REFERENCE ONLY

UNIVERSITY OF LONDON THESIS

Degree PhD Year 2005 Name of Author JIANG S.

COPYRIGHT

This is a thesis accepted for a Higher Degree of the University of London. It is an unpublished typescript and the copyright is held by the author. All persons consulting the thesis must read and abide by the Copyright Declaration below.

COPYRIGHT DECLARATION

I recognise that the copyright of the above-described thesis rests with the author and that no quotation from it or information derived from it may be published without the prior written consent of the author.

LOANS

Theses may not be lent to individuals, but the Senate House Library may lend a copy to approved libraries within the United Kingdom, for consultation solely on the premises of those libraries. Application should be made to: Inter-Library Loans, Senate House Library, Senate House, Malet Street, London WC1E 7HU.

REPRODUCTION

University of London theses may not be reproduced without explicit written permission from the Senate House Library. Enquiries should be addressed to the Theses Section of the Library. Regulations concerning reproduction vary according to the date of acceptance of the thesis and are listed below as guidelines.

A. Before 1962. Permission granted only upon the prior written consent of the author. (The Senate House Library will provide addresses where possible).

B. 1962 - 1974. In many cases the author has agreed to permit copying upon completion of a Copyright Declaration.

C. 1975 - 1988. Most theses may be copied upon completion of a Copyright Declaration.

D. 1989 onwards. Most theses may be copied.

This thesis comes within category D.

☑ This copy has been deposited in the Library of UCL

☐ This copy has been deposited in the Senate House Library, Senate House, Malet Street, London WC1E 7HU.

~~Mobile Support for IPv6-enabled Middleware~~

~~with Special Emphasis on its use in Grid Computing~~

MAKE GRID SYSTEMS IPV6-ENABLED AND
PROVIDE MOBILITY SUPPORT IN GRID SYSTEM
BASED ON MOBILE IPV6

(September 2005)

Sheng Jiang

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London

Department of Computer Science
University College London

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

During the last few years, systems have emerged to perform large-scale computation and data storage over IP-enabled data communication networks using Grid middleware technology. Grid middleware integrates the computational resources, which may be distributed geographically, over networks. These Grid implementations are currently supported only over IPv4. The next generation Internet Protocol – IPv6 - is replacing IPv4 with a number of improvements. Since IPv6 is expected to become the core protocol for next generation networks, Grid computing systems should be able to continue to work as the lower-layer network protocols migrate to IPv6.

Therefore, we studied in depth what needed to be done to integrate IPv6 functionality into Grid middleware; here we include both Grid middleware itself and its interface to the underlying networking environment. We have also given consideration to how a Grid implementation can be made to work in heterogeneous IPv4/IPv6 networks. We have used the Globus Toolkit as our working example of a Grid implementation. However, the mechanisms and approaches for integrating IPv6 into the Globus Toolkit are generic. It should cover the integration of IPv6 into most other Grid implementations and even to other IP-based applications.

Another aspect of my work relates to the provision of mobility support for Grid middleware, since a lot of Grid resources and users have to be mobile in the wide-area distributed computing environment. Amongst the many mobility solutions, Mobile IP we find the most suitable; it has two main advantages in its provision of mobility support in the lower-layer network infrastructure. Firstly, it separates the mobility operations from upper-layer applications, here the Grid middleware. No resultant changes are required in either the applications or the Grid implementations. Secondly, Grid hosts can maintain the identities, so that they can work continuously. The use of Mobile IPv6 rather than Mobile IPv4 is more efficient. This shows that our effort in making Grid middleware IPv6-enabled has brought advantages into the Grid computations.

The success of running Grid middleware over Mobile IPv6 builds up only the lower infrastructure for the mobile-enabled Grid by solving the transparent access and handover issues in mobility scenarios. The Grid needs to be modified and improved further in order to work effectively in the mobile environment. The study indicated the major Grid-relevant issue in mobility scenarios is that the status of the Grid changes frequently. Therefore, we introduce a dynamic Grid resource discovery mechanism. Then, we categorise these important characters into four aspects. They are monitored and parameterised dynamically allowing Grid middleware to assign Grid resources dynamically. Eventually, we provide Grid resource mobility functions.

Finally, while we have concentrated on the Grid environment, most of methodology and the generic approach apply equally well to other environments

Acknowledgements

Although written in the first person nominative plural personal form (also known as "we"), this thesis is all my own work. It would never have been finished, however, without the help of several people, whom I acknowledge here.

Peter Kirstein and Soren-Aksel Sørensen, University College London, have gone beyond the call of duty in providing supervision, guidance and encouragement.

Piers O'Hanlon, Manish Lad and Alan Modlen, University College London, and Tim Chown, the University of Southampton, have made substantial contributions to my work. Also thanks to Saleem Bhatti, John Andrews, Stefano Street, UCL, and David Mills, UoS for your invaluable help.

My PhD work has done under the aegis of the IST, 6NET project. The support of the European Communities is gratefully acknowledged. Also, many thanks to Universities UK, from which I received Overseas Research Students Award.

Last, but certainly not least, thank my families: my dear wife, Jiatao Xiong; my father, Xiansong Jiang, and my mother, Zhen Wei; for their enormous support.

Contents

List of Figures

List of Figures

Terms and Abbreviations

AH	Authentication Header
ALG	Application Layer/Level Gateway
ANL	Argonne National Laboratory
API	Application Programming Interface
AXIS	Apache Extensible Interaction System
BIND	Berkeley Internet Name Domain
Bioinf-UCL	the Bioinformatics Unit the Department of Computer Science at University College London
CDMA	Code Division Multiple Access
CN	Correspondent Node
CS	Computer Science
CS-UCL	the Department of Computer Science at University College London
CVS	Concurrent Versions System
DNS	Domain Name Service/Server
DHCP	Dynamic Host Configuration Protocol
DOCSIS	Data Over Cable Service Interface Specification
ESP	Encapsulating Security Payload

ECS-UoS	the School of Electronics and Computer Science at the University of Southampton
FA	Foreign Agent
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GASS	Globus Access to Secondary Storage
GGF	Global Grid Forum
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GRAAP	Grid Resource Allocation Agreement Protocol
GRAM	Globus Resource Allocation Manager
GriDM	Grid Distribution Manager
GRRP	Grid Resource Registration Protocol
GS	Grid Service
GSI	Grid Security Infrastructure
GSM	Group Special Mobile
GT	Globus Toolkit
GUI	Graphical User Interface
HA	Home Agent
HNR	Host Name Resolution
HTML	HyperText Mark-up Language

Terms and Abbreviations

HTTP	HyperText Transfer Protocol
ICENI	Imperial College e-Science Networked Infrastructure
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	IP security Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
JDBC	Java DataBase Connectivity
JPortal	Job Portal
JXTA	JuXTApose
JYDE	Job Yield Distribution Environment system
LDAP	Lightweight Directory Access Protocol
LMDS	Local Multipoint Distribution Service
MMJFS	Master Managed Job Factory Service
MN	Mobile Node
NAT	Network Address Translator
NAT-PT	Network Address Translator – Protocol Translator
ND	Neighbour Discovery

OASIS	Organisation for the Advancement of Structured Information Standard
OGSA	Open Grid Services Architecture
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
PING	Packet InterNet Groper
PKI	Public Key Infrastructure
PPP	Point to Point Protocol
QoS	Quality of Service
RFC	(Internet) Request for Comments
RFT	Reliable File Transfer
SDK	Software Development Kit
SGE	Sun Grid Engine
SGNP	Secure Grid Naming Protocol
SIP	Session Initiation Protocol
SOAP	Simple Object Access Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TME	Task Mapping Editor
TLS	Transport Layer Security

Terms and Abbreviations

TTL	Time To Live
UDP	User Datagram Protocol
UMTS	(Universal Mobile Telecommunications System)
UNICORE	Uniform Interface to Computer Resources
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
VO	Visual Organisation
VPN	Virtual Private Network
W3C	World-Wide Web Consortium
WCDMA	Wideband Code Division Multiple Access
WG	Working Group
WS	Web Service
WSDL	Web Service Definition Language
WSRF	Web Service Resource Framework
WWW	World Wide Web
XML	eXtensible Markup Language

Chapter 1 Introduction

1.1. Scope of the thesis

During the last few years, Grid middleware [12], [22], [23], [24], [25], [38] has emerged to enable large-scale computation over IP-enabled data communication networks. They use distributed, potentially remote, resources to solve the shortage of local computation and storage resources. They have changed the way we think about computation, data storage, and network services.

The generic architecture of Grid middleware is illustrated in Figure 1-1.

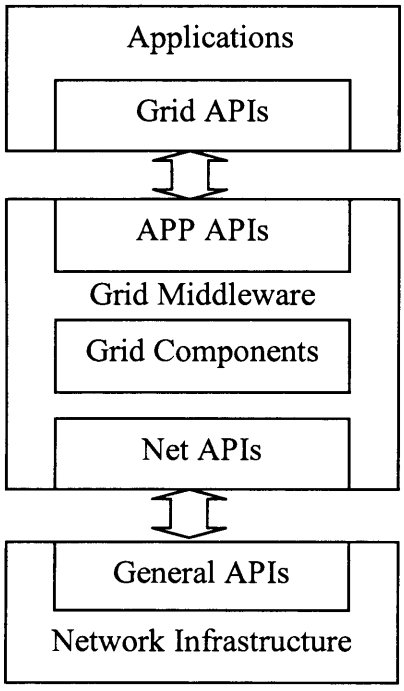


Figure 1-1: Schematic of Grid architecture

The vital differentiator between the Grid and others types of computation system is

the usage of middleware [2]. This is designed so that the applications of Figure 1-1 can be run on clusters of computers and on distributed computing systems. The aim of Grid middleware is to provide all the support functions that the applications need. Grid middleware separates user applications from the network infrastructure. It handles all network-relevant issues for user applications. Hence, the Grid architecture can be divided into three parts: network Application Programming Interfaces (APIs), which provides functions to interact with network infrastructure; Grid components, which provide certain services to user applications; and application APIs, which interacts with user applications.

The interest in Grid computing is so large that there is now an international body to standardise interfaces and services for the middleware systems – the Global Grid Forum (GGF, [72]). It handles some Grid-related standards, most notably Open Grid Services Architecture (OGSA [20]). In addition, the Web Services Resource Framework (WSRF, [131]), a new set of Grid-related standards, are being handled by the Organisation for the Advancement of Structured Information Standards (OASIS, [104]).

As network middleware, lying between applications and network resources, the Grid implementations are currently supported only over the Internet Protocol version 4 (IPv4) [31]. The next generation Internet Protocol – IPv6 [17], [56], [58] is replacing IPv4 in many environments with the potential of improvements in functionality.

As IPv6 is expected to become the core network-layer protocol for next generation networks [17], Grid computing systems should track the migration of the lower-layer network protocols to IPv6. If they fail to do so, Grid middleware would not be able to continue their success in the future networks.

However, the period of transition from IPv4 to IPv6 will not be short. Hence, it is important to make Grid middleware work over both IPv4 and IPv6, and to be able to communicate in heterogeneous IPv4/IPv6 networks.

While it is clear to those concerned with networks that IPv6 is an important development, most of those concerned only with Grid computing have shown little interest in the network level. This has resulted in some problems in the way that Grid

middleware has been structured. This causes some problems in the migration to IPv6.

While it is intended that Grid computing be carried out over the Internet or Enterprise Intranet, the requirements made by the networks on either the applications or the middleware are largely ignored. Our research here is designed to address this current gap. The problems that stop the Grid implemenations running over IPv6-enabled network infrastructure are investigated. Consistently, our research focuses on the network APIs of Grid middleware.

There is a substantial body of activity in the development of Grid computing. While there are many implementations of Grid middleware, the implementation of the Globus Consortium [26], [73] is one of the most popular systems among them. Therefore, we use the Globus Toolkit (version 3), which we call GT3, as our working example of a Grid implementation. Our endeavours to enable it to work over IPv6 networks are discussed. However, the mechanisms and approaches for integrating IPv6 into the Globus Toolkit are generic. It should cover the integration of IPv6 into most other similarly architected Grid implementations and even to other IP-based applications.

The GGF produces a wide range of standards, and has now started an IPv6 Working Group [92]. We have made important contributions to the deliverables of this Working Group.

Another aspect of our research relates to the provision of mobility support in the Grid. Up to now, most Grid research has focused only on fixed systems. On the contrary, some of the Grid sources and users are not fixed. They may work in a wide area and move around.

In order to extend the Grid nodes to be mobile-enabled, mobility support from the lower-layer network infrastructure is required. There are a few mobility solutions from different network layers: link-layer mobility, network-layer mobility and application-layer mobility. Our study has led us to choose Mobile IP [11]. One of the biggest advantages of Mobile IP is that it separates the mobility operations from upper-layer applications; in our case the latter are the Grid implementations. There is no need to change the Grid implementations, though some changes will make Grid middleware

work better in a mobile environment. Another vital reason that we chose Mobile IP is that with Mobile IP the hosts can maintain their identities. It keeps Grid services consistent while mobile-enabled Grid nodes move across networks. We chose the more advanced Mobile IPv6 [34] rather than Mobile IPv4. Running the Grid over Mobile IPv6 is based on our previous work making them IPv6-enabled.

The success of operating Grid middleware on mobile nodes over Mobile IPv6 builds up only the lower infrastructure for a mobile-enabled Grid. It allows a mobile Grid node, either a Grid client or a Grid server, to maintain the network connections while moving across networks. However, it solves only the transparent access and handover issues in mobility scenarios. Several more mobility-relevant issues need to be solved, such as choosing Grid resources dynamically and variable network connections. The Grid implementations need to be modified and improved further in order to be able to respond to these issues in the mobile environment.

In order to adapt automatically to the changes of the lower-layer mobility network infrastructure, Grid middleware should be able to discover Grid resources and their availabilities dynamically. Furthermore, Grid middleware should dynamically monitor and measure those elements that affect, or can potentially affect, Grid performance in a mobile environment. Grid jobs should be assigned according to dynamic status of Grid servers and their network connections.

In our approach, a dynamic Grid resource discovery mechanism is introduced. It uses multicast information to discover available Grid servers and requires update information periodically. This mechanism is not necessary in a fixed Grid, however, it is vital in a mobile-enabled one. It enables advanced Grid function, such as Grid resource mobility support. It also provides the fundamental information for these dynamical monitoring and measurement functions: which Grid hosts and network connections should be monitored and measured.

We have studied these elements that potentially affect Grid performance in the mobile environment. There are four aspects: the capability of Grid servers, the loading of Grid servers, the capability of the Grid client and the network connections between Grid clients and Grid servers. Each of these four aspects includes many parameters. Within different Grid services and different mobility scenarios, the major effective

parameters are varied. We have examined a number of parameters and their effects. Although we are not able to validate all of our analysis in our experiments, we do take some of them to construct a concrete dynamic job assignment function, using a specific application, eProtein, as an example.

We have added more functionality to the Grid resource discovery module that had been developed for eProtein. This module allows a Grid client to monitor and measure these relevant parameters dynamically. When the Grid client needs to assign certain Grid jobs, it can refer to the dynamic status information, in addition to the requirements of Grid jobs normally used. This fulfils a requirement to enable mobility of Grid resources. We have also defined a number of conditions, under which Grid servers should not be used.

By including all the research aspects mentioned above, we provide successfully a generic approach to integrate IPv6 into Grid middleware. Most mobile access issues in the Grid are solved by extending the IPv6-enabled Grid to operate over Mobile IPv6. Our research gives the generic consideration on how to provide support to Grid resource mobility in the mobile environment.

1.2. Limitation of the thesis

My research uses the Globus Toolkit as a concrete implementation of Grid infrastructure. Other existing Grid implementations are not considered.

Although we will discuss how to make the Grid implementations work in a heterogeneous IP environment, the mobility support for Grid nodes is based only on Mobile IPv6. Since the transition between Mobile IPv6 and Mobile IPv4 is difficult to implement, we ignore the latter in the PhD research.

For Mobile IPv6 support from the operating system, the existing Mobile IPL (Mobile IPv6 for Linux) is used. The development of a Mobile IPv6 implementation is not part of the PhD research.

The details of the implementation, such as the source code, are not included in this

thesis, though a write-up is available.

1.3. Outline of dissertation

- In **Chapter 2**, we introduce the relevant background technologies. We introduce the following three aspects independently: Grid middleware, IPv6 and mobility technologies. While there are many technologies details in each aspect, we discuss only these details are strictly relevant to our research.

- In **Chapter 3**, we introduce our motivation to bring the above three technologies together. We also state the research challenges that we will meet. The relevant work that has been done by others is reviewed in this chapter as well.

- In **Chapter 4**, we study what needs to be done in order to integrate IPv6 and the Grid together. We generalise our results so that the most Grid implementations and even to other IP-based applications, can use our approach. Our corresponding validation experiments are introduced in **Chapter 7**.

- In **Chapter 5**, we describe how to provide mobility support in the Grid. We examine existing mobility technologies according to the requirements of Grid middleware, and choose the most suitable one as our solution. We also save the corresponding experiments for **Chapter 7.**

- In **Chapter 6**, we extend our investigation into the impact of mobility on Grid middleware. We examine the parameters that potentially affect Grid performance in the mobile environment. We discuss also how Grid middleware should behave with respect to these parameters. Again, the corresponding experiments are introduced in **Chapter 7.**

- In **Chapter 7**, we introduce our experiments, which are validation of our work in Chapter 4, 5 and 6. They validate both our theory and approach.

Finally, in **Chapter 8**, we summarise of the main contributions of the research, and discuss the potential research that should be carried out in the future.

Chapter 2 Background

2.1. Introduction

In this chapter, we introduce the relevant background technologies. Our work spans three different fields: Grid middleware, IPv6 and mobility. They are introduced independently. The contents in each field are organised according to how they are relevant to our work. Our motivation in bringing these three fields together and filling in the gaps between them, will be introduced in the next chapter.

2.2. Grid middleware

2.2.1. The Grid overview

Today, islands of computing within organisations make inefficient use of resources. Systems are slow to change and expensive to maintain. Grid computing addresses these problems by providing an adaptive software infrastructure that makes efficient use of low-cost servers and modular storage, which balances workloads more effectively and provides capacity on demand. "Grid systems coordinate resources that are not subject to centralized control, using standard, open, general-purpose protocols and interfaces, to deliver nontrivial qualities of service.", by Ian Foster [21].

The term "the Grid" was coined in the mid-1990s. The central idea of Grid computing is that computing should be as reliable, pervasive, and transparent as a utility. The goal is to make computing a ubiquitous commodity. To an end-user or application, Grid Computing systems should look like one big virtual computing system.

"The Grid vision requires protocols (as well as interfaces and policies) that are not only open and general-purpose, but also standardised. It is standards that allow us to establish resource-sharing arrangements dynamically with any interested party and thus to create something more than a plethora of balkanized, incompatible, non-interoperable distributed systems. Standards are also important as a means of enabling general-purpose services and tools." [21] An international organisation of Grid standardisation – the Global Grid Forum was founded in 2000. It is a community-initiated forum of researchers and practitioners working on Grid computing. A number of working groups are producing technical specifications, documenting user experiences, and implementation guidelines. Since its foundation, it has led the standardisation of specifications in the Grid field.

There are many different descriptions of Grid architecture; ours is illustrated in Figure 1-1. The Grid, which include many Grid components, is operated over a network infrastructure. Grid services and Grid resources are made available through networks to serve end-users and user applications. Our research here focuses mainly on the interface between Grid middleware and the lower-layer networking.

In the following three sections, we review Grid services and Grid resources; Grid infrastructure and its components; and applications based on the Grid infrastructure in detail.

2.2.2. Grid services and Grid resources

Grid Services have emerged and have been formalised with the introduction of the OGSA. The OGSA integrates key Grid success factors with the Web services mechanism, providing a service-oriented view of Grid Architectures. A Grid service is "a Web service that provides a set of well-defined interfaces and that follows specific conventions. The interfaces address discovery, dynamic service creation, lifetime management, notification, and manageability; the conventions address naming and upgradeability" [22].

A Grid resource is a collection of Grid services. It might be a cluster, storage system, database, or scientific instrument of considerable value that is administered in an organised fashion according to some well-defined policy. A Grid resource is also a

virtual resource in a Virtual Organisation. It does not need to be mapped onto certain physical Grid nodes. One can obtain the functionality of Grid services from different Grid nodes in order to serve the requirements of users' applications from different locations.

In this thesis, we use the term "Grid server" to represent a physical Grid device. This Grid device may be a single processor server or a cluster, which is organised as a single Grid resource.

2.2.3. Grid infrastructure and its components

An infrastructure is a technology that we can take for granted when performing our activities. To be useful, an infrastructure technology must be deployed widely. Therefore, it must be efficient, valuable, or both.

A Grid infrastructure needs to provide more functionality than the Internet upon which it rests, but it must remain efficient as well. Typically, a Grid infrastructure includes the following main components, see Figure 2-1:

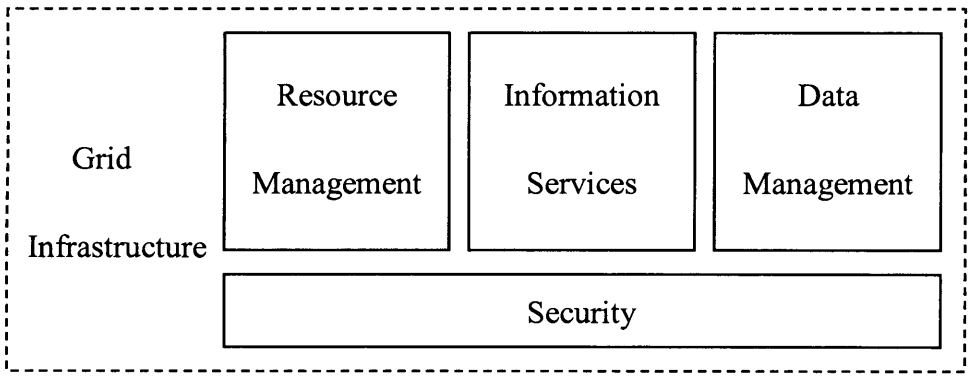


Figure 2-1: Main components of typical Grid infrastructure

- **Security**. Security is an important consideration in Grid computing. Each Grid resource may have different security policies that need to be complied with. A single sign-on authentication method is a necessity. A commonly agreed-upon method of negotiating authorisation is also needed.

- **Resource management**. When a job is submitted, a Grid resource manager is concerned with assigning a resource to that job, monitoring its status, and

returning its results.

- **Information services**. For the Grid resource manager to make informed decisions on resource assignments, the Grid resource manager needs to know which grid resources are available, as well as their capacities and current utilisation.

- **Data management**. Data management is concerned with how jobs transfer data or access shared storage.

All these above Grid components operate on the network functions. They invoke network functions directly. The Grid infrastructure provides a reusable platform for the upper-layer Grid applications. It solves the generic and common Grid problems.

There are many implementations of a Grid infrastructure, such as the Globus Toolkit [76], Imperial College e-Science Networked Infrastructure (ICENI) [83], Condor-G [70], Legion [123], Task Mapping Editor (TME) [108], and Uniform Interface to Computer Resources (UNICORE) [127]. Among them, the Globus Toolkit, as we will introduce in the next section, is one of the most popular. Some other Grid implementations are built on the Globus Toolkit, such as Job Yield Distribution Environment (JYDE [95]), which we will introduce in Section 2.2.3.2.

2.2.3.1. Globus and the Globus Toolkit

Globus is a concrete implementation of Grid middleware. There is a large current project, called the Globus Alliance, to develop Grid middleware. It is intended to achieve a vertically integrated treatment of application, middleware, and network. A low-level toolkit provides basic mechanisms such as communication, authentication, network information, and data access. These mechanisms are used to construct various higher-level metacomputing services, for instance, parallel programming tools and schedulers.

The Globus Toolkit, which is the concrete result of the Globus Alliance in the Argonne National Laboratory (ANL [68]), provides a middleware platform for the Grid. It provides essentially libraries and services for the computational Grids and the Grid applications, including resource managers, security protocols, information services, communication services, fault tolerance services, and remote data access

facilities.

The Globus Toolkit fulfils these main components of Grid infrastructure we introduced earlier:

- **Implementation of Security**. Security mechanisms underlie the operation of Grid. These security mechanisms are provided by the Grid Security Infrastructure (GSI), which enables the use of certificates to provide authentication and authorization services. GSI is a library for providing generic security services for applications that will be run on the Grid. GSI provides programs to facilitate login to a variety of sites, while each site has its own flavour of security measures.

- **Implementation of Resource Management**. The first pillar of the Globus Toolkit provides Resource Management, which involves the allocation of Grid resources. It includes such packages as the Globus Resource Allocation Manager (GRAM [75]) and Globus Access to Secondary Storage (GASS).

- **Implementation of Information Services**. The second pillar of the Globus Toolkit is for Information Services, which provide information about Grid resources. Such utilities include the GT3 Index Service.

- **Implementation of Data Management**. The third pillar of the Globus Toolkit is for Data Management, which involves the ability to access and manage data in a Grid environment. This involves such utilities as GridFTP [77] and the Reliable File Transfer (RFT) service, which are used to move files between Grid-enabled devices.

The edition of the Globus Toolkit with which I worked mainly was Version 3 (GT3). This provides a Grid service container, which is built on top of the OGSA primitives and protocols. Figure 2-2, from the Globus Alliance, provides a general view of Globus protocols and their relationship to external systems:

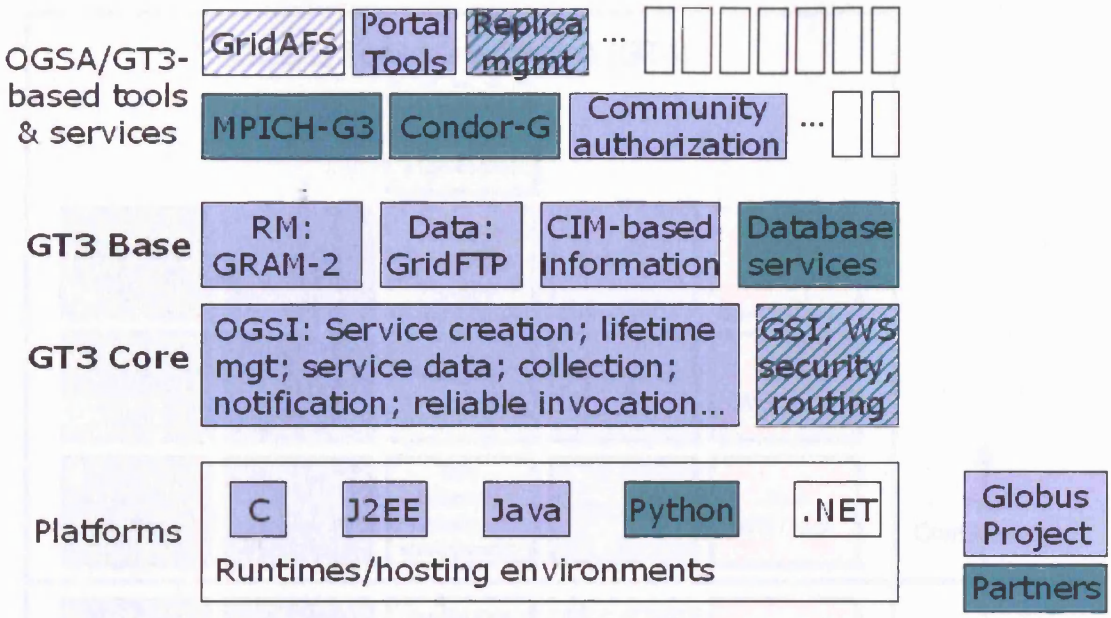


Figure 2-2: Globus Toolkit 3 hierarchy

A new version of Globus – GT version 4 – was released in April 2005. Most of GT4 services are implemented on top of WSRF. Figure 2-3, also from the Globus Alliance, provides a general view of GT4 components referring to whether they are Web Services (WS [130]).

