034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D.3.1

Use-cases, requirements and design of changes and extensions of the applications and middleware

Due date of deliverable: 2006-12-15
Actual submission date: 2006-12-22
Document code: <Phosphorus-WP3-D.3.1>

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
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)	

Abstract

The main objective of the Phosphorus project is to address some of the key technical challenges to enable on-demand e2e network services across multiple domains. The Phosphorus network concept will make applications aware of their complete Grid resources (computational, storage and networking) environment and capabilities, and leverage dynamic, adaptive and optimized use of heterogeneous network infrastructures connecting various high-end resources. Phosphorus will enhance and demonstrate solutions that facilitate vertical and horizontal communication among applications, middleware, existing Network Resource Provisioning Systems, and the proposed Grid-GMPLS Control Plane. These developments will be validated and demonstrated in a test-bed with a set of applications which access the new services via Grid middleware. To achieve this, existing Grid middleware has to be adapted and enhanced to support the new services provided by Phosphorus. Also, the applications that are going to be deployed in the test-bed have to be enhanced to make the best possible use of these services.

This document defines the use-cases of the applications that will be deployed in the test-bed and analyses the requirements that have to be fulfilled by the infrastructure of the test-bed to enable this deployment. This includes network, security, hardware, Grid middleware and other software requirements. From these requirements, the necessary design changes and extensions of both the middleware and the applications are derived.

Table of Contents

Table of Figures

Figure 4.1 Components of the current KoDaVis version and their interaction 32
Figure 4.2 Design of the planned extensions of the KoDaVis software. 34

0 Executive Summary

The main objective of the Phosphorus project is to address some of the key technical challenges to enable on-demand e2e network services across multiple domains. The Phosphorus network concept will make applications aware of their complete Grid resources (computational, storage and networking) environment and capabilities, and leverage dynamic, adaptive and optimized use of heterogeneous network infrastructures connecting various high-end resources. Phosphorus will enhance and demonstrate solutions that facilitate vertical and horizontal communication among applications, middleware, existing Network Resource Provisioning Systems, and the proposed Grid-GMPLS Control Plane. These developments will be validated and demonstrated in a test-bed with a set of applications, which access the new services via Grid middleware.

This document defines the use-cases of the applications that will be deployed in the test-bed and analyses the requirements that have to be fulfilled by the infrastructure of the test-bed to enable this deployment. This includes network, security, hardware, Grid middleware and other software requirements. From these requirements, the necessary design changes and extensions of both the middleware and the applications are derived

The analysis of the requirements of the applications indicates that most of the middleware extensions are related to UNICORE: the VIOLA MetaScheduling Service will be enhanced to make use of the new Phosphorus services and will be integrated into the new UNICORE 6 system. However, it is planned to integrate the MSS into Globus Toolkit 4 also, in order to broaden the applicability of the results. Another major outcome of the use-case and requirement analysis of this document is the formulation of a consolidated set of requirements of work package 3 towards the test-bed (work package 6) and towards work packages 1 and 2, which will define, design and implement the new network services. These requirements are also going to enter the deliverables of these respective work packages, in particular the test-bed design document D6.1.

1 Introduction

The main objective of the Phosphorus project is to address some of the key technical challenges to enable on-demand e2e network services across multiple domains. The Phosphorus network concept will make applications aware of their complete Grid resources (computational, storage and networking) environment and capabilities, and leverage dynamic, adaptive and optimized use of heterogeneous network infrastructures connecting various high-end resources. Phosphorus will enhance and demonstrate solutions that facilitate vertical and horizontal communication among applications, middleware, existing Network Resource Provisioning Systems, and the proposed Grid-GMPLS Control Plane. These developments will be validated and demonstrated in a test-bed with a set of applications which access the new services via Grid middleware. To achieve this, existing Grid middleware has to be adapted and enhanced to support the new services provided by Phosphorus. Also, the applications that are going to be deployed in the test-bed have to be enhanced to make the best possible use of these services.

The purpose of this document is to lay the foundation for the planned extension of both middleware and applications as well for their deployment in the test-bed. In particular, the following topics are addressed:

1. Use-cases and requirements of the application:
   In the technical annex of the project, four applications are provided as demonstration applications. These applications have been selected to cover a broad spectrum of different communication demands and usage scenarios. To allow for a smooth deployment in the test-bed, it is essential to define the use-cases of the applications that shall be executed in the test-bed and to specify their prerequisites. For that purpose, we have developed a questionnaire that tries to collect all relevant information, including the use-case description and the concrete requirements regarding networking, security, hardware (computing, storage, and visualization), middleware and applications. The questionnaire is provided in Appendix A for reference, the results are given in section 2.

2. Middleware design changes and extensions:
   From the use-cases and application requirements the requirements regarding middleware have been extracted and analysed. Two applications will rely on UNICORE and access the new Phosphorus services via UNICORE. Therefore one focus of the middleware extensions will be a deep integration of the MetaScheduling Service that provides resource orchestration for the applications into UNICORE. The other applications will access the Network Resource Provisioning System (NRPS) directly. Therefore no middleware extensions are required for them. The planned design changes and extensions of the middleware are described in section 3

3. Application design changes and extensions:
   In order to make use of the new Phosphorus services, the applications have to be extended. As already mentioned, two of them, WISDOM and KoDaVis will rely on the Grid middleware for that purpose. Therefore the major task for these applications is to interface them with the middleware.
   DDSS/GridFTP relies on Globus Toolkit 4. Since the network is the only resource, that this application has to allocate, it is planed that is accesses the NRPS directly. The other applications (DDSS/TSM and TOPS) do not rely on a particular middleware and will also access the NRPS directly. The planned design changes and extensions of the applications are described in section 4 of this report.

The rest of this report is organized as outlined above. The description of the use-cases and requirements of the applications is followed by a thorough description of the planned design changes and extensions of the middleware and the applications.

2 Use-case descriptions and requirements

2.1 WISDOM

2.1.1 Users perspective of the Use-case

WISDOM (Wide In Silico Docking On Malaria) is a docking workflow/service which allows the researcher to compute millions of compounds of large scale molecular dockings on targets implicated in diseases like malaria (in silico experimentation). It can compute up to 46 million docked ligands in one month, depending on the Grid-infrastructure. On classical computer cluster the result is 10.000 docked ligands. In silico docking enables researchers to compute the probability that potential drugs will interfere with a target protein and it is one of the most promising approaches to speed-up and reduce the cost to develop new drugs to treat diseases such as malaria. So WISDOM presents both a compute and data challenge.

Use-case: A scientist wants to perform a molecular docking to propose new inhibitors for the targets implicated in malaria by using a high-throughput docking approach. By deploying an application that generates a large date flow, the output could be more than 1TB.

There is a pre-staging phase where the software and the input-data have to be transferred to the sites where the docking simulation will be running (estimated size of data: 1/2GB) and a post-staging phase, where the output-data is gathered and stored in a common data-base (estimated size: 1/2TB).

To get a reliable service for pre- and post-staging, the user may also request to reserve network bandwidth between the site with the input data, the site with the result data-base and all compute-sites.

2.1.2 Topology and general requirements

Which sites are involved?
Initially Fraunhofer SCAI in Germany, Physics Laboratory (CNRS/IN2P3) in France, PSNC in Poland, University of Essex, Great Britain. Later also other sites of the PHOSPHORUS testbed

How often and how long to you intend to test/use?
We would propose a test phase in the early stage of the project, e.g. 2 weeks to test, and the correct set up and an execution phase for the real data processing, e.g. 4 weeks.

Define preferred/possible timeslots for tests
No special requirements.

2.1.3 Local resources at participating sites

Define hardware requirements
500 Mb main memory, license server (FLEXlm), database server (Oracle)

Define OS & software requirements
Linux, pre-installation of FlexX/Autodock

Are additional/separate instances of the OS required?
No.

Accounts required?
One normal user account.

Permanent resource usage?
Except for the disk space for the data-base, recourses are required only during tests.

2.1.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?
No requirements.

Are special network layouts required?
No.

Do you require special protocol support?
No.

Are there any performance requirements on service setup, teardown and recovery?
No.

