# IPv6 and gLite: a roadmap proposal

**Xavier Jeannin (CNRS UREC – EGEE-II SA2), on behalf of the IPv6 group**

***JRA1 All Hands Meeting – 2007-10-24, CERN***

**www.eu-egee.org**

Information Society and Media

# Outline

- IPv6 why now?
- Status of gLite
- IPv4/IPv6 interoperability
- Roadmap to IPv6
- The example of BDII: 2 different versions of dual stack server BDII
- Testbed and test methodology
- Conclusions
- Discussion

# Why IPv6 now?

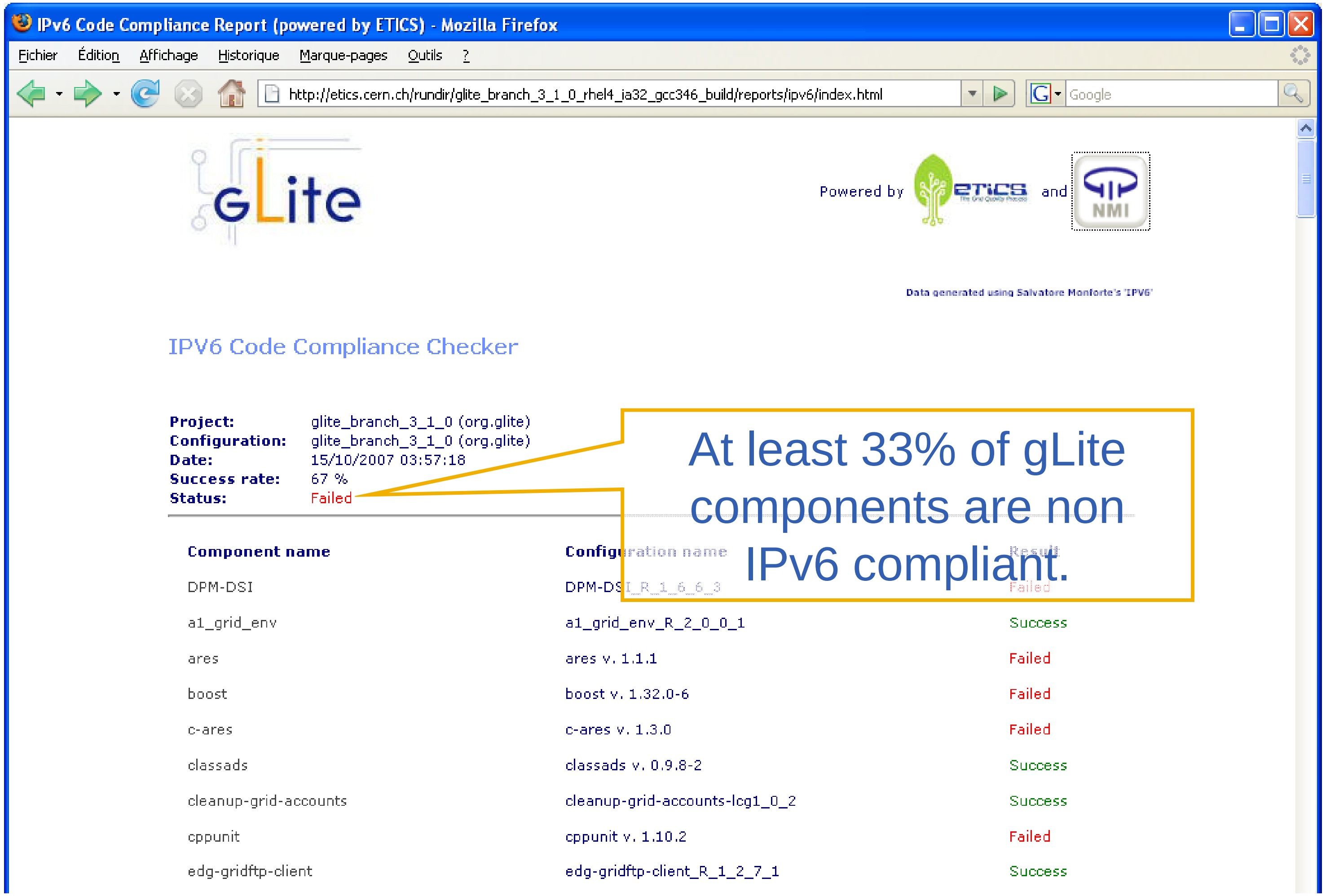
- **The need for a “new IP” has been identified since many years:**
  - **Larger address space** (solve IPv4 address exhaustion);
  - **Many advantages**: auto-configuration, security, multi-cast, support for ad-hoc network, routing scalability, simpler header structure, improved protocol extensibility, etc.
- **Why now?**
  - IPv4 address space may be fully allocated to Regional Internet Registries by IANA around mid-2010; (http://ipv4.potaroo.net/)
  - IPv4 address pool in RIRs may be exhausted around mid-2011;
  - No more IPv4 address at all (assuming that allocated but unused prefixes come back in the “IPv4 market”) in 10 years!
- **IPv6 deployment will speed up in the remaining years:**
  - **Only 4 years remaining before IPv6 will become mandatory!**
  - Being non IPv6 compliant will be seen as a blocking factor for large scale new software deployment.
  - Both IPv4 and IPv6 will be available during the transition period for a long time