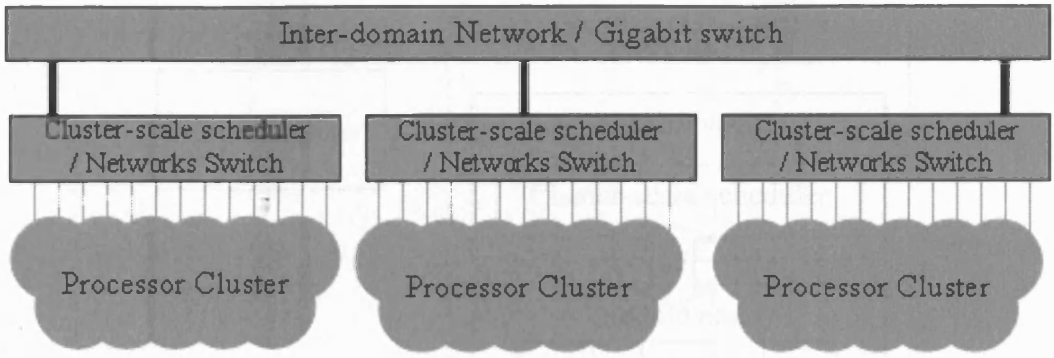


Figure 2-4: hierarchy model of sub-clusters in JYDE system

As shown in Figure 2-4, the JYDE system uses a cluster-scale scheduler to structure and manage the processor cluster. Between the clusters, a Grid implementation – Globus toolkit is used to remotely submit Grid jobs.

JYDE system first used a constant Grid resource table to assign Grid jobs. Later, an advanced function – dynamical Grid resource discovery mechanism was implemented in order to discover the existence and availability of Grid resources dynamically.

Within many Grid, including JYDE system, cluster-scale schedulers are used widely in order to integrate effectively computational cluster nodes to be a single Grid resource. Several cluster-scale schedulers, such as Sun Grid Engine (SGE) [107] and Condor [116], have been developed to manage the Grid jobs within a cluster.

These cluster-scale schedulers provide centralised control of their managed hosts: it has complete knowledge of system state and user requests, and complete control over individual components. As shown in Figure 2-5, a cluster-scale scheduler is mainly operated on a master node, which communicates and manages all execute nodes within the cluster. The execute nodes receive Grid jobs from the master node and return the results to it. Only the master node is able to communicate and receive Grid jobs from the "outside" applications. With this approach, the cluster is a single Grid resource.

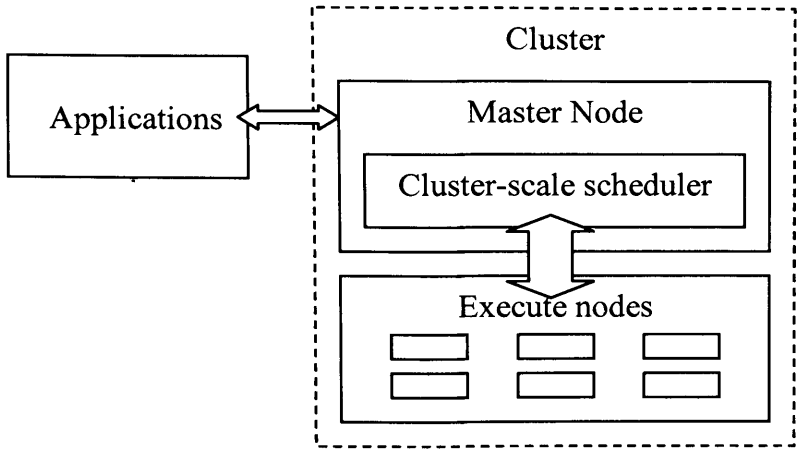


Figure 2-5: Schematic of cluster-scale scheduler

The cluster-scale schedulers are good for organising a single control cluster. They are also ideal to be the bottom layer of a hierarchical Grid. However, they work only in a single managed domain; they cannot work across networks.

2.2.4. Applications based on Grid infrastructure

As shown in Figure 1-1, the applications based on a Grid infrastructure invoke Grid resources through a Grid infrastructure. These applications are user-oriented. They are developed to meet specific requirements from users by invoking the accessible and re-usable Grid services provided by Grid resources. Users can submit jobs to these applications. Then, these applications interact with Grid resources by invoking APP APIs of Grid middleware.

2.3. IP moves to next generation

Most of network applications build on either the Transmission Control Protocol (TCP [30]) or the User Datagram Protocol (UDP [32]). They are based on the core Internet Protocol – IP, as shown in Figure 2-6. IP is also the interface to make the whole of lower-layer protocols or devices, such as Ethernet [80] and Point to Point Protocol (PPP [66]), to be global accessible. It is the core of the Internet.

In the last several years, the next generation Internet Protocol – IPv6 has been developed aiming to replace the current IPv4. We will review IPv6 and its relevant

works in this section.

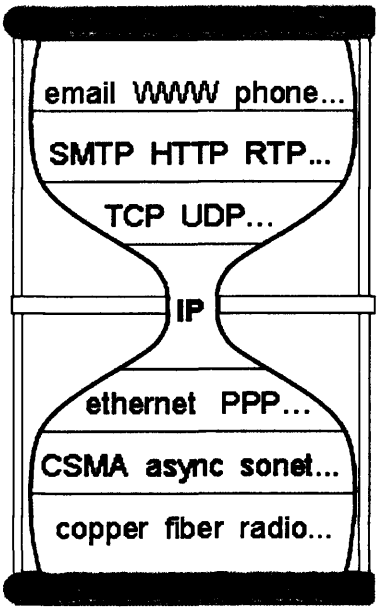


Figure 2-6: IP - hourglass model of networking

2.3.1. IPv4 review

Before we review IPv6, it is necessary that we first briefly review its predecessor – IPv4.

The TCP/IP suite was first developed in the mid-1970s and has become the network standard by the early-1980s. They are the world's most popular open-system protocol suite, because they can be used to communicate across any set of interconnected networks and are equally well suited for LAN and WAN communications.

IP is a network-layer (Layer 3) protocol that contains addressing information and some control information that enables packets to be routed. IP is the primary network-layer protocol in the Internet protocol suite. Along with the TCP, IP represents the heart of the Internet protocols. Almost all the other TCP/IP protocols and functions [3], [13] are constructed by layering atop IP.

2.3.2. IPv6 overview

The rapid growth of the current Internet causes problems with IP addressing space depletion. In addition, global IP mobility creates demand from consumers for value-

added services, and much better QoS is being required by new IP network application. The next generation IP - IPv6 is designed to replace IPv4 with a number of improvements. In the early 1990s, the Internet Engineering Task Force (IETF) [84] began an effort to develop IPv6 as a successor to the IPv4 protocol and the bulk of the IPv6 standards were ratified by the IETF in 1998. IPv6 fulfils the future demands on address space, and also addresses other features such as multicast, encryption, auto-configuration [62], Quality of Service (QoS) [18], and better support for mobile computing. In comparison with the current IPv4 protocol family, IPv6 offers a number of significant advantages. The IPv6 data format does not really provide most of these advantages by itself though it expands address length to 128 bits from the 32 bits of IPv4. However, the design of the IPv6 protocol suite has taken the opportunity to re-design the relevant protocols with a better and more logical system.

At present, standards on IPv6 are far enough along, that vendors have already committed to a considerable number of development and testing projects. Although study on IPv6 has been under way for several years, there are still some debates, more research and study are desirable. The potential of IPv6 is still not fully discovered and fulfilled. Many more advantages of IPv6 will become apparent when the study and application of IPv6 are progressed further.

2.3.3. IPv6 advantages over IPv4

As a solution to the future Internet, IPv6 has the following enhanced features in comparison with the current IPv4:

- **Expanded Addressing Capability**

  IPv6 increases the size of IP address to 128 bits to accommodate enough address space. In addition, this larger addressing space enables a flexible, hierarchical global routing architecture [48], [49].

  Moreover, the size of IPv6 addresses allows more addressing hierarchies. The IPv6 global aggregation addressing architecture splits addresses into two parts. The high-order 64 bits identify the network, and the low-order 64 bits identify the node. The improved addressing hierarchy allows smaller routing tables and faster lookups.

With the enlarged address space, IPv6 allows full, global IP connectivity for IP-based machines as well as upcoming mobile devices like PDAs and cell phones – all can benefit from full IP access through end-to-end services. There can be multiple addresses for a single interface, where the addresses can be used for different functions. The larger address space allows for simpler end-to-end security, IPv6 renumbering mechanism, separated addressing and routing [48], etc.

Whilst the larger address space provides the potential to individual address for more end systems, the existence of Network Address Translators (NAT [55]) cannot be ruled out. Given their prevalence in IPv4 networks, NATs may well exist in IPv6 networks. However, their necessity is reduced, and factors such as their management overhead and limited throughput may help alleviate their deployment. In addition, in terms of security, the larger address space reduces the ease of certain address scanning techniques. Consequently, IPv6 systems should not discount the existence of NATs in their design.

- **IP Security (IPSec)**

The IP security architecture (IPSec [36]) is mandatory for IPv6 implementation, but not required for IPv4, though it is increasingly being made available. It provides integrity, authentication and encryption of IP traffic. In IPSec, a set of security services can be provided through the use of two traffic security protocols, the Authentication Header (AH) and the Encapsulating Security Payload (ESP), and cryptographic key management procedures and protocols.

With IPSec, all IP traffic between two nodes can be handled without adjusting any applications. Using IPSec, all applications on a machine can benefit from encryption and authentication, and policies can be set on a per-host (or even per-network) basis instead of per application/service.

- **Mobile IPv6**

Mobile IP is a solution to support Mobile Nodes maintaining their network connection while moving cross the networks. The details of both Mobile IP and Mobile IPv6 are introduced in the next section.

- **Simplified header and flexible extension**

  IPv6 simplifies the header of the packet to reduce the cost of processing and to save network bandwidth. Less header fields can expedite the processing rate of routers. The chain-type of IPv6 extension headers makes the IPv6 header more flexible for future use.

- **Neighbour Discovery and Auto-configuration**

  Neighbour Discovery (ND) protocols [58], [62] are proposed to determine information dynamically about the directly attached networks. This information includes local network parameters, resolution of IPv6 addresses to layer 2 address, route redirections and status of neighbouring nodes. In addition, the fully plug and play protocol in IPv6 includes advanced facilities to support auto-configuration of network interfaces on hosts, based on the information advertised by neighbouring routers.

- **Quality of Services (QoS)**

  QoS is introduced into the Internet with the aim at supporting real-time services and providing a network client with a range of service-quality levels at a range of prices. The approaches of offering QoS require that different types of traffic be treated differently by intervening routers in Internet. To achieve this goal, IPv6 has an elusive definition of a "flow" as a header field that allows labelling of packets belonging to particular flows.

2.3.4. IPv6 support and development

All of the major router vendors (Cisco, Nokia, Bay Network, Digital, 3Com, etc.) have already added IPv6 capabilities to their products. End-System vendors, such as Apple, Digital Equipment, Hewlett-Packard, International Business Machine, Microsoft, Silicon Graphics and Sun have also delivered IPv6 on desktop machines and servers. In addition, many organisations are working on IPv6 drivers for the popular UNIXs including the BSD and Linux operating systems. IPv6 support has been provided on most of the current operating systems [87].

In fact, IPv6 is being deployed worldwide today. A number of IPv6 commercial and

science activities are operational. In Asia Pacific, the Widely Integrated Distributed Environment (WIDE [129]), KAME [122] and Test and Verification for IPv6 (TAHI [126]) projects have made important contributions to the IPv6 development. In Europe, a few projects, such as 6bone [110], IPv6 Wireless Internet IniTiative (6WINIT [91]), and 6NET [112] have devoted themselves to IPv6 deployment. In North America, there are many activities related to IPv6, in terms of both standardisation and deployment/testing, such as 6REN [113] and 6TAP [114]. IPv6 is proposed for use since Universal Mobile Telecommunications System (UMTS [128]) version 5 and is planned for Data Over Cable Service Interface Specification (DOCSIS [117]) version 3 as well.

2.3.5. IPv6 application porting

While the lower layer is changing, any applications based on IP communication should track the migration from IPv4 to IPv6 [54]. A few projects, such as Kame and Long [124], have made massive efforts on IPv6 porting [88]. A number of applications have already been ported to be IPv6-enabled [85].

The migration from IPv4 to IPv6 will be a gradual and flexible transition. The mechanisms for transition have been designed so that there are as few dependencies as possible between the different parts of the transition. However, for most of applications, some modifications must be made to make the applications IPv6-enabled, while still allowing IPv4 to work.

In general, the following aspects have been identified as potential IP dependencies. They need to be modified to be IP-independent. They are relevant on both standardisation and implementation.

- **IP Address Representation**. The most obvious differences between IPv4 and IPv6 lie in the address size and format itself. In IPv4, addresses are 32 bits, represented as a dot-delimited decimal quad address, while in IPv6 they are 128 bits, represented as colon-delimited hexadecimal address.

- **Storage and Display of IP Addresses**. There are different storage requirements for addresses in each protocol. These may affect specifications and implementations where text representations of addresses are being handled. An

IPv4 address may be up to 15 characters long (12 digits plus three dots), while an IPv6 address may be up to 39 characters long (32 characters plus seven colons). The minimum length of a displayed IPv4 address is seven characters (four digits plus three dots), and an IPv6 address is three characters (two colons and a digit) for a regular address, or two characters ('::') for the unspecified address. The '::' notation indicates one or more groups of 16 bits of zeros. In many operating systems and development kits, IP-independent formats and data structures are presented. In implementations, they should be used instead of both IPv6-only and IPv4-only data structures.

- **Use of Fully Qualified Domain Names**. Some applications may pass IP addresses in the payload of their data. In the case of IPv6 it will be commonplace for hosts to have multiple IPv6 addresses, and potentially for more renumbering events to occur. There are also additional IPv6 host addresses for hosts implementing IPv6 Privacy Extensions [57]. As a result, there is a strong argument for hosts to exchange fully qualified domain names (FQDNs) rather than IP addresses, especially given the FQDN is an IP-independent identifier for the host. It is currently not uncommon practice for applications to exchange IP addresses as data for communication endpoints.

- **Handling Literal IPv6 Addresses**. In IPv4, the common delimiter for address and port representation is a colon. Since IPv6 addresses contain colons, a new method for expressing address:port pairs is required where literal addresses are used. The method adopted to handle this problem in application or context-dependent Uniform Resource Identifiers (URIs [6]) is the format specified in [29], i.e. [address]:port, e.g. http://[2001:0DB8:a0:1::1]:8080. The '[]' solution of [29] can be used for other situations, e.g. in SIP-based applications.

- **Documentation Examples**. There is an IETF proposal to use a common documentation prefix in specification documents [27], namely 2001:0DB8::/32. Specifications should use this prefix where address examples are given.

2.3.6. Related IP transition work

The migration of IPv4 to IPv6 will not happen overnight [16]. Rather, there will be a

period of transition when both protocols are in use over the same infrastructure. One of the major problems for the development of IPv6 is to undertake an appropriate transition from IPv4 towards IPv6. It is expected to be a long transition period, during which it will be necessary for IPv4 and IPv6 nodes to coexist and communicate with each other.

A few projects, such as 6WINIT and 6NET, have made massive efforts on the transition mechanisms between IPv4 and IPv6. A number of transition tools [90] are available as we will introduce in the following sections. Different transition schemes may be used at different transition stages. Basically, the transition schemes may be classified into three categories: dual-stack, tunnelling and protocol translation.

Before we introduce these transition mechanisms, it is necessary to mention two important concepts: hostname and IPv4-compatible IPv6 address format, as we will introduce in the next section.

2.3.6.1. Dual-stack

Dual-stack [47] is the essential component in IP transition scenarios. In all IP transition scenarios, some dual-stack gateway must be operated on the border between IPv6 network and IPv4 network. With this approach, nodes implement both IPv6 and IPv4. These IPv6/v4 nodes can communicate to dual-stack node, IPv6-only node and IPv4-only node.

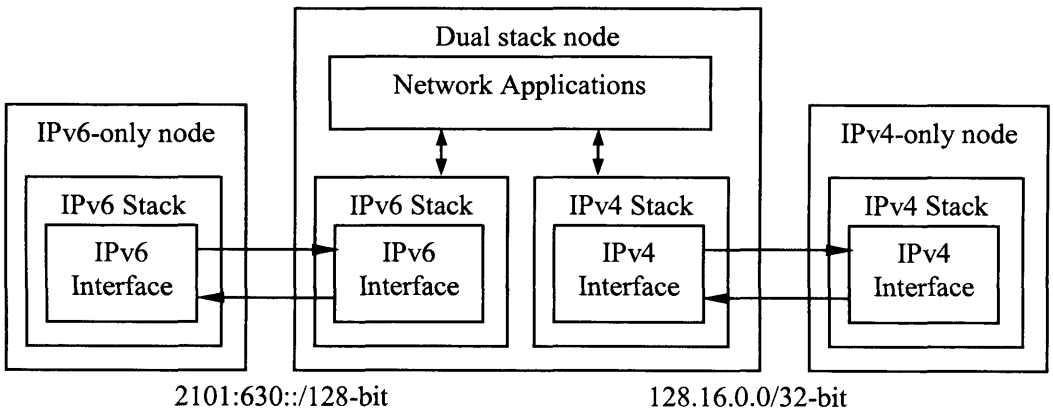


Figure 2-7: dual-stack node is able to communicate with both IPv6-only and IPv4-only nodes/networks

In Figure 2-7, we can see that a dual-stack node has the capability to communicate with both IPv6-only and IPv4-only nodes. It can also communicate with another dual-stack node over either IPv6 or IPv4. However, it does not enable the communication between IPv6-only and.IPv4-only nodes.

2.3.6.2. Hostname and IPv4-compatible IPv6 address format

In the heterogeneous IP environment, the hostname has a very important role. It is the key that brings IPv4 and IPv6 together. A hostname is independent from its IP addresses. A dual-stack host can have both IPv4 address and IPv6 address, which are binding to the same hostname. In the Domain Name Server records, as shown in Figure 2-8, IPv6 address is in AAAA (IPv6 Address Record) and IPv4 address is in A (IPv6 Address Record) query type. This allows a node starts a communication requirement without the awareness of the IP status of the destination node. The DNS service itself is not necessary to be linked to the query type. In another word, DNS can respond to an A query within an IPv6 connection or an AAAA query within an IPv4 connection.

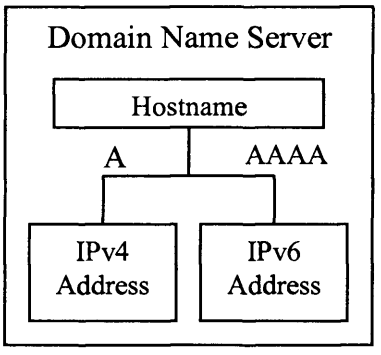


Figure 2-8: Hostname binding to both IPv4/IPv6 address in DNS

However, it is important to note that one should not assume IPv6 connectivity by the presence of an IPv6 DNS record (an AAAA record). The target host may have no or only some IPv6 services actually enabled.

The concept of IPv4-compatible IPv6 address format was first introduced in the automatic tunnelling [47] scenario. An IPv4-compatible address is identified by an all-zeros 96-bit prefix, and holds an IPv4 address in the low-order 32-bits, such as ::w.x.y.z. In practice, the IPv4-compatible address can use any 96-bit prefix, if they

are only used in the local network. In our network, we use 2001:630:13:1b2::/96. Hence, the IPv4-compatible IPv6 address in our network is 2001:630:13:1b2:w.x.y.z.

2.3.6.3. Tunnelling

An alternative to the dual-stack approach is known as a tunnelling transition mechanism [5], as shown in Figure 2-9. The aim of this mechanism is to connect IPv6 hosts/routers within IPv4 networks when there is no IPv6 backbone provided by an Internet Service Provider (ISP).

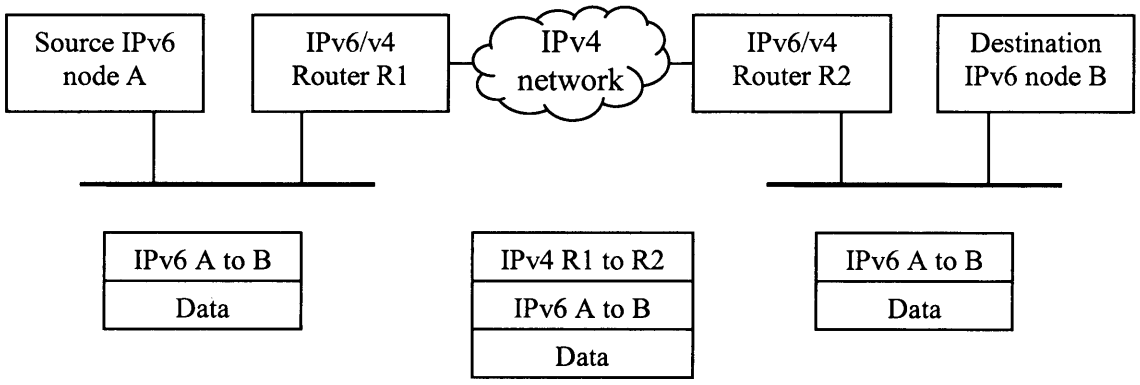


Figure 2-9: IPv6-over-IPv4 tunnelling scenario

2.3.6.4. Protocol translation

The situation becomes complicated when an IPv6-only node requires access to an IPv4-only node or IPv4-only node requires access to an IPv6-only node. The protocol translation mechanisms [64] allow the communication between IPv6-only and IPv4-only nodes.

Work on developing standards for IPv6 transition has been undertaken in the Next Generation Transition (ngtrans [81]) Working Group of IEFT, and its successor – a new IPv6 Operations (v6ops [82]) Working Group since the summer of 2002. The methods adopted by v6ops are: Transition Mechanisms [47], Stateless IP/ICMP Translation (SIIT [19]), Network Address Translator and Protocol Translator (NAT-PT [28]), and 6to4 [8].

2.4. Mobility Technologies

2.4.1. Mobility technologies overview

At its most basic, being mobile is being capable of moving from place to place. For networking, that means maintaining network connections while moving from network to network. The mobility technologies provide the underlying technologies and services that enable this possibility to the users.

In our research, mostly we consider the support for mobile devices and the communication with IP-based networks.

In the following Figure 2-10, we show a schematic of mobility technologies in a specific layered architecture. Our goal in describing this architecture is not to provide a complete enumeration of all relevant technologies, but rather identify the necessary components to fulfil mobility support for the upper layer application.

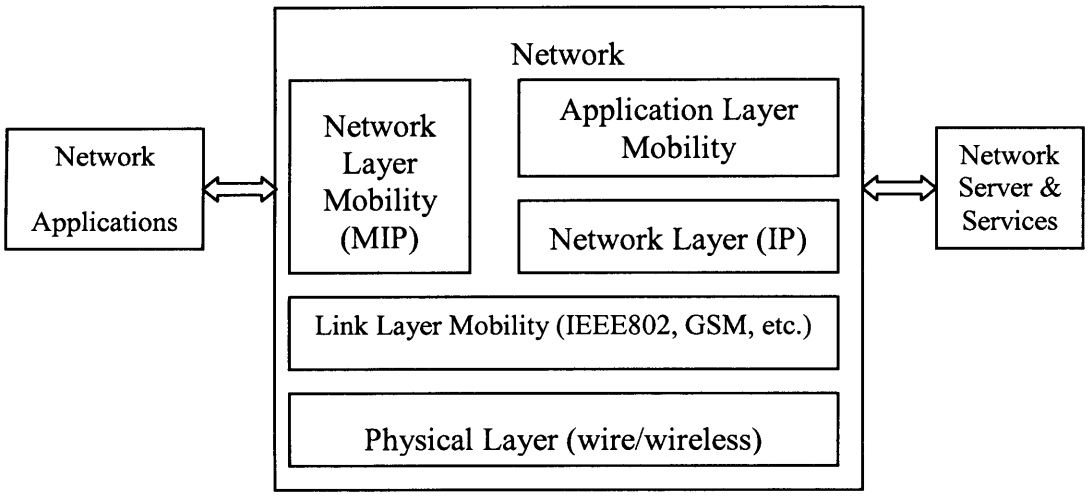


Figure 2-10: Schematic of mobility technologies

Mobility means freedom from the restrictions on locations and networks. It is also the aim of wireless communication technologies – freedom from the constraints of wires. It makes the wireless communication technologies the best match physical-layer infrastructure for the mobility. Today, many mature wireless communication technologies have been widely used already, such as Bluetooth [101], radio.

However, in some other mobility scenarios, such as portable computing, wire communication technologies can be used as well. Also, in many mobility scenarios, wireless communications are only part of network communications. The rest of networks are linked together by cables.

2.4.2. Link-layer mobility

The goal of cellular mobility standards is to provide global connectivity, where the IP layer is not involved in the mobility management. This is called the link layer or layer 2 mobility. It supports the movement of wireless devices.

By term link layer, we refer to these mobility protocols that are below the IP layer. Many protocols have been widely adopted today. In Wireless Wide Area Networks, Group Special Mobile (GSM [78]), Code Division Multiple Access (CDMA [69]) and UMTS are used. In Wireless Local Area Networks, there are IEEE 802.11 [79], Digital Enhanced Cordless Telecommunications (DECT [71]), High Performance Radio Local Area Network (HIPERLAN [120]) and Local Multipoint Distribution Service (LMDS [96]).

In link-layer mobility, it is not possible to access directly the Internet, which is based on IP. The access to IP networks is through one specific IP router – home Gateway GPRS Support Node (GGSN). These applications that do not need access to the Internet can directly build on the link-layer mobility. In some scenarios, applications can use IP over the link-layer mobility while the mobile device's point of attachment to IP network remains the same. In such scenarios, the link-layer mobility solutions are more or less link technology specific.

2.4.3. Network-layer mobility

The increasing variety of wireless devices offering IP connectivity, such as Personal Digital Assistants (PDAs), handhelds, and digital cellular phones, are beginning to change our perception of the Internet. Today's Internet is ubiquitous, always available, and always on, but it is not possible to plug in any device at any location, and is not invisible as well. Basically, the problem regarding to anything being plugged in at any location is that the initial TCP/IP protocol assumes that end-users and their devices and IP addresses would all be found at the same location and would all be tightly

coupled. But the truth is that users are nomadic. The moving of the Internet nomads [37] means that ubiquitous access to the Internet from anywhere at anytime is necessary. In dynamic environments, it is necessary to find ways to share resources in an adaptive fashion. The task to provide better mobility support in IP network [10] was reasonably emerged.

Mobility support by Dynamic Host Configuration Protocol (DHCP [46]) solves temporary IP address assignment issue for mobile devices. DHCP servers can dynamically assign IP addresses to client machines in their managed network. When a mobile device attach to a network that has a DHCP server, it will automatically get an IP address, and some other configuration information, such as the subnet mask and default router.

Mobile IP is a special protocol for maintaining transparent network connectivity to mobile hosts. Mobile IP is first introduced by IETF in the middle of 1990s. It was designed to solve this problem by allowing the mobile node to use two IP addresses: a fixed home address and a care-of address that changes at each new point of attachment.

Mobile IP allows network packets sent to the home address to be delivered to the mobile node. In addition, Mobile IP can hide any address changes from the transport and application layers, enabling the mobile terminal to roam seamlessly between different access networks.

As IPv4 and IPv6 will co-exist for a long time, correspondingly, there are two Mobile IP protocols – Mobile IPv4 [9] and Mobile IPv6.

2.4.3.1. Mobile IPv6 vs. Mobile IPv4

Principally, mobility support is possible and standardised for both IP versions, IPv4 and IPv6. However, while IPv6 appears to replace IPv4, the IPv6 protocol suite has taken the opportunity to re-design the Mobile IP with a better and more logical system. Mobile IPv6 is more efficient than Mobile IPv4 [34]. We list briefly the main advantages of Mobile IPv6 compared with Mobile IPv4 as the following:

- Mobile IP has to assign global IP addresses to a mobile node on each point it attaches to the Internet. On the links which serve for mobile nodes, a set of IP

addresses assigned as mobile node care-of addresses has to be reserved. Due to address shortage in IPv4, there may be problems on some links to reserve enough global IPv4 addresses. However, for IPv6, there are enough addresses available.