Are there MTU requirements?
No.

Are there special requirements for firewall configuration?
License information has to be tunnelled.
Maybe, for the database access.

Do you require (and is it possible) to separate the resources and the test-bed network from the local network?
Not required, but possible.

2.1.5 Middleware requirements

Is the application tied to a special Grid-middleware?
The last experiments (EGEE Data Challenge) have been performed using the EGEE Middleware stack. The application (environment) has to be migrated to the initial UNICORE environment of PHOSPHORUS.

Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?
The reason for failure should be logged and depending on the reason the user should be noticed or the job should be rescheduled automatically.

2.1.6 AAA requirements

Are there any, beyond what's provided from the middleware?
No.

2.1.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system?
Currently no. The current approach is to have a Monitor task that is in charge of submitting and tracking the submitted jobs.

2.1.8 Which parameters do you have to specify in your resource requirement?

+ explicit specification of resource
– broker to select resource according to requirements
+ workflow specification
+ local resources: number of CPUs
+ network resources: bandwidth between the site with the input-data,
the site hosting the data-base and all computing sites.
+ time constraints: duration, time-window(s)
* time domain: 1 min.,typical duration of usage: at least 5 min. (depends on performance)
– complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))
+ reservation of licenses, mechanism to tunnel license information through firewalls (if any)

legend: + must have, * nice to have, - not needed

2.2 KoDaVis – Distributed collaborative visualization

2.2.1 Users perspective of the Use-case

Climatologists handle huge collections of data, partly measured data (as provided by ECMWF), partly data coming from simulations performed on supercomputers that simulate different scenarios based on the measured data. A typical data-set contains 1 TB of data. Such data-sets are stored at the supercomputer site and not locally at the scientists' lab. For visual analysis, only part of the data is accessed, but it cannot be fully specified in advance, which part of the data needs to be accessed during the exploration session.

Use-case 1: A scientist wants to perform visual analysis of the data. He is e.g. using a high-end visualization facility at his own lab for the session, is visiting a remote site with a graphics workstation or may only have his laptop at hand for a demo at a conference. A session can be scheduled for a specific time or be spontaneous. In both cases the scientist starts his visualization application and requests a data-service from the system on which the data is stored. To get a reliable service, he may also request to reserve network bandwidth between the data service and his local visualization device. The request would be: I need data-service for one client for '1 hour' 'now' or 'tomorrow starting at any time between 14:00 and 17:00'. At the same time I need 700 Mbit/s reserved bandwidth between the data-service and my location.

Use-case 2: Scientists at two or more different sites want to collaboratively exploit the data. They may be using different hardware and software environments for that purpose. Nevertheless they want to communicate via video or at least audio and need to have a common, synchronized view on the data. The request would be: we need data-service for three clients for '1 hour' 'now' or 'tomorrow starting at any between 14:00 and 17:00'. At the same time we need 700 Mbit/s reserved bandwidth between the data-service and each of the three locations.

Remark: a more advanced request might also include the personal calendars of the scientists and reservation systems for the high-end visualization facilities to negotiate the earliest possible timeslot, where all the requirements can be met. But we do not expect this to be implemented in PHOSPHORUS.

2.2.2 Topology and general requirements

Which sites are involved?

The data-service is located at FZJ. The data consuming visualizations are located at FZJ and Amsterdam.
More consumer-sites would be nice to have (see call for participation)
Topology: star-like with the data-service in the centre of the star.

How often and how long to you intend to test/use?

After a setup-phase where the software is installed and tested, further use is mainly planned for demonstrations and test of new test-bed functionality.

Define preferred/possible timeslots for tests

Since the application is about human interaction, tests will mainly be during working hours.

2.2.3 Local resources at participating sites

Define hardware requirements

Data-service: PC-Cluster with at least 2 CPUs per visualisation client, more CPUs are used as data-cache and are therefore beneficial. Storage: 30 GB disk space sufficient for tests, up to 1 TB can be used, disk performance is not important, since the service uses a memory cache.
Visualisation clients: any system that can run AVS/Express, from laptop to VR-systems

Define OS & software requirements

Data service: any Linux/UNIX with MPI
Visualisation clients: Linux/UNIX and AVS/Express (license required !)

Are additional/separate instances of the OS required?

No, probably not feasible.

Accounts required?

One normal user account.

Permanent resource usage?
Except for the disk space on the data-service, recourses are required only during tests.

2.2.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?
Between data-service and each client (see q.2): 700 Mbit/s, ~50 msec latency
Between all client sites (video): ~2 Mbit/s for Access Grid or 25 Mbit/s for DVTS

Are special network layouts required?
No.

Do you require special protocol support?
In case of more than 2 clients: Multicast for AccessGrid

Are there any performance requirements on service setup, teardown and recovery?
1 Minute is sufficent for service setup and recovery, no requirements on tear-down

Are there MTU requirements?
Jumbo frames preferred for performance, but not strictly required

Are there special requirements for firewall configuration?
AccessGrid requires Multicast support in Firewall
Connection between data-service and clients requires a small number (~10) of unprivileged TCP-Ports, Port numbers are configurable.

Do you require (and is it possible) to separate the resources and the test-bed network from the local network?
Not required, but possible.
Data-Service in FZJ is located in normal LAN, visualization clients can be anywhere

2.2.5 Middleware requirements

Is the application tied to a special Grid-middleware?
Currently, it directly accesses the ARGON system for bandwidth reservation.
Reservation for the data-service is not yet included but will be added during the project.
Plans are to access the MetaScheduling service via UNICORE.

Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?
If the network service fails, automatic recovery by middleware or any other lower layer would be useful but is not a strict requirement.

2.2.6 AAA requirements

Are there any, beyond what’s provided from the middleware?
To be specified

2.2.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system?
See question 2.2.5: The Web-Service interface of ARGON is accessed via a small Perl-Library. From the MetaScheduling service (MSS) the ability for co-ordinated advance and immediate reservations of network connections and CPUs for the data-service is expected.

2.2.8 Which parameters do you have to specify in your resource requirement?

+ explicit specification of resource
– broker to select resource according to requirements
– workflow specification
+ local resources: number of CPUs
+ network resources: bandwidth & latency between data-service and all clients
+ time constraints: duration, time-window(s), immediate
+ time domain: granularity: 1 minute, typical duration of usage: 1 hour
– complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))

legend: + must have, * nice to have, - not needed

2.3 TOPS – Streaming of Ultra High Resolution Data Streams

2.3.1 Users perspective of the Use-case

A climatologist has performed large scale simulations of Earth's climate. The size of the simulation output is enormous, and needs to be analyzed. To do this, the climatologist evaluates the data that is visualized on large display devices (Tiled Panel Display, or TPD devices).

The climatologist is currently visiting a lab of a collegue, somewhere in Europe. This lab is equiped with a TPD. A remote visualization service is used to generate imagery on the TPD. The visualization service resides in close proximity with the data. The data remains secure in the data center, and only pixels of the visualization are streamed to the panel.

The user switches on a TPD device, and requests a visualization service from the data center. The user specifies as start of resource allocation: 'now', and duration '1 hr'. The reservation system comes back with the message that the resource is occupied, and suggests a different start time. The user accepts.

At the reserved start time, the user switches on the TPD again, and starts the TOPS system. TOPS makes reservations for the required bandwidth and graphics resources, and starts rendering, and sending pixels. The user pans/zooms in the climate simulation output. After one hr, the user is still using the application, but TOPS is terminated automatically as the resource allocation has expired.

2.3.2 Topology and general requirements

Which sites are involved?
The first sessions will be between SARA and FHG. At both sides, clusters need to be connected. One cluster at SARA, the other cluster at FHG. Additional sessions will be between Amsterdam and Barcalona, and also between Amsterdam and Prague.

How often and how long to you intend to test/use?
We expect roughly 1 hour sessions, that occur roughly weekly.

Define preferred/possible timeslots for tests
These will be at office hours.

2.3.3 Local resources at participating sites

Define hardware requirements
The sessions are always with two sites participating. At one site, a display facility, such as a Tiled Panel Display should be available. The other side will act as data source, and will require 3D graphics clusters for the 3D datasets.