# IPv6 and Grid

- **Why do we need to take care of IPv6?**
  - Sites starting to deploy IPv6 or new sites with IPv6 already;
  - Collaboration & inter-operability with other Grids already running on IPv6;
  - **Need of an IPv6 ready middleware to remain attractive**;
- **EGEE needs to be prepared for this evolution:**
  - **Training developers**;
  - Pushing IPv6 awareness within the community;
  - Knowledge of the implications of IPv6 deployment on:
    - System administration,
    - Security,
    - Third party software,
    - etc.

# Current status of gLite 1/2

## Internal dependencies:


IPv6 Code Compliance Report (powered by ETICS) - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
http://etics.cern.ch/rundir/glite_branch_3_1_0_rhel4_ia32_gcc346_build/reports/ipv6/index.html
Google
gLite
Powered by ETICS and NMI

Data generated using Salvatore Monforte's 'IPV6'

IPV6 Code Compliance Checker

**Project:** glite_branch_3_1_0 (org.glite)
**Configuration:** glite_branch_3_1_0 (org.glite)
**Date:** 15/10/2007 03:57:18
**Success rate:** 67 %
**Status:** Failed

| Component name | Configuration name | Result |
|---|---|---|
| DPM-DSI | DPM-DSI_R_1_6_6_3 | Failed |
| a1_grid_env | a1_grid_env_R_2_0_0_1 | Success |
| ares | ares v. 1.1.1 | Failed |
| boost | boost v. 1.32.0-6 | Failed |
| c-ares | c-ares v. 1.3.0 | Failed |
| classads | classads v. 0.9.8-2 | Success |
| cleanup-grid-accounts | cleanup-grid-accounts-lcg1_0_2 | Success |
| cppunit | cppunit v. 1.10.2 | Failed |
| edg-gridftp-client | edg-gridftp-client_R_1_2_7_1 | Success |

At least 33% of gLite components are non IPv6 compliant.

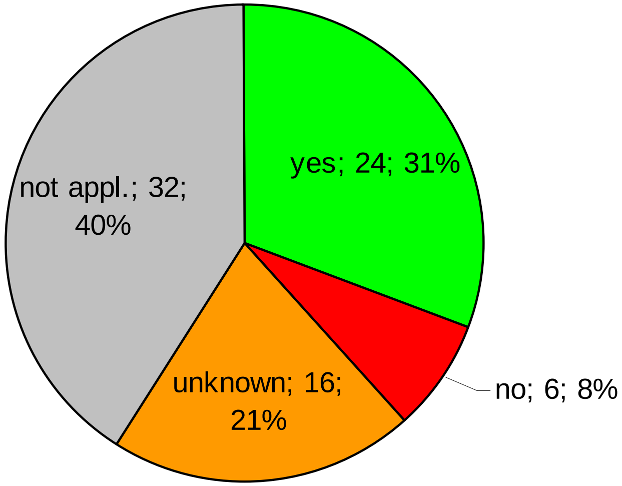
http://etics.cern.ch/rundir/glite_branch_3_1_0_rhel4_ia32_gcc346_build/reports/ipv6/index.html