- Using addresses of IPv6 enables multi-homing for mobile node. Mobile IPv6 makes efficient use of anycast mechanism for the Dynamic Home Agent Discovery mechanism by sending a Binding Update to the Home Agent anycast address and getting a response from exactly one of several Home Agents. IPv4 doesn't provide such an elegant solution.

- Using stateless address auto-configuration and neighbour discovery mechanisms, Mobile IPv6 needs neither DHCP nor foreign agents on foreign links to configure the care-of address of mobile nodes. In Mobile IPv4, there have to be Foreign Agent(s).

- To avoid waste of bandwidth due to triangle routing [34], Mobile IP specifies the mechanisms of Route Optimisation. While Route Optimisation is an additional functionality for Mobile IPv4, it is an integral part of Mobile IPv6.

- Mobile IPv6 can use IPSec for all security requirements, such as authentication, data integrity protection, and replay protection.

2.4.3.2. Mobile IPv6

The design of Mobile IPv6 has benefited from both the experiences gained from the development of Mobile IPv4, and the opportunities provided by IPv6. Mobile IPv6 thus shares many features with Mobile IPv4, but is integrated into IPv6 and offers many other improvements. As regards next generation mobile networks, IPv6 is mandated by the 3rd Generation Partnership Project [109].

Mobile IPv6 allows mobile nodes to remain reachable while moving around in the IPv6 Internet. Without specific support for mobility in IPv6, packets destined to a mobile node would not be able to reach it while the mobile node is away from its home link. In order to continue communication in spite of its movement, a mobile node can change its IP address each time it moves to a new link, but the mobile node would then not be able to maintain transport and higher-layer connections when it

changes location. "Mobility support in IPv6 is particularly important, as mobile computers are likely to account for a majority, or at least a substantial fraction, of the population of the Internet during the lifetime of IPv6" [34].

In Mobile IPv6, a mobile node is always identified with its home address regardless of whether it is at or away from home. While at a foreign network the mobile node has a care-of-address, which identifies its current point of attachment to the network. The home address is stored by the Home Agent (HA) in the home network. Having moved to a foreign network the mobile node registers the care-of-address with its HA by sending it a Binding Update. The Home Agent then, in turn, implements a proxy function that defends the mobile node's home address and intercepts all packages destined for the mobile node's home address and tunnels them to the mobile node using the care-of-address. Hereby the Home Agent mechanisms ensure that the mobile node is reachable by its home address regardless of its actual point of attachment to the internet and, moreover, this mechanism enables ongoing communications to continue in spite of the movement by the mobile node from one network to another. When the mobile node returns to home, the binding is erased from the Binding Cache and the proxy function is terminated.

In order to support a mobile node, there is one more support host needed – the Home Agent. The Home Agent in the home network is another node that operates Mobile IPv6. It has a registry list including these IPv6 home addresses of mobile nodes, and another binding cache list for the IPv6 care-of addresses of mobile nodes.

The mobile node needs to be registered in at least one Home Agent. On the other hand, the mobile node has the IPv6 address of Home Agent. Mobile IPv6 has also provided Dynamic Home Agent Address Discovery mechanism. However, we do not go that far in this thesis.

With Mobile IPv6 functions enabled on both mobile node and Home Agent, the mobile node is able to remain the network connections when it moves across the networks.

When the mobile node moves into a foreign network, shown in Figure 2-11, it receives the router advertisement from the IPv6 router in the network. According to

the IPv6 prefix information in the router advertisement, the address auto-configuration function on the mobile node forms its care-of address. All packets addressed to this care-of address will reach the mobile node on the current link.

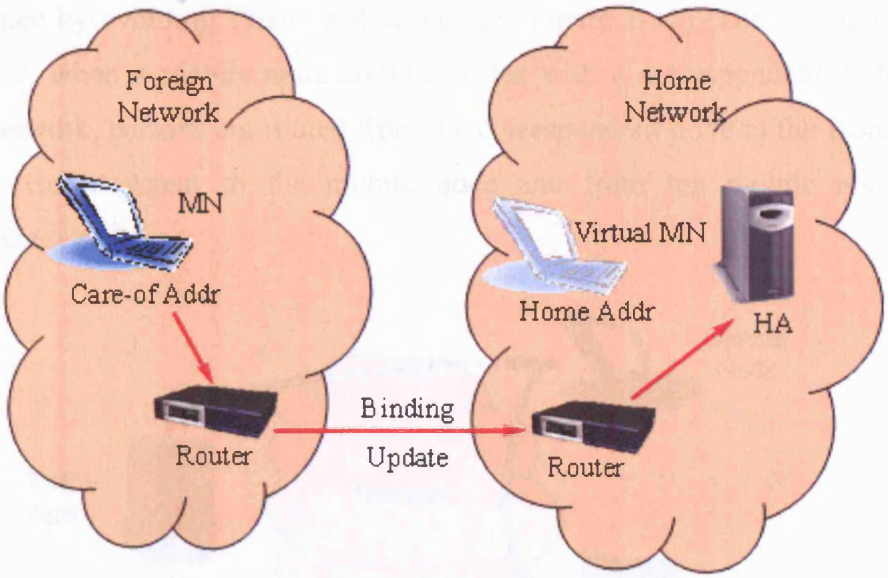


Figure 2-11: Binding Update to Home Agent – Mobile IPv6

Then, the mobile node sends a packet containing a "Binding Update" destination option (red arrows in the Figure 2-11) back to its Home Agent. After the Home Agent receives the "Binding Update" packet, it maps the care-of address and the home address of the mobile node. After the Home Agent Registration, the Home Agent uses proxy Neighbour Discovery and also replies to Neighbour Solicitations on behalf of the mobile node. It intercepts packets on the home link destined to the mobile node's home address, encapsulates them, and tunnels them to the mobile node's registered care-of address.

The above operations create a virtual mobile node with its home address in its home network. It allows the mobile node transparently maintaining all of its present connections and remaining reachable on the static home address to the rest of the Internet.

A mobile-enabled Grid node supported by the above functions and operations is able to move across the network and transparently maintain the connections with other Grid nodes. On the other side, the other Grid nodes are able to access the mobile-

enabled through its home address.

Through the above operations does not require any changes for the other Grid nodes, operating these Grid nodes as Mobile IPv6 correspondent nodes can improve the performance by avoiding Triangle Routing (see Figure 2-12). The Triangle Routing anomaly is, when a mobile node communicates with a correspondent node from a foreign network, packets are routed from the correspondent node to the Home Agent, from the Home Agent to the mobile node and from the mobile node to the correspondent node.

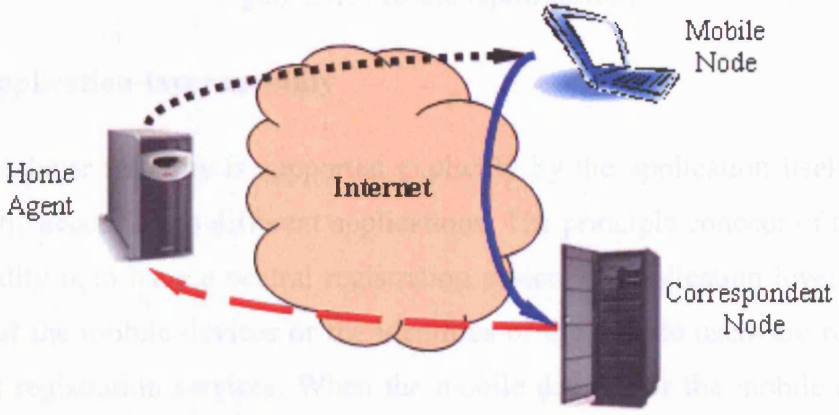


Figure 2-12: Triangle Routing of Mobile IP

By operating Mobile IPv6 functions on the correspondent Grid nodes, these nodes maintain their own binding cache. When the mobile-enabled Grid node is in a foreign network, it can send correspondent Binding Update packets (blue arrow in Figure 2-12) to these correspondent Grid nodes. This allows correspondent IPv6 Grid nodes to cache the current care-of address and send packets directly to the mobile-enabled Grid node (black arrow in Figure 2-13).

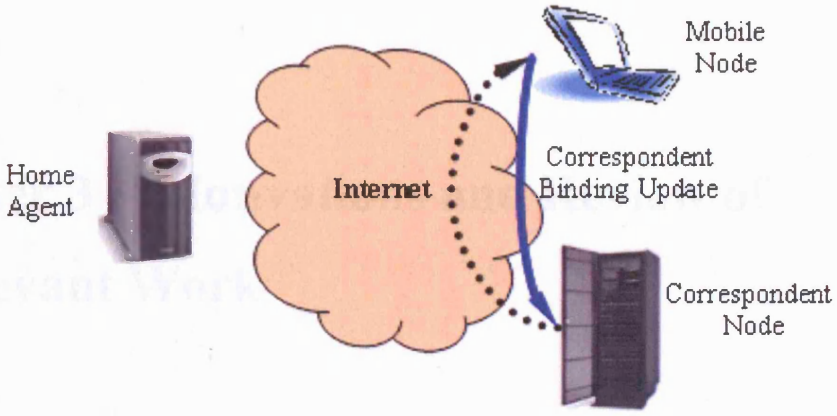


Figure 2-13: Route Optimisation

2.4.4. Application-layer mobility

Application-layer mobility is supported explicitly by the application itself. It can be very specific according to different applications. The principle concept of application-layer mobility is to have a central registration system in application level. Either the identities of the mobile devices or the identities of the mobile users are registered in the central registration services. When the mobile devices or the mobile users move into new places, they report their new status, which can be new IP addresses or new devices, associated with their identities to the central registration services. With this mechanism, the mobile devices or the mobile users can keep their identities when they move across network and their lower-layer network infrastructure changes.

The most mentioned example is application-layer mobility using Session Initiation Protocol (SIP [33], [60]). It provides support to mobile Internet multimedia applications.

2.5. Summary

In this chapter, we have reviewed Grid middleware, IPv6 and mobility technologies individually. Our motivation to bring them together will be detailed in the next chapter.

Chapter 3 Motivations and Review of Relevant Work

3.1. Introduction

There are many technologies that have changed and will continue to change our world. Grid computing; IPv6 and mobility are among them. However, they should not be isolated from each other; their combination provides greater benefits.

The efforts to combine them together can be divided into three steps: firstly we need to make sure that the Grid implementations can run over IPv6; then, and only then, we can provide the mobility support in Grid middleware by deploying Mobile IPv6 underlying them; ultimately, we focus on how to provide effective Grid service in the mobile-enabled Grid environment.

There is a lot of research needed in these fields. We will first describe our motivations and research challenges in this chapter and review the relevant work that has been done by others. We will introduce our solutions and efforts, and validate them by experiments in the following chapters.

3.2. Moving Grid middleware into IPv6 era

As a large system over network, Grid middleware lies on IP protocol. The network communication requirements of the Grid include transport, routing, and naming. Although alternatives certainly exist, the current Grid implementations use only the protocols drawn from the TCP/IP protocol stack: specifically, the Internet (IP and

ICMP), transport (TCP, UDP), and application (DNS, HTTP, XML, SSL/TLS, etc.) layers of the Internet layered protocol architecture.

As we have introduced in detail in Section 2.3, IPv4 is expected to be replaced by IPv6. Grid middleware must track the migration of the lower-layer network protocols to IPv6. Moving Grid middleware into the IPv6 era will also bring a few advantages into Grid middleware as well.

In our research, we need to investigate most of the Grid communication aspects for IPv6 purposes. Both general protocols that the Grid implementations invoke and Grid-specific protocols need to be examined. As an example, at least one particular Grid implementation needs to be ported to IP-independent and tested in IPv6 and IP heterogeneous environment. We also need to abstract our approaches to be generic knowledge so that most of the Grid implementations and even to other IP-based applications, can use our mechanisms.

The Globus Toolkit, our concrete example of the Grid implementation, was designed to work with IPv4, though many aspects were compatible with IPv6. However, the IPv6 and IP-independent considerations were never tested in this type of environment. Therefore, it must be modified to be fully IPv6-compatible and tested in the right environment.

We first started porting the Globus Toolkit version 2 (GT2) to an IPv6-enabled form in the middle of 2002. This almost reached the state of a working system by the end of that year. However, in January 2003, an alpha release of GT version 3 was produced. This was a radical departure from GT2. Therefore, we abandoned our earlier work and switched our focus onto GT3. A Japanese project 6Grid [111] claimed an IPv6-enabled GT2 by the end of 2003.

3.3. Bring mobility support into Grid middleware

In the wide-area distributed computing, a lot of Grid devices, either Grid clients or Grid servers, are not fixed or required to move. They need mobility support to be able to wander among networks.

3.3. Bring mobility support into Grid middleware

There are a few mobility-relevant research aspects within Grid fields: access to the Grid through mobile devices [39], [43], [67]; support for user mobility [15], [52], such as a mobile-enabled Grid user moving from one Grid device to another Grid device; using mobile devices for Grid resources, [35], [41], [43], [132].

In our research, we first investigate the general mobility technologies giving considerations to both using mobile devices to access the Grid and using mobile devices for Grid resources. Although most of general mobility technologies can provide some kind of mobility support in Grid middleware from the lower-layer network infrastructure, some certain mobility technology can serve the Grid purpose better than others. There are two major factors that we used to examine general mobility technologies: one, when Grid devices move from a network into another, they should be able to access the new network and access to the Internet, particular the IP world, through the new network; two, Grid devices should also be identified as the same so that their previously linked Grid nodes could return the job results.

Only after choosing the suitable lower-layer mobility solution, can we focus on the effects that the mobility brings into Grid middleware. The current Grid middleware has no consideration of or even awareness about the network topology changes caused by Grid devices moving. They are not able to respond to any necessary network changes. As a result, some Grid services may be broken or become unavailable. Therefore, in our research, we will discuss how Grid middleware can respond to these network changes without going deep into the lower-layer network infrastructure. This means Grid middleware should be able to discover Grid resources dynamically. Then, no matter what network topology changes, Grid middleware can automatically adapt. We take advantage of others' work on dynamic resource management, such as Agile computing [42] and runtime dynamic scheduling [14]. Agile computing is geared toward a more dynamic and volatile environment with shifting resources and requirements, runtime dynamic scheduling is a system dynamically reschedules running processes over a network of computing resources via automatic decision-making and process migration. These mechanisms result in the provision of some similar functions to ours. In our implementation, we use broadcast and "request and respond" in our dynamical resource discovery mechanism. Furthermore, these parameters that could affect Grid performance in the mobile environment need to be

monitored and measured dynamically, so that Grid middleware can have the right information to choose certain Grid servers. We will also discuss how Grid middleware should behave in relation to these parameters.

3.4. Summary

We have been motivated by the wide open research to bring Grid, IPv6 and mobility together. Many research issues need to be solved. We are going to study these issues deeply and give our solutions in the next three chapters.

Chapter 4 Integrating IPv6 and Grid middleware

4.1. Introduction

As we have introduced in Section 3.2, most of the current Grid implementations use IPv4-based network resources to serve the upper layer applications and users. They might have been designed with IPv6 in mind, but were never tested in an IPv6-enabled environment. Like all widespread applications, Grid middleware should be prepared to move into the IPv6 era. While we take advantage of IPv6 features, we also give thought to communication in heterogeneous IPv4/IPv6 networks. Even more modifications might be required to make Grid work in a mixed IP environment even when special IP transition network services are provided. It is important to make Grid middleware work on both IPv4 and IPv6 and be able to communicate in heterogeneous IPv4/IPv6 networks, see Section 4.6.

We take the Globus Toolkit version 3 as our concrete working Grid example and IPv6-enabled it. However, we did not limit ourselves to just the Globus Toolkit. As generic research, our approach to integrate IPv6 into Globus introduced in this chapter would also allow the integration of IPv6 into other Grid implementations and even to other IP-based applications. In fact, both the Globus Toolkit version 4 and the JYDE system have followed our guidelines when they were designed and implemented, and they have been tested working over IPv6.

In this chapter, we will first give the IPv6 specific consideration in Grid middleware. Then, we will discuss the IPv6 host support and network environment that Grid needs. We will also introduce our experience of integrating IPv6 into Grid implementations.

We keep the consideration as general as we can, so that others can use our approach in other Grid implementations, or even in other upper layer applications. We also give consideration to communication of the Grid in heterogeneous IPv4/IPv6 networks. We will introduce the current status of global Grid IPv6 standardisation, in which we have participated and made important contributions.

4.2. IPv6 benefits for Grid middleware

Grid is the most promising technology that may provide an infrastructure for both business and science. This kind of infrastructure needs to be established on a world-wide scale. The following facts are part of the reasons why IPv6 is required by the Grid.

The full availability of IPv6 allows provision of better Grid services. We address here three major benefits that Grid can obtain from IPv6 in detail: bigger address space, mobility support and security support, along with a brief description of many other advantages and potential benefits.

4.2.1. Larger address space for Grid aggregation

With its larger address space and much better address aggregation properties, IPv6 potentially makes massive scaling of Grid networking possible; this is important in view of the aims to deploy Grid computation globally.

The nature of Grid applications requires a huge number of computational resources, such as processors and data storage, to be linked together for problem solving. IPv6 provides such possibility: each device can have global IPv6 address(es); they can communicate with each other directly.

4.2.2. Better mobility support in distributed networks

Until recently, most Grid research has focused only on fixed systems. However, the mobility support within Grid middleware will be needed as mobility takes an even more important role in modern life. As we have discussed in the previous Section 2.4.3.1, Mobile IPv6 provides better mobility support than Mobile IPv4. Mobile IPv6

can provide mobility support in distributed networks or global-scale networks. In our research, we actually utilise Mobile IPv6 to provide the mobility support in Grid middleware.

4.2.3. Built-in security enables tighter Grid security

While scalability, performance and heterogeneity are desirable goals for any distributed system, including Grid middleware, the characteristics of computational Grids lead to security issues. Although the security improvement from IPv6 does not solve all the security problems, Grid middleware can benefit from IPv6's security features. The IPv6 security and Grid Security Infrastructures are running at different levels. They can be employed together to provide better security granularity.

Full IPSec security operates over IPv4 today when there is a full end-to-end connectivity. If private IP addresses are used rather than global IP addresses, as often occurs in IPv4 networks but are not needed in IPv6 ones, it is not possible to fully deploy IPSec on the end-to-end communications. These considerations are not too significant in the current Grid development, but would be serious if the security mechanisms intended for IPv6 were adopted.

4.2.4. Other IPv6 advantages for Grid benefit

A number of IPv6 advantages for Grid benefit have been stated in [51]: IPv6 allows separation of addressing and routing; it provides better routing scalability compared to IPv4. It also simplified the management of Virtual Organisations (VO); the simplified packet format, router functions and header processing for normal packet handling in IPv6 should allow improvements in performance for Grid middleware; the auto-configuration mechanisms in IPv6 would allow Grid users to construct and use more easily the IP-based Grid infrastructure without having to have detailed network-specific knowledge; consistent use of IPv6 multicast and anycast will allow powerful new mechanisms for Grid community; QoS-sensitive Grid-based applications could benefit from using the IPv6 flow label to identify individual end-to-end flows.

4.3. IPv6 considerations in Grid middleware

As with network middleware, the Grid depends upon its network environment for access to network services. Therefore, before we can work out how to make a Grid implementation IPv6-enabled, we must provide the right IPv6 environment. This includes both the IPv6 support from the hosts and the IPv6 network services, such as DNS. Grid middleware depends highly on network services, some of which are vital for IPv6-enabled Grid.

The Grid implementations are quite complicated. They invoke many generic network protocols, such as HTTP, WSDL, FTP, and SOAP. In addition, they also utilise Grid-specific protocols, such as the Secure Grid Naming Protocol (SGNP), the Grid Resource Allocation Agreement Protocol (GRAAP), and the Grid Resource Registration Protocol (GRRP). In order to make a Grid implementation IPv6-enabled, we have to examine these protocols first. In certain cases, the actual protocol may need modification, though in general only the Grid implementation requires changes.

In order to run in the heterogeneous IP networks, Grid middleware needs to be IP independent – the capability to run over both IPv6 and IPv4. There are potentially extra network services required, such as NAT-PT and Domain Name Server – Application Level Gateway (DNS-ALG).

In the following sections, we are going to meet the above-mentioned IPv6 consideration in Grid middleware.

4.4. IPv6 environment for Grid middleware

As a distributed system, Grid middleware runs over distributed hosts that are linked by networks. Therefore, in the first step of integrating IPv6 with Grid middleware, the hosts must be IPv6-enabled. The IPv6-capable application API libraries are required so as to run the IPv6-enabled applications or IP-independent applications over IPv6. All network-associated applications, such as network-sharing database applications and web containers, need to be IPv6-enabled to run over IPv6. In order to run a Grid implementation over an IPv6-enabled network rather than only on local hosts, IPv6

supports on the network is essential. We will discuss how to build up an IPv6 environment for Grid middleware step-by-step.

4.4.1. IPv6-enabled Operating System on hosts

The IPv6 support on hosts depends on the Operating System (OS) and its kernel [87].

For Linux distributions [89], from Linux Red Hat 8 (kernel 2.4.18) or Fedora (kernel 2.4.22), IPv6 support is provided as a kernel loadable modules. In the earlier distributions of Linux, users had to re-compile the kernel to get the IPv6 support. Peter Bieringer's online "IPv6 & Linux - Current Status – Distributions" [105] webpage has provided the detail information of IPv6 support on different Linux distributions.

On Windows 2000 [97], an IPv6 preview package is available with limited functionality. The IPv6 support in Windows XP [98] provides better IPv6 functionality and comes as part of the standard distribution, though it requires an explicit installation.

4.4.2. IPv6-capable application API libraries

IPv6-capable application API libraries need to provide support for upper-layer applications.

With the IPv6-compatible kernel and the IPv6 module loaded, Linux system libraries provide a few IPv6 data structures, such as sockaddr_in6, in6_addr and in6addr_loopback, and a few IPv6 system functions, such as inet_ntop ( ) and inet_pton ( ), are available to be used. But they are not IP-version-independent. To be IP-independent, other IP-independent data structures, such as addrinfo and sockaddr_storage, and functions, such as getaddrinfo ( ) and getnameinfo ( ), should be used on dual-stack servers and server applications. For IPv6 support, the Java-based Grid implementations should use data structure and function provided by the Java Software Development Kit (SDK [93]).

As a platform-independent runtime environment, the Java SDK 1.5.0 provides the full IPv6 support on Solaris, Linux, WinXP and Windows 2003 server; whilst the IPv6 support in Java SDK1.4.X is only provided on Linux and Solaris. Within Java SDK 1.4.0 and above, the class java.net.InetAddress has two direct subclasses:

java.net.Inet4Address and java.net.Inet6Address. These provide the support for IPv4 and IPv6 addresses respectively. The InetAddress class uses the Host Name Resolution (HNR) mechanisms to resolve host names to their appropriate host address family. Additionally, there are various Java system preferences that can influence protocol preferences, such as perferIPv6Addresses and perferIPv4Stack.

For IPv6 support in the Java-based Grid implementations, Java SDK 1.4.0 and above should be used. Java SDK 1.5 is recommended by us for the time being.

There is as yet no definition within Java for advanced API functions [56], e.g. writing a Flow Label field from a Java application. There needs to be activities within the Java community to investigate and specify advanced API functionalities where required, including handling of IPv4-mapped addresses. Before that happens, the Java-based Grid implementations could not benefit from this function of IPv6.

4.4.3. Associated applications required to be IPv6-enabled

The Grid implementations also utilise external applications. All network associated applications need to be IPv6-enabled as well. For example, in GT3, the Java DataBase Connectivity (JDBC), which is used for Reliable File Transfer (RFT), need to be IPv6-enabled. Postgresql, one of the JDBC implementations, is required to be version 7.4.0 or higher. As recommended by the Globus Implementation Group, Jakarta Tomcat is used as the web container for the Grid services on a Grid server. The container environment needs to provide IPv6 Web services for Grid services. Tomcat version 5 has been tested with full IPv6 capabilities. One of the libraries shipped with GT3, the Axis library from the Apache project [115], is also IPv4-only. It is used to produce URIs. We have reported it to the Apache project and helped modify it to rectify the problem.

4.4.4. Networking services that IPv6-enabled Grid requires

In order to run IPv6 tests over an IPv6-enabled network rather than only on local hosts, IPv6 support for networking is essential. It requires IPv6-enabled routers, which provide the auto-configuration of IPv6 addressing, forwarding and dynamic routing, and support from IPv6-enabled network services, such as IPv6 DNS, and Web services.

A number of major router manufacturers now provide basic IPv6 support and are beginning to provide more advanced support such as hardware forwarding, as we have introduced in Section 2.3.4. They may be used to construct a native IPv6 network and enable the IPv6 communication with other networks.

Support for IPv6 in the DNS provides hostname and IPv6 address resolution, which may be provided over IPv4 and/or IPv6 connections. It may not be necessary when a Grid implementation is operated in an IPv6-only network since most hosts can resolve hostname from the local host resolving data. But for communication in heterogeneous IPv4/IPv6 networks, it is essential that there is some hostname resolution to dynamically assign temporary IP addresses. This needs consideration when building an IPv6 environment within or around current global IPv4 networks.

According to the latest GGF specifications, Grid services are all Web services. Therefore, a web service container, which is used to contain Grid services, needs to be IPv6-enabled. Many web-server providers, such as Jakarta Tomcat, and IBM Websphere, have already added IPv6 capabilities to their products.

4.5. Integration of IPv6 into Grid middleware

The integration of IPv6 into Grid middleware starts with finding IP dependencies in the network protocols. The implementation of network APIs within Grid middleware may involve a few IP-dependent functions. We will introduce our methodologies in the following sections using Globus Toolkit as an example. A number of modifications need to be made for IP-dependent operation.

4.5.1. IP dependencies in the standards and specification of Grid middleware

In Section 2.3.5, we have introduced the common potential IP dependencies. There are many IP dependencies in the standards and specifications of Grid middleware. Their correspondent implementation should be modified. We have participated in the GGF IPv6 Work Group activity to examine all GGF specification for IP dependencies. We will introduce the GGF IPv6 documentation and Grid IPv6 standardisation status in Section 4.7. Of the GGF protocol documents that were found to contain IPv4

dependencies, about 60% of them failed to reference RFC 2732 [29] when mentioning URIs. A quarter contained some form of IPv4 biased textual explanations, while the remainder contained other minor dependencies.

There are many generic network protocols used in GT3, including HTTP, XML, LDAP, SOAP, TLS, and WSDL. In the GT3 implementation, they all operate over TCP. These network protocols are IP-independent. However, some implementations of them need to be modified. For example, in GT3 stand-alone web container implementation, the IP functions were invoked directly to generate a URL for HTTP; then, the HTTP implementation has to be modified in order to be able to handle a different IP.

In the GT3, GridFTP is an IP dependent protocol. It needs a modification in a way similar to FTP ("FTP extend for IPv6 & NATs" [4]). Correspondingly, the specific implementations of these protocols need modification as well. Within the Globus project, GridFTP is implemented in standard C. The original code was not well organised. It went too deep by directly invoking the IP layer functions from the application layers. Therefore, it was not IP-independent and was very difficult to port to be IP-independent. A new IP-independent network module known as globus_XIO has been developed for use by GridFTP. It has been released within GT4, and tested working over IPv6, as we will introduce in Section 4.8.

4.5.2. Methods of finding IP dependencies in Grid implementations

In order to find out exactly which lower-layer protocols and APIs were IP dependent, two approaches were taken: firstly, the 'top down' approach where we executed some upper layer Grid applications. Secondly, the 'bottom up' approach where we monitored all the network traffic between nodes and on the loopback interface.