At the display side the incoming pixels will be displayed via the graphics card of the receiving machine. An input device is needed to allow the user to interact with the displayued data (i.e. camera position).

Define OS & software requirements
Required OS is GNU/Linux. At SARA we use Debian GNU/Linux, but we expect other Linux distributions to work just fine. No commercial software licenses are required. Build dependencies for TOPS include the following software packages:

- swig
- python2.4-dev
- liblzo-dev

- libtiff4-dev
- sdl-dev
- libxxf86dga-dev

Are additional/separate instances of the OS required?
None

Accounts required?
A user account is required with SSH access. We need to be able to start processes on the two clusters using the SSH authorized_keys approach.

Permanent resource usage?
Roughly 20Gbyte per cluster node is req'd.

2.3.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?
Preferably a 10 Gbit/s bandwidth. Absolute minimum for testing is 1 Gbit/s. The latency should be 100ms roundtrip or better. The connection should support Jumbo frames with 9000 byte MTU.

Are special network layouts required?
No

Do you require special protocol support?
Only UDP and TCP

Are there any performance requirements on service setup, teardown and recovery?
We require a 1 minute setup time, or better.

Are there MTU requirements?
9000 bytes or better.

Are there special requirements for firewall configuration?
Some ports should be opened on the firewall. Local security policies must allow to login to the display machine from outside.

Do you require (and is it possible) to separate the resources and the test-bed network from the local network?
Probably not. Local security policy does not allow to access machines from the outside which are in the internal network. A setup has to be found that separates internal data or networks from access through the phosphorus

network. At least for the testing phase access from outside to the display machine must be possible. For the final scenario the display resource should operated only from the display site network.

2.3.5 Middleware requirements

Is the application tied to a special Grid-middleware?.
No

Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?
The connectivity should be restored ASAP, as this is an interactive application.

2.3.6 AAA requirements

Are there any, beyond what's provided from the middleware?
We currently use SSH authentication. It has to be evaluated what are the consequences on the security policies of each site.

2.3.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system?

Not yet, but this is desirable.

2.3.8 Which parameters do you have to specify in your resource requirement?

+ explicit specification of resource
– broker to select resource according to requirements
– workflow specification
– local resources: number of CPUs
+ network resources: bandwidth & latency between data-service and all clients
+ time constraints: duration, time-window(s), immediate
+ time domain: granularity: 1 minute, typical duration of usage: 1 hour
– complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))

legend: + must have, * nice to have, - not needed

2.4 DDSS – Distributed Data Storage System

2.4.1 Users perspective of the Use-case

Distributed Data Storage Systems (DDSS) are widely used to transport, exchange, share, store, backup/archive and restore data in many scientific and commercial applications. Proposed test scenarios include two DDSS use cases: data transfers performed with use of educational, open-source GridFTP application and backup/archive/restore operations made by a commercial application.

Use case 1: GridFTP is a high performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area IP networks. It works in the client-server architecture. GridFTP server can store or retrieve the data from/to single or multiple clients (many-to-one). Data transmission between client-server pair can be done over 1-128 parallel streams (stripes). Typical users of GridFTP are end-users that need to transport the data to/from the remote computing system (e.g. input/output data in Grid systems) as well as the Grid systems and applications that use the GridFTP services as lower-layer data services in order to move input/output/intermediate data between system nodes, sites etc. Typical GridFTP session is started by end-user (typically through the command-line interface) or it is started by the Grid system/application typically through command-line or API interface.

Normally, no advance resource reservation is made for GridFTP sessions. The new Phosphorus services could be used to minimise and/or guarantee the total time of the data transmission. Bandwidth reservation feature would be useful for movement of large files. Network delay warranty feature could speed-up transfers of multiple smaller files. Reservation of multiple network paths at the same time could be exploited in many-to-one data transfer scenarios (e.g. data distribution or gathering).

Use case 2: Backup/archive application (B/A application) is used for performing automatic, centralised backups and/or archive copies of data that are originally stored on the client machines. Data are collected on clients by a client B/A modules and send through TCP/IP connections to B/A server module. B/A server manages storage pools that can include disks, tapes and other storage media and stores user data in these pools, according to the defined policy. Backup/archive copies are performed according to some typical schemes: full, incremental or cumulative copy. Typical setup contains single B/A server and multiple B/A clients. Multiple server setups are difficult to deploy, therefore they are not going to be tested. In the test-bed, we plan to exploit IBM Tivoli Storage Manager as the B/A software.

Typical users of B/A application are system administrators that intend to protect the end-users data against damage of data and/or archive the data for longer periods. B/A applications' functionality allows automating and optimising this process. B/A clients are installed on workstations and computing servers. B/A server is typically run on the designated storage server with huge storage resources connected to it, e.g. tape libraries, large disk matrices etc. B/A session is typically started on the client machine by a scheduler (central or local one that runs on the client machine). B/A session can also be initiated 'on-demand' by the end-user or the B/A application administrator

2.4.2 Topology and general requirements

Which sites are involved?
Use case 1: GridFTP
Grid FTP server(s) – in PSNC, FZJ, Fraunhofer (one per site or many per site).
Grid FTP clients – in PSNC, FZJ, Fraunhofer, + others ... (many per site)

Use case 2: B/A application
B/A server: PSNC and FZJ (using PSNC's and FZJ's server licenses)
B/A clients: basically PSNC and FZJ (using PSNC's and FZJ's client licenses); more client sites welcome

More clients-sites would be good. A call for participation has been issued for that purpose.

Topology:	
Use case 1: GridFTP:	one-to-one, many-to-one: -> STAR
Use case 2: B/A application:	one-to-one, many-to-one: -> STAR

How often and how long to you intend to test/use?
After preparation of some automation mechanisms, scenarios can be run constantly, periodically or on-demand (during demos, tests of network functionality).

Define preferred/possible timeslots for tests
Since the application does not require human interaction, tests can be performed during either working or off-hours.

2.4.3 Local resources at participating sites

Define hardware requirements
Use case 1: GridFTP
Client side: (one-to-one, many-to-one)

PC machines 1-2 CPUs, 512+ MB RAM or similar
at least one 1GEth interface
at least 100 GB of disks; more welcome – if more space easier it will be easier to manage multiple tests
the disk subsystem should be able to read/write at least 40MB/s,
the more performance, the more “spectacular” measures, but above is enough

Server side (many-to-one setup):

PC machines 2+ CPUs, 512+ MB RAM or similar
one or two 1GEth interfaces,
efficient PCI bus (PCI-X or PCI express)

at least 400 GB disks,
disk system able to read/write with speed = ((number of clients) x (40 MB/s))

Use case 2: B/A application:
Client side: like for GridFTP

Server side (many-to-one setup):

like for GridFTP, PLUS:
1 or more Fiber Channel interfaces connected to SAN or directly to FC matrix
High-end disk matrix and/or tape subsystem - able to write ((number of clients) x (40 MB/s))

Define OS & software requirements
Use case 1: GridFTP:
Clients/servers:
Linux, preferably RedHat/Fedora or SuSE
Globus Toolkit v. 2.4 or higher (at least Data Management modules including: GridFTP server and Globus-url copy tool),

Use case 2: B/A application:
Clients:
Linux, preferably RedHat/Fedora or SuSE
Tivoli Storage Manager Backup/Archive Client

Servers:
depending on what partners have “under” the TSM server.
Tivoli Storage Manager v5.2, v5.3 etc.

Are additional/separate instances of the OS required?
Use case 1: GridFTP:
Client side: not necessary
Server side: would be convenient (to not to interfere with existing system configuration etc.) but is not necessary.

Use case 2: B/A application:
Client side: not necessary
Server side: perhaps impossible – TSM servers used in test-bed are 'productive'

Accounts required?
Use case 1: GridFTP
Client side: access to normal (root not needed) for PSNC people
Server side: access to root account

Use case 2: B/A application:
Client side: access to normal (root not needed) for PSNC people
Server side: no

Permanent resource usage?
Use case 1 & 2: Not needed, during tests only.