Enabling Grids for E-sciencE

# Current status of gLite 2/2

- **External dependencies:**

IPv6 compliance of external components



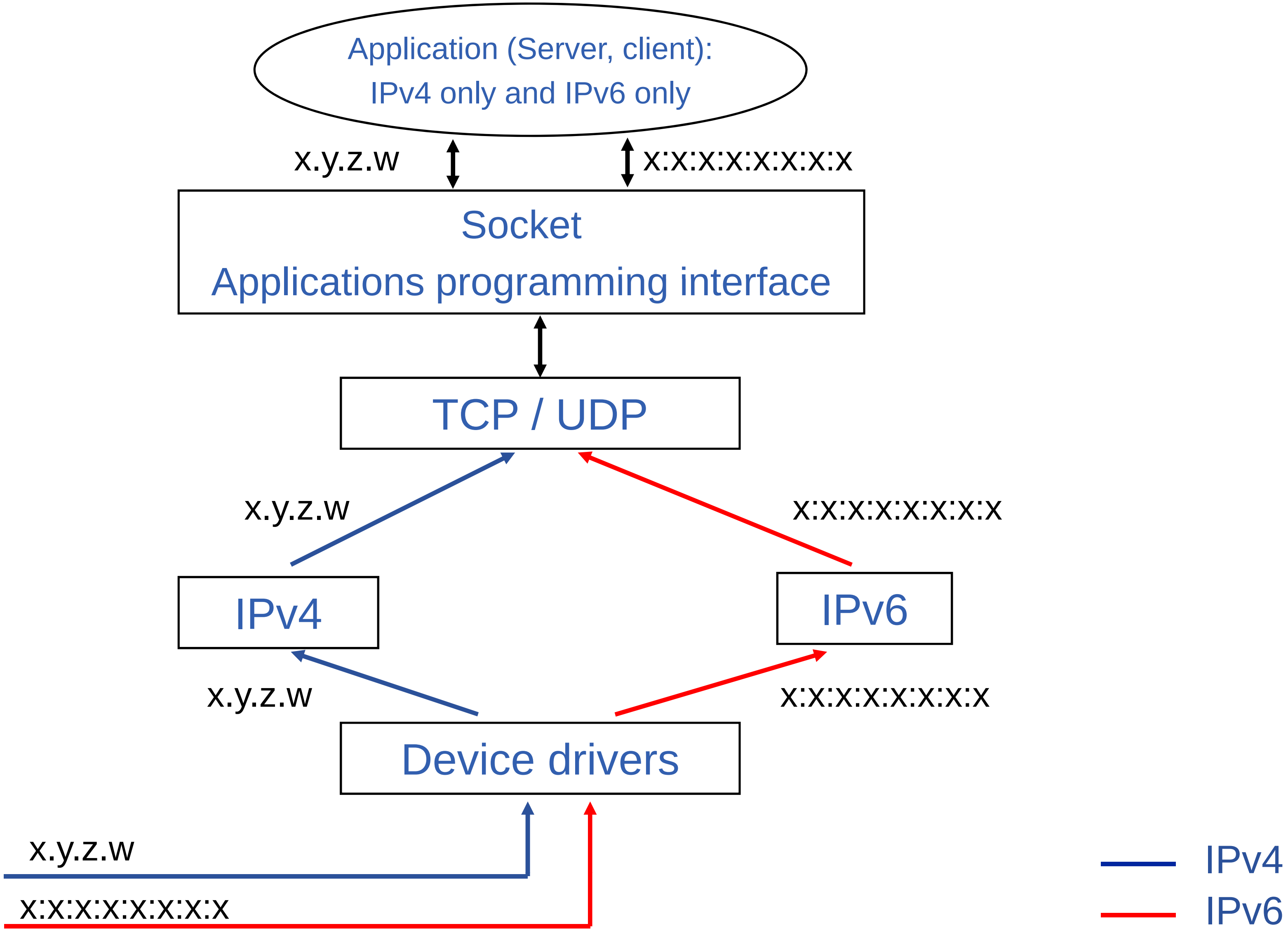
- **Non compliant packages:**

| | |
|---|---|
| **condor** | condor v. 6.8.4 |
| **dcap** | dcap v. 1.2.38 |
| **edg-gridftp-client** | org.edg.gridftp-client.v1_2_5 |
| **mysql-client** | mysql-client v. 4.1.20 |
| **mysql-devel** | mysql-devel v. 4.1.20 |
| **udpmon** | udpmon v. 1.1.2 |