In the 'top down' approach, we dealt with Grid components separately. We started from the central components, which were invoked by other Grid components, such as Globus stand-alone Web container in GT3. Then, we analysed the independent Grid components and their related protocols one by one, such as GRAM, GridFTP, and OGSA Service Browser.

In the 'bottom up approach, a number of network dump and analysis tools can be used,

such as tcpdump and ethereal. They provide a number of functions that help to capture and analyse network communications.

4.5.3. IPv6 relevant implementation issues

In this section, we will discuss these implementation-oriented issues in Grid middleware and their solutions. We aim to keep our discussion as generic as possible, and our porting efforts in GT3 implementation are also introduced as the concrete sample.

4.5.3.1. IPv6 APIs

The introduction of IPv6 requires changes to the APIs. There are currently two main programming platforms supporting IPv6, namely C and Java.

The new APIs and data structures for TCP/IP sockets (as used in the C programming language) are defined in the Basic Socket Extensions for IPv6 [61] and the Advanced Socket API for IPv6 [56].

These specify the socket address structures, address-conversion functions, socket options and name-resolution functions. The definitions include IP-independent functions, as well as those for IPv6-only applications. In the current state of IPv6 deployment, IP-independent applications are preferred so that they can operate in the presence of either or both protocols.

The IPv6 support by Java APIs has been introduced in Section 4.4.2.

It is easier to port a Grid implementation to be IP-independent if these network-relevant components are modular and well-isolated, and do not make assumptions about the IP version (e.g. representing an IP address by four integer values); the same principle should apply to new implementations. However, most Grid implementations have been developed by those who have no considerations for IPv6 or IP-independency. Therefore, the Grid implementations need to be examined and modified. For the new Grid implementations, the developer should limit themselves on these IP-independent APIs rather than go deep and invoke IPv4-only or IPv6-only APIs.

As we have introduced in Section 4.4.2, IPv6 support in Java is only available since Java SDK 1.4. In the early stages of our IPv6 GT3 experiments, we mostly used Java SDK 1.4.2, which has an IPv6 reverse look-up bug. It caused our IPv6 GRAM experiments to fail. In early 2004, Java SDK 1.5 was released with more comprehensive IPv6 support. It fixed the bug in Java SDK 1.4. Since then, we mostly used Java SDK 1.5.0 in our experiments. Recently, the Java SDK 1.6.0, which is still being developed, was also tested and used in our IPv6 Grid experiments.

4.5.3.2. Storage of IP addresses

In the C implementation, for the sockets API, there are data structures that may be used for IPv4 or IPv6 applications – sockaddr_in() and sockaddr_in6() – but also a generic IP independent structure sockaddr_storage() that hides the specific structure that the application is using. The latter should be preferred for IP-independent applications. For IP address storage we have in_addr (IPv4-only), in6_addr (IPv6-only), and addrinfo (IP dependent). Again, the latter is preferred. Any Grid implementation with the above IPv4-only or IPv6-only data structures/applications needs to be rewritten by using IP-independent data structures/applications.

In the Java implementation, the IP relevant java.net.InetAddress class has been upgraded to be IP-independent since Java SDK 1.4. It does not force any changes in the upper-layer applications. Therefore, there are minimum changes required in these Java-based Grid implementations. That is also the case for the Java-implemented GT3.

4.5.3.3. Representation of hosts and resolution

As described in Section 2.3.5, IPv4 and IPv6 have different literal representations; they are IP-dependent anyway. Any hard-coded IP literal addresses should be avoided in the Grid implementations. As we have described in Section 2.3.6.2, the IP-independent hostname should be used instead. Also the Grid implementations should not resolve a hostname and use the IP address for communicating with itself. It should leave the hostname to the lower-layer network services and resolve the hostname at run time, unless it has parallel IPv4-only and IPv6-only functions running. This may increase the complexity of the operation of Grid middleware by involving domain name resolution. However, this allows for a single Grid implementation to operate

both IPv4 and IPv6 environments.

There are differences in special addresses, e.g. the loopback/localhost address is 127.0.0.1 in IPv4 and ::1 in IPv6. Use of localhost by name abstracts that difference.

Regarding reverse DNS lookups, there is an ongoing transition at the time of writing from the ip6.int to the ip6.arpa namespace [44]. Some transitional address space (e.g. under the 6to4 prefix of 2002::/16) has no defined reverse lookup namespace.

The reverse DNS lookup is not required widely. It has been used in Grid security mechanism. The failure of reverse DNS lookup can result in Grid security check fails. Thus the reverse lookup of all addresses must be functional. In fact, in some experiment stage, reverse DNS lookup failure did cause Grid job failures, see Section 7.2.3.3 (IPv4-only Grid client submitted jobs to IPv6-only Grid server).

In the original GT3 implementation, when the Grid services were initiated, the local IPv4 address was found and then passed between GT3 components. Every network-relevant component was associated with the local IPv4 address. When communicating with other GT3 nodes, the IPv4 address was passed in the network payload.

Our modification indicates that "localhost" or particular hostname is used in both the Globus configuration file and Globus initial functions. Then, IP-independent functions (InetAddress.getByName in Java, getaddrinfo in standard C) are used wherever a hostname needs to be translated into an IP address. However, in most implementations, we try to avoid the translation from hostname to IP address. The address should only be resolved at run-time by the lower-layer network functions.

4.5.3.4. Format for Literal IPv6 Addresses in URLs

Both literal IPv6 addresses and syntax of Uniform Resource Locators (URL [7]) use ":" character as delimiters; they are incompatible. To use a literal IPv6 address in a URL, the literal address should be enclosed in "[" and "]" characters, for instance, http://[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:80/index.html.

There are a few places in GT3 that URLs and URIs are generated using String. It works only when the hostname and IPv4 literal addresses are used. But it will fail if literal IPv6 addresses are used. All URL and URI generating functions have been

modified to use java.net.URL and java.net.URI classes to generate URLs and URIs. These functions automatically detect literal IPv6 addresses and put them into the right format.

4.5.3.5. Conversion functions

The choice of preferred protocol, and address selection mechanisms, are defined in [45], by which a returned address list can be inspected to select addresses for source and destination addresses. The Grid implementations that have no consideration for IPv6, of course, have no consideration to the choice of IP protocol. However, in the modified Grid implementations that have both IPv6 and IPv4 available, the option of the choice of IP version should be possible.

In GT3, a few configuration options are available to allow the user to choose the start-up IP bindings. To use hostname instead of IP address, the user needs to set up the configuration option "publishHostName" be "true" in the globalConfiguration section of server-config.wsdd and client-server-config.wsdd. If there are other IP addresses that are associated with a host, such as a particular IPv6 hostname, the user needs to set up which hostname is used with the configuration option "logicalHost".

By default, the Java implementation gives higher priority to the IPv6 protocol on dual-stack machines. To operate in an IPv4-only network or set IPv4 to be higher priority, the Java system properties "perferIPv6Addresses" should be set as "false" or "perferIPv4Stack" set as "true".

4.5.3.6. Parsing and Displaying IP address

New code will be required to parse IPv6 addresses where entered as input or parameters. Such code will need to be aware of the IPv6 address formats, including conventions such as the '::' shortcut for multiple zero address sequences. However, it is not necessary. In such situations, FQDNs could be used.

Additionally, the different formats and lengths of IPv4 and IPv6 addresses might have a significant impact on the layout of addresses fields when presented in graphical user interfaces.

4.5.3.7. Hard-coded IP addresses

In many implementations, IP addresses are hard-coded in configuration files, even in programs, particular the unique IPv6 loopback address – 127.0.0.1. In order to be IP-independent, all hard-coded IPv4 addresses should be replaced by "localhost" or hostnames, which are provided by network functions. Then IP-independent functions should be used to look up the IP address when translation from hostname into IP address is needed.

In the early GT3 implementations (alpha version, 3.0.1 and 3.0.2), there are ten Java source files that include at least one IPv4 loopback address. They are replaced by "localhost".

4.5.3.8. Cooperating with Globus Alliance

We worked with the Globus implementation group in ANL [68] to include the IPv6 modifications in the official release. We have reported our IPv6 modifications through the Globus online Bugzilla [74]. We attached a list of these reported porting bugs as Appendix B. For GT3 version Alpha and 3.0.x, we provide modified binary library online to support IPv6 functions. Since GT 3.2, our IPv6 modifications have been included in the official GT3 code. We also provide an online guideline "How-to IPv6 in GT3" [102]. It is quoted as one of GT3's official technical documentation.

4.6. Grid communication in heterogeneous IPv4/IPv6 networks

As we have introduced in Section 2.3.6, there will be a period of IP transition. Thus, consideration must be given to an interim coexistence of IPv4 and IPv6 [16]. While Grid middleware take advantages of IPv6 features, we also give consideration to the Grid communication in heterogeneous IPv4/IPv6 networks.

Our effort to integrate IPv6 into Grid middleware takes an IP-protocol independent [54] approach, i.e. it supports both IPv4 and IPv6. The IP-independent server running on dual-stack is able to respond to client calls according to the IP family that the client

uses. In other word, the client decides which version of IP is to be used. In operation, an IP-independent Grid server on a dual-stack machine starts and listens on both its IPv4 and IPv6 interfaces. When an IPv4 client connects over IPv4, the Grid server uses its IPv4 interface to call back; only IPv4 communication takes place – similarly with IPv6. With dual-stack servers, the client can choose which IP family is the default or preferred. In order to run Grid services on the dual-stack server, the following fundamental network services need to be dual-stack as well: HTTP, FTP, DNS, SSL, routing etc.

For communication in heterogeneous IPv4/IPv6 networks, the environment is significantly more complex, as illustrated in the following figure.

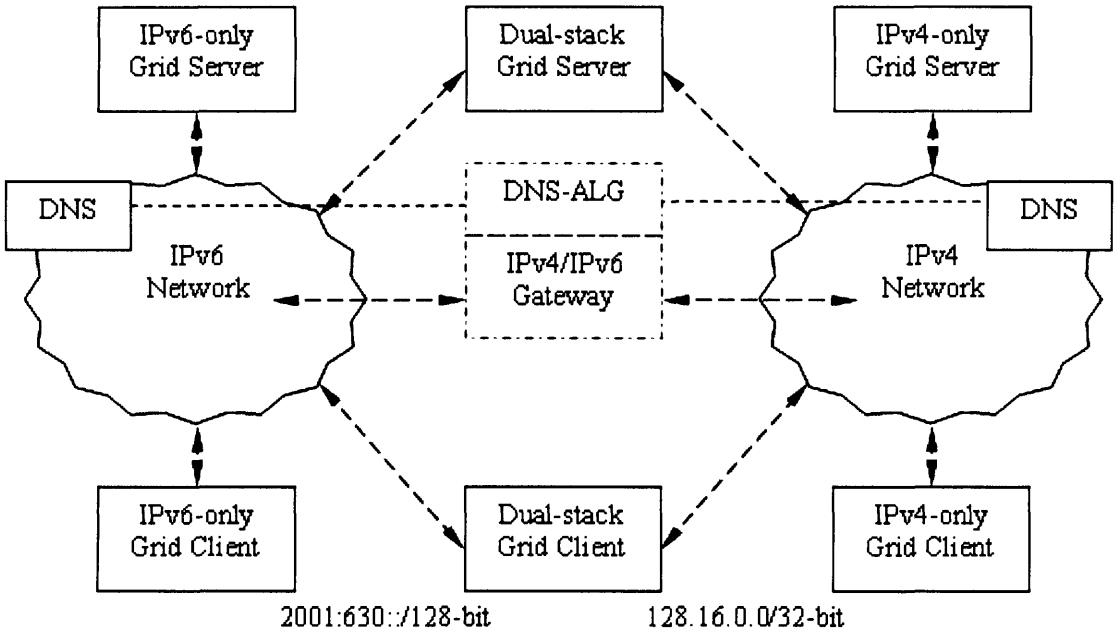


Figure 4-1: IP Transition in heterogeneous Grid networks

With the IP-independent Grid services running on the Dual-stack Grid server, an IPv4-only Grid client, IPv6-only Grid client and dual-stack Grid client can access it. Certainly, the IPv4-only server is accessible by an IPv4-only client, and the IPv6-only server is accessible by an IPv6-only client.

The situation becomes complicated when an IPv6-only client requires access to an IPv4-only server or an IPv4-only client requires access to an IPv6-only server. Therefore, IP-transition network services are required. A standard NAT-PT gateway is necessary to provide network level transition of Grid traffic. DNS-ALG is needed if

the temporary IP addresses need to be assigned dynamically rather than statically.

For the above scenarios to succeed, Grid middleware should use only hostnames in the content of the payload rather than any IP addresses. If any IP addresses are passed in the packets' content, it would lead to later failure if that IP address was used.

4.7. Grid IPv6 standardisation status

Since February 2003, when the GGF and the IPv6 Forum [86] announced a liaison relationship to drive New Generation Applications deployment worldwide, IPv6 has become relevant to Grid activities. Since then, an IPv6-Grid Working Group, GGF-IPv6-WG [92], has been set up in the GGF.

While many working groups of the IETF devote much of their efforts to the impact of IPv6, most do not consider the requirements of specific applications. The GGF-IPv6-WG is specifically tasked with considering the impact that IPv6 may have on Grid computing, with regards to development and implementation of standards and protocols. The GGF-IPv6-WG has been addressing two tasks: Firstly, a survey of the current GGF standards with respects to their IPv4 dependencies; secondly, a guide on how to develop and implement IP independent specifications. We have participated in and made contributions to both activities.

"IP version dependencies in GGF specifications" [50]. It has been published as Grid Forum Document 41 in December 2004. The report surveyed some 88 protocols for IPv4 dependencies. It concluded that about one third had such dependencies. Surveying for IP version dependence is also happening with the IETF.

"Guidelines for IP independence in GGF specifications" [63]. It has been published as Grid Forum Document 41 in December 2004. The document serves two functions. Firstly, it describes how to avoid IPv4 dependencies in GGF specifications. Secondly, it outlines new, IPv6-specific issues for Grid designers and implementers. The idea is that it should be used by all GGF WGs and as a checklist for document approval.

4.8. Sharing generic IPv6 porting experiences to other Grid implementations

Besides our participation in the GGF-IPv6-WG, we have devoted ourselves to help other Grid implementations to be IPv6-enabled with our experience.

The Globus Toolkit version 4 (GT4) is not a simple update of GT3; it applies the new WSRF standards. Many components in GT3 are totally re-implemented. GT4 has been designed with IPv6 in mind. It also benefits from our IPv6 porting on GT3. There are only a couple of IP-dependencies found in the GT4 implementation. At the time of writing, we have demonstrated successfully IPv6 functionalities for WSRF core and WS GRAM on the GT4 alpha version, also known as GT 3.9.4. We have also published an online guideline "How-to IPv6 in Globus Toolkit 4" [103].

The re-implemented GridFTP in GT4 is based on the Globus_XIO, which is an extensible input/output library for the Globus Toolkit. It provides a single POSIX-like API (open/close/read/write) that supports multiple wire protocols, with protocol implementations encapsulated as drivers. The XIO drivers distributed with 4.0 include TCP, UDP, file, HTTP, GSI, GSSAPI_FTP, TELNET and queuing. Although the IPv6 driver is still under test, it is also distributed with Globus_XIO. We have tested it working in a proper dual-stack environment.

We started to become involved when the JYDE system was still in the design stage. As a result of our advice and suggestions, the implementation of the JYDE system is IP-independent. We also investigated how to run the JXTA, which has been used in JYDE implementation, in the IPv6-enabled environment. Although JXTA already supports IPv6, some modifications have to be made in JXTA source code in order to successfully run JYDE system over IPv6.

4.9. Summary

The mechanisms of integrating Grid middleware and IPv6 together, introduced in this chapter, can be carried out successfully. The example of providing GT3 with IPv6

support was used to demonstrate these mechanisms.

Among many IPv6 advantages that Grid middleware can take after we have made them IPv6-enabled, we have highlighted mobility with Mobile IPv6. In our next step, we are going to provide the mobility support in Grid middleware by applying Mobile IPv6 underneath.

Chapter 5 Moving Grid middleware into the Mobile environment

5.1. Introduction

Up to now, most Grid research has focused only on fixed systems. However, some Grid nodes are not fixed. They may work in a wide-area and move around. During their movement, they may cross network boundaries and change their attachment points. They need the support from the lower-layer network infrastructure in order to remain network connection and their communication with other Grid nodes.

There are many benefits from providing such support. For example, in a large disaster relief zone, there may be no pre-established resource infrastructure or previous infrastructure may have been ruined. There would be many moving agents, such as rescue teams, working on their own. Lacking a fixed infrastructure, the individual mobile units may have very limited resources. If they can have mobility support with Grid middleware, they could access to remote resources, or they could cooperate with other units in the area – utilising the equipments as a single whole resource. With mobile-enabled Grid middleware, it is extremely easy to link together all equipment belonging to different organisations.

In this Chapter, we will discuss what is necessary to extend Grid middleware to be mobile-enabled. We will discuss the specific requirements of the mobile-enabled Grid middleware. In order to meet these specific requirements, we investigated and analysed existing mobility solutions, including link-layer mobility, application-layer mobility, and network-layer mobility. As a result, we chose Mobile IPv6 to provide mobility support in Grid middleware with the capability that link-layer mobility

technologies can also be used underlying it. Although we did not design or implement Mobile IPv6 ourselves, it is crucial that Mobile IPv6 be introduced into the mobile-enabled Grid middleware. With IPv6 auto-configuration functions, Mobile IPv6 solves most generic mobile access problems, such as allocating care-of address, stateless reconfiguration, home address tunnelling, avoiding triangular routing, etc. It provides automatic and transparent service for Grid middleware. The combination of Mobile IPv6 and Grid middleware is based on our previous efforts of integrating IPv6 and Grid middleware.

5.2. Requirements of the mobile-enabled Grid middleware

Grid middleware operates over a number of networking hosts. In the mobile-enabled Grid, some of these hosts need to move, to be mobile-enabled. Our investigation focuses firstly on what kind of mobility support the lower-layer network infrastructure should provide to mobile-enabled Grid nodes. It can be separated into two parts: generic mobile access issues; and Grid-specific mobility requirements.

As distinct from the normal nodes, the status of the mobile nodes may change frequently. When the mobile nodes move across network, their location, access point, attach network, bandwidth, networking neighbours, etc. are changing as well. Therefore, it requires some extra support from networking. We abstract these particular requirements as the following:

- To support a nomad's motions effectively, the mobile systems must be able to be reconfigured dynamically and automatically. Conversely, the networks and correspondent nodes may need to be changed as well;
- The networking neighbours of the mobile node may change. Correspondingly, the network connection may need to change or be reset;
- The mobile node should be accessible at its home address all the time, so that the remaining networks can access or communicate with it;
- There should be no forced changes for most of the networks, while the changes on

correspondent nodes may provide improved performance;

- All communication should be secure;
- The above-mentioned operations should be automatic and transparent to the upper-layer applications;

Further from above mobility requirements, mobile-enabled Grid devices have more requirements referring to Grid particularities.

- Grid middleware mainly works in IP networks. Therefore, the mobility solution in Grid middleware should allow mobile-enabled Grid nodes to communicate with other Grid nodes through IP. When a mobile node moves in a foreign network, it needs to be assigned a temporary IP address. It will be used for all connections to the IP networks;
- A Mobile-enabled Grid devices should be identified as the same, no matter to which network they are attached, so that their previous linked Grid nodes could return the job results;
- There should be minimal modification required to the existing Grid implementations.

5.3. Mobility solution for Grid middleware

Most existing mobility solutions have given enough considerations to generic mobile access issues. Therefore, in this section, we examine mobility solutions referring to the above-mentioned Grid-specific mobility requirements.

5.3.1. Link-layer mobility is not sufficient for the mobile-enabled Grid

As we introduced in Section 2.4.2, link-layer mobility is used mostly to provide wireless communication. In order to communicate with IP networks, the mobile devices need to remain the same access point. If the mobile devices that require IP communication change their access point, their temporary IP address must be re-assigned. To re-assigned the IP address automatically and transparently for moving

mobile devices, some support mechanisms must be provided in the network layer – the IP layer. Only link-layer mobility is not sufficient for IP-based applications.

Grid middleware is IP-based; it requires that moving Grid nodes maintain the same identities even when their temporary IP addresses changes. Therefore, some support mechanisms must be provided in the network layer or layers above it.

However, in most mobility Grid scenarios, link-layer mobility is useful to provide wireless support. It allows mobile-enabled Grid nodes to move around in certain ranges. It can always be used no matter what mobility support Grid middleware requires from those layers above it.

In some other mobility Grid scenarios, link-layer mobility may not be necessary. For instance, a mobile-enabled Grid node may just be unplugged in one network and plugged in another network using cables.

5.3.2. Application-layer mobility is not suitable to Grid middleware

If there is some support mechanism in the network layer that can solve the temporary IP assignment issue, such as mobility support using DHCP, the application-layer mobility, introduced in Section 2.4.1, may be used to maintain the same identity requirement of Grid middleware.

In order to fulfil application-layer mobility in Grid middleware, a central registration service is required to be implemented and operated. Within this solution, all Grid nodes should register their identities (it can be hostname) and IP addresses to this central registration service. When a mobile-enabled Grid node changes its IP address, it reports the new IP address with its identity to the centre registration service. Afterwards, Grid middleware uses the new IP address to communicate with the mobile-enabled Grid node.

However, there are two major problems in the use of application-layer mobility in Grid middleware. Firstly, the above modification requires considerable code. There is actually another mobility solution, which does not require modification in the Grid implementations. We will introduce this mobility solution in the next section.

The second problem is vital: a central registry is against the concept of Grid

middleware, which is to coordinate resources without any centralised control. On the other hand, a central registration service is the core of the application-layer mobility solution. The solution of the next section avoids both problems; it is much better suited to Grid middleware than application-layer mobility.

5.3.3. Using network-layer mobility in Grid middleware

5.3.3.1. Mobility support using DHCP cannot remain the host identity

Mobility support using DHCP aims to solve the temporary IP address assignment issue. It allows mobile hosts to move across IP networks transparently. However, more than this, Grid middleware requires to maintain the identity of mobile-enabled Grid hosts. Therefore, mobility support using DHCP is simply not sufficient.

5.3.3.2. Mobile IP meets the requirements of the mobile-enabled Grid

Mobile IP is not the only mobility solution, but it is the most suitable mobility solution for Grid middleware. It provides a direct connection between the mobile device and the IP-based internet. It is completely transparent for all layers above IP, e.g. TCP, UDP and of course for all applications, including Grid middleware.

One of the main benefits of Mobile IP is that there is no forced change to the upper-layer application, in our case the latter are the Grid implementations. Mobile IP provides mobility support from the IP layer, which is lower layer to Grid middleware. It separates Grid middleware from the mobility operations. As long as the Grid implementations can communicate over IP, which they are, they can potentially be operated over Mobile IP.

Another main benefit is that Mobile IP keeps the identities of mobile hosts. When mobile hosts attach in a foreign network, it has its temporary IP address to communicate with IP hosts. Meanwhile, it is also accessible by its home IP address. Mobile hosts are identified as the same as they were home network.

5.4.Providing Mobility Support in the Grid middleware from the Lower-layer Network Infrastructure

Since the IP is in the process of migrating from IPv4 to IPv6, correspondingly, there are two Mobile IP protocols – Mobile IPv4 and Mobile IPv6. Both of them can meet Grid-specific requirements. As we reviewed in Section 2.4.3.1, Mobile IPv6 is more advanced. Therefore, we choose Mobile IPv6 as our mobility solution for Grid middleware.

5.4. Providing Mobility Support in Grid middleware from the Lower-layer Network Infrastructure

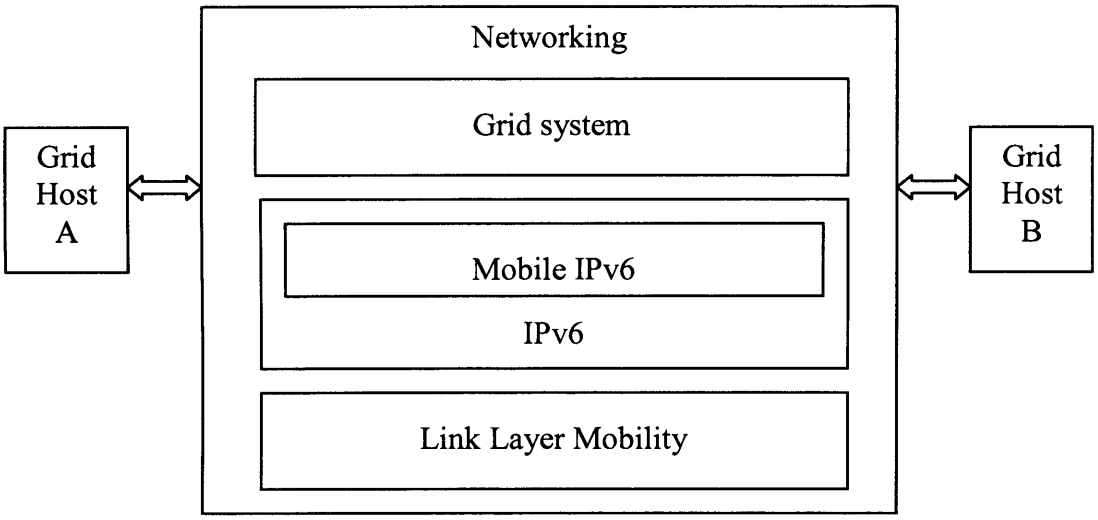


Figure 5-1: Lower-layer network infrastructure provide mobility support in Grid middleware

As a result of our analysis in previous section, we have the mobility support in Grid middleware provided from the lower-layer network infrastructure, shown in Figure 5-1.

5.4.1. IPv6-enabled is the prerequisite of Mobile IPv6

Mobile IPv6 is accessible using general IPv6 APIs, appearing transparent to the application layer. Thus, in an IPv6 implementation, there is potential support for roaming across networks, with global notification when you leave one network and enter another. In order to operate the Grid over Mobile IPv6, they should firstly be

IPv6-enabled. In Chapter 4, we have introduced how to make a Grid implementation IPv6-enabled.

5.4.2. Operating the Grid over Mobile IPv6

Mobile IPv6 meets most of the requirements that we have raised in Section 5.2. In this section, we will describe how to operate Grid middleware over Mobile IPv6.

Firstly, the mobile-enabled Grid host needs to be IPv6-enabled and it needs to have Mobile IPv6 function loaded. This host should use address auto-configuration mechanism, rather than manual address configuration. The address auto-configuration mechanism enables the host to configure its IPv6 address automatically according to its own identity and the IPv6 router advertisement. It may not be necessary when the host only remains in one network. However, it will become a critical mechanism to allow mobile node to get its care-of address(es) automatically, when it moves across the networks. The Mobile IPv6 function on the mobile-enabled Grid nodes should be set up as a Mobile Node model.

Secondly, a Home Agent needs to be set up. It should have Home Agent model of Mobile IPv6 function operated on it. Mobile-enabled Grid nodes should also have their Home Agent IPv6 address in their configuration and be registered on their corresponding Home Agent.

Thirdly, optionally, the fixed Grid nodes can be operated as Mobile IPv6 correspondent nodes. It can avoid the triangle-routing between these nodes and the mobile-enabled Grid nodes.

5.5. Summary

The mobility support from Mobile IPv6 provides the lower-layer network infrastructure for Grid middleware. It does not require any changes in the Grid implementations. All Grid middleware need to do is to operate the mobile-enabled Grid node(s) over Mobile IPv6 with the support from Home Agent(s). It provides the transparent Grid services to the mobile Grid users. It hides the operation details of

middleware and network layers away from the mobile Grid users.

However, further research is needed in the mobile-enabled Grid middleware in order to improve their performance. We will discuss this in the next Chapter and try to provide generic solutions, which can serve most Grid mobility scenarios.

Chapter 6 Support for Grid Resource Mobility

6.1. Introduction

Running Grid middleware over Mobile IPv6 successfully impacts only the lower network infrastructure. It solves only transparent access, handover, and consistent identity in a mobile environment. Although there is no need to change the implementation of the Grid software itself, some changes will make Grid middleware works more reliably and effectively in the mobile environment. In this chapter, we are going to analyse the effects that mobility brings into Grid middleware and how Grid middleware should be changed to respond to these effects.