2.4.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?
Bandwidth: max. 40-60MB per data stream; no special requirements on
Latency: no special requirements – as long as it does not compromise bandwidth

Are special network layouts required?
No

Do you require special protocol support?
No

Are there any performance requirements on service setup, teardown and recovery?
Use case 1 and 2: general requirement is to start and/or recover the service ASAP, no strict time conditions. Exception is use case 2 (B/A application), where the backup/archive window can be limited by some external factors, however this applies to the application in general, and is not valid in PHOSPHORUS test-bed.

Are there MTU requirements?
Jumbo frames preferred for performance, but not required

Are there special requirements for firewall configuration?
Use case 1: GridFTP:
Client side: output to GridFTP server port + 50-100 high ports open
Server side: input to GridFTP server port + 50-100 high ports open

Use case 2: B/A application:
Client side: output to TCP port 1500 needed.
Server side: input to TCP port 1500 from selected IPs needed.

Do you require (and is it possible) to separate the resources and the test-bed network from the local network?
Use case 1: GridFTP:
Server side: will be put into DMZ (in PSNC, what about the others?)

Use case 2: B/A application:
Server side: will be put into DMZ in PSNC, is in local network at FZJ

2.4.5 Middleware requirements

Is the application tied to a special Grid-middleware?.
Use case 1: GridFTP:
Tied to Globus Toolkit. However, the only resources that will be demanded by the application are the network links. Therefore the network resources will be reserved by the Grid FTP clients using Network Resource Provisioning System (NRPS) application interface (API).

Use case 2: B/A application:
Not tied to Grid middleware. Functionality expected: bandwidth and/or latency warranty, link “stability” warranty may be achieved by direct use of NRPS API from the application.

Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?
In case of service failure, the service should be recovered ASAP. In ideal situation, the lower level failures should not compromise TCP sessions. This would be useful e.g. for GridFTP transfers with multiple data streams (considerable multiple stream recreation overhead in case of TCP failure) and for B/A jobs (in case of TCP session failure, the whole backup operation can have to be restarted).

2.4.6 AAA requirements

Are there any, beyond what’s provided from the middleware?
No.

2.4.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system?
Use case 1: GridFTP:
No.

Use case 2: B/A application:
No.

What you expect from it (functionality, protocols, programming language for API, …)?
Use case 1&2: GridFTP & B/A application:
Advance or on-demand bandwidth and/or latency reservation,
Coordinated link establishment/reservation for many-to-one setups

Use case 2: B/A application:
“stable” link feature reservation,
advance link reservation (e.g. for scheduled backups)

2.4.8 Which parameters do you have to specify in your resource requirement?

+ explicit specification of resource
– broker to select resource according to requirements
– workflow specification
* local resources: number of CPUs
+ network resources: bandwidth & latency between data-service and all clients
+ time constraints: duration, time-window(s), immediate
+ time domain: granularity: 1 minute, typical duration of usage: 1 to some hours
– complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))

2.5 INCA

2.5.1 Users perspective of the Use-case

There has been a constant raise in the awareness that the handling, in terms of storing and transmitting, the ever growing amount of data will be fundamental in the future. From informal discussion with ISP techies to invited keynote speakers in international conferences, even only loosely linked to SAN / NAS technologies, there is a fil rouge that links their thoughts: how to handle data, and move it somehow, somewhere, in the best possible way, in order to have a set of desired properties, such as seamless, secure, fast, scalable access to storage. Tony Hey, the former director of the UK e-Science program, now at Microsoft, call it data deluge. Citing his words:

> In the next 5 years, the e-science project ALONE will produce more scientific data that has been collected in the whole of human history.

With the current network technology (in terms of equipment, software, and protocols) is it hard to sustain that transfers of very large amount of data (PB+) in a reasonable time is a feasible task, especially sharing the same medium with packets of a few bytes. Of course, we can hope that having a bigger pipe will solve everything, but it is quite clear that we need something more than that. In addition to this, not all data are equal on the network. The multiplication of data types and the infrastructure consolidation trends confirm the need for a better alignment of data networking and data storage requirements.

Use-case 1: Video on-Demand application: This scenario can be also generalised into delivering any kind of streaming video to a device, either fix or mobile. In these scenarios data can be pre-cached and dispatched, and any storage device would be holding the data until the mobile device consumes them. Just to make a practical example, VoD data such as news can be flooded into the network, and then downloaded by final users; the number of replicas and their positioning in different domains would provide necessary QoS level to make this service interesting for both providers and clients.

Use-case 2: Bio-Informatic and pharmaceutical research: Another area in which there is a clear need of a new solution is the bio-informatics and pharmaceutical one. As a concrete example, human genome researchers in France face the problem of having to perform operations such as pattern matching on a large set of files, ranging from megabytes to gigabytes. The biggest centres have the storage capabilities to hold the whole set, while smaller institutions or single researchers cannot afford such storage farms. But even for research centres with sufficient storage capacity, the pattern matching jobs require significant temporary data buffering placed in different locations. The security and confidentiality of the transmission are also a key issue in this field of research. Having a solution that allows a shared access to the entire set of data, keeping in mind that the data has to be extremely secure to forbid unwanted access, is therefore vital for their research. The IBCP (Institute of Biology and Chemistry of Proteins), based in Lyon, France, is currently using a prototype developed by the author in the past, in order to securely manage the data set; according to their director of Computer resources, currently there is no "commercial" solution for their needs. As a mean to scale up their local environment to allow remote consulting and pattern matching analysis through their web-portal, IBCP has to result to a local NFS cluster with a RedHat/Systina base, with a small SAN in the back-end. This is typical of a data business environment where not only there is a need to scale-up locally to provide a service on-line, but also there is a requirement to scale-out with different application spaces and data repositories geographically distributed.

2.5.2 Topology and general requirements

Which sites are involved?
Ideally, we should have 4 or 5 sites with storage capacity, and around the same number (but not the same sites, possibly) for clients. The middleware will manage the data and dynamically choose the most appropriate sites for the data.

Topology: mesh network, not necessarily fully connected.

How often and how long to you intend to test/use?
After a setup-phase where the software is installed and tested, further use is mainly planned for demonstrations and test of new test-bed functionality.

Define preferred/possible timeslots for tests
There are no particular time constraints, although, to prove the functionalities of the middleware, tests should be run both on working hours and at night time.

2.5.3 Local resources at participating sites

Define hardware requirements
Standard PCs with Linux are OK. About space requirements, to make sense we should have around 4 TB disk space in the data nodes, while for the client no particular needs.

Define OS & software requirements

Data service: any Linux/UNIX

Are additional/separate instances of the OS required?

No.

Accounts required?

Debugging might need root powers. Standard operations would need a normal user account.

Permanent resource usage?

Except for the disk space on the data-service, recourses are required only during tests.

2.5.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?

Between data nodes and each client: no particular constraints. The more, the better ☺

Between data nodes. No particular constraints. Again, the more the better.

Are special network layouts required?

No.

Do you require special protocol support?

No.

Are there any performance requirements on service setup, teardown and recovery?

No.

Are there MTU requirements?

Jumbo frames preferred for performance, but not strictly required.

Are there special requirements for firewall configuration?

Port numbers are configurable.

Do you require (and is it possible) to separate the resources and the test-bed network from the local network?

Not required, but possible.

2.5.5 Middleware requirements

Is the application tied to a special Grid-middleware?

No.

Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?
No.

2.5.6 AAA requirements

Are there any, beyond what's provided from the middleware?
No.

2.5.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system?
No.

2.5.8 Which parameters do you have to specify in your resource requirement?

+ explicit specification of resource
+ broker to select resource according to requirements
– workflow specification
– local resources: number of CPUs
+ network resources: bandwidth & latency between data-service and all clients
+ time constraints: duration, time-window(s), immediate
+ time domain: granularity: 1 minute, typical duration of usage: 1 hour
– complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))

legend: + must have, * nice to have, - not needed

3 Middleware design changes and extensions

The initial middleware layer in PHOSPHORUS is based on the infrastructure and developments used in the German VIOLA project: UNICORE and the MetaScheduling Service. In the first phase of the project UNICORE will be used as Grid middleware stack, in the second phase GT4 will be included as additional middleware stack.

