

EGEE-II

GLITE IPV6 COMPLIANCE

TEST METHODOLOGY AND TESTBED ARCHITECTURE

Document identifier:	EGEE-II-SA2-TEC-810278- IPv6_test_methodology-v2.8.doc
Date:	18/12/2007
Activity:	SA2: Network Management
Document status:	DRAFT
Document link:	https://edms.cern.ch/document/810278/

Abstract:

In this document a testing methodology is proposed to check the gLite IPv6 compliance. A general approach to deal with the IPv6 compliance is sketched and different levels of IPv6 compliance are defined, according to the level of support provided by the middleware to the IPv6 protocol.

Copyright notice:

Copyright © Members of the EGEE-II Collaboration, 2006.

See www.eu-egee.org for details on the copyright holders.

EGEE-II (“Enabling Grids for E-science-II”) is a project co-funded by the European Commission as an Integrated Infrastructure Initiative within the 6th Framework Programme. EGEE-II began in April 2006 and will run for 2 years.

For more information on EGEE-II, its partners and contributors please see www.eu-egee.org

You are permitted to copy and distribute, for non-profit purposes, verbatim copies of this document containing this copyright notice. This includes the right to copy this document in whole or in part, but without modification, into other documents if you attach the following reference to the copied elements: “Copyright © Members of the EGEE-II Collaboration 2006. See www.eu-egee.org for details”.

Using this document in a way and/or for purposes not foreseen in the paragraph above, requires the prior written permission of the copyright holders.

The information contained in this document represents the views of the copyright holders as of the date such views are published.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE MEMBERS OF THE EGEE-II COLLABORATION, INCLUDING THE COPYRIGHT HOLDERS, OR THE EUROPEAN COMMISSION BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE INFORMATION CONTAINED IN THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Trademarks: EGEE and gLite are registered trademarks held by CERN on behalf of the EGEE collaboration. All rights reserved"

Document Log

Issue	Date	Comment	Author/Partner
1	29-01-2007	First draft	Jean-Paul Gautier, Mathieu Goutelle, Xavier Jeannin / UREC
2	08-02-2007 09-02-2007	Improvement following internal UREC comments	Xavier Jeannin, Jean-Paul Gautier
3	15-02-2007	Added various parts	Mario Reale / GARR
4	23-03-2007 05-05-2007	Correct definition and complete parts, spread changes and corrections	Xavier Jeannin/ Mario Reale
5	26-07-2007	Corrected and extended section 8 (ETICS)	Marian Zurek / CERN

Document Change Record

Issue	Item	Reason for Change

TABLE OF CONTENTS

1. INTRODUCTION	4
1.1. PURPOSE	4
1.2. DOCUMENT ORGANIZATION	4
1.3. APPLICATION AREA	4
1.4. REFERENCES	4
1.5. DOCUMENT AMENDMENT PROCEDURE.....	5
1.6. TERMINOLOGY.....	5
2. IPV6 Compliance	7
3. Network Level Mechanisms	7
3.1. DUAL STACK MECHANISM	8
3.2. TRANSLATION MECHANISM NAT-PT	9
4. TEST METHODOLOGY	10
4.1. SEVERAL SITUATIONS.....	10
4.1.1. Test of single external component from an IPv6 client	11
4.1.2. Test of single external component from an IPv4 client through the translation mechanism.....	11
4.1.3. Test of a distributed gLite component from an IPv6 client.....	11
4.1.4. Test of a distributed gLite component from an IPv4 client through the translation mechanism.....	11
4.2. A SINGLE METHODOLOGY	11
5. IPV6 COMPLIANCE	12
5.1. DEFINITION OF IPV6 COMPLIANCE FOR A GLITE COMPONENT	12
5.2. DEFINITION OF IPV6 COMPLIANCE FOR A GLITE PROFILE (METAPACKAGE)	13
6. Distributed TESTBED	14
6.1. UREC SITE.....	14
6.2. GARR SITE	15
7. CASES Studies	16
7.1. INFORMATION SYSTEM: BDII	16
7.2. THE WORKLOAD MANAGEMENT SYSTEM	17
8. Production Validation Service (SA3, JRA1, ETICS)	17
9. ANNEXES	19
9.1. THE GLITE CODECHECKER FROM EUCHINAGRID COLLABORATION	19
9.2. NETWORKING CONFIGURATION SCRIPTS	19
9.3. NAT-PT MECHANISM	19
9.3.1. Connection established from the NAT-PT network to IPv4 Internet	19
9.3.2. Connection established from IPv4 Internet to the NAT-PT network	20

TABLE OF TABLES

Table 1: Table of references	4
---	----------

1. INTRODUCTION

1.1. PURPOSE

IPv6 is the next generation IP protocol [R1], whose gradual introduction in the next years is expected to imply a large number of advantages in both the management and usage of the Internet.

IPv6 benefits from a set of enhancements and improvements with respect to the currently widely used version 4 of the internet protocol. It also introduces new features, like hosts auto configuration, which are completely absent in IPv4. IPv6 will first of all make available a much larger total number of Internet addresses, the basic IP address being made up of 128 bits instead of 32.

Thanks to the different structure of its header, it will also improve the routing performances and embed security and mobility features, which have currently to be added to IPv4. Describing into detail the various advantages of IPv6 is out of the scope of this document. We would only like to state here that any Grid middleware being produced for the forthcoming Grid projects should seriously take into account IPv6 compliance of its produced middleware, allowing for a gradual introduction of this new protocol all over the internet, without breaking the provided Grid functionalities.

1.2. DOCUMENT ORGANIZATION

This document describes a proposed methodology to test the gLite IPv6 compliance:

- Section 2 defines the IPv6 compliance from our point of view
- In section 3 the existing mechanisms at the network level which can help to define a test methodology are briefly described
- Section 4 describes the methodology
- In section 5, the level of IPv6 compliance are defined
- In section 6, the distributed testbed is described
- Section 7 is reserved to cases studies
- In section 8, how a service validation by other activities could be envisaged

1.3. APPLICATION AREA

This document applies to the implementation and the support of the gLite middleware. It deals with the IPv6 compliance of the gLite middleware itself. It is mainly dedicated to the developers and the service support and testing communities.