Our research has led us to focus on providing Grid resource mobility in the mobile-enabled Grid. It requires a few specific functions. These requirements motivate us to introduce a dynamic Grid resource discovery mechanism. In order to be able to choose Grid resource dynamically, the capability of Grid servers and Grid client needs to be considered. Besides them, both loading of Grid servers and the network connections between Grid servers need to be monitored and measured as well. We analyse in depth these four aspects and their relevant parameters – though we cannot validate all of them in our experiments. Several functions are introduced in order to assign Grid jobs dynamically according to the above parameters. With these functions in operation, the mobile-enabled Grid client can choose the optimum mix of Grid devices as its Grid resources. These functions are necessary for the mobile-enabled Grid although the fixed Grid can benefit from them. In addition, these functions also enable mobile Grid devices to be Grid resources. In our discussion here, we use eProtein application as a

concrete example of upper-layer applications.

6.2. Grid resource mobility and its requirements

Unlike in normal fixed networks, the status of the mobile-enabled Grid client always changes dynamically. When the mobile-enabled Grid device moves, its location, access point, IP address, bandwidth and network neighbours all change. While the mobile-enabled Grid device seems to keep the same network connectivity with its correspondent nodes, these network connections have been reset in reality, it is just that the loss of connectivity is sufficiently brief that it is not apparent at the application level. Mobility influences the upper-layer Grid middleware as well. Although Grid middleware may not need to be aware of how the lower-layer mobile infrastructure changes, some modifications in them can make them work more effectively in the mobile environment.

Our investigation shows that a fixed Grid resource assignment is not suitable in the mobile-enabled Grid. For example, a mobile-enabled Grid node may move into a new access network, which had no connections to the network to which it was previously linked; this would be impossible with a fixed assignment of Grid resource. In this case, the mobile-enabled Grid client receives no service, even some other Grid devices and their services are available. There is another more common problem scenario. Once the mobile-enabled Grid node moves on, its network topology changes. Although it is able to use the constant assigned Grid resource, there are other Grid devices, which can provide better Grid services with their more powerful hardware and better network connections with the mobile-enabled Grid node.

Therefore, the Grid resource should be able to move along with the mobile-enabled client. We call this Grid resource mobility.

In order to fulfil Grid resource mobility in our research, we break down its specific requirements as described in the following:

- The mobile-enabled Grid nodes need to be able to discover what Grid resources are available dynamically. When the mobile-enabled Grid nodes move across

networks, some new Grid devices may appear and some previous Grid devices may disappear. It is essential that the mobile-enabled Grid nodes dynamically find these change and adapt to them;

- Conversely, the fixed Grid nodes also need to be able to discover what Grid resources are available dynamically as some mobile-enabled Grid resources may appear or disappear;

- A Grid client, either mobile-enabled or the fixed, should be able to compare its discovered Grid servers, and assign its Grid jobs with some proportion, which should be decided according to the dynamic information associated with each Grid server;

- In order to provide such comparable information, four aspects of Grid servers should be parameterised: the capability of Grid servers, loading of Grid servers, the network connections between Grid services and the mobile-enabled Grid client, and the capability of the mobile-enabled Grid client;

- Since the mobile systems change frequently, the above information should be measured dynamically;

- The above-mentioned measurement of network connections should not force Grid implementations to become too deeply involved in the lower-layer network infrastructure;

- Job assignment depends on the characters of specific Grid jobs; therefore, a Grid job should be also characterised and parameterised;

- All the above operations in the mobile-enabled Grid should be automatic and transparent to end-users.

According to these requirements, more functions will be introduced in the mobile-enabled Grid.

6.3. Dynamic Grid resource discovery mechanism

In order to meet the requirements of discovering dynamically the availability of Grid resources, a dynamic Grid resource discovery mechanism is introduced into the mobile-enabled Grid. The dynamic Grid resource discovery mechanism uses a broadcast mechanism and "request and respond" model to discover dynamically the existence and their status of Grid devices.

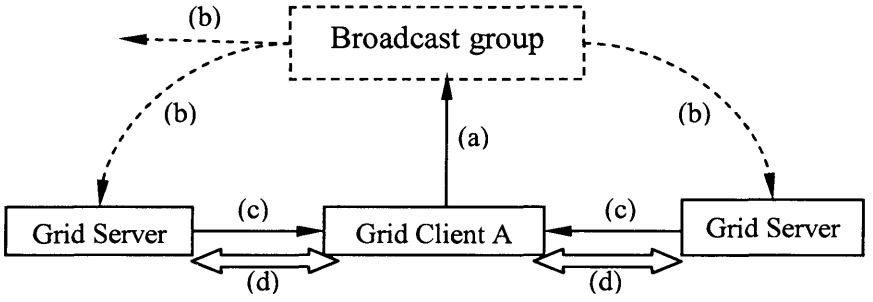


Figure 6-1: Schematic of Dynamical Resource Discovery

As shown in Figure 6-1:

a). Grid client A, which needs to discover Grid devices, sends out a broadcast discovery request with its address to a certain broadcast group by multicast, it requires all Grid servers, which provide the request Grid services, to respond;

b). Broadcast packets are sent to all nodes in the Time-To-Live range of the broadcast group;

c). After Grid servers receive the discovery request packets, they respond and report themselves to the source – Grid client A; some of broadcast packets, which are received by none-Grid node or Grid servers without the required services, are just dropped silently;

d). Afterwards, the connections between Grid client A and Grid servers can be built, update information will exchange between them periodically. The broadcast discovery request is sent periodically in order to monitor the dynamic change of Grid servers.

As we have described in the previous section, both mobile-enabled Grid nodes and fixed Grid nodes need to be able to discover Grid resources dynamically, while the network neighbours in mobile-enabled Grid are changing dynamically. Therefore, the dynamic Grid resources discovery function needs to be operated on all Grid devices, both the mobile-enabled and the fixed.

The dynamic Grid resource discovery mechanism may not be an essential component of the fixed Grid though it can improve the performance. However, it is necessary in the mobile-enabled Grid because there are many more changes compared to the fixed Grid. In the fixed Grid, Grid clients can be bound to certain Grid servers because the changes of these nodes are small and the connections between these nodes are stable. Within the mobile-enabled Grid, it is very unreliable if the mobile-enabled Grid nodes are bound to other Grid nodes. As we have introduced in previous section, in such scenarios, the Grid jobs may be broken by the movement of mobile-enabled Grid node.

6.4. Parameterising the Grid and network connections

The dynamic Grid resource discovery mechanism meets some of the requirements of Grid resource mobility. It provides the essential function for Grid resource mobility. It builds the connections between the mobile-enabled Grid client and the available Grid servers. However, the existence and availability information is not sufficient yet. In order to provide the right information for the mobile-enabled Grid client, we need to extend dynamic Grid resource discovery considerations further.

Four aspects of a Grid middleware should be parameterised: the capability of Grid servers, loading of Grid servers, the network connections between Grid services and the mobile-enabled Grid client, and the capability of the mobile-enabled Grid client. These aspects need to be monitored and parameterised into information, to which a Grid can refer. In each aspect, there are a number of parameters can be used and make affects in different Grid scenarios. We are going to describe and examine them in this section one by one.

6.4.1. Capability of Grid servers

The capability of Grid servers can be described by different parameters according to different requirements of different Grid services. Therefore, we list the following parameters and their affected Grid scenarios.

- **Number of processors**: how many processors a Grid service has or manages. This is the most useful parameter for computational Grid services. It decides how many Grid jobs a server can execute parallel. Clearly, the more processors a Grid server has, the more Grid jobs it can handle.

- **Power of processor**: how fast a processor can execute a Grid job. This is another important parameter for computational Grid services. Clearly, the more powerful a processor is, the quicker it completes a Grid job.

- **Data storage capability**: this can be separated into several parameters: the size of the data storage, the speed of input/output, etc. These parameters are useful for data Grid services. Although most of computational Grid services also require some amount of data and data storage, the amount is normally relatively small and far below the capability of data storage.

- **Bandwidth**: the amount of data that can pass through a network interface over time. This is particular useful for data Grid services or other Grid services that require large-scale data transfer. However, for most of fixed Grid devices, bandwidth is not the bottleneck. It may need to be monitored when using the mobile-enabled Grid devices.

Most of above-mentioned parameters are constant or rarely changed once a Grid server has been formed. They can be sent to a Grid client as the initial information. However, our research does take into account the change of these parameters, such as a certain processor going down. Therefore, these parameters are also measured dynamically on Grid servers. They are reported to Grid clients as the update information.

6.4.2. Loading of Grid server

The parameters of loading of Grid servers reflect their usage. They represent how much Grid resource is currently available. They change from time to time. Hence,

they must be dynamically measured and updated.

- **Processor loading**: describes the available computational resources. It is important parameter for computational Grid services. However, the loading parameter of certain processor does not reflect the usage of a multiple-processor Grid server. For a multiple-processor Grid server, the loading parameter of all processors must be collected.

- **Queue state**: is an indirect parameter of processor loading. On a multiple-processor Grid server, managed by a cluster-scale scheduler, the queue state of the cluster-scale scheduler can reflect the loading of all processors of Grid server.

- **Available data space**: how much data space is available for use. This parameter is the most useful for data Grid services.

These above-mentioned parameters keep changing during the usage of Grid servers. Therefore, they must be measured in real time, and reported to Grid clients as the update information.

6.4.3. Network connections between Grid servers and client

The network connections between them and the mobile-enabled Grid client can affect the performance of Grid resource, particularly in mobile-enabled Grid scenarios. Therefore, we give considerations to the following network parameters:

- **Response time**: using control message to test how fast the connections are. This describes the network delay between two nodes. In mobile-enabled Grid, the network delay between mobile-enabled Grid nodes and other Grid servers can be very variable. The network delay is one of most probable causes for Grid jobs failure. The response time parameter is useful for most Grid services.

- **Hop count**: how many hops a packet has passed through from the source node to the destination node. It describes the network distance between two nodes. If the capabilities of Grid servers are the same, the closer Grid servers provide better performance.

- **Packet loss**: packet loss can cause failure of Grid services. This consideration is

necessary if a Grid service is based on UDP. However, most of the current Grid services are based on TCP. TCP can recover from a certain amount of packet loss. Within the TCP recoverable range, in most Grid scenarios, the packet loss only causes network traffic delay.

- **Stability**: how stable the network connections are. The average and maximum value of connection response time for a period can reflect this. Apparently, Grid client should choose Grid servers that have more stable network connection.

A suitable time to measure the network connection is when the node receives the update information from the Grid servers that it has discovered. The update information confirms the existence of source Grid server. Then, the network connection measurement adds more parameters into the information data that describes the source Grid server.

6.4.4. Capability of the Grid client

The capability of the Grid client restricts the maximum number of parallel Grid job submissions. In some Grid scenarios, the client itself is also used as Grid resource. In particular, when the mobile-enabled Grid client has no network connection in some movement stage, it has to use itself or its managed processors, if there is any, for all Grid jobs.

- **Maximum number of parallel Grid job submissions**: the maximum number of Grid jobs the Grid client can parallel handle. This parameter is mostly decided by the capability of the client and certain Grid implementation. The number may reduce when the CPU and memories of the client are occupied by applications, such as user software, local executed Grid jobs, etc. Also the bandwidth could be the bottleneck and reduce the number.

- **Capability parameters of Grid client**. The processor and the amount of memory affect the maximum number of parallel Grid job submissions. When the client uses itself or its managed processors as Grid resource, it actually treats itself as a Grid server. It needs the similar parameters of capability of Grid servers, such as the number of processors and the power of processor. However, in this scenario, data storage capability and bandwidth parameters are not needed since there is no

data transmission or network traffic.

- **Loading of Grid client**. The loading of the Grid client can affect the maximum number of parallel Grid job submissions as well. When the client uses itself or its managed processors as Grid resources, the loading of Grid client also affects how many Grid jobs and how fast the Grid client can handle them.

In most Grid scenarios, only the maximum number of parallel Grid job submissions needs to be separated among these above-mentioned parameters. The other parameters can be measured and treated as server parameters because the Grid client may also be a Grid server for another Grid client at the same time.

6.5. Dynamic Grid jobs assignment

All the parameters that we have introduced in the previous section provide the comparable information for a Grid client. They are used to assign dynamically Grid jobs among Grid servers.

6.5.1. Characterising a Grid job

Obviously, job assignment depends also on the characteristics of the specific Grid jobs. A Grid job is characterised by the amount of computation required by each job (COM), the amount of data (DAT) that must be stored and the amount of data that must be transferred (TRAN). In general, we have a function with all three factors of the Grid job as the following:

$$JOB=F_j(COM, DAT, TRAN)$$

Again it depends on the specific computation, and on the way that it is organised, whether each job uses data which must be transmitted with it, whether all the data is stored at each site, or whether data needs to be fetched on demand from another site. The parameters associated with each job may remain constant throughout the computation, or may vary as a result of the computations already carried out.

6.5.2. Maximum number of job submissions for each Grid server

6.5. Dynamic Grid job assignment

With both the information of Grid jobs and Grid servers, we can now decide the maximum number of job submissions for each Grid server. In general, we can construct the following function:

$$j_n=F_s\{JOB; (C_n, L_n, N_n)\}$$

In above formula, j_n is the number of jobs that can be submitted to Grid Server$_n$; JOB is the synthesis of the parameters associated with each job; C_n is the synthesis of the parameters associated with the capability of the Grid server; L_n is the synthesis of the parameters associated with the loading of the Grid server; N_n is the synthesis of the parameters associated with the network connection between the Grid client and the Grid server. We assume that at the time of decision make, these parameters are known to the job assignment function.

The job assignment function (F_s) can be very complicated and different for different applications. It should be constructed to include the most influential parameters in the specific Grid scenarios. It can be simplified according to the specific application and its requirements.

In this section, as an example, we will construct a concrete job assignment function for the particular case of eProtein. It will be used and validated in our experiments in Chapter 7.

In the eProtein application, a large amount of preparation data, a database of around 400 MB, needs to be transmitted to the Grid servers. We have distributed the database on all our servers in advance. Therefore, we do not need to be concerned with data transmission in the job assignment function. The dynamic data is very small, only around 300 Bytes/job. We do not need to be concerned with this in the function. The computational requirements are large here. It is a much more important concern. Typically, a real eProtein job can be cut into several hundreds, even over one thousand sub jobs. It takes a 1 GHz processor around 200 seconds to complete a single sub job. Therefore, in this case, the most important parameters are the "number of processors" (p_n) and the loading of processors. In order to reflect the loading of all the processors within a single Grid resource, we use "queue state" (q_n) in our function. Network connections are initially measured by using the combination (n_n) of parameter

"response time" and parameter "packet loss rate". Now, we have a concrete job assignment function as below:

$$j_n = p_n * q_n * n_n$$

Each parameter has its specific adjustment. For "number of processors", we define 1 GHz processor as standard, and any processor greater than 1 GHz is counted as 1, 0.6 GHz processor is only counted as 0.6. For "queue state", it has the maximum value 1. It means we use the full capability if there is no loading. It is adjusted using the formula $(N_{processor} - N_{job}) / N_{processor}$. $N_{processor}$ is the number of processors. N_{job} is the number of current jobs in queue. For "response time", it also has the maximum value 1. It means we use the full capability if the connection is the best. It is adjusted using the formula $(T_{maxusable} - T_{response})/T_{maxusable}$. $T_{response}$ is the actual response time. $T_{maxusable}$ is the maximum usable response time, which shows the correspondent server can be used. It is also the unusable condition, which we will discuss in next section. Any servers having longer response time than $T_{maxusable}$ should not be used. The similar adjustment is applied to "packet loss rate". It has the maximum value 1. It is adjusted using the formula $1 - (L_{loss}/L_{maxusable})$. L_{loss} is the actually packet loss rate. $L_{maxusable}$ is the maximum usable lost rate. When L_{loss} equals or even greater than $L_{maxusable}$, we reach the unusable condition.

In the mobile-enabled Grid, parameters associated with the network connections could become the bottleneck and the most influential. For example, if the mobile Grid client has attached to a low qualify access point and has bad connection to the fixed Grid servers. The job assignment function should reduce the Grid jobs assigned to the fixed Grid servers, and use the local Grid resources, or mobile Grid resources with better connections.

6.5.3. Conditions for Grid servers to become unusable

In general, we discuss the common extreme scenarios for all application – unusable conditions in this section. In these scenarios, if specific Grid servers are restricted by certain bottlenecks, the job assignment function should not use them at all. The unusable conditions are also relevant to different applications. Therefore, we discuss unusable scenarios referring to different categories of Grid jobs.

For data Grid jobs or computational Grid jobs that require a large amount of data, if the available data storage space on a certain Grid server is less than what has been required, this Grid server should be removed from the usable resource list, no matter how much computational resource it has, till some more data storage space is released.

For computational Grid jobs, if the loading on a certain Grid server is too high, for instance, jobs in the queue are already more than the number of processors, this Grid server should be removed from the usable resource list, till some jobs are completed and processors are released.

For most Grid jobs, if the network connection with a certain Grid server is too bad, for instance, response time is longer than 400 milliseconds, or the packet loss rate is greater than 20%, this Grid server should be removed from the usable resource list, till the network connection become better.

In the specific case of eProtein, we have three unusable conditions: one, the number of jobs in queue is more than the number of processors; two, response time is longer than 400 ms; and the packet loss rate is greater than 20%.

6.5.4. Proportion of submission among Grid servers

Based on the formula in the previous section, the sum of j_n is the total number the client can submit to these discovered Grid servers, J_s:

$$J_s = \Sigma j_n$$

However, the above formula considers only the server side of a Grid. On the other side, restricted by the hardware or network connection, the Grid client may only be able to handle J_c parallel Grid job submissions:

$$J_c = F_c\{JOB; (C_c, L_c, N_c)\}$$

In most scenarios, J_c depends on only the bottleneck parameter. For example, if the job submission needs a lot of memory space on the client devices, J_c may be decided only by how much memory space the client devices have; if the job submission needs a lot of data transmission and the client has limited bandwidth or bad network connections, J_c may be decided only by the data transmission rate; if the client devices

have loaded much computation on itself, J_c may be decided only by how much computation power are available.

In the specific case of eProtein, J_c mainly depends also on the memory spaces because each parallel job submission through GT3 needs around 30 MB memory spaces.

The actual maximum number of job submissions depends on the smallest value of the above two variables (J_c, J_s) and the number of Grid jobs (J_j):

$$J=\min\{J_c, J_s, J_j\}$$

If J_s is the minimum variable, all Grid servers will be submitted the maximum number j_n of Grid jobs. The rest of Grid jobs queue on the Grid client and will be submitted in the later cycles. The submission proportion is not needed.

If either J_c or J_j is the minimum variable, Grid jobs should be assigned according to the following formula:

$$J_{sn}=\min\{J_c, J_j\}*(j_n/J_s)$$

J_{sn} is the actual number of Grid jobs that are submitted to Grid $server_n$.

If $J_j > J_c$, the rest of Grid jobs queue on the Grid client and will be submitted in the later cycles.

6.5.5. Job recovery and re-assignment

In mobile situations, there are some inevitable breakdowns in communication. They can cause job submission failure or job result transmission failure. In both cases, the assigned Grid jobs fail. Our solution is to put these failed Grid jobs back into the queue on the Grid client. They will be re-assigned according to the dynamic situations. If the network connection keeps down, the mobile Grid device can eventually execute all Grid job using its own resource with maybe much longer time.

6.6. Transparency to the end users

Our Grid resource mobility mechanism provides the automatic and transparent service to the Grid end-users travelling with mobile-enabled Grid devices. The mobile-enabled Grid automatically discovers the available Grid resources, no matter where they are. The system can also assign the user Grid jobs intelligently according to dynamic information we have introduced above.

There is no need for end-users to know how the mobile-enabled Grid infrastructure works. The end-users could even assume all jobs are executed locally, but with a much powerful device, which can move along with them.

However, it may be possible for the Grid end users to access and affect the mobile-enabled Grid infrastructure. For example, our mobile-enabled Grid provides Grid end-users a configurable option, in which Grid end-user can decide the value of disqualified network connection. It overwrites the default value in the mobile-enabled Grid infrastructure.

6.7. Summary

Based on the effort in the previous chapter, we have considered how to provide Grid resource mobility in this chapter. We have introduced a dynamic resource discovery mechanism and examined a few relevant parameters. A framework has been introduced in order to fulfil dynamic job assignment. Although we have used eProtein application as our concrete example, our Grid resource mobility approach is generic and can be carried out in any mobile-enabled Grid, though the implementation may be variable.

Chapter 7 Experimentation and Validation

7.1. Introduction

In this chapter, we will validate the approach and solutions that we introduced in the previous chapters. For networking-relevant experiments, one of the most important things is that they are run in the correct environments. In our opinions, simulation environments may omit some potential influential aspects, comparing to the real system. In order to build a simulation environment, only important characters/parameters are included. It may able to show or emphasize certain aspects. But some potential characters/parameters, which may become important in certain circumstances, are ignored. On the other hand, the efforts on the testbed can be extended into the real usage easily, while pure simulation is difficult to transfer to real usage. Therefore, all our experiments have been carried out on real systems rather than in the simulation environment.

Therefore, we will introduce also the building of our testbeds for each experimental purpose. We try to make our testbeds as generic as possible so as to include most scenarios in our experiments. Our work in Chapter 4, 5 and 6 is generic to network-relevant services. However, we have limited ourselves to a Grid environment in our experiments. The following three sections describe the experiments corresponding to Chapters 4, 5 and 6. These experiments also validate our contributions.

7.2. Grid IPv6 experiments

Our experiments to make Grid implementations IPv6-enabled were carried out on our Grid IPv6 testbed, which was introduced in Section 7.2.1. In Section 7.2.2, we

introduced our efforts to port GT3 implementation. We validated our IPv6-enabled Grid infrastructure by successfully running Grid applications over it, as introduced in Section 7.2.3. We also ran the Grid implementation in a heterogeneous IP environment, as introduced in Section 7.2.3.3.

7.2.1. Building the IPv6 environment for Grid middleware

7.2.1.1. IPv6-enabled hosts

For the time being, we restrict ourselves to the LINUX/PC platform since GT3 only works fully on these systems. When we started our experiments in the middle of 2002, we set up an IPv6-enabled Grid testbed, which included one node running Linux Red Hat 8 and two nodes running Linux Red Hat 7.3 with the re-compiled IPv6-enabled kernel. The Red Hat 7.3 nodes were updated to Linux Red Hat 8 later. Our IPv6-enabled Grid testbed now consists of ten nodes running Linux Red Hat 8 and two nodes running Linux Fedora Core 1.

7.2.1.2. IPv6-capable application APIs

The GT3 stand-alone web service container is our default testing web container. Making it work over IPv6 is part of modifying GT3, which we will introduce in the next section.

We also use the Apache Jakarta Tomcat, the recommendation of the Globus implementation group, as our GT3 OGSA service container. Tomcat 4.x works well for most IPv6 functions. However, there is a software bug that Tomcat 4.x cannot respond correctly with literal IPv6 address calls. It has been identified as the well-known IPv6 address handling problem, which adds square bracket to a literal IPv6 address when using an IPv6 address in the URL. The Tomcat implementation group provides IPv6 support more fully on Tomcat 5.

7.2.1.3. Native IPv6 networks

An IPv6 router is an essential component in a generic IPv6 network. It connects the IPv6 networks together. It also sends out Router Advertisement that allows the stateless address auto-configuration on IPv6-enabled hosts.

Since we did our experiments in the Mice-net network (the main experimental network in the Computer Science Department of University College London), we used the main router of the department – a Cisco 7206 (7206VXR (NPE400) processor). It had the IPv6 prefix 2001:630:13:101::/64. It also performed IPv6 multicast forwarding. We also used other devices for IPv6 routing: a FreeBSD and a Linux machine. Both of them had 2 interfaces and were used to build two extra test networks for the mobility scenarios, see Section 7.3.1.

We used the DNS server in CS/UCL department for IPv6 look-up. However, we also ran our own DNS with the Berkeley Internet Name Domain (BIND) software in order to obtain specific hostname resolution in specific scenarios, such as resolving temporary IPv6 addresses for IPv4-only hosts.

7.2.2. Modifying GT3 implementation to be IP-independent

GT3 is mainly written in Java though GridFTP and its underlying communication module globus_io are written in standard C. We focused mostly on the Java part of GT3, since GridFTP was going to be re-implemented using Globus_XIO. Within the Java part, the following items have been examined and modified:

- The relevant network protocols and their IP-dependencies
- Functions that generate IP addresses
- Functions that generate URLs and URIs with IP addresses
- Hard-coded IP addresses
- Externally linked IPv4-only libraries

The details of the modifications have been reported to the Globus implementation group through Globus online Bugzilla. The list of all reported porting bugs is attached as Appendix B.

7.2.3. Validation of IPv6-enabled GT3

As it is middleware, a Grid infrastructure can be validated only by the usage of the upper-layer Grid applications. A few upper-layer services have been run over IPv6-

enabled Grid middleware. The success has confirmed the adaptation between IPv6 and the upper-layer applications.

7.2.3.1. IPv6 test scenarios and porting stages

The GT3 IPv6-enabled tests and porting started from an IPv4-only test on dual-stack hosts. This might differ from the IPv4-only environment. Then, IPv6-only porting was carried out with minimal modifications. During this stage, as much as possible, IP-independent functions and data structures had been used instead of IPv4-only or IPv6-only ones. The situation became much more complicated for IPv4/IPv6 dual-stack environments. In the dual-stack environment, parallel independent support for both IPv4 and IPv6 must be provided. The Grid server starts with the IP-independent hostname and responds to the client calls according to the IP family that the user uses. After the modifications, we ran IPv4-only tests as well, since most Globus users are obviously still IPv4.

7.2.3.2. IPv6-enabled GT3 service container and shipped applications

After our modifications, we are able to configure GT3 stand-alone service container to start with literal IPv6 address, as shown in Figure 7-1. In the command lines, we can see all the Grid services bound with IPv6 address.