For several reasons, e.g. interoperability with GT4 in the second phase, we plan to use the new WS-based UNIICORE version in the PHOSPHORUS environment. While the major changes in UNICORE already have been developed and implemented in other projects the corresponding changes of the VIOLA MetaScheduling Service will be done in POSPHORUS.

The current integration of the MetaScheduling Service and the UNICORE environment is denoted in Figure 3.1. The MetaScheduling Service is an external service communicating to the UNICORE client and performing the negotiation of resource usage and reservation of resources, e.g. for co-allocation at a common timeslot, for the job described by the user in the UNICORE client.

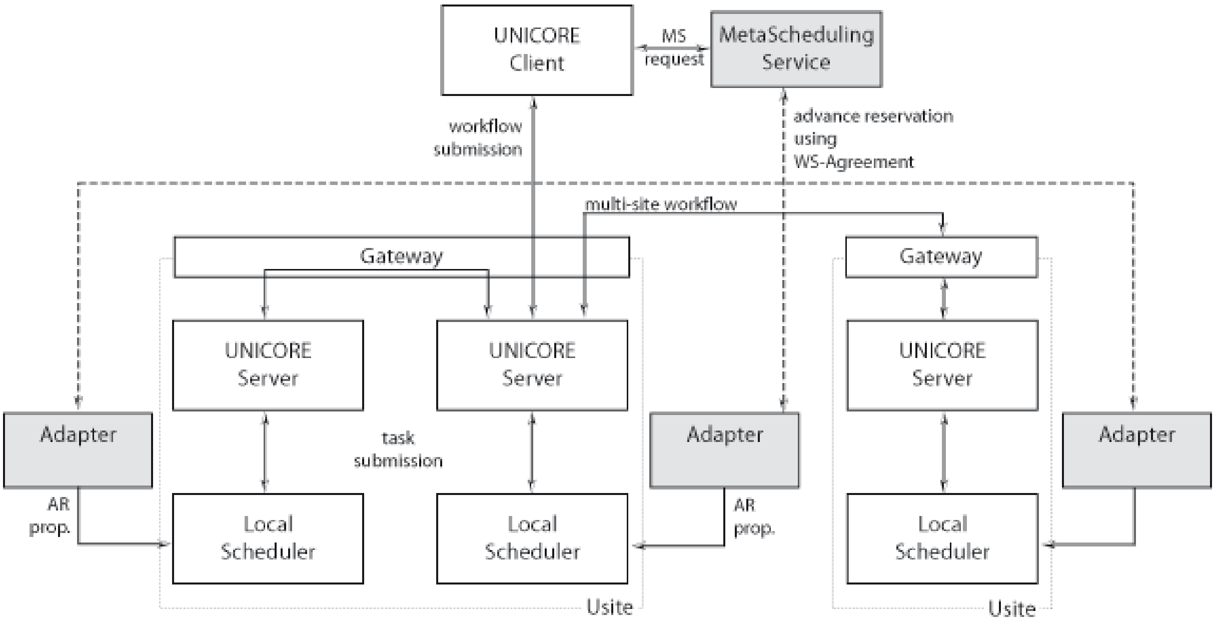

Figure 3.1 VIOLA UNICORE MetaScheduling integration

In contrast to the solution implemented for the integration in the VIOLA UNICORE environment we plan another integration in the WS-based UNICORE environment in PHOSPHORUS. Figure 3.2 sketches the new architecture of the MetaScheduling Service integration. The figure shows the situation for the KODAVIS use-case. In the PHOSPHORUS environment the MetaScheduling Service becomes a distinct UNICORE server responsible for and providing a single service: the negotiation of resource usage and reservation of resources.

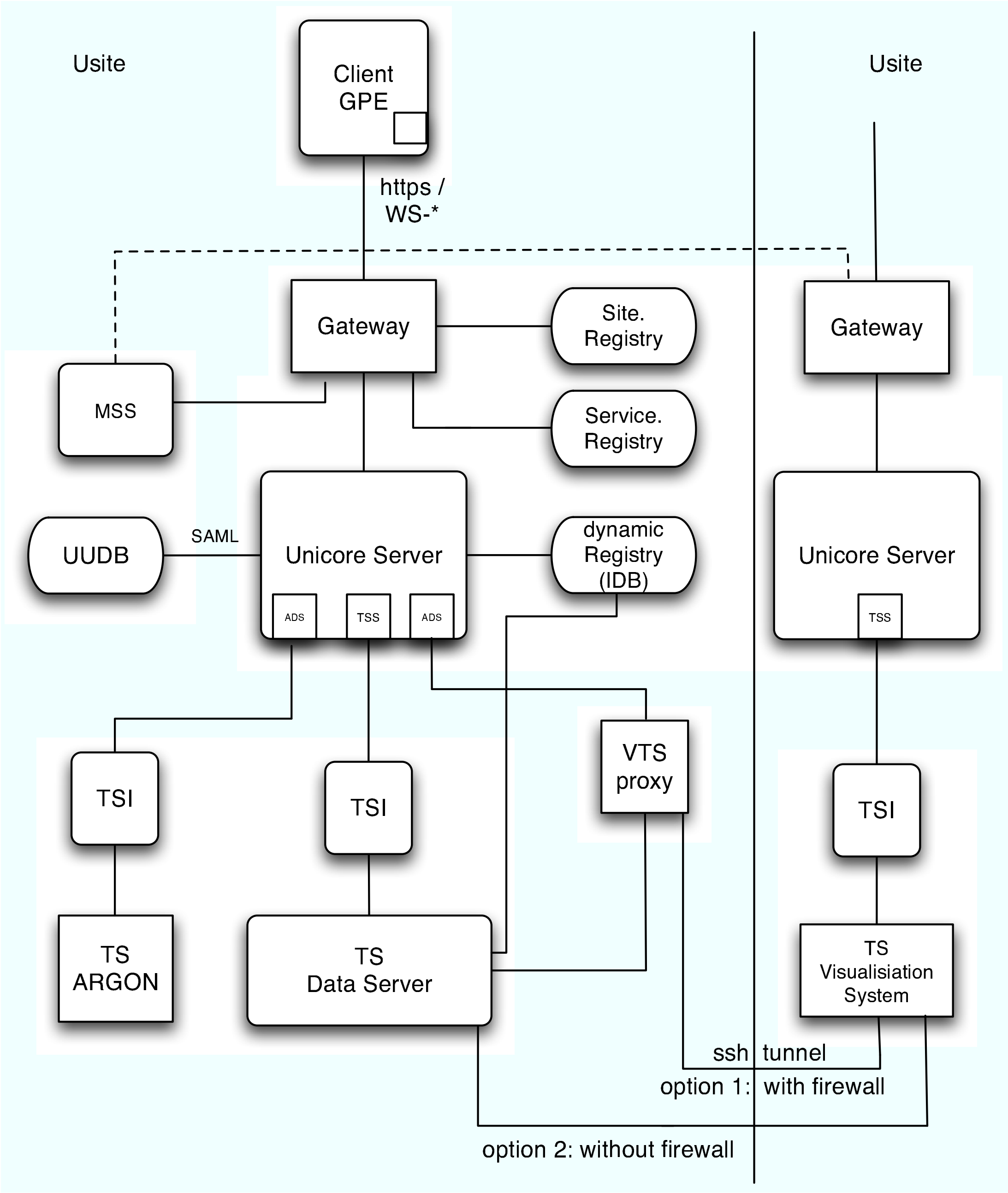

Figure 3.2 Planned Phosphorus UNICORE 6 MetaScheduling integration

The major effort changing the middleware is the modification of the MetaScheduling Service to become a UNICORE Server. Further we need extensions to the protocol between the UNICORE Server and the Target Systems in order to allow the negotiation protocol of the MetaScheduling Service flow along with the other UNICORE protocols. Naturally, this also implies changes in the Target System Interface allowing enabling the negotiation protocol for the Target Systems and the adapters respectively. These adapters are not depicted in Figure 3.2 but will be used for providing the MetaScheduling Service a single interface to the heterogeneous Target Systems. Finally, an extension of the existing protocol between the different UNICORE Servers located in different administrative domains (Usites) has to be implemented. This extension will be used by the MetaScheduling Service for negotiation involving Target Systems located in other Usites.