1.4. REFERENCES

Table 1: Table of references

R1	IBM redbook TCPIP 243376, TCP/IP Tutorial and Technical Overview
R2	Testbed proposal, https://edms.cern.ch/document/810285/
R3	Report on implication of IPv6 usage for EGEE Grid (EGEE I-DJRA4.3), https://edms.cern.ch/document/603955/
R4	RFC 3493 - Basic Socket interface extensions for IPv6
R5	RFC 2893 - Transition Mechanisms for IPv6 Hosts and Routers
R6	RFC 4038 - Application Aspects of IPv6 Transition

R7	IPv4-Mapped Addresses on the Wire Considered Harmful draft-itojun-v6ops-v4mapped-harmful-02.txt jun-ichiro itojun hagino
R8	Programming guidelines on transition to IPv6 T. P. de Miguel E. Castro
R9	Guidelines for IP version independence in GGF specification T Chown
R10	Survey of IPv4 in Global Grid Forum specifications R. Sofia
R11	Build, Configuration, Integration and Testing Tools for Large Software Projects: ETICS, Marc-Elian Bégin Contact Information, Guillermo Diez-Andino Sancho, Alberto Di Meglio, Enrico Ferro, Elisabetta Ronchieri, Matteo Selmi ¹ and Marian Žurek,, Lecture Notes in Computer Science , Springer Berlin / Heidelberg

1.5. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the authors. The procedures documented in the EGEE “Document Management Procedure” will be followed: <http://egee-jra2.web.cern.ch/EGEE-JRA2/Procedures/DocManagmtProcedure/DocMngmt.htm>.

1.6. TERMINOLOGY

This subsection provides the definitions of terms, acronyms, and abbreviations required to properly interpret this document. A complete project glossary is provided in the EGEE glossary <http://egee-jra2.web.cern.ch/EGEE-JRA2/Glossary/Glossary.html>.

Definitions

Grid node	A grid node is a machine that hosts a part of the middleware taking part in a Grid
gLite meta-package (a.k.a.. gLite profile or gLite deployment module)	A gLite meta-package (also known as gLite profile) is a collection of software packages (RPM or other format) installed on a machine. It provides a specific service to GRID users or to other Grid services. For instance, the gLite-CE or the gLite-BDII are examples of gLite metapackages. A gLite metapackage is often also called gLite deployment module or gLite profile. The concept of gLite meta-package is in general related to the middleware release integration process and installation.
gLite subsystem	A gLite subsystem is a developer notion. A logical partition of the system architecture providing a consistent, more specific subset of the required functionality. A subsystem is made of middleware components. For instance org.glite.wms is an example of gLite subsystem. We call therefore gLite subsystem a set of software modules belonging to a unique functional unit, normally referred to as a CVS subsystem. The gLite middleware system is made up of different subsystems. See https://edms.cern.ch/document/446241/

<p>Middleware component or generic gLite component</p>	<p>A middleware component defined as a collection of objects in electronic format (source files, binary files, configuration files, resources, documents, etc) providing together a well defined, more specific functionality within the system architecture. All JRA1 CVS modules are components of the gLite system. For instance org.glite.wms.ns-client. The middleware is composed by either internal or external components. See https://edms.cern.ch/document/446241/</p>
<p>Component of the Grid middleware</p>	<p>A component of the Grid middleware can be:</p> <ul style="list-style-type: none"> • A gLite meta-package • A gLite subsystem • A middleware component (generic gLite component) • A required external component used in the gLite middleware like an LDAP server, GridFTP etc. <p>The term “component” without any further information refers in this document to a component of the Grid middleware.</p>

Glossary

--	--

2. IPV6 Compliance

Since the beginning of EGEE, the interest for an IPv6 compliant gLite middleware has grown and not only because of the increasing collaboration EGEE built up with various countries promoting IPv6. There is indeed an incentive to provide a pervasive and sustainable middleware and as such the support for IPv6 should be fostered. A report on the implications of IPv6 usage for EGEE has been provided by JRA4 in EGEE [R3]

The gLite middleware is however a complex software stack. Producing IPv6 compliant software requires therefore a comprehensive IPv6 environment and a detailed methodology to test software in such an environment. Testing software is twofold: a developer should be able to assess the behaviour of his/her software in various IPv6 environments (pure IPv6 connectivity, IPv4/IPv6 connectivity, presence of different translation or migration mechanisms) and she/he should also be able to test if the modifications she/he made to her/his software do not change the expected behaviour (so called regression tests). It includes also partial tests, for instance when the client has been modified for IPv6 whereas the server has not yet been migrated or installed. The proposed methodology and testbed should encompass all these requirements.

IPv6 compliance is a complex concept and should therefore be clearly defined (see section 5.1). We will define different levels of IPv6 compliance, depending on the maturity degree of the tested software in various IPv6 environments. The different levels range from the installation and configuration of the software to the normal running in a mixed IPv4/IPv6 environment.

However, we will focus on two IPv6 scenarios. The first one is a pure IPv6 network; the second one is a mixed IPv4/IPv6 network, where most of the servers should run in dual stack mode (i.e. with both IPv4 and IPv6 configured on the node; see §3) and where the clients can be either IPv4 or IPv6. This transitional scenario is currently the preferred one because of its easy deployment and scalability. Software will be considered full IPv6 compliant if it is able to run with all the expected features on IPv6 like on IPv4 in a dual stack environment. Moreover the software will be considered dual stack compliant if it is able to provide all the features in IPv6 and IPv4 simultaneously, irrespective of the underlying internet protocol version.

Performance and stress tests are currently beyond the scope of this study being beyond the issue of full IPv6 compliance. Comparing the performance and quality of gLite between IPv4 and IPv6 will require qualifying and quantifying the quality of the full layered stack between these two protocols, from the network equipments to the operating systems to the network services (e.g. DNS) to the gLite components. Such exhaustive tests are as such out of the scope of EGEE.

3. Network Level Mechanisms

The Grid middleware is a distributed software, the middleware of a Grid is scattered on different servers that provide a service. A Grid node deploys a gLite metapackage like the glite-WMS, BDII, the gLite-CE, etc. A Grid node must provide its service either for IPv4 or IPv6 clients. The dual stack technology [R5] which is a well-proven transition mechanism will be used to provide an IPv4 service and an IPv6 service on a single host.

Due to the relevant amount of work required to port the middleware to IPv6 and the geographical spread of the different development communities, the gLite middleware components will not be IPv6 compliant all together at the same time. As a consequence, a component of the Grid middleware will have to be tested independently of the IP environment under which the other components will be running. A dual stack machine using IPv4 mapped IPv6 address [R6] can allow an IPv6 server to handle IPv4 request. Nevertheless, this feature could not be not available on some OS or the implementation of the code server does not use this feature therefore you need a mechanism making possible communication between an IPv6 node and an IPv4. Such mechanism is called a *translation mechanism*; among them NAT-PT is probably the most popular one.

Different interactions on the network will be solved during the migration phase:

- An IPv6 server of an external component used with the fitted IPv6 client → Dual Stack mechanism.
- An IPv6 server of an external component used in gLite from the fitted IPv4 client → Dual Stack mechanism if the server uses IPv4 mapped IPv6 address.
- An IPv6 server of an external component used in gLite from the fitted IPv4 client → Translation mechanism (NAT-PT) if the server does not use IPv4 mapped IPv6 address.
- An IPv6 comprehensive Grid node with an IPv6 cooperative Grid node → Dual Stack mechanism.
- An IPv6 comprehensive Grid node with an IPv4 cooperative node → Dual Stack mechanism if the gLite meta-package installed on the grid node uses IPv4 mapped IPv6 address.
- An IPv6 comprehensive Grid node with an IPv4 cooperative node → Translation mechanism (NAT-PT) if the gLite meta-package installed on the grid node does not use IPv4 mapped IPv6 address.

