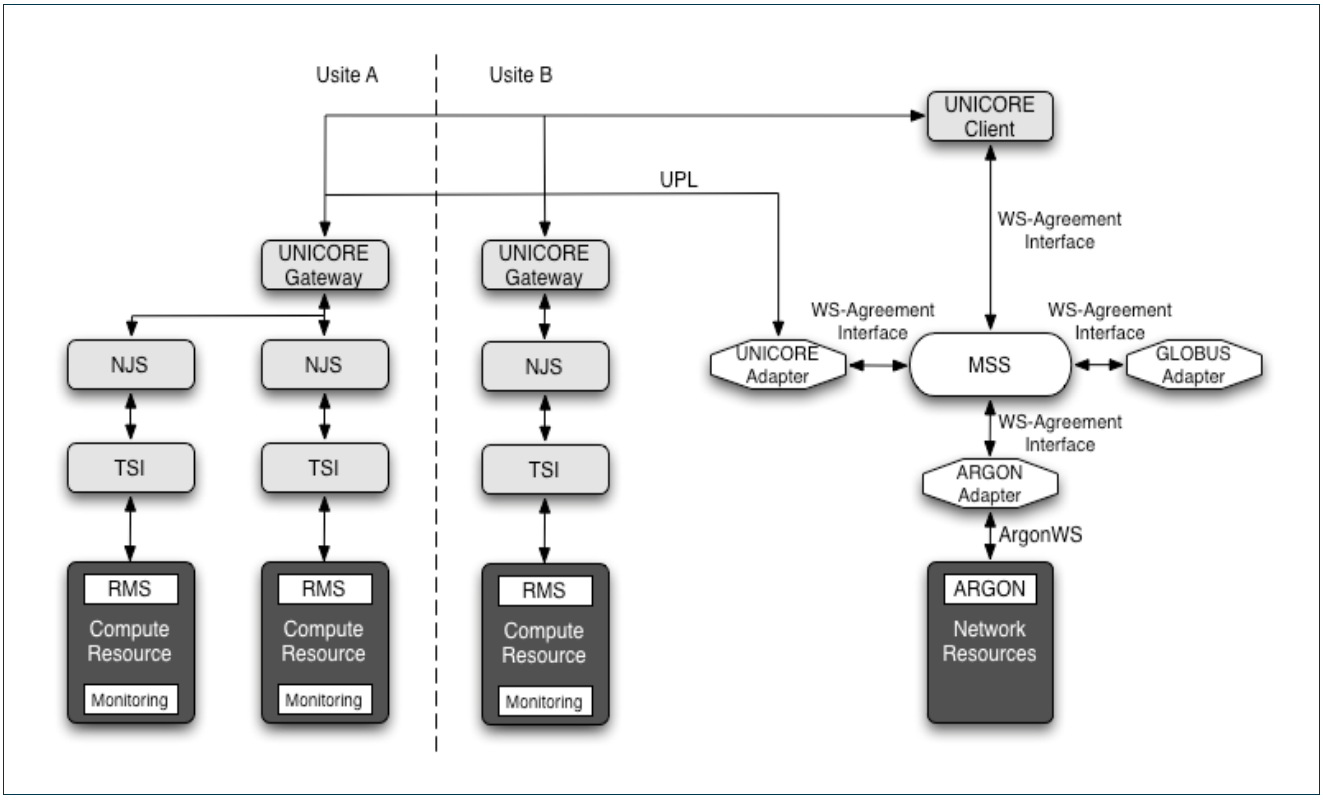


Figure 5.1: Integration of the Metascheduling Service and UNICORE

Parallel to this activity the resource selection mechanism based on semantic matching of application requirements and resource capabilities was outlined and the requirements for an interface of resource selection service (RSS) and the choices for an implementation of the RSS with the MSS were evaluated. It was decided to use the evolving standard of the Open Grid Forum (OGF) for RSS, which is currently under development in the OGF OGSA-RSS working group.

Based on the WP1 specification of the network service plane (NSP) the MSS adatpter towards the ARGON NRPS was replaced by an adapter for the NSP. Figure 2 shows the current architecture that was used for the demonstrations during the first project review in December 2007 and which is now used as part of the regular PHOSPHORUS testbed infrastructure. As Figure 2 depicts the different NRPS available in the PHOSPHORUS network environment are have an interface to the NSP while the request for network resources from the MSS is handled by the NSP and transparently forwarded to the respective NRPS.

Figure 2 also shows the interface with the control plane and the G2MPLS developed by WP2 in the PHOSPHORUS project. The implementation of these interfaces will be mostly done after month 18. More details about the architecture can be found in chapter 4 (Task 3.3 – Interfacing between G²MPLS and Resource Management System via G-OUNI).