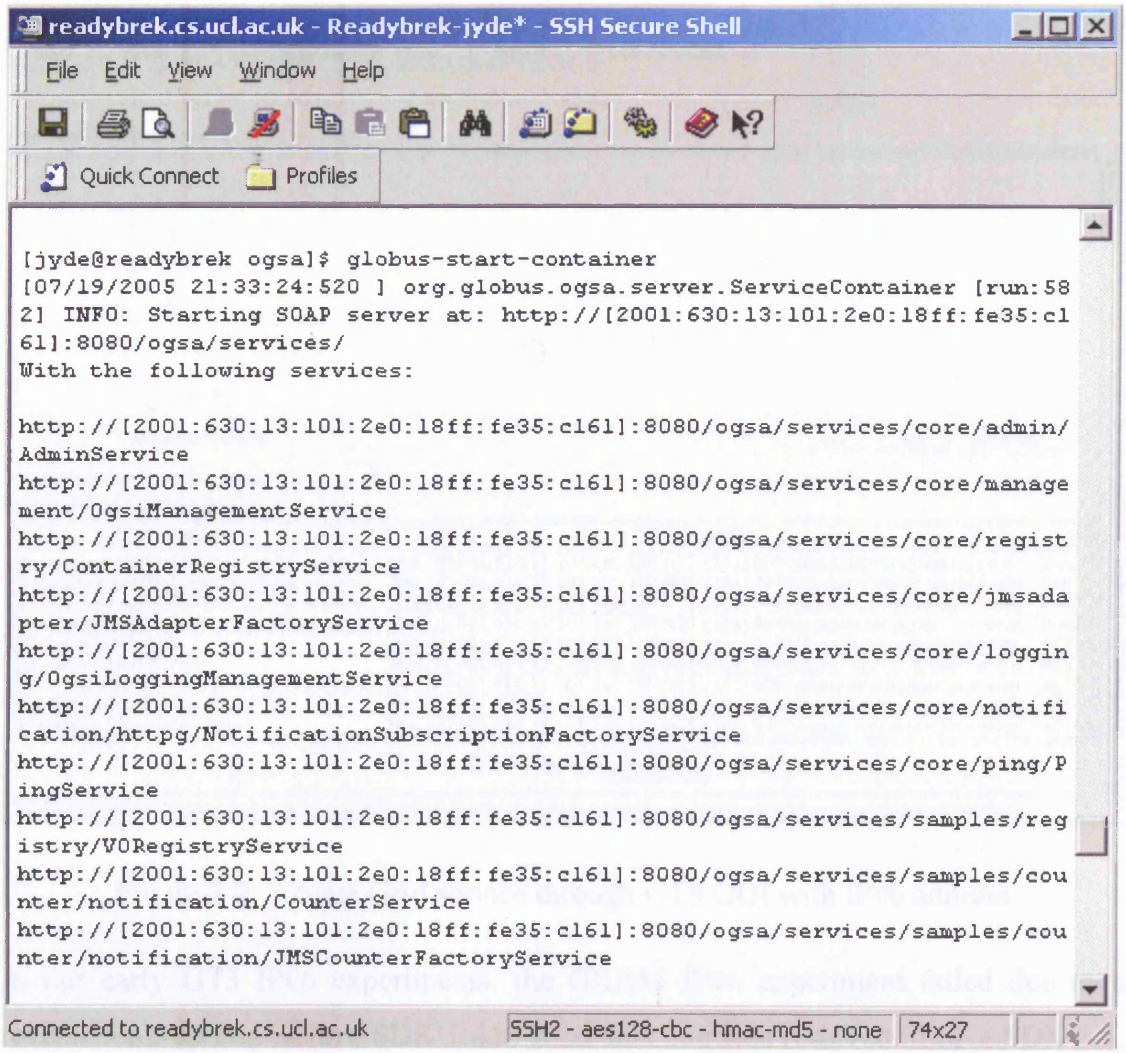


Figure 7-1: IPv6-enabled GT3 service container

The applications distributed with GT3 are used as general initial test services in our test scenarios. We managed to access Grid Services through IPv6 interfaces by using the OGSA Graphical User Interface (GUI) service browser, as shown in Figure 7-2.

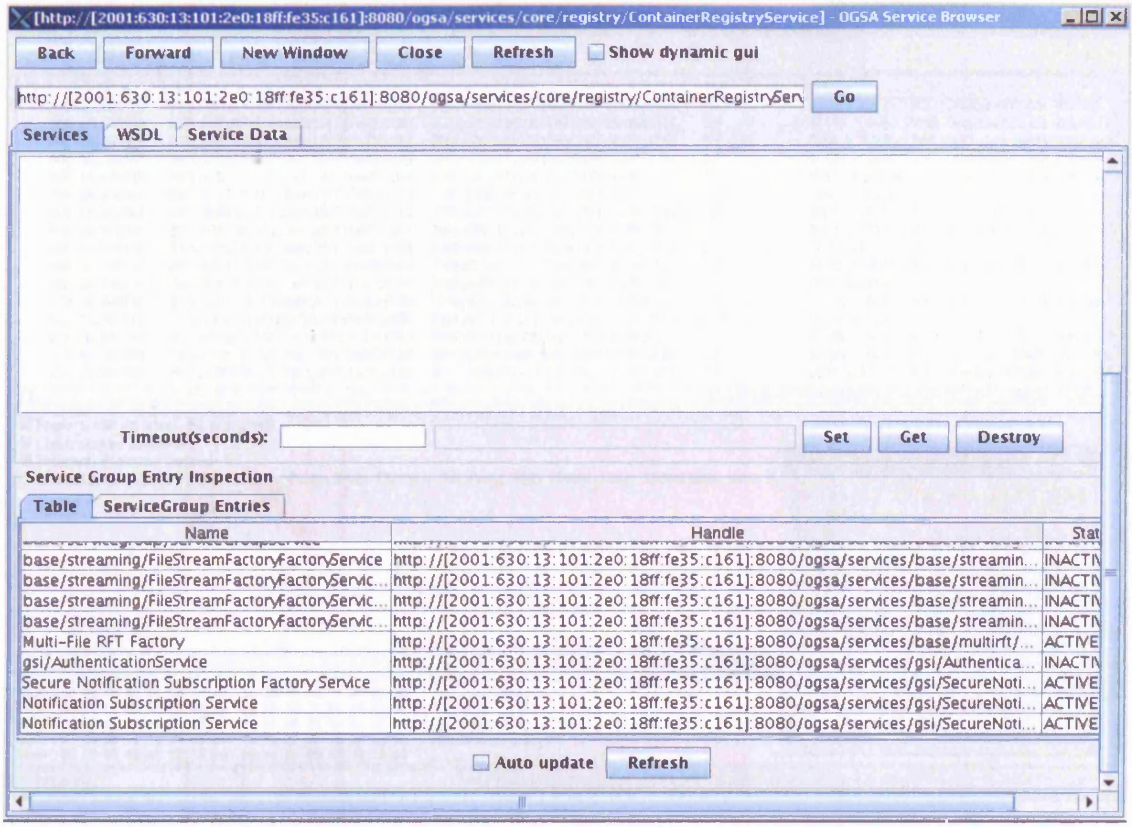


Figure 7-2: Access Grid service through GT3 GUI with IPv6 address

In our early GT3 IPv6 experiments, the GRAM IPv6 experiment failed due to a reverse look-up bug in Java SDK 1.4.x. After this bug was resolved in Java SDK 1.5, we succeeded in submitting remote GRAM jobs through IPv6 interfaces. During the remote job submission, we use ethereal software to capture the network traffic between the Grid client and the Grid server. Pure IPv6 communication was observed, as shown in Figure 7-3. In this figure, we can see packets are IPv6 and they use IPv6 address (red arrow. Although only part of packets are shown in the current roll window, we can see in the ethereal capture window that all packets are shown as "Other" (green arrow), while IPv4 TCP or HTTP packets are categorised as "TCP".

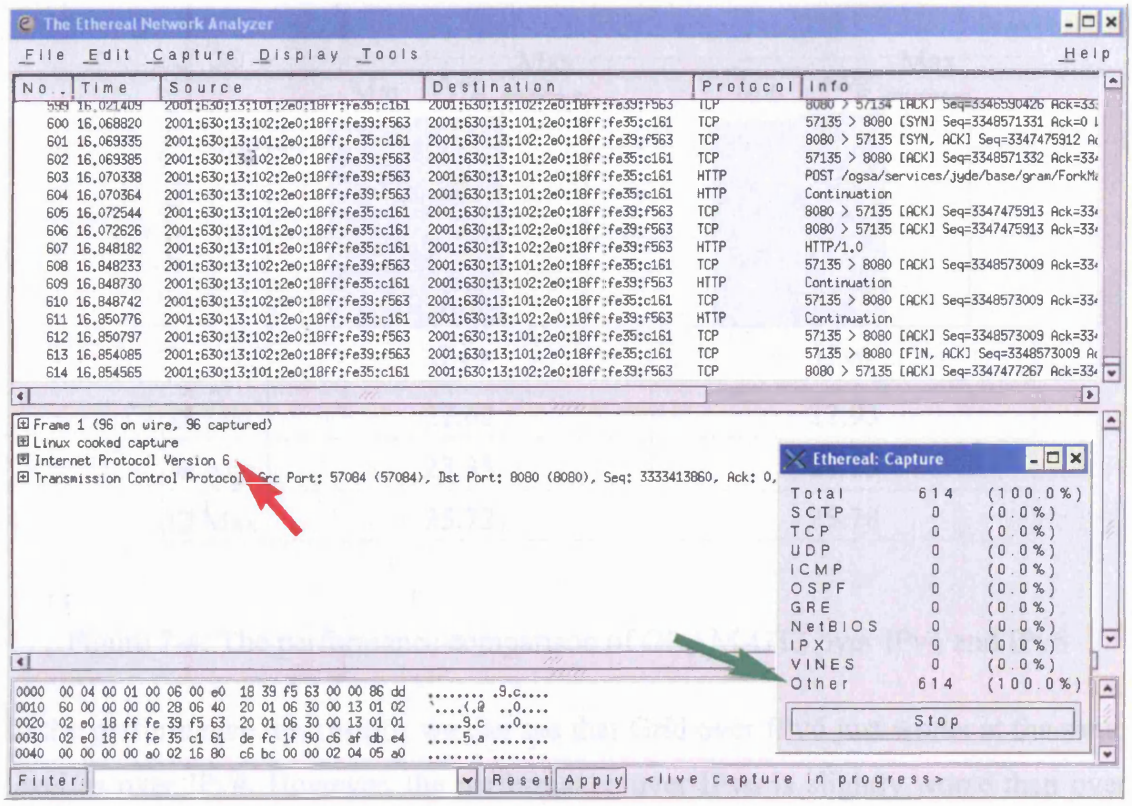
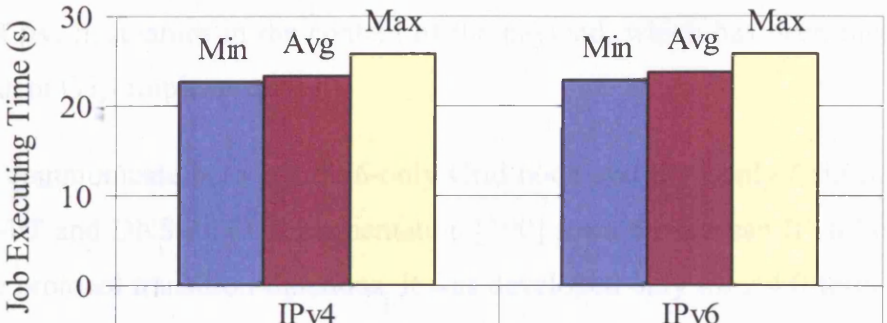


Figure 7-3: Ethereal capture pure IPv6 communication with GRAM

In order to test the performance of GRAM-GT3 over IPv6, we submitted remotely a minimum execution function, which includes a "date" command only, from a Grid client to a Grid server over IPv6, comparing to the remote submission over IPv4 with the same executing function and the same nodes. We measured the job completion time. In order to reduce the random factors, we ran it 100 times for each of IPv4 and IPv6. The minimum computation function reduces the affects of computational execution on the server. The usage of the same Grid nodes ensured that the tests had the same hardware and the same network topologies. The test results are shown in the following figure and table.



	IPv4	IPv6
Min	22.62	22.93
Avg	23.35	23.85
Max	25.72	25.78

Figure 7-4: The performance comparison of GRAM-GT3 over IPv4 and IPv6

In the performance test results, we can see that Grid over IPv6 just works at the same level as over IPv4. However, the performance over IPv6 is slightly worse than over IPv4. In fact, it has already been proved that IPv4 performance is better when there are smaller packets and simpler network topologies [53], [65].

7.2.3.3. Grid experiments in the heterogeneous IP environment

Our GT3 IPv6-enabled modifications have removed all IP-dependencies. With the configuration options "publishHostName", GT3 servers can start with IP-independent hostnames. On the IPv4-only Grid server, the Grid services are binding to an IPv4-only socket; on IPv6-only Grid server, the Grid services are binding to an IPv6-only socket; and on dual-stack Grid server, the Grid services are binding to an IPv6-enabled socket, which is listening to both IPv4 and IPv6 requests.

We have successfully demonstrated that all IPv4-only Grid clients, IPv6-only Grid clients and dual-stack Grid clients can access the dual-stack Grid server. Dual-stack Grid clients can access to dual-stack Grid servers through both IPv4 and IPv6 connections given the appropriate configuration. Of course, the IPv4-only server is accessible by an IPv4-only client, and the IPv6-only server is accessible by an IPv6-only client.

The experiments between IPv6-only nodes and IPv4-only nodes, which involve the

NAT-PT and DNS-ALG, will only succeed under one circumstance: Grid middleware should only use hostnames in the content of the payload, which has been met in our IP-independent GT3 implementation.

In order to communicate between IPv6-only Grid node and IPv4-only Grid node, we ran a NAT-PT and DNS-ALG implementation [100] from the Korean IPv6 Forum. It fulfilled the protocol transition functions. It was developed only for 2.4.0-test9 kernel. However, we ran it successfully on 2.4.18 kernel on a Redhat 8 node.

We assigned a number of IPv4 addresses (128.16.67.51-54) for the V4 address pool of the NAT-PT server. Another IPv4 address (128.16.67.55) was assigned for the DNS-ALG. It was mapped with the IPv6 address of an IPv6-enabled DNS server.

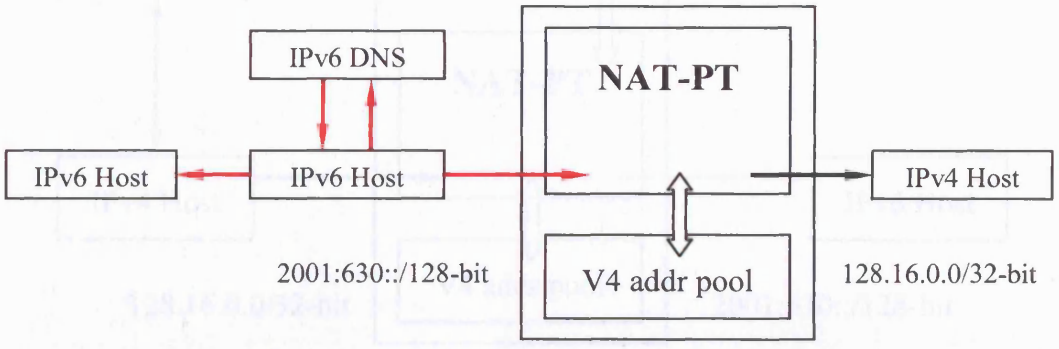


Figure 7-5: IPv6 node access IPv4 node through NAT-PT Gateway

In Figure 7-5, a specially configured IPv6 DNS server is also involved. When an IPv6-only node launches a communication requirement referring to a hostname, it sends a name search request to the specially configured IPv6 DNS server. In the normal DNS behaviour, if the DNS server finds the matching IPv6 address, it returns the address to the node. Then the IPv6 connection between the IPv6 client and the IPv6 server is built up.

If there is no matching IPv6 address, the specially configured DNS server returns a special IPv6 address, which includes the IPv4 address of the destination node in the IPv4-compatible IPv6 address format. With a specially configured routing on the IPv6 node, all the packets with the particular prefix, such as 2001:630:13:1b2::/96 in our experiments, in their destination addresses are sent to the IP transition gateway – NAT-PT gateway. The gateway, which has both IPv6 and IPv4 interfaces, receives the

IPv6 packets through its IPv6 interface and the IPv4 destination address as well. It picks up an unused IPv4 address from its V4 address pool and maps the IPv6 source address with it. Then the gateway re-packages the packet payload using the IPv4 address as source address, and sends them towards the IPv4-only destination server through its IPv4 interface. A mapping section that records this pair of the source IPv6 address and the allocated gateway IPv4 address is kept on the NAT-PT gateway for a while before it is removed or renewed. Typically, the configurable section live time is approximately a hundred seconds. The return packets go through the IP transition gateway using the information recorded in the mapping section.

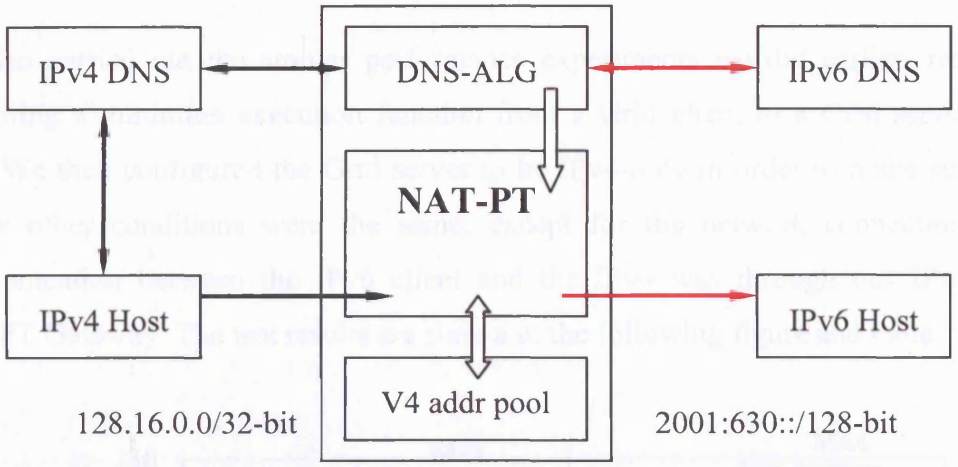


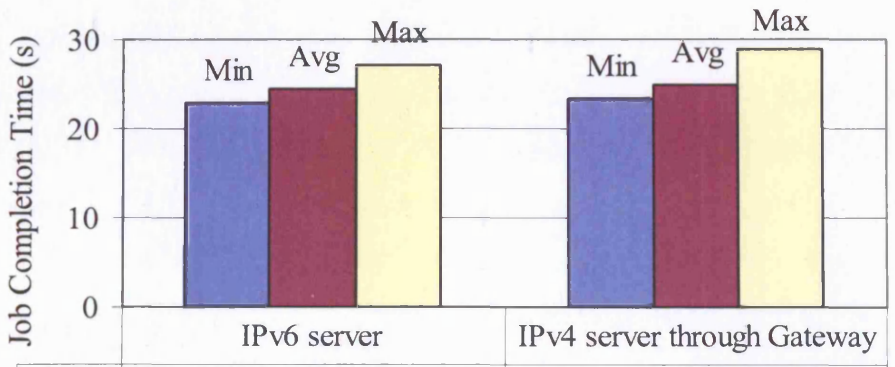
Figure 7-6: IPv4 host access IPv6 host through NAT-PT Gateway with DNS-ALG

In the opposite direction, the access experiment from an IPv4-only node to an IPv6-only node is fairly similar. When an IPv4-only node launches a communication requirement referring to a hostname, it will not receive a valid IPv4 destination address from the DNS server. The DNS server returns one of the NAT-PT gateway addresses to the IPv4-only node, while the NAT-PT gateway has a number of IPv4 addresses in its V4 address pool. Every gateway IPv4 address can be statically mapped to an IPv6-only node. For the dynamic address mapping, a DNS-ALG is required, as shown in Figure 7-6. According to these maps, the NAT-PT gateway re-packages all IPv4 packets and sends them towards the IPv6-only destination node. The dynamic mapping section that records this pair of the allocated gateway IPv4 address and the destination IPv6 address is kept on the NAT-PT gateway for a while before it is removed or renewed. The return packets go through the IP transition gateway using the information recorded in the mapping section.

The operation in these transition scenarios is generic as the IPv4/IPv6 transition for any applications.

With the NAT-PT and DNS-ALS in operation, we managed to access an IPv4-only GT3 server from an IPv6-only GT3 client for both GUI access and GRAM job submission. We succeeded to access an IPv6-only GT3 server from an IPv4-only GT3 client using GUI. However, our experiment failed when an IPv4-only GT3 client submitted jobs to IPv6-only GT3 server, because the NAT-PT implementation failed for those temporary IPv4 addresses reverse lookup. This problem is not serious conceptually, but illustrates the problems that must be resolved in this sort of activity.

We also carried out the similar performance experiments we did earlier: remotely submitting a minimum execution function from a Grid client to a Grid server over IPv6. We then configured the Grid server to be IPv4-only in order to make sure that all the other conditions were the same, except for the network connection. The communication between the IPv6 client and the IPv4 was through our IPv6/IPv4 NAT-PT Gateway. The test results are shown in the following figure and table.



	IPv6 server	IPv4 server through Gateway
Min	22.97	23.39
Avg	24.44	24.98
Max	27.06	28.96

Figure 7-7: The performance comparison of GRAM-GT3 from IPv6 client to IPv6 server and IPv4 server through NAT-PT gateway

In the performance test results, we clearly see that the IPv4 server through the gateway has the worse performance. Potentially, the NAT-PT gateway may cause more problems or even job failures due to the temporary IPv4 addresses reverse

lookup failure in this specific implementation.

7.2.3.4. eProtein applications running on GT3-based JYDE system

We tested the IPv6-enabled GT3 systems with externally developed GT3 services as well. In the case of the Grid applications, these are normally large applications, which require significant amount of resource. For this reason, it is possible to demonstrate only applications that have already been written to run in the environment envisaged here. We took one such application from the eProtein project, which was set up to conduct large protein analysis, from UCL.

Within the eProtein project, a Grid job distribution system – JYDE system is provided as the underlying platform for the protein analysis applications. As we have described in Section 2.2.3.2, the JYDE system uses a hierarchical architecture of sub-clusters. Within each network, it uses a cluster-scale scheduler to manage the local resource. Between the networks, existing Grid infrastructure implementation, such as IPv6-enabled GT3 in our experiments, is used to distribute jobs. The architecture of the GT3-based JYDE system is illustrated in the following figure.

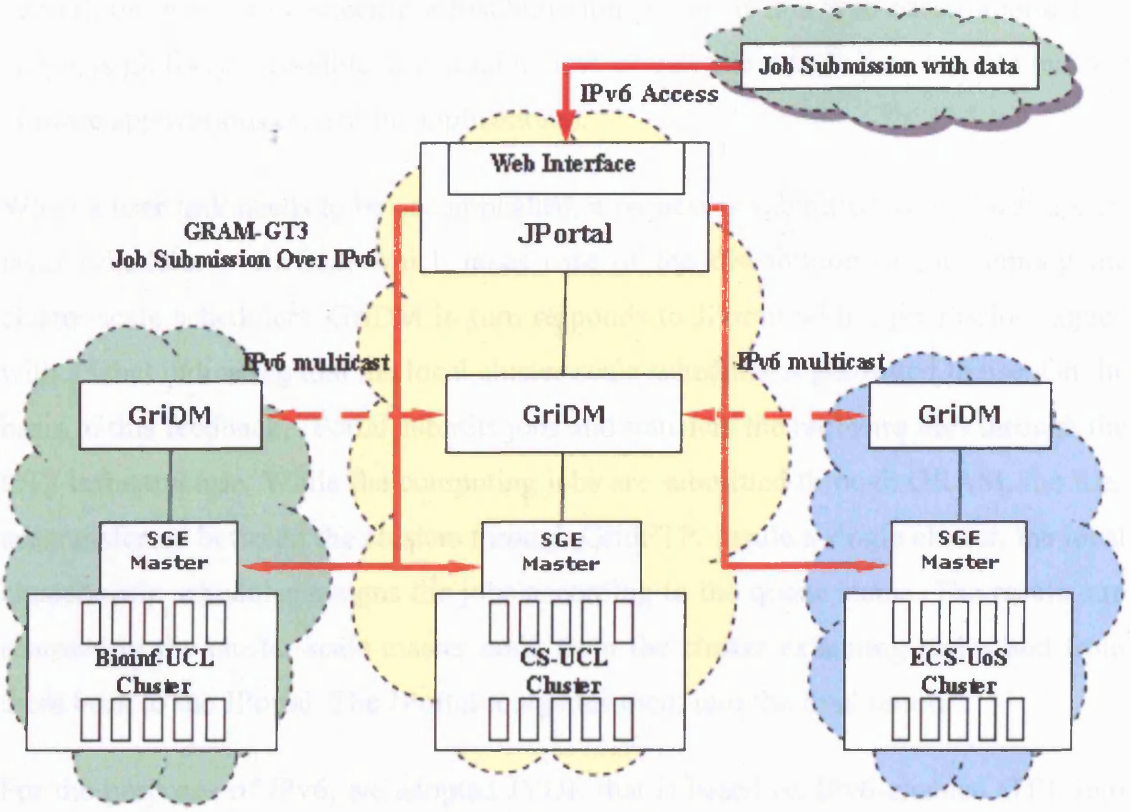


Figure 7-8: Schematic of GT3-based JYDE system

In order to demonstrate our system in real networks, we cooperated with the School of Electronics and Computer Science at the University of Southampton (ECS-UoS) and the Bioinformatics Unit the Department of Computer Science at University College London (Bioinf-UCL). In the above figure, we had a cluster located at ECS-UoS, with 4 processors; a cluster located at Bioinf-UCL, containing up to 170 processors; and a cluster located at the Department of Computer Science at University College London (CS-UCL), which was our GT3-IPv6 testbed with up to 12 processors. All the communications between these clusters were IPv6 only while IPv4 communication was filtered by gateways or firewalls. We also dedicated two of our nodes to become an IPv4-only Grid server in one of our experiment scenarios. Through the NAT-PT gateway, it worked similarly to the other IPv6-enabled Grid servers with slightly slower performance. However, we were not able to establish the Grid client on the IPv4-only nodes to access IPv6-only Grid servers due to the implementation restriction, which we have discussed in the previous section.

In Figure 7-8, the access authentication and user job handling are performed by

JPortal, an application-specific job submission portal. It is a web-based application, hence, is globally accessible. It is used to authenticate users and allow them to use one or more applications (scientific applications).

When a user task needs to be accomplished, a request is submitted to the local upper-layer scheduler – GriDM, which takes care of the distribution of jobs among the cluster-scale schedulers. GriDM in turn responds to JPortal with a permission tagged with a label indicating that the local cluster-scale scheduler is permitted to use. On the basis of this feedback, JPortal submits jobs and transfers the requisite files through the GT3 infrastructure. While the computing jobs are submitted through GRAM, the files are transferred between the clusters through GridFTP. Inside a single cluster, the local cluster-scale scheduler assigns the jobs according to the queue status. The results are returned to the cluster-scale master node from the cluster executing nodes and from there back to the JPortal. The JPortal integrates them into the final return.

For the purposes of IPv6, we adopted JYDE that is based on IPv6-enabled GT3, into our testbed. JPortal uses GRAM-GT3 to submit jobs remotely over IPv6. In the experiments, we used Secure SHell (SSH) over IPv6 to transfer files as GridFTP-GT3 is not IPv6-enabled. As we have introduced in Section 4.8, GT4 has both GRAM and GridFTP IPv6-enabled. However, we are not able to use GT4 in our experiments due to the fact that GT4 is not working with the cluster-scale scheduler – SGE.

A specific user application – GenTHREADER has been successfully run over our GT3-based JYDE system. We also monitored pure IPv6 communications between Grid servers using ethereal software. We have demonstrated the eProtein system working lively over an IPv6 infrastructure at the GARR conference, May 2005 [119].

The GriDM in this experiment assigned Grid jobs according to a fixed resource table. We observed that some Grid servers completed their jobs faster than others. The system had to wait for the slowest Grid server to complete its jobs. Although we can change the resource table manually to get a better balance among Grid servers, we are not able to change the table every time to adopt the changes of system. Furthermore, there are many dynamic changes, to which the fixed resource table cannot respond, such as the loading of clusters.

In our experiments, we also created some extreme situations. We shut down one of the Grid servers in one experiment scenario and disconnected its network connection in another. In both scenarios, the job assignment function still sent jobs to the unusable server. All these jobs failed as expected. Even during the job recovery, some jobs were still sent to it. Although the job failure rate reduced to a very low level after a few iterations of failure and re-submission, some jobs may never get executed if the job assignment function keeps assigning them to the unusable Grid servers. In other words, they are trapped in a "dead loop".

7.3. Mobile-enabled Grid experiments

To carry out our mobile-enabled Grid experiments, first we built up a mobile environment for Grid middleware, which we will demonstrate in Section 7.3.1. Then we ran a Grid implementation – GT3 in this environment, without the MIPv6 support, introduced in Section 7.3.2; and with MIPv6 support, introduced in Section 7.3.3. The comparison has validated our theory analysis in Chapter 5. At the end, we will shortly discuss the handover issue we meet in our experiments.

7.3.1. Building a generic Mobile environment

In order to be able to run Grid middleware in the mobile environment, our first step was to construct a mobile infrastructure. We adopted Mobile IP infrastructure to provide the support from the lower layer of networks. Although both Mobile IPv4 and Mobile IPv6 provide support for our system, only Mobile IPv6 is deployed in our experiments. This is because there are a number of improvements, which distinguish Mobile IPv6 from Mobile IPv4, as details have been introduced in Section 2.3.3. Underlying the network layer, some link-layer mobility technologies are used as well.