3.1. DUAL STACK MECHANISM

This technique provides complete support for both Internet protocols IPv4 and IPv6 in hosts and routers by integrating IPv6 itself. Using this method, a host or a router is configured with both IPv4 and IPv6 protocol stacks in the operating system (Figure 1). Those dual stack hosts defined in RFC2893 [R5], need applications, TCP/IP modules and addresses for both IPv4 and IPv6.

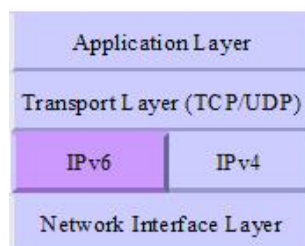


Figure 1: A dual IP layer architecture

For a host, an application communicates with IPv4 nodes through the IPv4 stack, and with IPv6 nodes through the IPv6 stack (Figure 2). The selection of the stack can be deduced from the IP destination address (IPv4 or IPv6) given, for example, by the user or as the result of a DNS resolution.

It is important to make Grid systems work with both IPv4 and IPv6, and to be able to communicate in heterogeneous IPv4/IPv6 networks. In this case, the IP version-independent server has to be able to respond to client calls according to the IP version that the client uses. So, the client decides which version of IP has to be used. For instance, an IP version-independent Grid server on a dual-stack machine starts and listens on both its IPv4 and IPv6 interfaces.

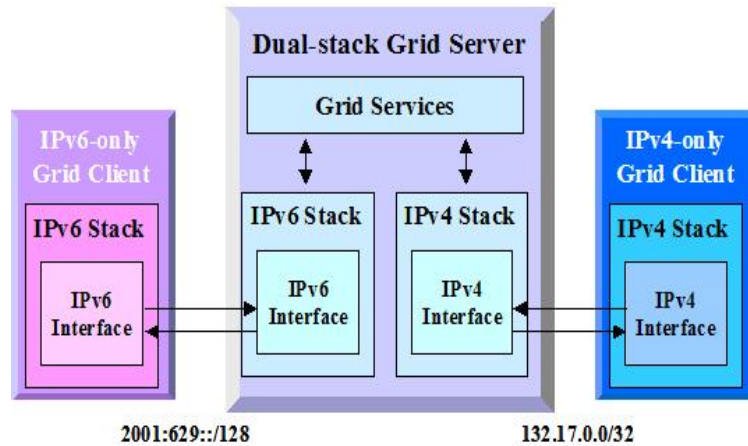


Figure 2. IP communication of Client/Server in simple heterogeneous IPv4/IPv6 networks

3.2. TRANSLATION MECHANISM NAT-PT

The general approach to the problem is to set up a component of gLite middleware on a Grid node first on IPv4 and after in a second step try to run this service on IPv6. A component of the Grid middleware can be a complex set of programs that needs to interact with other Grid nodes. In order to make the different tests by component, a translation mechanism between IPv4 network and IPv6 network like NAT-PT or dual stack with IPv4 mapped IPv6 address, will be used. Using this method a node running on IPv6 can communicate with other part of the IPv4 Grid.

A dual stack machine can redirect IPv4 request for a server to IPv6 stack using IPv4 mapped IPv6 address [R6]. For example 192.0.2.128 is mapped in ::FF:C000:280. This mechanism allows a server to open a single listening socket to handle connections from both IPv6 and IPv4. Not all stacks support this mechanism, and for security reasons [R7] this feature can be disabled by default. Moreover like translation mechanism, any IPv4 address in the payload is not translated. On the other hand, server application can be written to open two listening sockets to handle separately IPv4 and IPv6 request [R8].

Dual stack with IPv4 mapped IPv6 address can not handle all situations, therefore we have to resort to an external translation mechanism NAT-PT

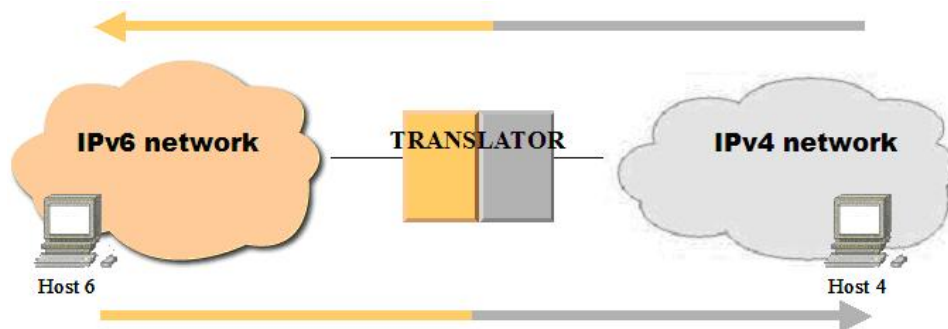


Figure 3. Translation between IPv6 network and IPv4 network

The limitation of the NAT-PT mechanism to translate certain protocols like FTP (RFC2663 and 2766) can be overcome by using an application layer gateway (ALG). Translation mechanism has to be used for testing only: the scalability of the NAT-PT is not proved and could raise issues in production.

4. TEST METHODOLOGY

As it has been mentioned before, the idea behind the methodology is first to test the gLite middleware component by component. A component of the grid middleware can be:

- A gLite Metapackage
- A gLite Subsystem
- A middleware component
- An external component used in gLite middleware like an LDAP server, GridFTP, etc.

Developers need a testbed to migrate their code to IPv6 or to test in real condition a code they just have enabled for IPv6. The methodology aims to test in general the feature level provided by a program, i.e. the general behaviour of the program and not specific aspects of the code. The testbed will be used to mainly test the interaction between a component under testing and the rest of the Grid. However, performance test is beyond the scope of this document.

The general approach is to install the component on IPv4, to configure it on IPv4 in order to work properly with the rest of the testbed Grid and then migrate the machine to IPv6 in the NAT-PT network. The IPv4 network interface is then disabled; one tests if the service delivered by the component is still able to start on an IPv6 only network and works properly with the rest of the IPv4 Grid through NAT-PT or directly according to the code specification. At this step, developers should patch the code and test it until the IPv6 compliance is achieved.

Note that this test procedure can be applied to both to structured components like the BDII server and simpler components like the LDAP server, GridFTP, etc.