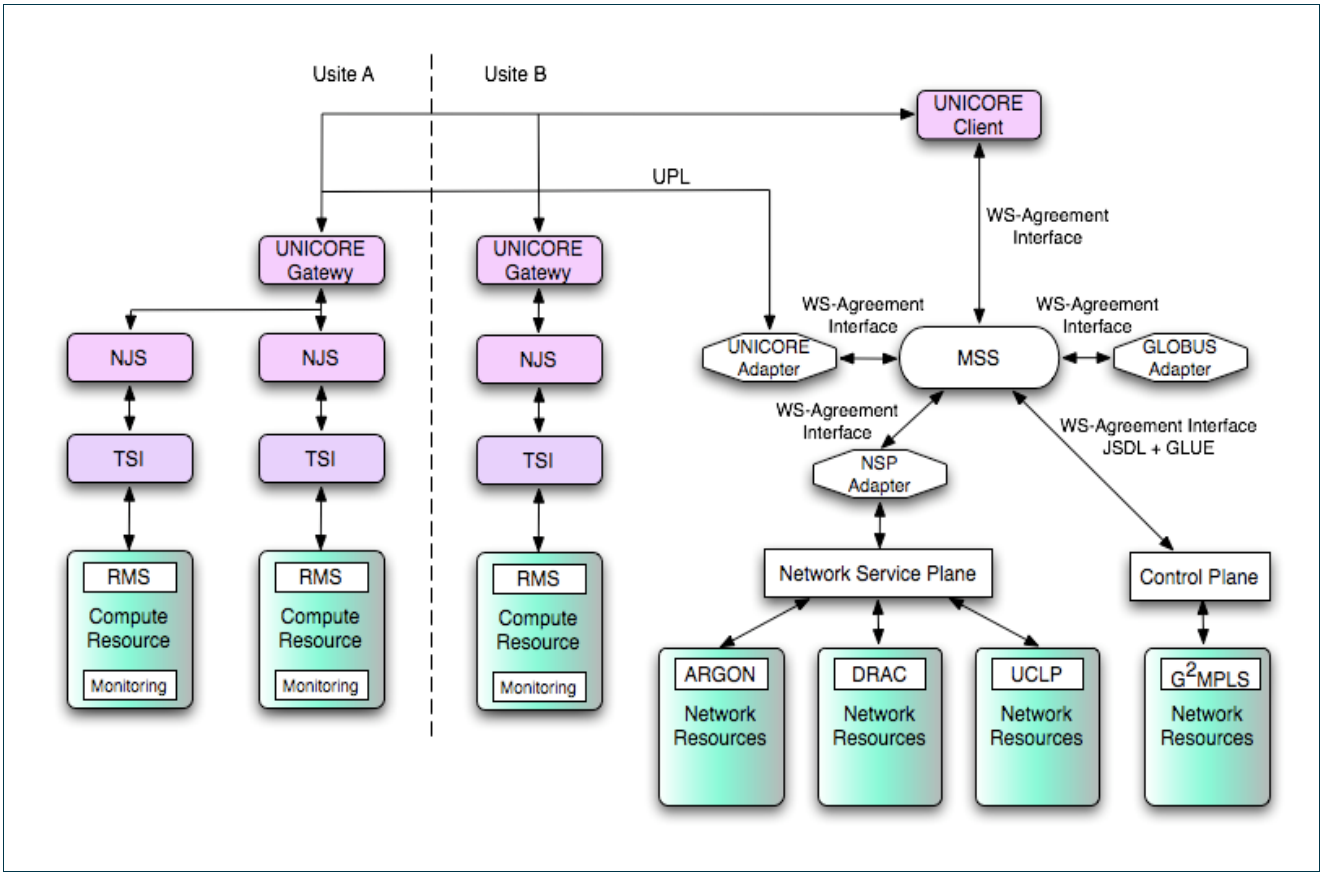


Figure 5.2: Integration of the MetaScheduling Service and the Network Service Plane and the Control Plane

The final modifications of the MSS concerned the integration in the new, web services-based UNICORE 6, which replaced UNICORE 5 as Grid middleware in the testbed. The modifications concerned mainly the new security model of UNICORE 6, which made changes in the MSS interfaces to the UNICORE client and the UNICORE gateway necessary. As a result, the middleware in the PHOSPHORUS testbed is following the WS-Security standard now.

At month 18 of the project the MSS installation in the testbed is stable and ready to be used in the second phase of testbed experiments, which is scheduled starting month 19. The implementation of the interface to the network control plane to provide middleware service to the network layer will be completed as foreseen month 24 of the project.

5.3 Globus

From March 2007 to September 2007, PSNC worked on installing and configuring the Globus Toolkit middleware in the sites that participate in DDSS applications testing. These sites include PSNC (installation and configuration), FZJ (configuration), FhG (configuration) and Essex (installation and configuration). In December 2007, some reconfiguration actions were necessary in Essex – as the testing nodes were changed. In February 2008, Globus installation in FZJ was moved to another machine – again some reconfiguration works were required.