- **Packages with an unknown status:**

| | | | |
|---|---|---|---|
| **bcprov-jdk14** | bcprov-jdk14 v. 1.22 | **hsqldb** | hsqldb v. 1.7.2.3 |
| **boost** | boost v. 1.32.0-1.rhel4 | **Jglobus** | jglobus v. 1.1 |
| **bouncycastle** | bouncycastle v. 1.34 jdk 1.5 | **joram** | joram v. 4.1.2 |
| **db** | db v. 4.2.52 | **lcg-info-templates** | lcg-info-templates-lcg1_0_15 |
| **edg-mkgridmap** | org.edg.mkgridmap.v2_6_1 | **libhj** | libhj v. 4.1.3 |
| **egee-ant-ext** | egee-ant-ext v. 0.4.0 | **sunxacml** | sunxacml v. 1.2 |
| **exist** | exist v. 1.1.1 | **unixodbc** | unixodbc v. 2.2.11 |
| **gssklog-cern** | gssklog-cern.HEAD | **wsi-test-tools** | wsi-test-tools v. 1.1 |

# IPv4 /IPv6 interoperability



Application (Server, client):
IPv4 only and IPv6 only

Socket
Applications programming interface

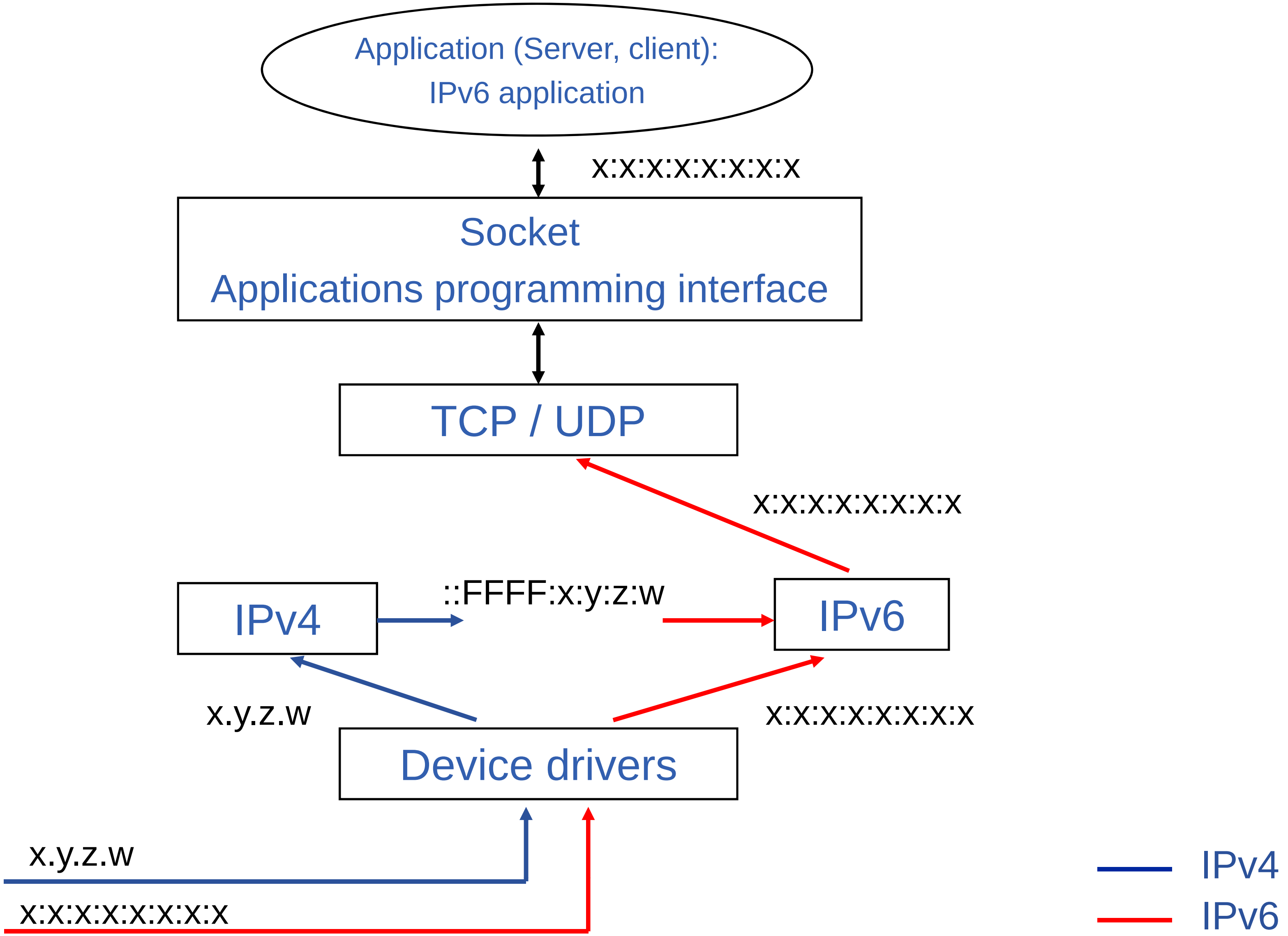
TCP / UDP

IPv4

IPv6

Device drivers

# IPv4 /IPv6 interoperability

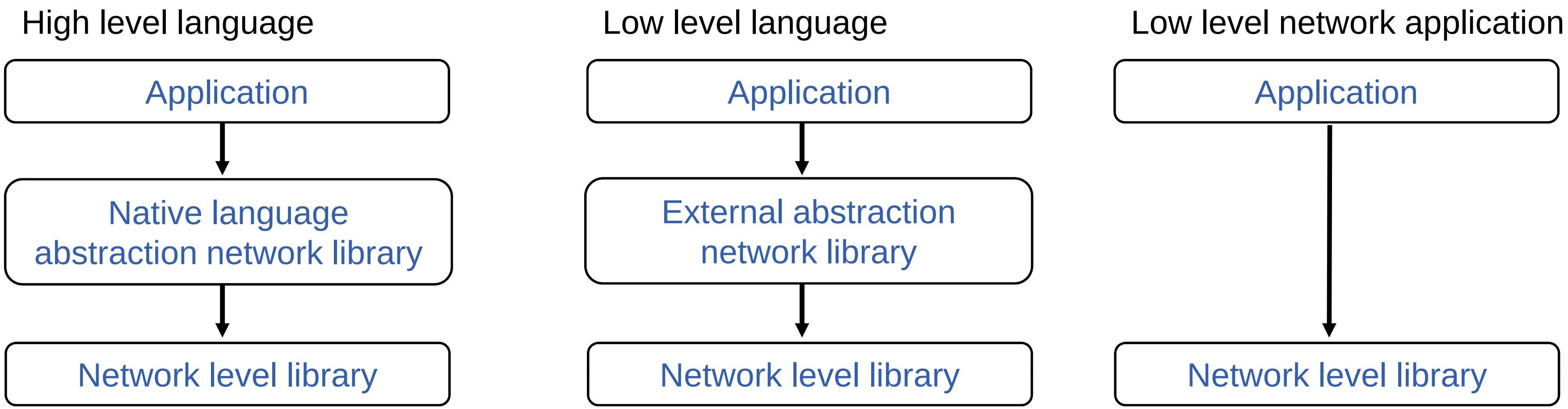


# Roadmap to IPv6

- **Make a list of gLite components with priority**
  - Various criteria: easy to port (Java, Python), or nearly ready, or network interaction, or…
- **Dependencies analysis of the selected gLite component**
  - **Internal gLite component dependencies**
    - Code checker of ETICS building;
  - **External dependencies**
    - List of status of external component (SA2, EUChinaGrid);
    - Upgrade to an IPv6 compliant;
    - Try another component with similar functionalities.
- **Implementation choices for the selected component**