3.1 Semantic Annotation of Application and Services

In the second phase of the project we want to enhance the Grid's knowledge of each resource request in PHOSPHORUS, so that each service can automatically decide whether it can fulfill the demand or not.

The general practice today is that the user, who wants to submit a request, has to specify the need for the resources a job (i.e. the application, service or workflows composed of applications or services) is demanding in order to work correctly. Thus the user is forced to provide the knowledge of both the applications need and the resources capabilities in order to be able to request the execution of an application in the Grid. To overcome this time-consuming and tedious work we will annotate the applications semantically and provide the MetaScheduling Service on the other hand with the information on the resources' capabilities. Based on these information the MetaScheduling Service is able to select resources matching the applications' need and start negotiating the usage with these suitable resources. If a broker will become available in the PHOSPHORUS environment, this resource pre-selection can already be done on a broker level.

The semantical enrichment of both the applications and the MetaScheduling Service will be done by using Protégé and TUAM (Tool for Universal Annotating and Mapping). Protégé is a free available ontology editor, which was developed at Stanford, USA. The mapping tool, TUAM, is a development of the Fraunhofer Institute of Scientific Computing and Algorithms.

We will use Protégé to build up an ontology, which can be used by the whole Grid middleware environment (e.g. UNICORE Client, GridSphere Portal) to make use of knowledge to describe the specific resource requirements.

We will use TUAM to enrich the application semantically with its proper need of resources. This will be done by annotating the individual application and services (which are the building blocks for jobs and workflows) with their specific requirements.

Submitting an application then implies automatic communication of its resource need to the MetaScheduling Service. The MetaScheduling Service, in turn, can contact the local agreement manager by sending the information to them and gather the answers. The knowledge is provided by a scheduling ontology.

No changes of the application have to be made, as the annotation is kept external to the applications. This approach requires the creation or adoption of an ontology describing resources used in the Grid environment. Additionally some logic has to be either included in the MetaScheduling Service for the mapping of resource demand and resource capabilities or to be provided by an broker instance.

4 Application design changes and extensions

4.1 WISDOM

The applications foreseen to be used for the WISDOM experiments in PHOSPHORUS are FlexX and AutoDock. Both applications are powerful tools for molecular docking simulations, AutoDock being available through a non-commercial license for research use while FlexX is commercially licensed.

There is no need to change the applications for using them in the PHOSPHORUS environment. However, as FlexX is designed for single administrative domain cluster environments rather than multi administrative Grid environments an appropriate mechanism for making the licenses for using FlexX available throughout the PHOSPHORUS test-bed is needed. The most suited approach currently seems to be a license server or, depending on the firewall policies of the resource providers, multiple license servers.

To actually perform the docking simulations a workflow was defined in the EGEE Data Challenge. In EGEE the purpose of this workflow has been pre- and post-staging of input data and results, launching and monitoring of the applications is needed. The WISDOM-workflow of the EGEE Data Challenge was designed for the EGEE LCG environment and is implemented in Perl and Java.

The output of the docking simulations is a large number of potential candidates for creating malaria medicaments, which have to be analysed in a post-processing step after the simulations. In contrast to the

EGEE Data Challenge where the results have been stored in distributed flat files across the EGGE infrastructure, the resulting molecule structures are stored in a single database this time.

Thus the major effort changing the WISDOM application environment will be

1. adopting the EGEE LCG workflow for the use in the PHOSPHORUS environment,
2. setting up the database for the resulting molecular structures,
3. implementing a firewall-aware and transitioning licensing mechanism for the commercially licensed application

4.2 KoDaVis – Distributed collaborative visualization

The current version of the KoDaVis software is sketched in Figure 4.1. It consists of the following components:

- visualization applications,
- a parallel data-server that distributes fragments of data selected by the clients,
- a collaboration server that synchronizes the clients, and
- a control GUI that monitors client activity and interacts with the ARGON [ARGON] system to handle immediate network connection requests.

The VISIT (Visualization Interface Toolkit) software [VISIT] is used as a communication library between the various components. While the KoDaVis system is fully functional, it lacks the ability to make scheduled synchronous reservation requests of its resources: compute capacity on the data server, the visualization systems that run the visualization application where applicable and network bandwidth and QoS between data server and clients. Access to a collaborative visualization session is currently controlled via the simple password mechanism included in VISIT. The main objective in Phosphorus is to enhance KoDaVis with respect to resource reservation and access control. This will be achieved by adapting it to the Grid middleware UNICORE [UNICORE]. Figure 4.2 sketches the design of the planned system. Components that will have to be newly implemented or modified are drawn in red. In the current system (Figure 4.1), all parts of the distributed application are started separately and interactively, requiring a user to log into the machines where the data and collaboration server are running, and start them. After that, the visualization applications are started and the communication connections are established.

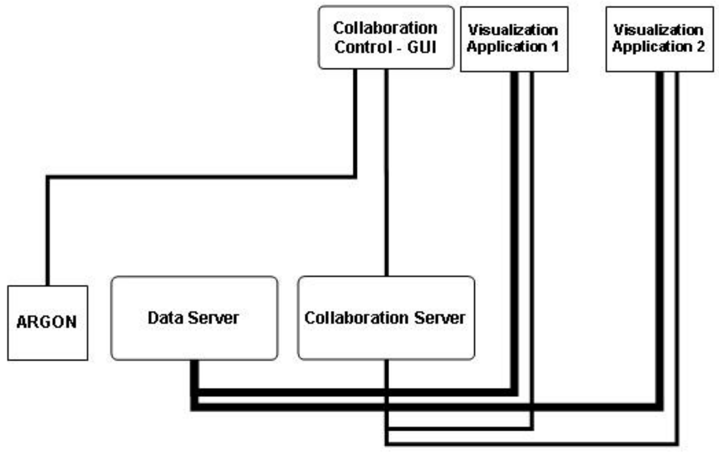
Collaboration Control - GUI
Visualization Application 1
Visualization Application 2
ARGON
Data Server
Collaboration Server

Figure 4.1 Components of the current KoDaVis version and their interaction

The major benefits of the new system (Figure 4.2) are that these steps will be fully automated, including the startup of the visualization applications, data and collaboration servers as well as the establishment of communication connections. When planning a collaborative session, a user starts his UNICORE client (named Grid Programming Environment (GPE) Client in UNICORE 6) and selects the required resources: the locations of the visualization applications and the data server. He also specifies the desired duration of the session and suitable time intervals. He does this using a UNICORE client plugin that is tailored for KoDaVis and provides exactly this functionality. One or more of the visualization systems may have reservation systems that offer interfaces for automated reservations and application startups. For such systems, an interface to UNICORE will be implemented that allows including them in the negotiation process carried out by the MetaScheduler System (MSS). When the user has completed the specifications, he submits it to the MSS via a UNICORE gateway. The MSS in turn performs the negotiations and reservation of those resources. At the reserved starting time, all reserved resources are allocated and the application is started: the reserved bandwidth between the participating systems, the data and collaboration servers, and the visualization applications, where applicable. All the required exchange of bind information (e.g. IP-addresses) is handled by the middleware as described in the chapter on middleware design changes and extensions. To summarise, the following components will have to be implemented or modified:

- A VISIT/KoDaVis plugin for the UNICORE client will be developed. This plugin will offer an easy to use interface for resource specification and control of the data and collaboration servers.

- A VISIT/KoDaVis additional target system service (ADS) for the UNICORE server will be developed. This service starts up the collaboration server and controls the exchange of status information and commands between the UNICORE clients and the data and collaboration servers.

- For visualization systems that offer interfaces for automated reservation, specific target system interfaces (TSI) will be developed, that allow the MSS to include these systems into the negotiations.