Our Mobile IPv6 testbed includes three IPv6 networks (Home, Foreign A and Foreign B) and one laptop moving across them (see Figure 7-9). It is generic enough to include most mobility scenarios:

a. Mobile node moves from home network into foreign network;

b. Mobile node moves from one foreign network into another foreign network;

c. Mobile node moves from foreign network back into home network.

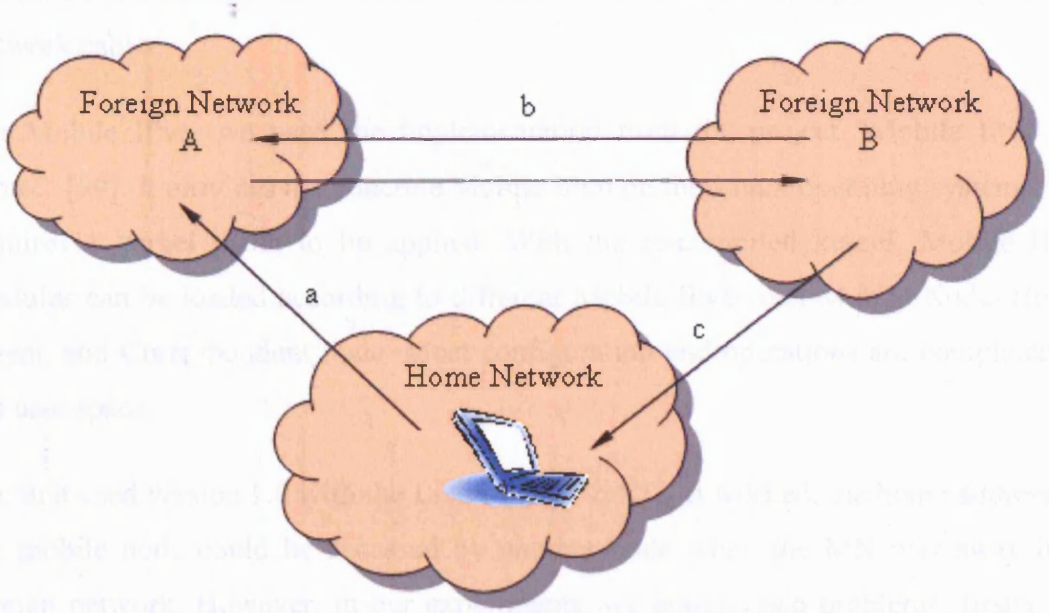


Figure 7-9: Mobile testbed that covers generic scenarios

For wireless communications, link-layer mobility technologies are used. In our main network, we used a set of Lucent base stations, Buffalo airstation, and Cisco aironet station. They are mainly support IEEE 802.11b wireless communication standard.

To set up Foreign Network A, we used a Redhat 8 node with two network interfaces. One interface remained the connection with our main network (Home Network, IPv6 prefix 2001:630:13:101::/64). On the other interface, we ran Linux IPv6 Router Advertisement Daemon (radvd). It sent Router Advertisement messages, specified by RFC 2461, with IPv6 prefix 2001:630:13:1b0::/64. We also ran Pim6dd as the IPv6 multicast routing daemon.

In Foreign Network B, we used a two-interface FreeBSD node. Its first interface was connected to our network. The second interface had the IPv6 prefix 2001:630:13:1b1::/64. Route6d and pim6sd were run on it.

The laptop, the mobile node in our experiments, had a wireless interface to communicate over link-layer mobility technologies in the home network. However, we were not able to demonstrate MIPv6 and link-layer mobility technologies working

together because there was no wireless communication support in our foreign networks. The laptop used a pin-cross cable or a small switcher to attach to the networks. It was switched between networks manually by unplugging and plugging network cable.

For Mobile IPv6, we used the implementation from the project "Mobile IPv6 for Linux" [99]. It provides the concrete Mobile IPv6 on the Linux operating systems, but requires a kernel patch to be applied. With the re-compiled kernel, Mobile IPv6 modules can be loaded according to different Mobile IPv6 role: Mobile Node, Home Agent, and Correspondent node. Most configuration and operations are completed in the user space.

We first used version 1.0 with the Linux kernel 2.4.22. It worked; the home address of the mobile node could be accessed by another node when the MN was away in a foreign network. However, in our experiments, we noticed two problems: firstly the MN could not access its own home address. Secondly, the MN always used the home address as its source address, even when it tried to access the nodes in the foreign network. This caused some unnecessary traffic diversion.

Later, we switched to version 1.1 with Linux kernel 2.4.26. The problems mentioned above had been resolved. Therefore, we mainly used this version in our mobility experiments.

7.3.2. Mobility in the Gird without MIPv6 support

First, we moved our mobile node between networks without MIPv6 in operation. Within our main network, the mobile Grid node was supported by the link-layer mobility. It could move around within the range of our wireless network. All communications, including Grid job submissions, were through the wireless interface.

Then, we moved the mobile node into a foreign network. Since the mobile node was supported by the IPv6 auto-configuration mechanism, it could get a temporary IPv6 address automatically when it attached to the foreign networks. It was able to communicate over IPv6. However, the mobile node was identified as a different host from what it was in home network. It was because different IPv6 addresses were used. All the communication referring to the hostname of the mobile node failed because

the home IPv6 address was not accessible any more.

When we ran a Grid implementation on the mobile node, in principle, it became a mobile-enabled Grid node. We took both IPv6-enabled GT3 system and the JYDE system onto it. Unfortunately, without the support of MIPv6, the mobile-enabled Grid node did not work properly. Every time when the mobile-enabled Grid node moved from one network into another, its previous Grid jobs broke. All remotely submitted Grid jobs were not able to return to the mobile-enabled Grid node. The mobile-enabled Grid node had to re-launch all jobs again. Furthermore, because the Grid servers cannot map the temporary IPv6 of the mobile-enabled Grid node with any hostname in their authorized hosts list, the remote GRAM job submission from the mobile-enabled Grid node in foreign network failed. Under this circumstance, we had to manually change the DNS records or hosts files to make the GRAM-GT3 on the mobile-enabled Grid node workable.

7.3.3. Operating the Grid over the Mobile IPv6 infrastructure

As we introduced in Section 5.3, if the Mobile IPv6 infrastructure functions correctly, it can be transparent for any IPv6-enabled applications above it. The same Grid applications that we introduced in Section 7.2.3 have also been taken into the Mobile IPv6 environment for the validation of Grid middleware.

We first set up both Grid services and a client on the laptop when it was still in the home network. It successfully accessed Grid services on other Grid nodes, and the Grid services on it were accessed by other Grid nodes. We also set up a Home Agent in the home network and registered the mobile-enabled Grid node's home address on it. However, MIPv6 was not really functional when the mobile-enabled Grid node was still in the home network. Within the home network, with the support of link-layer mobility technologies, the Grid services on the mobile node were accessible through the wireless interface. The mobile-enabled Grid node was free to move within the support range of our wireless base stations.

We then moved the mobile-enabled Grid node into the Foreign Network A. A Binding Update packet from the mobile-enabled Grid node to the Home Agent was captured. On the Home Agent, the mapped pair of home address and care-of address of the

mobile-enabled Grid node was observed in the binding cache list. The mobile-enabled Grid node successfully accessed these Grid services on other Grid nodes. Its Grid services were accessed by other Grid nodes through both home address and care-of address. Without any changes made in the DNS server, all the other Grid nodes used the hostname of the mobile-enabled Grid node to access it.

In the MIPv6 scenarios, the mobile-enabled Grid was always identified as the same host, regardless of its moving location or changing of its attach point. It was always accessible through its home address. Most Grid jobs, except for those who tried to communicate during the handover interval, were not affected at all. We will discuss the handover issue in next section.

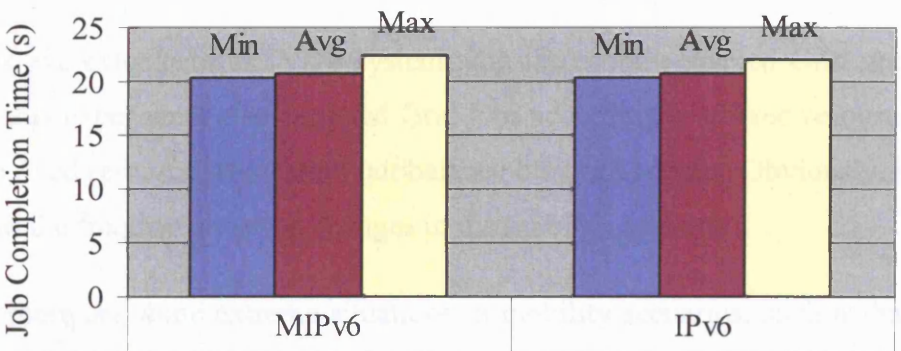
Without the further operation, the triangular routing packets were observed between the mobile-enabled Grid node and other Grid nodes. We then operated one of our Grid nodes as a Mobile IPv6 correspondent node. The communication between this node and mobile-enabled Grid node was direct after a Correspondent Binding Update packet was captured. Since our experiment network topology was simple, we did not observe performance improvement here. In a more complicated network topology, the mechanism of avoiding triangular routing will improve performance greatly.

After the mobile-enabled Grid node moved from the Foreign Network A into the Foreign Network B, new Binding update packets carrying the new care-of address were captured from mobile-enabled Grid node to the Home Agent and Mobile IPv6 correspondent Grid node(s). Afterwards, the home address was mapped to the new care-of address. Grid services on the mobile-enabled Grid node were accessible through both home address and its new care-of address.

Without any modification in the Grid implementations, we have run them successfully over the Mobile IPv6 environment. This result actually validates another main motivation for choosing Mobile IPv6: no forced changes for Grid implementations were required.

We also carried out the similar performance experiments we did earlier: remotely submitting a minimum execution function from a Grid client to a mobile Grid server over IPv6. We then moved the Grid server into Foreign Network A, in which it

worked over MIPv6. All the other conditions remained the same, except for the network connection. The test results are shown in the following figure and table.



	MIPv6	IPv6
Min	20.40	20.34
Avg	20.83	20.70
Max	22.33	22.01

Figure 7-10: The performance comparison of GRAM-GT3 over IPv6 and MIPv6

It is shown in Figure 7-10 that the performance over MIPv6 was slightly worse than over normal IPv6. However, this is explained by all communication over MIPv6 being through one more Gateway.

Afterwards, we extended our JYDE system with the mobile-enabled Grid node. The GriDM in this experiment also assigned Grid jobs according to a fixed resource table. Again, the fixed resource table could not balance between servers. Obviously, it could not adapt to the frequent situation changes in the mobility scenarios.

Naturally, there are some extreme situations in mobility scenarios, such as handover interval, breakdown time, bad connections, etc. These extreme situations happen frequently and are inevitable. Once they happen, many Grid jobs will fail. If these extreme situations do not improve themselves, the current Grid middleware even cannot recover as they always assign jobs to the unusable Grid servers.

7.3.4. Handover issue in our mobility experiments

Since we manually switched the mobile-enabled Grid node between networks, there was an inevitable short period for handover. Hence, some Grid jobs failed if the Grid servers tried to return the results to the mobile-enabled Grid client during such period.

Our recovery mechanism, introduced in Section 6.5.5, is robust enough to handle this kind of situation. All failed jobs will be re-assigned using the dynamic discovered resources. Even in the worst scenario, if the mobile node would not get any network connections, the dynamic job assignment function could use the mobile node itself as the only resource to complete all jobs.

On the other hand, there are many technologies aimed to solve the generic handover problems. Soft domain handover using shadow registration [60] can be used with Mobile IP approach together. Seamless handover [40], [59] technologies in wireless communication have become mature and ready for practise usage.

7.4. Grid resource mobility experiments

In Chapter 6, we introduced our approach to providing Grid resource mobility. In this

section, we will fulfil this approach in experiments. Firstly, we will introduce the dynamic Grid resource discovery implementation and operation in Section 7.4.1. Then we will discuss how we provided the dynamic parameter information in Section 7.4.2. We will describe our dynamic job assignment function with chosen parameters in Section 7.4.3. In particular, we will examine these network parameters. In Section 7.4.4, we will shortly discuss fulfilling Grid resource mobility in heterogeneous IP environment. At last, we will briefly discuss the issue between the remote submission procedure delay and the actual resource availability in Section 7.4.5. Our experiments in this chapter also involve our mobile Grid testbed, introduced in Section 7.3.

7.4.1. Dynamic Grid resource discovery experiments

As we have already experienced in the previous experiments, the fixed assigned Grid resource function did not work well when the situation of the Grid changed. It lacks the flexibility and intelligence to adapt to frequent situation changes; particularly, when extreme situations happen frequently in mobility scenarios. It is necessary for us to introduce the dynamic job assignment function in our experiments.

In practise, in order to fulfil dynamic Grid resource discovery function, we applied GriDM implementation, which was provided by eProtein project. It solves a general Grid issue – discover dynamically the existence and availability of Grid devices. It also monitors local Grid devices dynamically. GriDM is implemented based on JXTA [121] – a platform-independent collection of protocols for peer-to-peer networking.

In the GriDM implementation and operation, resource information update function has been separated from resource discovery function. The former uses TCP over normal IP and the latter is over multicast. One of the benefits we found in the experiments was: when the mobile-enabled Grid node moves into a network without multicast routing, it can still restore and retain the previous connected resources over TCP, though it cannot re-discover these resources through multicast.

GriDMs are operated independently on Grid clients and every Grid server. They are bound to the local Grid resource management functions, such as SGE. The multicast in JXTA is used to discovery Grid resources. A GriDM sends out a broadcast request every few seconds. All the other GriDMs that receive the broadcast information

respond with their own information to the source GriDM. By this mechanism, a GriDM can discover the existence of all the other GriDMs in multicast TTL range. The existence of the Grid servers found is stored in the GriDM local database. After a foreign GriDM has been found, the source GriDM sets up a communication pipe with it over TCP. Every cycle, the source GriDM sends an information update request to all its associated GriDMs. The respondent information from these GriDMs updates their latest availability. It is also the confirmation of their continue existence. The Grid resource information is also updated through the GriDM updating.

If one previous existing GriDM fails to respond to the update request, the GriDM will remove it from the list of available Grid devices, but continue to send it the update request for a number of cycles. If there is no update response, this GriDM will be removed from the local database.

The first problem we met turned out to be a generic Java problem: Java has default value 1 for TimeToLive (TTL) all multicast packets. This means that none of these multicast packets will go out the local network if TTL does not change. MulticastSocket.setTimeToLive function can change the TTL. However, we also observed, by default, this function only works for IPv6 multicast packets on a dual-stack host [94].

As Java-implemented software, JXTA also has the default TTL value 1 for its multicast packets. There is no TTL configuration option in the JXTA implementation. And in its default configurator, it has hard-coded IPv4 multicast "224.0.1.85". In order to enable JXTA-implemented GriDM to work across networks, we made modifications to the JXTA source code. We added a new "MulticastTTL" configuration option and its correspondent methods into the JXTA configuration data structure. We also changed the default multicast address to be IPv6 multicast "FF0E::4".

In our experiments, with the dynamic Grid resource discovery function in operation, the Grid client only submitted jobs to these Grid servers whose existence and availability had been confirmed. It can handle some of extreme situations, such as Grid server disappearing.

When the mobile-enabled Grid node moved from its home network into a foreign network, the dynamic Grid resource discovery function on it can still find normal Grid servers; and these Grid servers can find it as well. Then they can submit jobs to each other.

In the next section, we will introduce our approach to provide more dynamic parameter information based on Grid resource discovery function. The information will be used in the dynamic job assignment function later.

7.4.2. Providing dynamic parameters information

As discussed in Section 6.5, we have constructed a concrete job assignment function for the particular case of eProtein. Four parameters have been taken into our function: "number of processors", "queue state", "response time" and "packet loss rate".

As shown in Figure 7-11, after the mobile-enabled Grid client discovers Grid server A dynamically, on Grid server A, underlying cluster-scale scheduler provides the information of the number of the managed processors and the number of jobs in queue to GriDM A. GriDM A updates this information to GriDM B on the mobile-enabled Grid client dynamically. GriDM B then measures the response time and the packet loss rate from itself to the Grid server A. All four parameters are stored in the dynamic-updating Grid resource database on the client. Additionally, if any parameter associated with Grid server A is too bad, reaching the unusable conditions introduced in Section 6.5.3, the Grid server A will be removed from the Grid resources database.

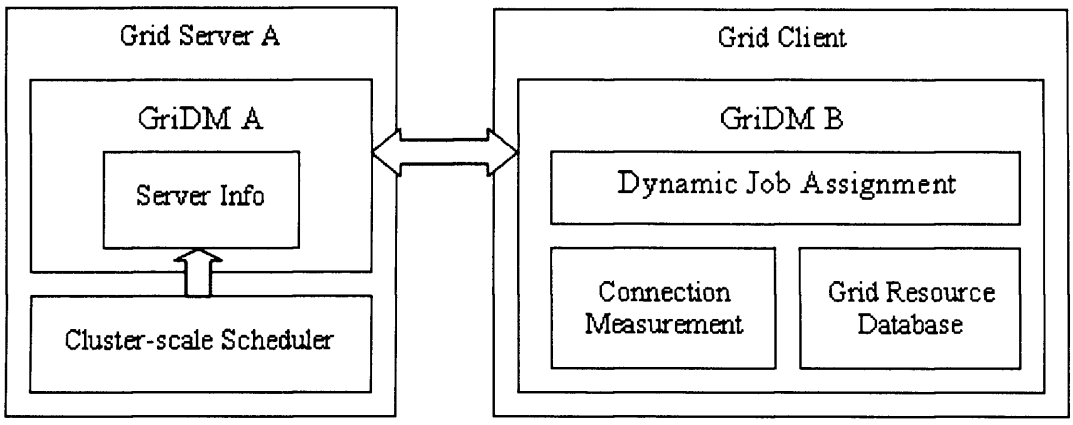


Figure 7-11: Dynamic parameters for Grid resource mobility

When the Grid client requires a certain Grid service, the dynamic job assignment function has all the needed Grid resource information from the Grid server database. With our concrete function, a certain number of jobs are assigned to a certain server. This dynamic job assignment function has enough "almost-real-time" information associated with both Grid servers and network connections. Hence, it can handle almost all the extreme situations.

In our experiments, SGE was used to provide the "number of processors" and "queue status" information to the local GriDM. The existing network utility, ping6 [106], was used to measure both the packet loss rate and the response time between the local node and any destination nodes. Ping6 uses the ICMPv6 [1] protocol's mandatory ICMP6_ECHO_REQUEST datagram to elicit an ICMP6_ECHO_REPLY from a host or gateway.

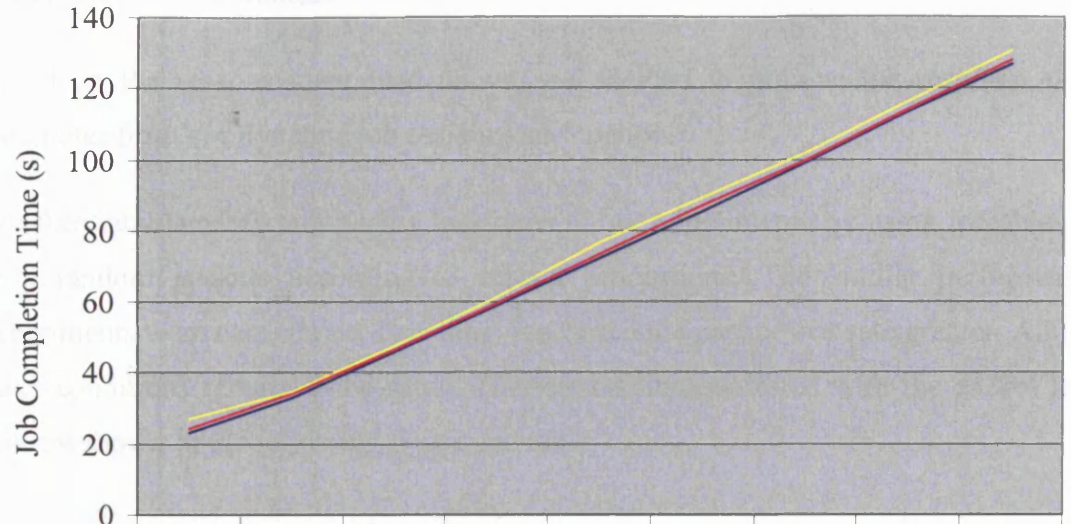
We had the fixed unusable values in our experiments for the usable line. We also provided a user configuration option for the unusable value.

7.4.3. Grid resource mobility experiments with chosen parameters

Since our research mostly focused on the affects of network infrastructure, we particularly examined these network parameters in our experiments.

In the real network environment, we used ip6tables [125] utility to simulate specific network connections, such as certain latency and certain packet loss.

We also carried out the similar performance experiments we did earlier: remotely submitting a minimum execution function from a Grid client to a Grid server over IPv6. This time, we varied the network latency. All the other conditions remained the same. The test results associated with response time are shown in the following figure and table.



	0ms	30	60	90	120	150	180	210	240
Min	23.43	33.33	45.57	58.78	72.27	85.50	99.04	113.40	126.55
Avg	24.51	33.98	46.26	59.60	73.43	86.86	100.00	114.24	128.04
Max	26.79	34.93	47.36	61.36	76.56	89.48	101.95	116.35	130.48

Figure 7-12: Grid performance associated with the response time

In the above figure, the response time was measured before Grid job submissions when the local processor was idle. In the test results, we can see the Grid performance was slowed down nearly linearly associated with the response time. On average, while the response time increases 10 ms, the Grid job complete time increases approximately 4.5 seconds, and the rate shows an increasing trend.

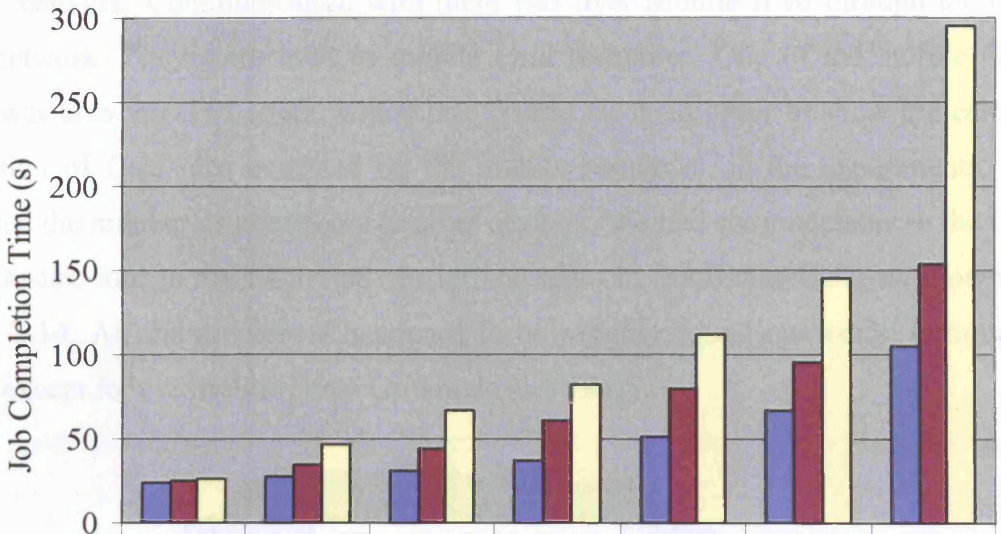
However, there are two important differences from our expectations. Firstly, the large latency did not cause Grid jobs to fail. Even when the response time increased to 500 ms, all the Grid jobs executed successfully, though it took average 280 seconds. Secondly, ping6 utility did not reflect packet loss within response time. Its response time only calculated the success acknowledgement. It had a separate packet loss rate parameter. Hence, this parameter is not as vital as we thought.

We also found that the measurement function of response time could be affected by many factors. For example, when the local host had high CPU loading, the measured response time could increase dramatically: up to 10 times relative to when the CPU was totally idle. This measurement did not reflect the network connection

performance as we wanted.

Based on the reasons mentioned above, we decided to remove the response time parameter from our dynamic job assignment function.

We then simulated certain packet loss rates in our experiments by using ip6tables to drop random packets according to certain proportions. The similar performance experiments were carried out. This time, we varied the packet loss rate greater. All the other conditions remained the same. The test results associated with the packet loss rate are shown in the following figure and table.



	loss 0%	4%	8%	12%	16%	20%	24%
Min	23.43	27.89	30.51	37.49	51.09	66.40	104.55
Avg	24.51	35.02	43.89	60.49	79.66	94.96	153.71
Max	26.79	46.90	66.20	82.30	112.34	145.29	295.70

Figure 7-13: Grid performance associated with the packet loss rate

In Figure 7-13, the packet loss rate was our simulation value. The test results indicated that the Grid performance was getting steadily worse as the packet loss rate increased. When the packet loss rate was higher than 20%, we observed a number of job failures.

We also noticed that the network measurement utility, ping6, showed different measurement results from our simulation values. The loss rate in ping6 is greater than the loss rate of connection. This is because ping6 uses loop back mechanism. The

packet loss happens in both directions – output and input. For example, when the connection has 5% packet loss rate, only 95% packets go through on both direction. In theory, the ping6 could measure loss rate around 10% ($1 - 0.95^2$).

Of course, due to the restriction in the number of ping6 attempts (we used 10) for every measurement, the measurement results are not always the same as theoretical results. There are mistakes sometimes.

We then applied the dynamic job assignment function into our experiments. The function was relevant to "number of processors", "queue status" and "packet loss rate". We structured our experiment system with two mobile Grid nodes, which sit on a foreign network. Communication with them was over Mobile IPv6 through the CS-UCL network. They were used as mobile Grid resources. One of the mobile Grid nodes was also the Grid client, which had JPortal on it. In order to show the certain proportion of Grid jobs executed on the mobile resources, in the experiments, we restricted the number of processors in other clusters. We had six processors in the CS-UCL cluster, four in the ECS-UoS cluster and eight in the Bioinf-UCL, as shown in Figure 7-14. All the processors happened to be roughly equally powerful (around 1 GHz), except for our mobile client Grid node (1.8 GHz).

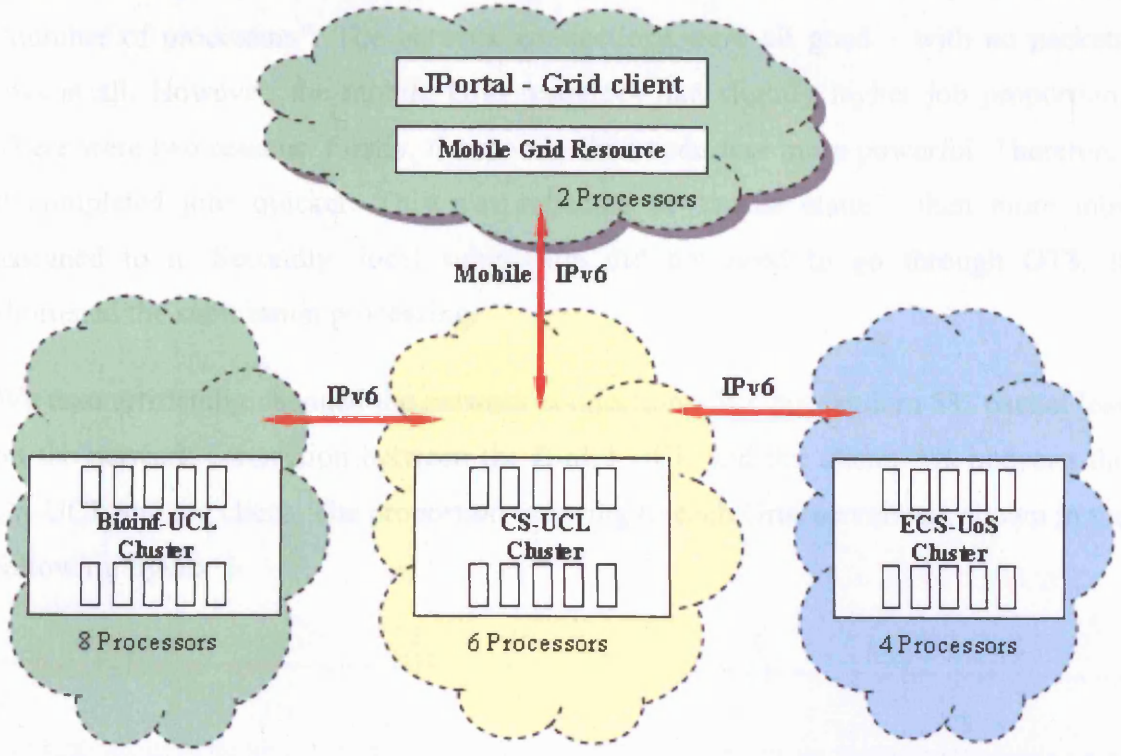


Figure 7-14: Schematic of structure of Grid resource mobility experiments

We then submitted a number of Grid jobs on the mobile Grid client without any artificial effects, and recorded the location where these jobs were actually executed. The proportion referring to each Grid servers are shown in the following figure.