  - Mapping IPv4 address in IPv6 or not? Depending on the targeted OS and the gLite component implementation (PERL, Java…);
  - Write a **network level independent** code: IP.
- **Port the code of the component on IPv6**
- **Test the IPv6 compliance on the testbed**
  - Installation and configuration;
  - Basic features;
  - Test the interplay with other components of gLite.

# Network level independent

- High level language (Java, Python)
  - High level language should hide to the developer all the management of the network stack;
- Low level language (C, PERL)
  - To avoid the complexity of management of the network stack, a library should be used;
- Application that need low level call
  - Use cases: tuning or debugging;
  - Need to manage network stack option;



# An example IPv6 programming: a dual stack server BDII (1/5)

We port the code for the BDII server and tested them following the roadmap and the test methodology:

1. Dependencies
   1. External dependencies analysis: openldap-clients-2, openldap-servers-2; These packages are IPv6 compliant.
   2. Internal dependencies analysis: bdii-3.9.1-4.noarc, this package (PERL) is not IPv6 compliant (see code checker)
2. Implementation choice
   - Two versions as an example
     - An IPv6 server only, that opens one socket using IPv4 mapped address and that listens on the IPv6 address.

       u.x.y.z → ::FFFF:u.x.y.z
     - An IPv4/IPv6 server that opens 2 sockets, one that listens on the IPv4 address and the other one that listens on the IPv6 address.