5.4 INCA

As for the first layer (**Data Management Layer**) we have implemented and evaluated the following functionalities:

1. Node insertion and removal
2. Distributed routing table and DHT zone maintenance
3. Application layer routing mechanism
4. Metadata insertion and removal
5. Distributed metadata base maintenance
6. Metadata insertion and data placement
7. Metadata location and data retrieval

Additionally for the second layer (**Storage protocol layer**) we have evaluated:

1. Point-to-point bandwidth utilization

2. ACK strategies during high rate transmissions.

3. Traffic Congestion.

4. Rate Adaptation.

5. Implementation of an Authentication Checksum in order to secure both the non-corruption of packets and the non-spoofing.

Through implemention we have done useful observations that have positively affected our **Data management layer**. For the maintenance of a DHT and the routing tables special attention has to be paid in order to keep the zones and the routing tables consistent. During node insertion and removal temporary inconsistency takes place and system has to handle this situation.

The implementation of the routing mechanism affects the performance, during the routing process, and the stability of the system in dynamic conditions. We have implemented a routing algorithm that infuses these two parameters in INCA.

The hash function that used in the DHTs towards the balanced metadata placement performs well. As for the balanced data placement small data chunks have to be used due to the variant file size of the files that inserted in the system.

At last as replication is the only way towards a fault tolerant system special attention has to be paid on it in order to avoid inconsistency and data loss.

As for the **Storage protocol layer**. Through our experiments we have pointed out some issues on the testbed, as we achieve performances that are 50% less the one achieved on another testbed. The issues are partially linked to the kind of access and hardware, but this brings in evidence some key points:

1. The need of jumbo frames in order to exploit the gigabit link (ok, this is a well known issue but it seems to not be taken in account in other systems);

2. A good tuning of nodes is needed;

3. In absence of that some development has to be done to deal with kind of issues and to try to counter that in **storage protocol layer** (precisely HITP).

This development is vertical to ACK strategies and rate adaptation strategies and of course is focused in circumventing bandwidth waste due to unsuitable nodes setup.

5.4.1 Plan for implementation and deployment

As it is obvious from the aforementioned description suitable environment for the implementation of our system can be any platform that can manage and support a storage space and network cards. The hole INCA consists only from software components that apart both layers that presented earlier.

The deployment could also be done in any site of the project and additionally in any other site. We are currently negotiating for more nodes in order to evaluate our system in a condition where more INCA nodes will be present.

6 Deliverables and Milestones

6.1 D3.1 – Report on use-cases, requirements and design of the changes and extensions of the applications and middleware

This report contains use-cases for all applications and describes the design changes and extensions to the middlware and applications that are required to implement the use-cases on the test bed. The deliverable was due in project month 3, December 2006 and completed in time.

6.2 D3.2 – Report on middleware extensions and implementation

This report describes the status of the changes and implementation of the first version of the middleware as defined for milestone M3.2. It's main input originates from D3.1, which lays out the design of changes and extensions to applications and middleware. The deliverable was due in project month 12, September 2007.

6.3 D3.3 – Report on initial adaption of applications

This report describes the status of the changes and implementation of the first version of the applications as defined for themilestone M3.3. The deliverable was due in project month 12, September 2007.

6.4 D3.4 – Report on planned integration and evaluation

Originally due in month 14, the due date moved to month 16, January 2008, during renegotiation. This report describes the requirements derived from the first test-bed results and the planned further optimisations of middleware during the second phase of the project – as defined in milestone M3.4.

6.5 D3.5 – Final report on achievements and results of the first 18 months' period

Deliverable 3.5 (this document) summarizes the achievements of WP3 during the first 18 month period, including inparticular the second version of middleware and aplication as defined in the milestones M3.5 and M3.6. The deliverable is due in project month 18, March 2008.

6.6 M3.1 – Use-cases, requirements, and design defined

M3.1 was due in month 3 and finished with the submission of D3.1. Use-cases that will be the basis for test-bed demonstrations have been defined. The middleware and application enhancements required to carry out those demonstarations were specified. A report on the use cases, requirements and the design of the middlware and application changes and extensions has been completed, see D3.1.

6.7 M1.3 – Prototypes for interoperability between the Service Plane and the Service layer

M1.3 was a joint milestone with WP1. It was delivered in project month 12. Details can be found in the M1.3 description of WP1.

6.8 M3.2 – First version of the middleware completed

A first functional implementation of the changes and extensions of the middlware to carry out the application use-cases has been completed with this milestone. This implements the changes and extensions defined in D3.1. This milestone has been completed in month 12.