It should be noted, that for the sake of simplicity, Figure 4.2 shows only one of two options for the communication between the visualization applications and the server systems. In the Phosphorus test-bed, these connections are dedicated links, typically without firewalls between the end systems. However, the system will also support configurations, where firewalls inhibit direct socket connections. This is achieved by ssh-tunnels from the visualization applications to the system, where the data- and collaboration-servers are located. In that case, the VISIT ADS will support the establishment of the tunnel by providing the required ssh-keys. Such a system has been successfully demonstrated for collaborative steering of simulation [COVS] and will be adapted to the KoDaVis application during the course of the project.

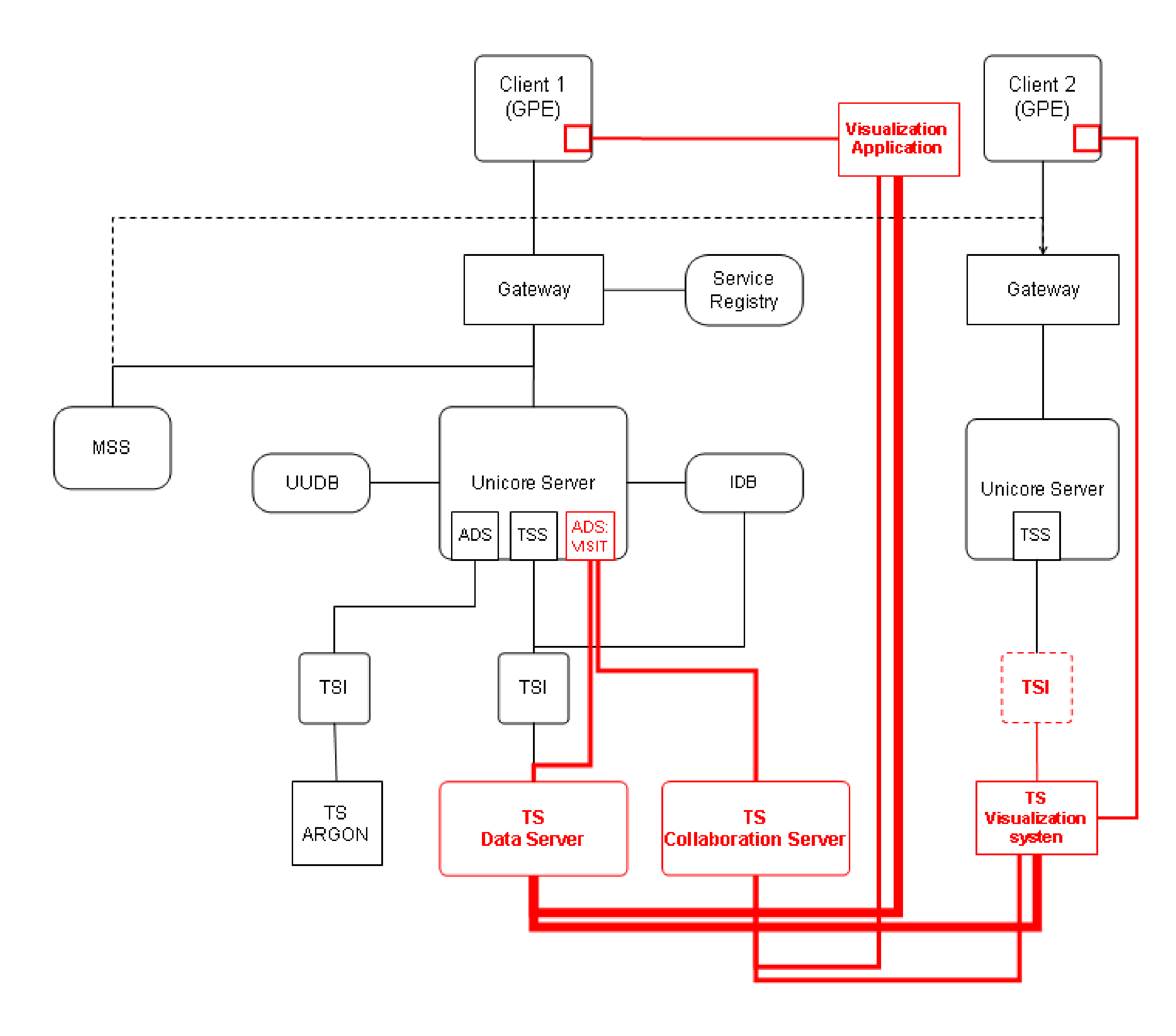

Figure 4.2 Design of the planned extensions of the KoDaVis software.

4.3 TOPS – Streaming of Ultra High Resolution Data Streams

This sections describes what changes need to be made to the existing code base of TOPS so that it may serve as a test-case for the Phosphorus framework.

4.3.1 Overcome Hardware Limitations

The current code-base of TOPS has some hard-coded constraints. These constraints (or limitations) have a historic nature, and were caused by the homogeneous nature of existing Tiled Panel Displays. Historically, target platforms for TOPS where the Tiled Panel Displays that where in service at the following locations:

- SARA Amsterdam
- UIC/EVL Chicago
- UCSD San Diego

Even though the number of the panels and the layout of the panels differ from site to site, they all used 1600x1200 pixel TFT monitors. TOPS can work with any layout of the panels, as we have tested 1x1, 2x2, 4x4, 5x3 and even 11x5 setups. The 1600x1200 per-tile resolution is hard-coded though. As more target platforms at more sites need to be serviced, TOPS should drop its 1600x1200 dependency.

4.3.2 Overcome Functional Limitations

TOPS currently generates and displays monoscopic imagery. IAIS Fraunhofer has a stereoscopic display system that they would like to deploy for TOPS use. This means that stereoscopic functionality has to be added to TOPS, both render-side and display-side.

Stereoscopic displays can broadly be classified in the categories 'active stereo' and 'passive stereo'. For both technologies, TOPS will need to be adapted in the way it generates the images render- side, and also in the way it displays them at the display-side. Passive stereo is a bit easier to do display-side, because often it is a case of simply running two instances of the same software, one for each projector. For active-stereo, things are trickier, and need a more unified approach to insure proper syncing of alternating L/R images.

4.3.3 Add Lambda Control

Raison d'etre for the entire Phosphorus project, is making the applications (or users) control the lambda's. It is of no surprise of course, that the main change in TOPS will be just this very aspect. TOPS needs to use a Phosphorus-supplied API that lets the application control these lambda's (or network resources). Instead of just assuming that the communication channel is already there, TOPS will have to set it up, using the future API. TOPS will specify, using the API, what bandwidth, latency, mtu it expects between two specified end-points.

4.3.4 Add Scheduling Dialog to User Interface

The lambdas, or the bandwidth, can be considered a scarce resource that needs scheduling. For this, TOPS must be able to schedule this bandwidth-on-demand. The user of TOPS must have a method of specifying the details of her reservation. Potentially, rendering capacity and display-facilities could be treated as a scarce resource, requiring scheduling. At this moment however, it seems premature to include this in the scope of this project.

At the start-up of the TOPS application, a dialog for scheduling should be presented to the user. Also, the status of the network (lamdas) must be presented to the TOPS user. E.g. an 'in-service' status light that indicates that the scheduled lambda's are up and running, and the TOPS visualization session can be performed.

4.4 DDSS – Distributed Data Storage System

DDSS test scenarios include two different use cases:

- GridFTP application (open source application (Apache license))
- B/A application (commercial - closed source application)

The adaptation and design changes of DDSS applications will focus on making them able to use Network Resource Provisioning System (NRPS) of PHOSPHORUS. For that purpose, the applications will make direct calls of application interface (API) of PHOSPHORUS NRPS.

To make it possible some changes have to be made in the application design. They are discussed in details in following points.

4.4.1 GridFTP application

Grid FTP is tied to Globus Toolkit middleware. However, the only resources that are considered as required for Grid FTP transmission and should be reserved in advance to actual data transmission are the network links. Therefore, Grid FTP application will use direct calls of the PHOSPHORUS NRPS API functions instead of metascheduling services.

The NRPS API calls will be put into the code of the Grid FTP client. The client code is open (in the confines of the Apache license) therefore such modification can be made with no formal limitations.

Typical GridFTP session is performed by the command-line application. This application can be run be the user, another application or Grid Broker. Therefore, some command-line options will be added to the application in order give the application user the possibility to specify some network links-related requirements.