# An example IPv6 programming: a dual stack server BDII (2/5)

- The first version, an IPv6 server only, that opens one socket using IPv4 mapped address, is easier to develop:
  - Two files and few lines were patched

- Only 4 lines in “bdii-fwd” were patched:

```
[root@quarks sbin]# diff bdii-fwd.map ../sbin.sav/bdii-fwd
10,11d9
< use Socket6;
< use IO::Socket::INET6;      # UREC/CNRS EGEE-SA2 xj
61c59
<     $proxy_server = IO::Socket::INET6->new(@proxy_server_config)
>     $proxy_server = IO::Socket::INET->new(@proxy_server_config)
169c167
<     $remote_server = IO::Socket::INET6->new(@rs_config)
>     $remote_server = IO::Socket::INET->new(@rs_config)
```

# An example IPv6 programming: a dual stack server BDII (3/5)

- **Only 5 lines in “bdii-update”:**

```
[root@quarks sbin]# diff bdii-update.map ../sbin.sav/bdii-update
17d16
< use IO::Socket::INET6;      # UREC CNRS EGEE SA2 xj
416c415
<      my $s = IO::Socket::INET6->new(@port2skip);
>      my $s = IO::Socket::INET->new(@port2skip);
445c444
<      my $guard = IO::Socket::INET6->new(@port2keep);
>      my $guard = IO::Socket::INET->new(@port2keep);
570c569
<      my $s = IO::Socket::INET6->new(@port2skip);
>      my $s = IO::Socket::INET->new(@port2skip);
639c638
<      system("$bdii_fwd_prog --local :: --service $bdii_port_read"
>      system("$bdii_fwd_prog --local 0.0.0.0 --service $bdii_port_read"
```

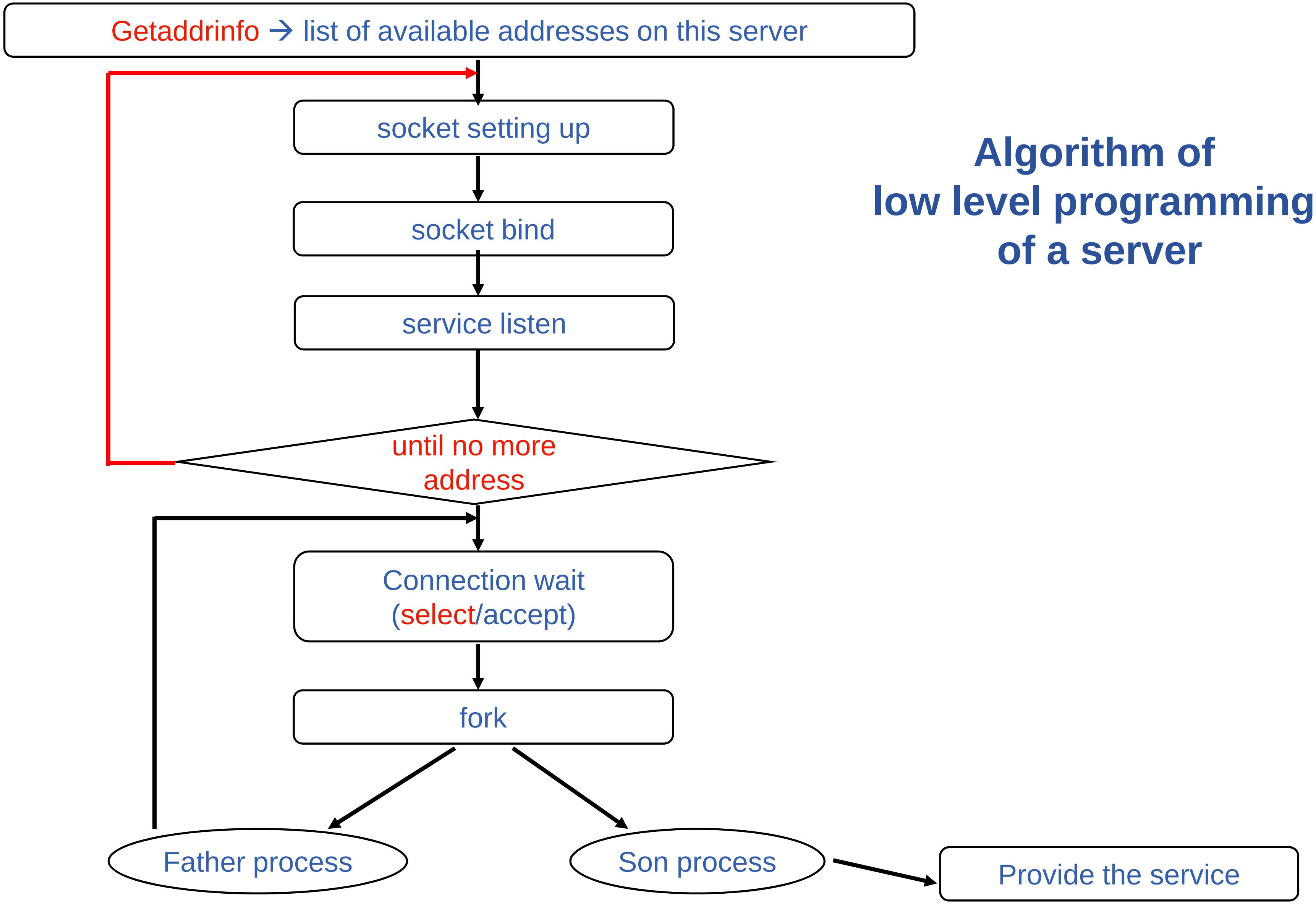
- **We provided this version to BDII developers.**