4.1. SEVERAL SITUATIONS

Since the whole process of porting the entire gLite middleware to IPv6 will necessarily take some time to the developer's community, it is more than likely that we will be at some stage faced to a situation in which some components will be already IPv6 compliant and some others not. This means that there will definitely be a phase in which it will make sense to test the IPv6 compliance only for some gLite components, while others will have to stay deployed at IPv4. To fully test the IPv6 compliance of a component, we will have to test the interplay with essentially all other related ones. Therefore we will need to be able to test a single component under IPv6 in its interplay with other components under IPv4. The same consideration will apply for the external dependencies, whose IPv6 compliance will not necessarily be provided at the same time, but rather, most likely, with a gradual introduction of a restrict number of IPv6 compliant external dependencies.

Moreover, not only different components will be provided with IPv6 compliance at different times, but even within a single gLite software component, the server part will eventually be available (IPv6 compliant) before the client part or vice versa. This is something likely to happen.

Therefore we'll have to face for a while a double-folded, hybrid IPv6/IPv4 operational scenario: within different gLite components and — in a single component — within its client and server parts.

To cope with this hybrid, provisional scenario, which will eventually last for quite some time, and prevent an immediate, direct test of the full gLite under IPv6 (which is — of course — an asymptotic test case, once full IPv6 compliance will be officially provided to a finally IPv6 blessed gLite code), we have envisaged a set of different test cases (here below) to be a reference for the IPv6 compliance test activities.

A final remark probably useful to the section above, is that having to deal with such a widespread operational scenario, will not only be useful for the transitional phase in which IPv6 compliance will be progressively delivered to the middleware, but also in the real world where we will have to live in the next years. To be more precise, as already mentioned in section 3.1, what we (taking for a moment the

side of the community of Grid users, sys-admins and deployers) are aiming for is a gLite middleware (and a Grid middleware in general, including of course also the required external dependencies) made independent of the particular underlying network layer which sits underneath. This is our definition of a perfectly operational dual stack world. We will have to live in an hybrid IPv4/IPv6 world for probably quite a while (probably many years), where we will see IPv4 and IPv6 nodes having to interoperate next to each other. This will mean that different gLite components will probably have to interplay using different IP protocols and that IPv6 clients will have to query IPv4 servers or vice versa.

If Grid middleware developers will stick to the recommended guidelines (RFC 3493) [R4] for using more abstract interfaces to the underlying network stack, we will at some point have available a middleware which will allow a gLite metapackage on a node to communicate with another gLite metapackage on another node using IPv4 or IPv6, according to what is the support protocol by the gLite metapackage/node to be reached.

This is much more than simply making sure that everything works if everything is running under IPv6. It's a major revolution in the way we will have to write the middleware, configure it, and get it running.

4.1.1. Test of single external component from an IPv6 client

If an IPv6 client of a given component (like an LDAP server, or a gridFTP server) is available, one can test the server directly using its corresponding IPv6 client, by placing both under IPv6 and testing the client/server interaction, making sure the provided functionality under IPv6 is exactly what has to be there, i.e. the same one with respect to IPv4.

4.1.2. Test of single external component from an IPv4 client through the translation mechanism

Once the IPv6 version of an external component server is available, one can test it from an IPv4 client using a protocol translation mechanism, to make sure it works as expected.

4.1.3. Test of a distributed gLite component from an IPv6 client

If an IPv6 client has been made available by the developers together with the server part, one can verify that the observed behaviour is exactly what it would be expected using IPv4.

4.1.4. Test of a distributed gLite component from an IPv4 client through the translation mechanism

If an IPv6 client for a gLite component is not yet available, if dual stack IPv4 mapped address IPv6 is not possible, one can use NAT-PT to test it from an IPv4 client.

4.2. A SINGLE METHODOLOGY

All scenarios described in the previous section can be handled in one single approach:

1. Developers should first use a code checker to assess the IPV6 compliance of the code. In particular we refer to the EuChinaGRID IPv6 code checker, and to the corresponding ETICS plug in defining the "IPv6" metrics in the gLite build system quality checking.
2. The operating system is installed on the machine that will hold the component of the grid middleware. The component of the Grid middleware (for instance BDII server, LDAP server) is installed on IPv4.
3. The component of the Grid middleware is configured on IPv4 and tested to ensure it works properly on IPv4. The interaction with the other parts of the testbed Grid is checked, to make sure all services are properly set up and responding as expected.

4. The network interface of the machine is configured to support IPv6; the tester checks if the service delivered by the component is still behaving correctly on IPv4.
5. The component of the Grid middleware is configured and included in the IPv6 network behind the NAT-PT router. The IPv4 network interface is disabled. The tester checks if the service can be started properly.
6. The tester checks if the service delivered by the component works on IPv6 and if the tested component can work properly through NAT-PT or directly with IPv4 mapped IPv6 address with the rest of IPv4 Grid. The developers should patch the component code, if necessary. This step could to be made several times until the service works properly on IPv6.
7. If an IPv6 client is already available, developers should check if the service works on IPv6 using the corresponding IPv6 client.
8. Finally, the machine is shifted in a dual stack IPv6 and IPv4 network, effectively configured as a dual stack machine. Developers could need to patch their code in order to allow the component to provide the service either for IPv6 or IPv4. For this stage be successful, the gLite code must be IP version independent, i.e. fully independent of the underlying network layer. This means, by definition, that the code must be able the run over IPv4 only machine, IPv6 only machine, or a dual stack machine.

5. IPV6 COMPLIANCE

We call *gLite subsystem* a set of software modules belonging to a unique functional unit, normally referred to as a CVS subsystem. The gLite middleware system is made up of different subsystems. For example the Workload Management System (WMS) of gLite is a unique functional unit, represented in the gLite CVS terms by the org.glite.wms subsystem, containing different WMS components.

We call a *generic gLite component* a software module belonging to a given subsystem, normally referred to as a CVS component in middleware development and integration terms. For example org.glite.wms.ui is a gLite component hosted in the gLite CVS server. This corresponds normally to a specific functional entity in the gLite architecture.

We call a *gLite profile*, or a *gLite deployment module*, a service normally running on a node (or a set of nodes), normally as a unique functional block within the Grid architecture.

For example a gLite module (or profile) is the gLite WMS (Workload Management System) server, whose profile name is glite-WMS. This normally corresponds to the name of an RPM meta-package, containing RPMs corresponding to the build of various components and sub-components. In this section we provide a definition for IPv6 compliance for both: gLite components and gLite modules (profiles).

5.1. DEFINITION OF IPV6 COMPLIANCE FOR A GLITE COMPONENT

We say that a gLite component is *fully IPv6 compliant* if:

- a) In all its sub-component modules (all of its lines of code) there are no non-IPv6 compliant calls, i.e. there are no non-RFC 3493 compliant calls, (irrespective of the particular language in which the socket function and address data structures have used and the component implemented) and there's no hard-coding of IP addresses by number, not in the software nor in its configuration.
- b) The provided functionality and operation of the component in all its parts (server, client, API, etc...) is exactly the same with respect to the one provided if the component is deployed in IPv4 and if this being the case, the original IPv4 functionality is not broken nor affected in any way by the IPv6 compliance.