6.9 M3.3 – First adaption of applications completed

This milestone is the equivalent of M3.2 for applications. It was also delivered in month 12.

6.10 M3.4 – Requirements derived from test-bed experiments

Following the delivery of M3.2 and M3.3, test-bed experiments were conducted, from which the requirements and need for changes in applications and middleware were derived. This second run of

requirements are taken as a basis for further developments during the second phase of the project and define the functional range of intermediate versions to be delivered in M18.

6.11 M3.5 – Second version of the middleware completed

The enhancements of the middleware according to the requirements formulated in M3.4, part 1 are have been completed.

6.12 M3.6 – Second adaption of applications completed

According to the original project plan, this deliverable was due in month 18, March 2008. However, for the second phase of the project, it is due in month 22, July 2008 and won't be reported on here.

7 References

[D3.1] Report on use-cases, requirements and design of the changes and extensions of the applications and middleware

[D3.2] Report on Middleware Extensions and Implementation

[D3.3] Report on Initial Adaptions of Applications

[D3.4] Report on Middleware Extensions and Implementation

[D6.4] Running Demonstration Application

[ARGON] Allocation and Reservation in Grid-enabled Optic Networks
http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0051.pdf

[EASY] Extensible Argonne Scheduling system
http://www.springerlink.com/content/pn740068375677wt

[CIM] Common Information Model
http://www.dmtf.org/standards/cim/

[GT4] Globus Toolkit 4
http://www.globus.org

[GridFTP] GridFTP. http://www.globus.org/grid_software/data/gridftp.php

[GridFTP2] GridFTP: Key Concepts.
http://www.globus.org/toolkit/docs/3.2/gridftp/key/index.html

[OASIS] Organization for the Advancement of Structured Information Standards
http://www.oasis-open.org

[OGSA-RSS] Open Grid Service Architecture Resource Selection Services Working Group
http://www.ogf.org/gf/group_info/view.php?group=ogsa-rss-wg

[MSS] MetaScheduling Service
http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0010.pdf

[OGF] Open Grid Forum
http://www.ogf.org

[PELLET] http://pellet.owldl.com/

[PROTÉGÉ] http://protege.stanford.edu/

[SPARQL] http://www.w3.org/TR/rdf-sparql-query/

[TORQUE] http://www.clusterresources.com/pages/products/torque-resource-manager.php

[TSMBA] Tivoli Storage Manager. http://www-306.ibm.com/software/tivoli/products/storage-mgr/

[TUAM] Tool for Universal Annotation and Mapping
http://www.scai.fraunhofer.de/index.php?id=2384&L=1

[U@SOURCEFORGE] UNICORE Sourceforge Project

	http://sourceforge.net/projects/unicore/
[unicore]	Uniform INterface to COmputing REsources
	http://www.unicore.eu
[VIOLA]	Vertically Integrated Optical Testbed for Large Applications in DFN
	http://www.viola-testbed.de/
[WS-RF]	Web Services Resource Framework
	http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf - announcements

8 Acronyms

AAA	Authentication, Authorisation, Accounting
DDSS	Distributed Data Storage Systems
DDSS B/A	Backup/Archive application used in the PHOSPHORUS testbed as one of the DDSS applications
e2e	end to end
EGEE	Enabling Grids for E-sciencE (European Grid Project)
FC	Fibre Channel
FC-SATA	Fibre Channel to SATA technology (mixed technology used in disk matrices: disk matrix have Fibre Channel ports for hosts connectivity, but contains SATA disk drives)
GEANT2	Pan-European Gigabit Research Network
GEANT+	the point-to-point service in GEANT2
GMPLS	Generalized MPLS (MultiProtocol Label Switching)
G^{2}MPLS	Grid-GMPLS (enhancements to GMPLS for Grid support)
GT4	Globus Toolkit Version 4 (Web-Service based)
INCA	
KoDaVis	Tool for Distributed Collaborative Visualisation
MSS	MetaScheduling Service
NREN	National Research and Education Network
NRMS	Network Resource Management System
NRPS	Network Resource Provisioning System
PoP	Point of Presence
Protégé	Ontology Editor and Knowledge Acquisition System
QoS	Quality of Service
SNMP	Simple Network Management Protocol
TOPS	Technology for Optical Pixel-Streaming
TPD	Tiled Panel Display
TUAM	Tool for Universal Annotation and Mediation
UNI	User to Network Interface
UNICORE	European Grid Middleware (UNIiform Access to COmpute REsources)
VLAN	Virtual LAN (as specified in IEEE 802.1p)
VIOLA	A German project funded by the German Federal Ministry of Education and Research (Vertically Integrated Optical Testbed for Large Applications in DFN)
VPN	Virtual Private Network
WISDOM	Wide In Silicio Dockong On Malaria