# An example IPv6 programming: a dual stack server BDII (4/5)

- **The second version (IPv4/IPv6 server, 2 sockets) needs to use a lower level library and to modify more lines in the opening section of the server (bdii-fwd);**

- **But the same code can be re-used for every type of server (respectively for the client):**
  - See next slide.

# An example IPv6 programming: a dual stack server BDII (5/5)



Getaddrinfo → list of available addresses on this server

socket setting up

socket bind

service listen

until no more address

Connection wait (select/accept)

fork

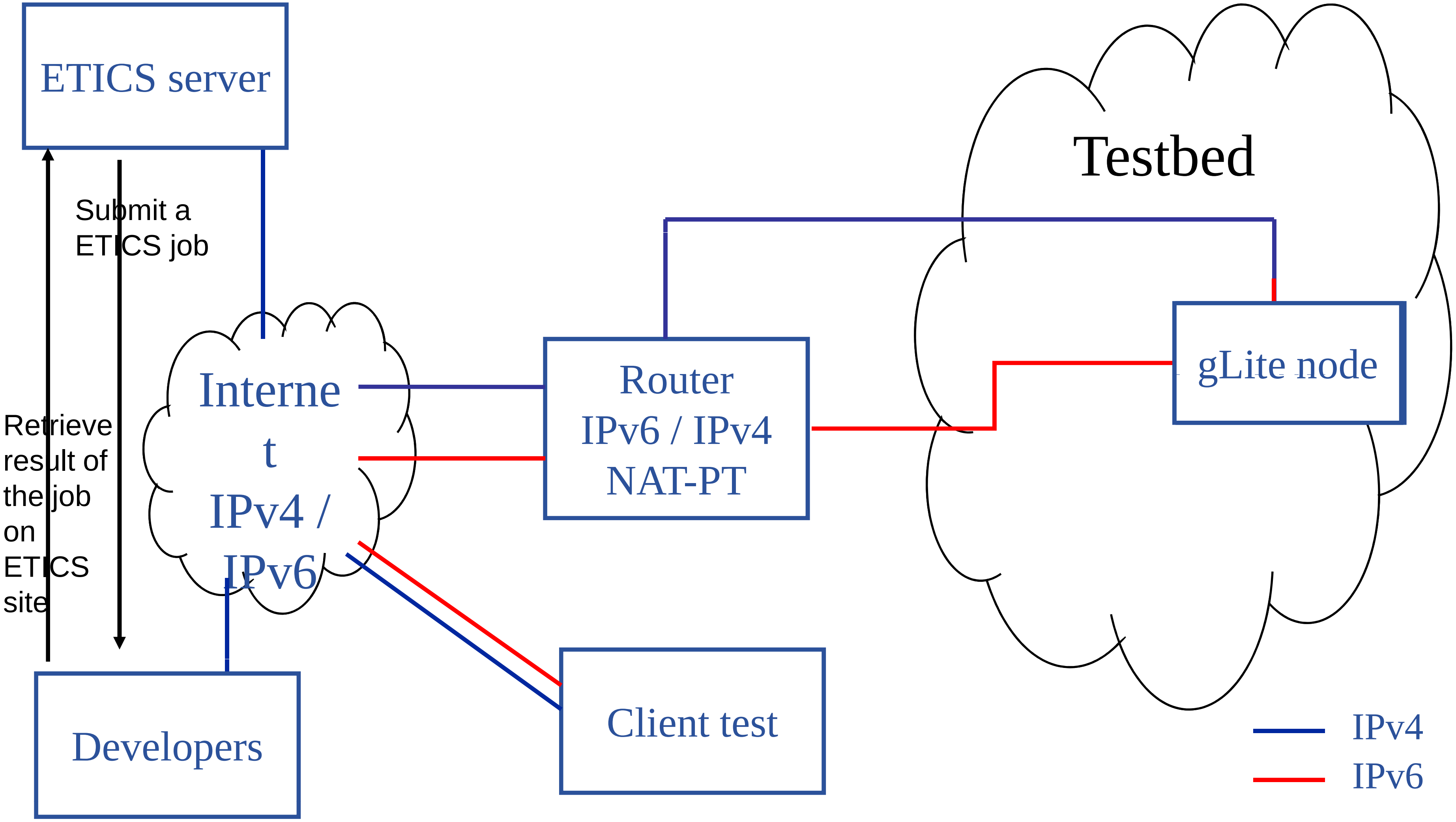
Father process

Son process

Provide the service

## Algorithm of low level programming of a server

# Test scenario and methodology



# Conclusions 1/2

## What can we offer to simplify the porting:

- An automatic code checker in the building tools to check internal dependencies, see ETICS;
- A state of the external dependencies;
- SA2 provides **a test methodology** **(https://edms.cern.ch/document/810278/)**
- **A testbed** (GARR and UREC testbed) providing:
  - IPv6 support;
  - Translation mechanism to test interactions between a gLite componenent and the other part of the operational EGEE grid: NAT-PT, Machines with IPv4 mapped address
- A tool to automate the testing → **ETICS**
  - install your gLite component automatically as much as possible on the IPv6 testbed machines, test it, and retrieve the result;

# Conclusions 2/2

- **IPv4 and IPv6 components will co-exist:**
  - Some gLite meta-packages will be IPv6 compliant while some others won't.