This is somewhat a conservative definition of IPv6 compliance, given that if condition a) is satisfied, than condition b) should also be satisfied. The two parts of this definition are however very different from the verification operational point of view, since the first one is more related to the execution of the proper set of code checking tools on the software, whereas the second one relates to verifying the functionality and operation.

Also, this definition of IPv6 compliance for a component should guarantee the correct operation of the system in Dual Stack mode. We say that a gLite subsystem is *fully IPv6 compliant* if all components belonging to it are.

5.2. DEFINITION OF IPV6 COMPLIANCE FOR A GLITE PROFILE (METAPACKAGE)

We define a gLite profile to be *fully IPv6 compliant* if its installation, configuration mode, functionality, operation and management using IPv6 is indistinguishable from the one it shows using IPv4, and this being the case, its original IPv4 based mode of operation is not affected in no way.

Under the general assumption that the IPv4 mode of operation is not affected at all, we define different *levels of IPv6 compliance* for the gLite profiles (also known as gLite deployment modules or gLite metapackages), in order to be able to quickly map the whole gLite middleware with respect to its IPv6 compliance:

A gLite profile (for example "glite-WMS" or "glite-UI") which installs fine using IPv6 has an IPv6 Compliance Level of 1 [**IPv6 CL = 1**]. This means that the whole installation procedure has been successful using IPv6, no conflicts are present in the RPM composition of the node and the middleware will be successfully placed on the node using the recommended operating system for gLite.

A gLite profile which configures successfully using IPv6 has an IPv6 Compliance Level of 2 [**IPv6 CL = 2**]. Configuring the nodes using the gLite XML & Python scripts (set [hostname]:port#, VOnames, etc. etc... in the XML files and then run the `/opt/glite/etc/config/scripts/<glite-machine-profilename> --configure` script has therefore to be successful).

A profile whose daemons all start fine under IPv6 has an IPv6 Compliance Level of 3 [**IPv6 CL = 3**]. This means that the start up of all services is successful and all daemons run OK. (Running the `/opt/glite/etc/config/scripts/<glite-machine-profilename> --start` script is therefore fully successful)

A profile for which the basic provided functionality works fine has an IPv6 Compliance Level of 4 [**IPv6 CL = 4**]. Invoking services from their corresponding Grid clients to access the basic provided functionality has to be fully successful. (In most cases invoke therefore the provided functionality from the CLI-based gLite UI, in some cases from the advertised web interfaces).

A profile for which all advanced functionality works fine has an IPv6 Compliance Level of 5 [**IPv6 CL = 5**]. This is the case if one can successfully exploit all advanced provided functionality by the system. This means successfully execute all possible commands and operations on the component (for example DAGs, long lasting jobs or job collections for the WMS system, large and structured R-GMA queries, full slots of FTS transfer jobs...).

A profile for which both the IPv4 server/IPv6 client and vice-versa (IPv6 server/IPv4 client) have been successfully verified using a protocol translation or a dedicated tunnelling has an IPv6 compliance level of 6 [**IPv6 CL = 6**]. This case reflects the final goal and operational scenario for the gLite middleware in an hybrid world (IPv4 only nodes, IPv6 only nodes, Dual Stack nodes)

Finally, a profile for which all this extended functionality in a multiprotocol environment over the WAN has been successfully verified has an IPv6 compliance level of 7 [**IPv6 CL = 7**]. This implies configuring dual stack on a multi domain WAN and successfully test every provided functionalities of a component in a hybrid, distributed IPv4 & IPv6 Grid environment. This includes the set up of many server and clients over the internet, using dual stack, IPv6 only and IPv4 only protocols.

6. Distributed TESTBED

The EGEE SA2 hybrid, distributed testbed consists of four parts, corresponding to the different role they play in the test methodology defined above.

1. A network of computers (to the right in Figure 4 here below) which contains all the necessary components of the Grid middleware to fully test a given, specific component. Generally, this part of the testbed contains all the services of an operational Grid and these services must be operational on IPv4. This part is therefore similar to a more standard testbed (like the EGEE SA2 certification testbed or the JRA-1 preview testbed), despite the fact that it may use a specific networking set up according to the needs (generally IPv4, but also IPv6 or dual stack if required),
2. At least one router that enables a NAT-PT mechanism.
3. A DNS service that answers for IPv4 or IPv6 address requests and supports both IPv4 and IPv6 protocols.
4. The component testbed (to the right in Figure 4 here below)
 - a. The component testbed contains an IPv6 network where is located a machine that holds the tested component. The IPv6 network can eventually contain other IPv6 clients. This IPv6 network is connected to the NAT-PT router.
 - b. The component tested can hold an IPv4 network in order to test dual-stack behaviour.

For the EGEE SA2 testbed, the sites contributing to the testbed are the CNRS UREC site in Paris and the GARR testbed site in Rome.