With use of these parameters, the user or another application will be able to set up the connectivity between the end-points of planned transmission.

Modified GridFTP client will contact PHOSPHORUS NRPS (through API) and request for network links, before starting the actual transmission. After the transmission, it will contact the NRPS again in order to release the reserved links.

Additional feature of GridFTP client provided for PHOSPHORUS testbed purposes will be the possibility of scheduling the Grid FTP transmission along with needed advance reservations of network resources for a given time. This will be possible using command-line interface of Grid FTP client.

4.4.2 B/A application

B/A application that will be used in DDSS tests is not tied to any Globus middleware. Moreover, similarly to DDSS Grid FTP, the only resources that should be reserved for B/A application are the network links. We assume that appropriate resources of B/A client and B/A server (storage pools, system accounts etc.) are in place. Therefore, B/A application will use NRPS API to reserve network links between transmission end-points.

B/A client application is a proprietary backup/archive application (Copyright by IBM Corporation) with closed source codes. This makes implementing the interoperability directly between B/A client and PHOSPHORUS NRPS impossible. Therefore, the B/A client will be wrapped with code that calls the PHOSPHORUS NRPS API functions.

The wrapper will reserve the network resources needed for the B/A job before running the standard B/A transmission and release them after B/A session finishes.

The modified B/A client will add some user interface (command line) options to the original version of B/A client application. They will be used for network resources reservation i.e.: bandwidth, latency etc. between transmission end-points. The user will be able to specify needed network parameters and the wrapper will make reservations or cancels of these reservations. Appropriate messages will be generated for the user in case if the specified requirements couldn't be met. The wrapper will also support for standard B/A configuration parameters (acquired from the command line or the B/A client's configuration files). Moreover, it will be possible to put network resources-related information to the standard B/A application configuration file. If specified, they will be interpreted by the wrapper in order to setup the end-to-end connectivity for the B/A session but they will be ignored by the original B/A application.

Additional feature of B/A client application provided for PHOSPHORUS testbed purposes will be the possibility of scheduling B/A session along with needed advance reservations of network resources for a given time. This will be possible using command-line interface of wrapped B/A application.

5 Conclusions

This document has defined the use-cases of the applications that will be deployed in test test-bed and that will serve for demonstration purposes and to evaluate the achievements of the middleware developments and the other work packages. The analysis of the requirements of the applications indicates that most of the middleware extensions are related to UNICORE: the VIOLA MetaScheduling Service will be enhanced to make use of the new Phosphorus services and will be integrated into the new UNICORE 6 system. However, it is planned to integrate the MSS into Globus Toolkit 4 also, in order to broaden the applicability of the results. Another major outcome of the use-case and requirement analysis of this document is the formulation of a consolidated set of requirements of work package 3 towards the test-bed (work package 6) and towards work packages 1 and 2, which will define, design and implement the new network services. These requirements are also going to enter the deliverables of these respective work packages, in particular the test-bed design document D6.1.

6 References

[ARGON] B.Bierbaum, C.Clauss, T.Eickermann, L.Kirtchakova, A,Krechel, S.Springstubbe, O.Wäldrich, W.Ziegler, *Reliable Orchestration of Distributed MPI-Applications in a UNICORE-Based Grid with MetaMPICH and MetaScheduling,* in B.Mohr, J.Larsson Träff, J.Worringen, J.Dongarra eds. Recent Advances in Parallel Virtual Machine and Message Passing Interface: 13th European PVM/MPI User's Group Meeting, Bonn, Germany, September 17-20, 2006, Berlin, Springer, 2006. (Lecture Notes in Computer Science; 4192). pp. 174 – 183.

[COVS] M.Riedel, W.Frings, S.Dominiczak, T.Eickermann, T.Düssel, P.Gibbon, D.Mallmann, *Requirements and Design of a Collaborative Online Visualization and Steering Framework for Grid and e-Science Infrastructures*, submitted to the German e-Science 2007, Baden-Baden.

[UNICORE] A.Streit, D.Erwin, T.Lippert, D.Mallmann, R.Menday, M.Rambadt, M.Riedel, M.Romberg, B.Schuller, and P.Wieder, *UNICORE - From Project Results to Production Grids.* L. Grandinetti (Edt.), Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing 14, Elsevier, 2005, pp. 357-376.

[VISIT] T.Eickermann, W.Frings, P.Gibbon, L.Kirtchakova, D.Mallmann, A.Visser, *Steering UNICORE applications with VISIT,* Philosophical Transactions of the Royal Society of London Series A, 363 (2005), pp. 1833, 1855-1865.

OGF GRAAP-WG, http://www.ogf.org/gf/group_info/view.php?group=graap-wg

7 Acronyms

AAA	Authentication, Autorisation, Accounting
DDSS	Distributed Data Storage Systems
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)
G2MPLS	Grid-GMPLS (enhancements to GMPLS for Grid support)
GT4	Globus Toolkit Version 4 (Web-Service based)
KoDaVis	Tool for Distributed Collaborative Visualisation
MSS	MetaScheduling Service
NREN	National Research and Education Network
NRPS	Network Resource Provisioning System
PoP	Point of Presence
QoS	Quality of Service
SNMP	Simple Network Management Protocol
TOPS	Technology for Optical Pixel-Streaming
TPD	Tiled Panel Display
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 Minitry of Education and Research (Vertically Integrated Optical Testbed for Large Applications in DFN)
VPN	Virtual Private Network
WISDOM	Wide In Silicio Dockong On Malaria

Appendix A Application Use-case Questionnaire

The following questionnaire has been developed and handed out to the application providers. It aims to collect information required for the design of the middleware extensions and for the work-packages that provide services for the applications, especially WP1, WP2, WP4 and WP6.

A.1 Users perspective of the Use-case

Please describe one or more typical uses of the application from the perspective of the users: who are the typical users; what do they intend to do; how is a typical session prepared and carried out (e.g. just batch-job-submission, appointment with colleagues + advance resource reservation for collaborative session), which of the new Phosphorus services will be used and how?

A.2 Topology and general requirements

Which sites are involved?
How often and how long to you intend to test/use?
Define preferred/possible timeslots for tests (e.g. only during the night, only during the day, anytime)

A.3 Local resources at participating sites

Define hardware requirements (PC / Cluster, # CPUs, Storage amount/performance, …)
Define OS requirements (Linux, Windows, …)
Define Software requirements (commercial licenses required ?)
Are additional/separate instances of the OS required (might be more convenient for resource providers, especially, if administrative rights are required)?
Accounts required (normal user, special administrative rights)?
Permanent resource usage (daemons running) or during tests only ?

A.4 Connectivity requirements between sites

Specify bandwidth, latency constraints, other QoS parameters , between which sites?
Are special network layouts required (e.g. Point-2-Point connections, virtual LAN, routed network)?
Do you require special protocol support (e.g. multicast, anycast)?
Are there any performance requirements on service setup, teardown and recovery?
Are there MTU requirements (e.g. Jumbo frames)?

Are there special requirements for firewall configuration?
Do you require (and is it possible) to separate the resources and the test-bed network from the local network (e.g. DMZ, physical separation. This may be a requirement from some resource providers)?

A.5 Middleware requirements

Is the application tied to a special Grid-middleware?
if yes - which one?
if no – what functionality do you expect from the middleware ?
Should any actions be taken by the middleware or other lower layers to provide recovery in case of service of job failure?

A.6 AAA requirements

Are there any, beyond what's provided from the middleware, (e.g. VO management)

A.7 Interface towards a resource reservation system

Has the application an interface towards a resource reservation system (either directly or through some middleware)?
if yes – which one?
if no – what you expect from it (functionality, protocols, programming language for API, …)?

A.8 Which parameters do you have to specify in your resource requirement

(please mark with: + must have, * nice to have, - not needed

explicit specification of resource
broker to select resource according to requirements
workflow specification
local resources: (#CPUs, Software, Storage, …)
network resources (if so, which?)
time constraints (duration, time-window(s), immediate, …)
time domain: granularity and typical duration of resource usage
complex combinations (e.g. like duration * #CPUs * FLOPS(CPU))
other (which?)