  - We need to test the interplay of a component under IPv6 with other components under IPv4.
- **Roadmap to port gLite on IPv6**
  - Make a list of gLite components with priority to port on IPv6;
  - Choose a strategy with regard to external components:
    - Update external component to IPv6 compliant one's;
    - Substitute external non IPv6 compliant component with IPv6 compliant one's.
  - Update the gLite code based on available programming guidelines → **network level independent** code;
  - Use the testbed to test and validate the updated components;

# Reference

## Links

- https://twiki.cern.ch/twiki/bin/view/EGEE/IPv6FollowUp (under construction)

## References

- RFC 1933 Transition Mechanisms for IPv6 Hosts and Routers
- RFC 3493 basic socket interface extensions for IPv6
- RFC 4038 Application aspects of IPv6 transition Programming guidelines on transition to IPv6, Miguel Castro
- IPv6 network programming jun-ichiro itojun hagino
- IPv4-Mapped Addresses on the Wire Considered Harmful draft-itojun-v6ops-v4mapped-harmful-02.txt jun-ichiro itojun hagino
- Guidelines for IP version independence in GGF specification T Chown
- IPv6 Guide for Windows Sockets Applications [Winsock] http://msdn2.microsoft.com/en-us/library/ms738649.aspx
- IPv6 Théorie et pratique Gisèle Cizault
- IPv6 Transition/Co-existence Security Considerations draft-ietf-v6ops-security-overview-06.txt P Savola
- Status for Java Developers Kit API for IPv6 IPV6_WG J. Bound
- Networking IPv6 User Guide for JDK/JRE 5.0 Sun
- UNIX Network Programming W. Richard Stevens
- Programming guidelines on transition to IPv6 T. P de Miguel, E. M. Castro