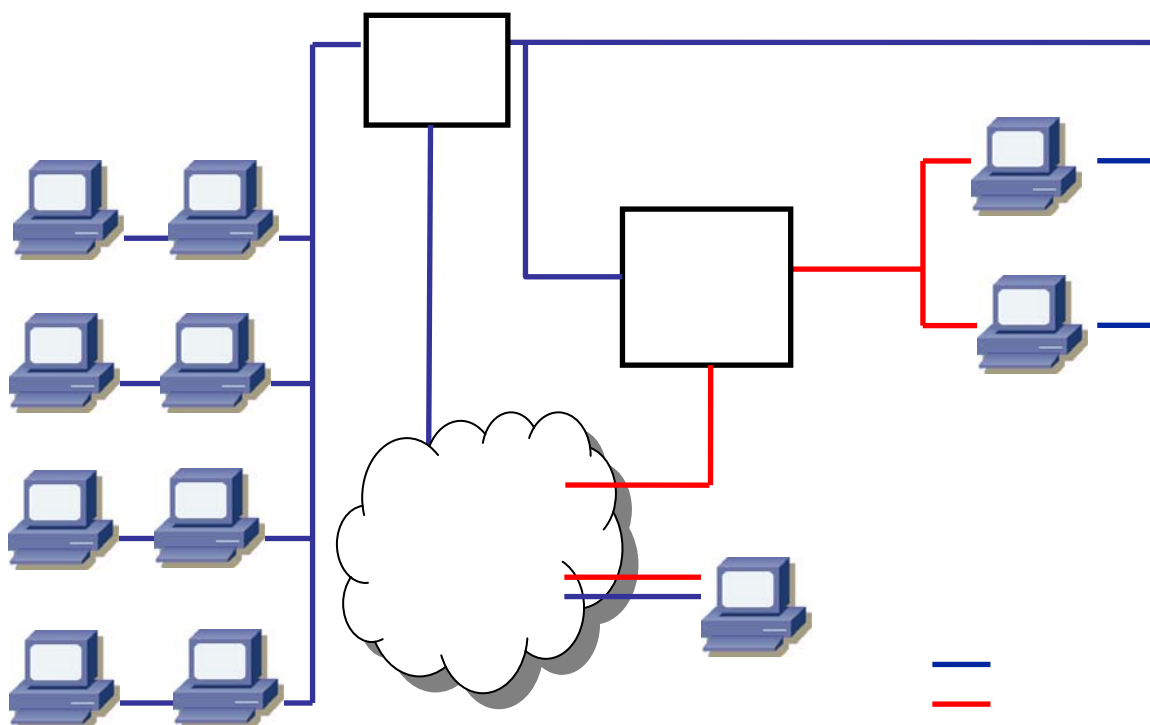


Figure 4: EGEE SA2 Testbed on February 2007

6.1. UREC Site

The UREC testbed is composed by a router with NAT-PT enabled and a dedicated IPv6 network. Two nodes running Scientific Linux 3.0.7 and deploying gLite 3.0.6 are included in this testbed network.

All the nodes can be dual stack according to the different tests that are run. A top level BD-II Server has been deployed and ported to IPv6 and are available.

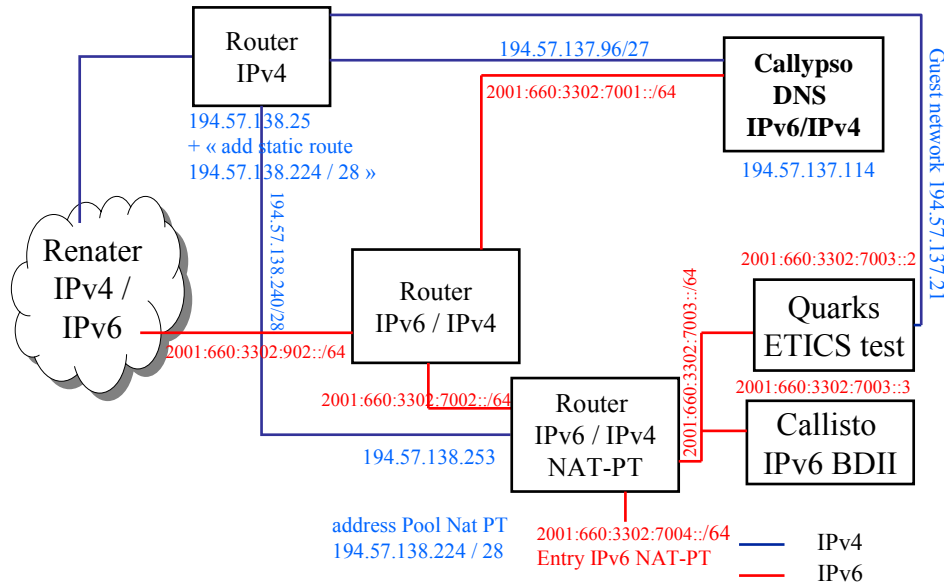


Figure 5: UREC/CNRS testbed on February 2007

6.2. GARR Site

The GARR testbed is represented by a dual stack DNS server and the following set of nodes (running Scientific Linux 3.0.8 and deploying gLite 3.0.6 + updates), all of which have both IPv4 and IPv6 addresses, and can be used in IPv4 only, IPv6 only or Dual Stack Mode:

- WMS Server
- Logging and Bookkeeping server
- RGMA Server
- BD-II Server
- Computing Element
- 2 Worker Nodes using Torque/Maui as a local resources management system queue
- 1 classical storage element
- 1 VOMS server

The supported VOs by the VOMS server are the following ones: egee, egttest, euchina, garr.

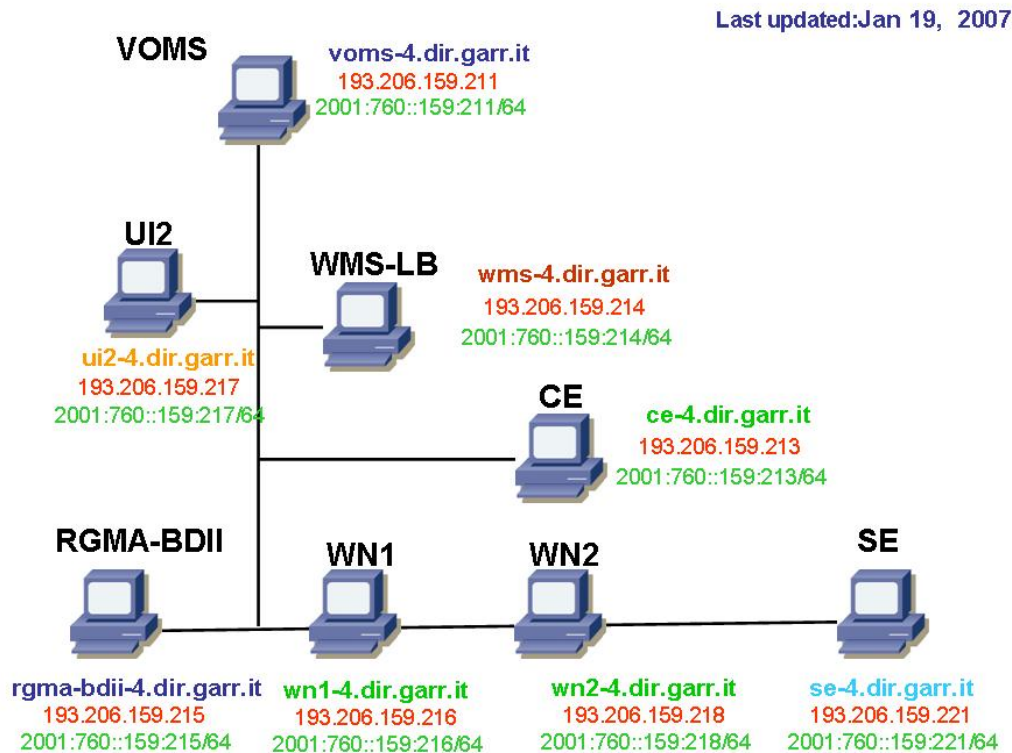


Figure 6: GARR testbed Site on February 2007

7. CASES Studies

We briefly report here the sequence of operations performed during the execution of two major case studies by means of which the test methodology has been refined and validated. The first refers to the sequence of operations performed to test the gLite BD-II metapackage and the second one refers to the gLite WMS system with the all included corresponding metapackages related to the execution of the jobs after user authentication and authorization.

A major difference between the two cases is that in the first case the full cycle of the defined test methodology leading to the verification of the gLite IPv6 compliance has been exploited, including a set of modification applied to the original code of the BDII server. At the end of the executed operations, a successful exploitation of the BDII functionally under IPv6 has been effectively, actually achieved. In the second case, on the contrary, only the first steps of the methodology have been applied, since after having set up and configured the services under IPv4, the switch to IPv6 (and the disabling of IPv4) has essentially broken all provided functionality and proper running of the required daemons and processes on the nodes. The second case is therefore the verification of a generalized failure under IPv6 for the WMS gLite middleware, and didn't imply anyhow any change in the original code, mostly since the amount of work to properly make the whole WMS IPv6 compliant was out of the scope and the knowledge of the testers involved in this preliminary task.

7.1. INFORMATION SYSTEM: BDII

1. Installation of Scientific Linux on IPv4 and installation of the BDII server on IPv4
2. Configuration of the BDII server on IPv4 and test if it works properly
3. The code checker is passed on the code to find IP dependencies
4. The server BDII is configured on IPv6 only and can be access behind the NAT-PT router

5. The BDII does not work properly
6. The code BDII server is patched
7. The BDII server collects information on operational EGEE IPv4 grid through NAT-PT. The BDII server can be accessed either on IPv4 through NAT-PT or directly on IPv6 to retrieve information.
8. One can retrieve information with the following commands.
 - With an IPv4 access through NAT-PT:

```
ldapsearch -x -P2 -h callisto-natpt.paris.urec.cnrs.fr -p 2170 -b "mds-vo-name=local, o=grid"
```
 - With an IPv6 access:

```
ldapsearch -x -P2 -h callisto.paris.urec.cnrs.fr -p 2170 -b "mds-vo-name=local, o=grid" OR ldapsearch -x -P2 -h 2001:660:3302:7003::3 -p 2170 -b "mds-vo-name=local, o=grid"
```
9. The code is installed on a dual stack machine quarks.paris.urec.cnrs.fr, the BDII server collects information on operational EGEE IPv4 through its IPv4 interface. One can retrieve information with the following command either with IPv4 client or IPv6 client:
 - With an IPv4 access

```
ldapsearch -x -P2 -h 194.57.137.100 -p 2170 -b "mds-vo-name=local, o=grid"
```
 - With an IPv6 access

```
ldapsearch -x -P2 -h 2001:660:3302:7003::2 -p 2170 -b "mds-vo-name=local, o=grid"
```

7.2. The Workload Management System

1. Installed a full WMS system (WMS, CE, WN, UI) and a VOMS server under IPv4
2. Configured it under IPv4.
3. Started all services and Verified the system successfully
4. Switched on IPv6 in Dual stack
5. Switched off IPv4
6. Restarted the services, after re-execution of the configuration scripts in some cases (where obvious configuration parameters had to be changed, like IP addresses, for instance)
7. Tested the provided functionality:
 - Most daemons have problems in the restart phase for different kind of failures (some possibly related to external components like Condor).
 - Every available functionality for the user, including authorization, have been completely broken (no voms-proxy-init possible, no glite-job-submit nor glite-job-status possible).

8. Production Validation Service (SA3, JRA1, ETICS)

In the framework of this activity carried out by EGEE SA2 aimed at defining and prototyping a well defined testing methodology for the gLite IPv6 compliance, reported by this document, a collaboration with the ETICS team has been established. A detailed description of the ETICS project is out of the scope of this section, and can be find in [R11]. The goal of this collaboration is to prototype a comprehensive scenario within ETICS for the automatic testing of the gLite IPv6 compliance. ETICS is focused on the automatic build, set up, configuration and testing of software in general, and the gLite middleware in particular, in the case we are considering.

The idea behind the establishment of this collaboration is to use ETICS, already the most natural place for developers for what concerns integration of their code within gLite, for the basic testing of the developed middleware subsystems and components, also for the tests related to the gLite IPv6 compliance. ETICS is already foreseeing the progressive inclusion of the automatic deployment of the various gLite metapackages as part of the automatic, daily procedures (build, integration process producing the gLite releases) carried out within the global process of the gLite integration process.

Therefore we aim at providing through ETICS at least a prototypal testbed for the test of of the gLite IPv6 compliance. This testbed will include gLite services running under IPv4 or IPv6 or dual stack, and all the required transition or tunnelling mechanisms to carry out a sensible test campaign inspired by the methodology we have defined in this document.

In particular, an ETICS test project devoted to the study of gLite IPv6 compliance has been created, named “gLite IPv6 compliance”¹ and a corresponding folder entry in the gLite CVS repository has been created². A test component called `org.glite.testsuites.ipv6` has been created and test methods for it have been defined. A demonstrative ETICS test job has been successfully querying the IPv6 top level BDII server based at UREC in Paris from an IPv4 client (an ETICS NMI node at CERN, SLC4) using the NAT-PT protocol translation at UREC. This is somewhat the proof of concept of providing an IPv6 testbed to the JRA-1 developers by means of ETICS and an IPv6, IPv4 and dual stack distributed testbed.

Together with the ETICS Team, two possible operational scenarios have been envisaged corresponding to different level of automation and delegation to the ETICS “machinery” for what concerns the deployment and configuration of the gLite middleware.

In one case, one can point to an already existing IPv6/IPv4 capable testbed, from an ETICS job on a Condor NMI worker node belonging to the ETICS pool. A test job will be launched from the NMI pool, exploiting the functionality under test on remote testbed servers. This scenario is already available and has been already partially exploited by the previously mentioned ETICS job query the top level BD-II in Paris.

In a second scenario, the delegation to ETICS is much higher, and ETICS will deploy all required services to perform a specific test and only once these have been successfully deployed (installed and configured), it will launch the test methods associated to the components to be tested. This scenario requires an improvement of the internal ETICS information system, which will need to allow inter job effective communication. More precisely, different ETICS NMI jobs will deploy all required services for a specific test to be executed, on different nodes, whose identity and address is initially unknown to the test job. There must therefore be a way of telling the final test job about the result of the related, previous jobs and the locations where all required services for the test have been installed.

Starting from the automated components/services deployment (installation and configuration handled by ETICS infrastructure) the compliance/interoperation test /test methods will be launched. The test reports will be published/made available at some specific web locaton. This scenario will make use of the co-scheduling (aka parallel test) mechanism developed for the advanced system tests. Among the basic functionality the ETICS will provide the mechanism for the information exchange and synchronisation between installed services (assuring the proper order of installation, configuration and information exchange - like hostnames, services end-points, etc.). For this purpose the job identifiers and some relevant environmental variable have to be made available by the ETICS information system to the final test job to be executed. This functionally is already undergoing tests at CERN and should be made publicly (initially to gLite community) available throughout the month of July/August.

¹ <https://etics.cern.ch:8443/etics/>

² <http://glite.cvs.cern.ch:8180/cgi-bin/glite.cgi/org.glite.testsuites.ipv6/>

A possible snapshot of this scenario is represented by the following workflow:

1. There is an ETICS server at CERN.
2. Users submit test jobs by means of the command line interface or the ETICS web application
3. Tests jobs will be flagged as requiring IPv6 connectivity, when required.
4. The server recognizes this requirement and sends these jobs to the NMI nodes where IPv6 is available (currently at GARR or UREC)
5. The test job lands on these IPv6 enabled nodes
6. Job declares it needs to set up a set of services (for example a UI, WMS, CE, WN, VOMS)
7. These services are set up (installed and configured) by a set of NMI jobs, which will then publish somewhere in the ETICS information system the location and addresses of the services they set up.
8. This information is accessed by the test job, test is performed and results are published

9. ANNEXES

9.1. THE GLITE CODECHECKER FROM EUCHINAGRID COLLABORATION

The EUChinaGrid project collaboration has developed a code checker tool aimed at spotting out non IPv6 compliant calls in the gLite middleware code, by looking for specific uncompliant patterns. Both the code checker and its documentation can be found at the following URL: http://www.euchinagrid.org/IPv6/cod_checker.html. It has been run on 52 gLite WMS CVS modules, and out this number, 16 failed, clearly showing non IPv6 compliance in the code itself.

Within the context of the EGEE, EuChinaGrid and ETICS collaboration, it has been ported to the ETICS build system as part of the quality metrics constantly monitored by it. A new metric called "IPv6" is therefore computed on the middleware components (gLite components) in the nightly gLite builds performed by ETICS, reporting on the degree of IPv6 compliance of the various gLite modules.

9.2. NETWORKING CONFIGURATION SCRIPTS

9.3. NAT-PT MECHANISM

NAT-PT stands for Network Address Translation and Protocol Translation. This mechanism translates the address and the protocol to allow IPv6 machine to communicate with IPv4 machine. It is available on different kind of routers.

NAT-PT establishes a translation table between IPv6 address and IPv4 address. To connect to a machine of IPv4 world from the machine behind the router NAT-PT on IPv6, in first step the DNS resolution should handle by the NAT-PT router, it is the purpose of DNS ALG. Be aware that in the described testbed the DNS is not behind the NAT-PT.

9.3.1. Connection established from the NAT-PT network to IPv4 Internet

Here are the stages of a connection, for instance the IPv6 machine (Quarks) want to connect `egee-sa2.web.cern.ch`.

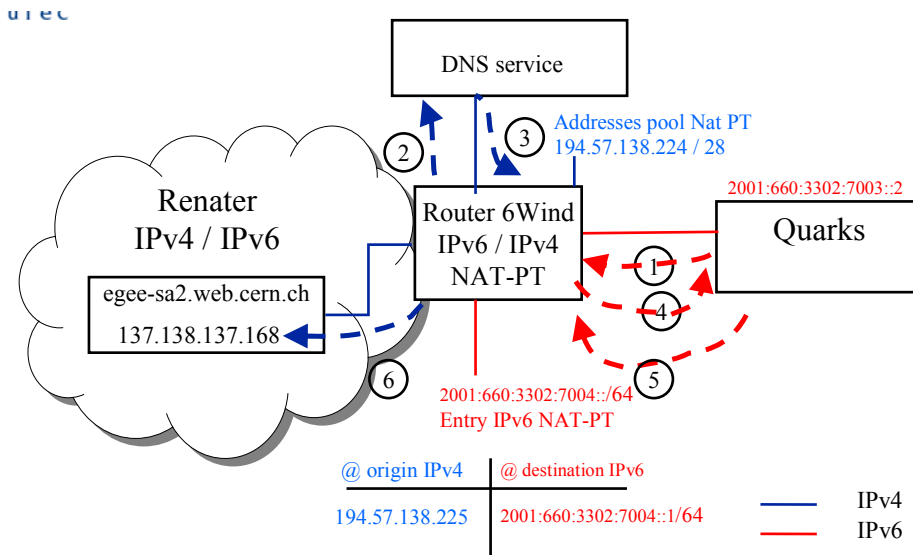


Figure 7: Connection through NAT-PT to IPv4

1. Quarks makes an IPv6 DNS (AAAA) resolution of egee-sa2.web.cern.ch.
2. The router recognizes that the request comes from a machine from the NAT-PT network. The router make an IPv4 (A) DNS request and an IPv6 (AAAA) DNS request.
3. If the DNS responds to the two requests, the router returns the IPv6 address. Otherwise, the router build an IPv6 address by appending to the NAT-PT prefix (2001:660:3302:7004::) the IPv4 address (137.138.137.168 = 898a:89a8) to obtain 2001:660:3302:7004::898a:89a8. A newline in the translation table of the router is added.
4. The router answers to Quarks the IPv6 address according to previous step.
5. The communication can be established now.

9.3.2. Connection established from IPv4 Internet to the NAT-PT network

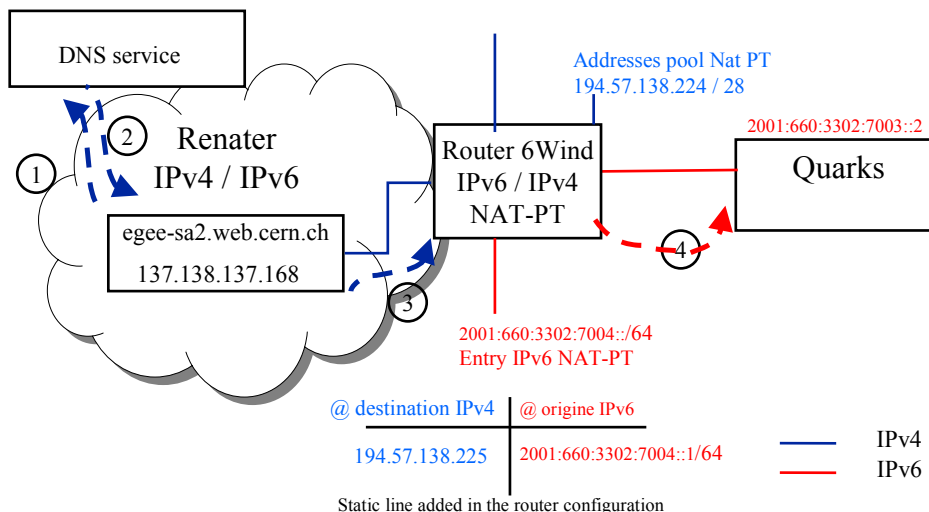


Figure 8: Connection from IPv4 through NAT-PT

Now an IPv4 machine egee-sa2.web.cern.ch wants to connect to Quarks an IPv6 machine. It is necessary then to add a static line in the table translation of the router (modify the configuration router by the hand) to allow egee-sa2.web.cern.ch to connect to an IPv4 address of NAT-PT pool. Then, the

router translates the packet and forwards it to Quarks faking this packet is coming from an address of IPv6 NAT-PT pool. The connection is established.