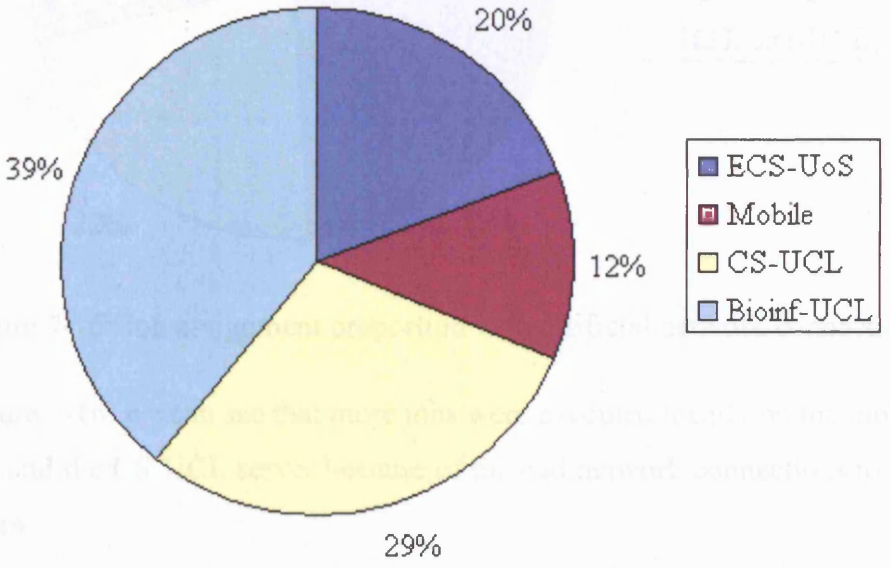


Figure 7-15: Job assignment proportion among normal servers

In the results, we can see the job assignment were almost the same proportion of "number of processors". The network connections were all good – with no packets loss at all. However, the mobile Grid resources had slightly higher job proportion. There were two reasons. Firstly, the mobile Grid node was more powerful. Therefore, it completed jobs quicker. This was reflected in "queue status", then more jobs assigned to it. Secondly, local submission did not need to go through GT3. It shortened the submission processing.

We then artificially changed the network connections. We put random 5% packet loss on the network connection between the Bioinf-UCL and the client; 5% between the CS-UCL and the client. The proportion referring to each Grid servers are shown in the following figure.

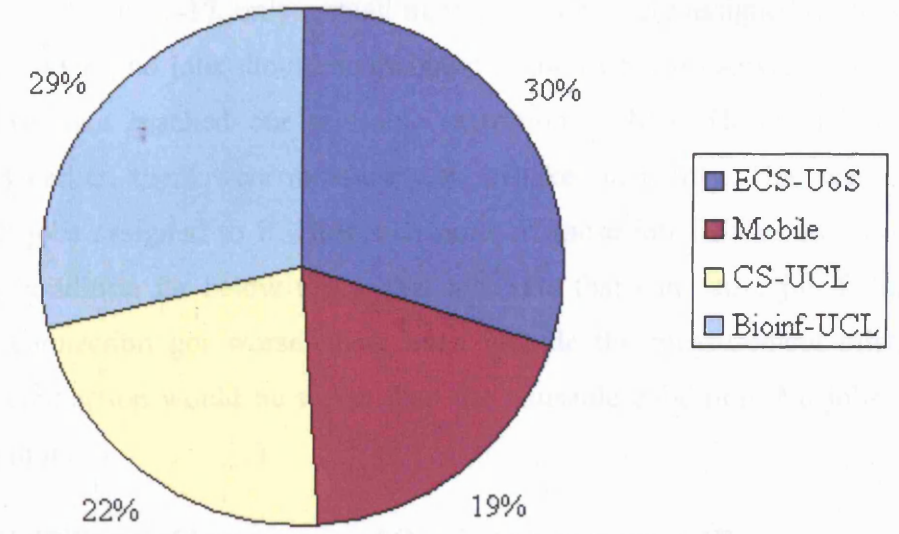


Figure 7-16: Job assignment proportion with artificial network connections

From Figure 7-16, we can see that more jobs were executed locally on the mobile Grid resources and the CS-UCL server because of the bad network connections to the other two servers.

We then artificially created extreme situations: random 10% packet loss on the network connection between the ECS-UoS server and the mobile Grid client; remaining 5% for the Bioinf-UCL; and 5% for the CS-UCL. The proportion referring to each Grid servers are shown in the following figure.

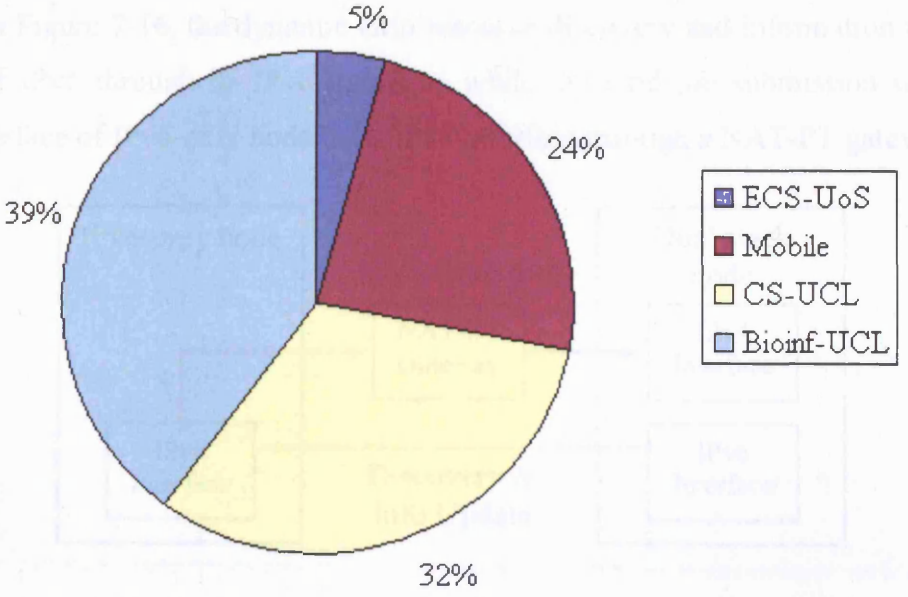


Figure 7-17: Job assignment proportion with artificial network connections 2

As shown in Figure 7-17, only a small number of jobs were assigned to the ECS-UoS server. In theory, no jobs should be assigned to the ECS-UoS server as its associated packet loss rate reached our unusable condition – 20%. However, as we have discussed earlier, there were measurement mistakes in reality. That was why there were still jobs assigned to it. However, none of these job failures as we chose the unusable condition far below the packet loss rate that can cause job failure. If the network connection got worse, then, even include the measurement mistakes, the network connection would be worse than the unusable condition. No jobs would be assigned to it.

7.4.4. Fulfilling Grid resource mobility in heterogeneous IP environment

In our experiments, we were not able to run the GriDM implementation in the heterogeneous environment. This was because the JXTA, on which GriDM was based, carried IP addresses in its payload. This problem depends on the specific implementation. With another implementation, which worked in the heterogeneous IP environment, we could fulfil our Grid resource mobility. We know how to modify JXTA to be able to carry out this aspect of the validation, but we did not feel that the effort that would be required would be justified.

In our experiments, we still tried to fulfil Grid resource mobility in heterogeneous IP environment by using the dual-stack node to pretend to reach an IPv4-only node. As shown in Figure 7-16, the dynamic Grid resource discovery and information updating was over IPv6 through its IPv6 interface, while the Grid job submission was from IPv6 interface of IPv6-only node to its IPv4 interface through a NAT-PT gateway.

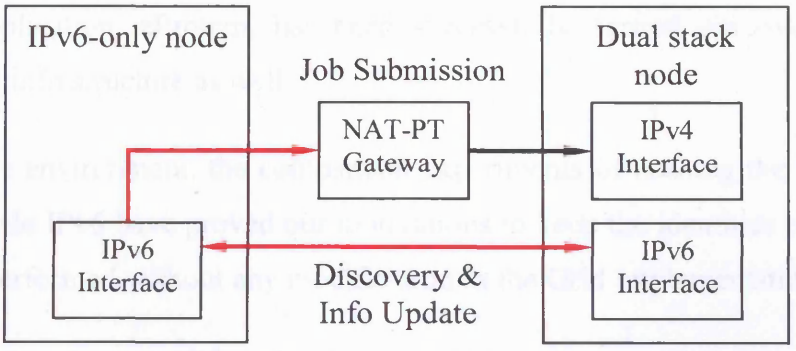


Figure 7-18: Grid resource mobility in heterogeneous IP environment

7.4.5. Remote submission procedure delay and the actual resource availability

When a Grid node discovers an available Grid resource, it can submit a job to the resource remotely. However, the remote submission takes time to arrive at and then occupy the remote resource. During this period, the resource is still seen to be available by all Grid clients and some jobs may be submitted to this resource. By the time that these jobs arrive at the resource, it may no longer be available. They have to wait for the resource to be released.

There are three solutions: one, which we used, is to set a query on the Grid servers. The later arrived jobs are executed according to the time order. However, with this approach, some jobs may sit in the queue for very long time before they are executed. Two, is to increase the amount of time between submission and further gathering. In our experiments, we use 30 seconds as the default interval, and for some scenarios we use 60, even 90 seconds. Three, when the job submission starts, its associated resource shows as occupied. This approach may also be ineffective in some scenarios.

7.5. Summary

In this chapter, we have validated our approach and mechanisms that we introduced in Chapters 4, 5, and 6. All our experiments have been carried out on real system rather than in the simulation environment.

We have modified a concrete Grid implementation, GT3, to be IPv6-enabled. It has been run successfully in both pure IPv6 and heterogeneous IP environments. A real scientific application, eProtein, has been successfully carried out over our IPv6-enabled Grid infrastructure as well.

In the mobile environment, the comparison experiments of running the Grid with or without Mobile IPv6 have proved our motivations to keep the identities of Grid hosts. It has been performed without any modification in the Grid implementation.

Furthermore, we have fulfilled the dynamic Grid resource discovery and dynamic resource assignment functions in the concrete system of eProtein. The Grid resource mobility mechanism provides more flexibility and intelligence so that mobile-enabled

7.5. Summary

Grid clients and servers can handle these extreme situations in mobile scenarios.

From the research and experiments mentioned above, we have reasonably validated our theories.

Chapter 8 Summary and further work

8.1. Summary of thesis

In this thesis, we have successfully built bridges between three research fields: IPv6, Grid and mobility. We investigated the possibility and benefit of running the Grid in an IPv6 environment. In our research, we provided generic approaches to move Grid middleware on to IPv6 networks. The transition mechanisms in a heterogeneous IP networks could also be carried out in many other mixed IP scenarios. We investigated the mobility requirements of Grid middleware. Our research leads us to provide lower-layer mobility support in Grid middleware using Mobile IPv6. It also validates the value of our Grid IPv6-enabled work. After using Mobile IPv6 to meet the general mobility problems in Grid middleware, we went further to investigate the specific requirements of mobile-enabled Grid. In order to realise Grid resource mobility, a few mechanisms have been introduced into the mobile-enabled Grid, such as dynamic Grid resource discovery, and parameterise Grid servers and network connections and choosing Grid resources according to these parameters. They bring more flexibility and effectiveness into the mobile-enabled Grid. With them, the mobile-enabled Grid is able to adapt any network changes. All the above-mentioned efforts were done within Grid infrastructure. It has limited influence to upper-layer Grid applications and is transparent for Grid users

All the experiments are running on real systems rather than in a simulation environment. They validate that our approach is feasible in the real network environment.

8.2. Contributions

This PhD work contributes to the generic porting issue from IPv4 to IP independence. It solves the porting issues for network-relevant applications and gives the experiences as generically as possible for the applications transition from IPv4 to IPv6. It applies to the Grid implementations. In this thesis, we make a Grid implementation, taking the Globus Toolkit as the example, work with IPv6, and demonstrate its operation in a heterogeneous IP environment.

This PhD work contributes an assessment of the embedded coupling of Grid computing middleware with implementation-specific underlying network layer functions and the subsequent development of a portable (IPv6 or IPv4) form of middleware. It is also extended into a more generic methodology for the design of Grid middleware.

The joint work on both Mobile IPv6 and Grid middleware has brought mobility support into Grid middleware from the lower-layer network infrastructure.

In order to provide resource mobility, in the mobile-enabled Grid, in collaboration with other researchers, we introduce a mechanism for the dynamic discovery of resources based on IP multicast addressing. The PhD work contributes a novel implementation and evaluation of mobility support within a Grid context, where either the clients and/or servers (and their functions) were able to move between domains.

A framework, which can be used to design the concrete dynamic resource assignment functions according to different requirement, has been developed also in this PhD work.

The mobile-enabled Grid is a generic system. It provides transparent services to the upper-layer Grid applications. Most of applications can be operated on the mobile-enabled Grid infrastructure with minimum modification.

8.3. Critique of work

If we had another chance to restart the whole work again, most of our investigation would have led us to the same or similar solutions/approaches.

However, we would do some of work in a different way. For example, in the Grid implementations, we would choose to have all components written in Java; it could make the process of IPv6 porting much smoother. We would implement Dynamic resource discovery functions using IPv6 multicast directly rather than using the IPv6 multicast function in JXTA. It would give us direct control of the packet payload. Then, we could rightly fulfil Grid resource mobility in a heterogeneous IP environment. However, this would mean that we would have to develop some security mechanism for the Grid peer authentication. We would like to have more control over the clusters we used and their network access. Although we have done what we wanted to achieve, the setting up, which has been done through other people, has slowed us a lot. We would also like more searching applications to have a richer set of resource allocation functions.

8.4. Relationship to other work

Based on our IPv6-enabled efforts, the University of Southampton has enabled IPv6-enabled GT3 in Websphere. They also developed some Grid applications, such as Weather Station over IPv6-enabled GT3.

We worked with the Globus implementation group at ANL to include the IPv6 modifications in the official release. Our on-line porting guide, entitled "How-to IPv6 in GT3", has become part of the official Globus technology reference documentation.

The eProtein Group in UCL is now adopting IPv6-enabled GT as the infrastructure of their JYDE system. We are also cooperating with them to integrate dynamic monitoring and measuring network connections and choosing Grid resources according to network connections' status functions into their future JYDE system.

8.5. Further work

With IPv6 functions enabled in Grid middleware, many advanced technologies can be applied just like the Mobile IPv6 that we have fulfilled in this thesis. Much research can be devoted to bringing more IPv6 advantages into Grid middleware, such as QoS flow, IPv6 anycast and multi-homing.

As a platform, the mobile-enabled Grid provides the potential to fulfil many user applications, such as distributed sensor networks, weather stations and tracking system. Much work can be devoted to develop some real user applications, which demand mobility support. Mobile routers can also enrich our mobile-enabled Grid research.

We can cooperate with other Grid researchers to disseminate our methodology of design portable Grid middleware, our dynamic resource discovery mechanism and our framework for dynamic resource assignment. Some of them may be standardized for generic Grid middleware.

Publications

The following publications are directly from the work of this thesis:

- Moving Grid Systems into the Mobility Environment
  By Sheng Jiang, Peter Kirstein
  Accepted by London Communications Symposium 2005

- The Combination of IPv6 and Grid Systems
  By Sheng Jiang, Piers O'Hanlon, Peter Kirstein
  In IPv6 and Broadband, Page 55 – 65.
  Published March 2005, Issued by IST 6Link

- Guidelines for IP version independence in GGF specifications
  By Tim Chown, Jim Bound, Sheng Jiang, Piers O'Hanlon
  Global Grid Forum Documentation 40, published January 2005

- Survey of IPv4 Dependencies in Global Grid Forum Specifications
  By Rute Sofia, Sheng Jiang, Christos Bouras, Dimitris Primpas, Kostas Stamos
  Global Grid Forum Documentation 41, published December 2004

- Web Intelligence Research Activities in the UK
  By Frank Zhigang Wang, Sheng Jiang, Yau Jim Yip
  In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence 2004, Page 807-808.
  Issued by IEEE Computer Society

- Moving Grid Systems into the IPv6 Era
  By Sheng Jiang, Piers O'Hanlon, Peter Kirstein
  Proceeding of Grid and Cooperative Computing 2003, Page 1033 - 1043.

Online Publications:

- How-to IPv6 in Globus Toolkit 4
  Online Guideline, 2005

- The Combination of IPv6 and Grid Systems
  Online Journal Grid Today, 2005 VOL. 4 NO. 2

- Final report on applications development and PoP deployment
  6NET public deliverable D5.15, 2005

- Progress report on applications development and PoP deployment progress report
  6NET public deliverable D5.13, 2005

- IPv6-enabled Globus Toolkit
  6NET public deliverable D5.12, 2004

- How-to IPv6 in Globus Toolkit 3
  Online Guideline, 2004

- Second phase of applications development and PoP deployment progress report
  6NET public deliverable D5.8, 2004

References

Note on references

The nature of the networking industry and community means that a number of the sources referred to in this dissertation exist only on the World Wide Web. All Universal Resource Identifiers (URIs) have been checked, however, their longevity cannot be guaranteed.

[1]. A. Conta, 1998. "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", Request for Comments 2463.

[2]. Aiken, B. 2000. "Network Policy and Services: A Report of a Workshop on Middleware," Request for Comments 2768.

[3]. Albitz, P. 2001. "DNS and Bind, 4th Edition", Published by: O'Reilly.

[4]. Allman M. 1998. "FTP Extensions for IPv6 and NATs", Request for Comments 2428.

[5]. B. Carpenter, 2001. "Connection of IPv6 Domains via IPv4 Clouds", Request for Comments 3056.

[6]. Berners-Lee T. 1998. "Uniform Resource Identifiers (URI): Generic Syntax", Request for Comments 2396.

[7]. Berners-Lee T. 1994. "Uniform Resource Locators (URL)", Request for Comments 1738.

[8]. C. Huitema, 2001. "An Anycast Prefix for 6to4 Relay Routers", Request for Comments 3068.

[9]. C. Perkins, 2002. "IP Mobility Support for IPv4", Request for Comments 3344.

[10]. C. Perkins, 1998, "Mobile networking through Mobile IP", IEEE Internet Computing, Vol. 2, 58-69.

[11]. C. Perkins, 1996. "IP Mobility Support", Request for Comments 2002.

[12]. Chervenak, A. 2001. "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Applications 23, 187-200.

[13]. Comer D. 2000. "Internetworking with TCP/IP, Volume 1, 4th edition," Published by Prentice Hall.

[14]. Cong Du, 2004. "A Runtime System for Autonomic Rescheduling of MPI Programs," Proceeding of the 2004 International Conference on Parallel Processing (ICPP'04).

[15]. Dario Bruneo, 2003. "Communication Paradigms for Mobile Grid Users", Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03).

[16]. Davies, J. 2002. "IPv6/IPv4 Coexistence and Migration," White paper of Microsoft Corporation.

[17]. Deering, S. 1998. "Internet Protocol, Version 6 (IPv6) Specification", Request for Comments 2460.

[18]. E. Crawley, 1998. "A Framework for QoS-based Routing in the Internet", Request for Comments 2386

[19]. E. Nordmark, 2000. "Stateless IP/ICMP Translation Algorithm (SIIT)", Request for Comments 2765.

[20]. Foster, I. 2005. "The Open Grid Services Architecture, Version 1.0", Global

Grid Forum Documents 30.

[21]. Foster, I. 2002. "What Is The Grid? A Three Point Checklist" Online Journal GRIDtoday, July 22, 2002: Vol. 1 No. 6.

[22]. Foster I. 2002. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Global Grid Forum.

[23]. Foster, I. 2002. "The Grid: A New Infrastructure of 21st Century Science," Physics Today Volume 55: 42-52.

[24]. Foster, I. 2001. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International J. Supercomputer Applications Volume 15(3).

[25]. Foster, I. 1998. "The Grid: Blueprint for a New Computing Infrastructure," Published by: Morgan Kaufmann.

[26]. Foster, I. 1997. "Globus: A Metacomputing Infrastructure Toolkit," Intl J. Supercomputer Applications Volume 11(2): 115-128

[27]. G. Huston, 2004. "IPv6 Address Prefix Reserved for Documentation", Request for Comments 3849.

[28]. G. Tsirtsis, 2000. "Network Address Translation - Protocol Translation (NAT-PT)" Request for Comments 2766.

[29]. Hinden R. 1999. "Format for Literal IPv6 Addresses in URL's", Request for Comments 2732.

[30]. ISI. 1981. "Transmission Control Protocol", Request for Comments 793.

[31]. ISI. 1981 "Internet Protocol DARPA Internet Program Protocol Specification", Request for Comments 791.

[32]. ISI. 1980. "User Datagram Protocol", Request for Comments 768.

[33]. J. Rosenberg, 2002. "SIP: Session Initiation Protocol", Request for Comments 3261.

[34]. Johnson, D. 2004. "Mobility Support in IPv6", Request for Comments 3775.

[35]. Junseok Hwang, 2004. "Middleware Services for P2P Computing in Wireless Grid Networks", Internet Computing, IEEE, Volume 8, Issue 4, pp 40-46.

[36]. Kent, S. 1998. "Security Architecture for the Internet Protocol", Request for Comments 2401.

[37]. Kleinrock, L. 2000. "Nomadic Computing and Smart Spaces," IEEE Internet Computing Volume 4: 52-53.

[38]. Laszewski, von G. 1999. "Grid Infrastructure to Support Science Portals for Large Scale Instruments," Proc. of the Workshop Distributed Computing on the Web (DCW).

[39]. Lee McKnight, 2004. "Wireless Grids – Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices", Internet Computing, IEEE, Volume 8, Issue 4, pp 24-31.

[40]. M. Endler, 2002. "General Approaches for Implementing. Seamless Handover," Proceeding of the second ACM international workshop on principles of mobile computing, pp.17-24.

[41]. Mauro Migliardi, 2002. "Mobile Interfaces to Computational, Data and Service Grid Systems", Mobile Computing and Communications Review 6(4): pp 71-73.

[42]. Niranjan Suri, 2003. "Agile Computing: Bridging the Gap between Grid Computing and Ad-hoc Peer-to-Peer Resource Sharing," Proceeding of the 3rd IEEE/ACM International Symposium on Cluster Computing and Grid (CCGRID'03).

[43]. Nirmalya Roy, 2005. "Enhancing Availability of Grid Computational Services to Ubiquitous Computing Applications", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05).

[44]. R. Bush, 2001. "Delegation of IP6.ARPA", Request for Comments 3152.

[45]. R. Draves, 2003. "Default Address Selection for Internet Protocol version 6

(IPv6)", Request for Comments 3484.

[46]. R. Droms, 1997. "Dynamic Host Configuration Protocol", Request for Comments 2131.

[47]. R, Gilligan. 2000 "Transition Mechanisms for IPv6 Hosts and Routers", Request for Comments 2893.

[48]. R. Hinden, 1998. "An IPv6 Aggregatable Global Unicast Address Format", Request for Comments 2374.

[49]. R. Hinden, 1998. "IP Version 6 Addressing Architecture", Request for Comments 2373.

[50]. Rute Sofia, 2004. "Survey of IPv4 Dependencies in Global Grid Forum Specifications", Global Grid Forum Documents 40.

[51]. S. Bhatti, "IPv6 - Good for Grid: a position statement from a technical viewpoint", Proc. 1st 6NET Workshop, May 2003.

[52]. Shang-fen Guo, 2004. "Grid Mobile service: Using Mobile Software Agents in Grid Mobile Service", Proceedings of the Third International Conference on Machine Learning and Cybernetics, pp 178-182.

[53]. Sherali Zeadally, 2003. "Evaluating IPV6 on Windows and Solaris", IEEE Internet Computing, Volume 7, Issue 3, pp. 51-57.

[54]. Shin, M. 2005. "Application Aspects of IPv6 Transition", Request for Comments 4038.

[55]. Srisuresh P. 2001. "Traditional IP Network Address Translator (Traditional NAT)", Request for Comments 3022.

[56]. Stevens, W. 2003. "Advanced Sockets Application Program Interface (API) for IPv6", Request for Comments 3542, obsoletes RFC 2292.

[57]. T. Narten, 2001. "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", Request for Comments 3041.

[58]. T. Narten, 1998. "Neighbor Discovery for IP Version 6 (IPv6)", Request for Comments 2461.

[59]. T. Pagtzis, 2003. "Proactive seamless mobility management for future IP radio access networks," Computer Communication, Volume 26, pp. 1975-1989.

[60]. Ted Kwon, 2002. "Mobility Management for VoIP Service: Mobile IP vs. SIP," IEEE Wireless Communications, Volume 9, no. 5, pp. 66-75.

[61]. Thomson S. 2003. "Basic Socket Interface Extensions for IPv6", Request for Comments 3493.

[62]. Thomson, S. 1998. "IPv6 Stateless Address Autoconfiguration," Request for Comments 2462, obsoletes RFC 2553.

[63]. Tim Chown, 2004. "Guidelines for IP version independence in GGF specifications", Global Grid Forum Documents 41.

[64]. Tim Chown, 2003. "Advanced Aids to Deployment", Deliverable 17, the 6WINIT Project.

[65]. Yi Wang, 2005. "Understanding Current IPv6 Performance: A Measurement Study", Proceeding of 10th IEEE Symposium on Computers and Communications, pp 71-76.

[66]. W. Simpson, 1994. "The Point-to-Point Protocol (PPP)", Request for Comments 1661.

[67]. Access to Knowledge through the Grid in a mobile World (Akogrimo): http://www.mobilegrids.org

[68]. Argonne National Laboratory: http://www.anl.gov

[69]. CDMA Development Group: http://www.cdg.org

[70]. Condor-G system: http://www.cs.wisc.edu/condor/condorg

[71]. Digital Enhanced Cordless Telecommunications (DECT) Forum:

http://www.dect.org

[72]. Global Grid Forum: http://www.ggf.org

[73]. Globus Consortium: http://www.globus.org

[74]. Globus online Bugzilla: http://bugzilla.globus.org

[75]. Globus Resource Allocation Manager: http://www-unix.globus.org/developer/resource-management.html

[76]. Globus Toolkit: http://www.globus.org/toolkit

[77]. GridFTP: http://www.globus.org/datagrid/gridftp.html

[78]. GSM World from GSM Association: http://www.gsmworld.com

[79]. IEEE 802.11 Working Group: http://www.ieee802.org/11

[80]. IEEE 802.3 Working Group (Ethernet): http://www.ieee802.org/3

[81]. IETF Next Generation Transition Working Group (ngtrans): http://www.ietf.org/html.charters/OLD/ngtrans-charter.html

[82]. IETF IPv6 Operations Working Group (v6ops): http://www.ietf.org/html.charters/v6ops-charter.html

[83]. Imperial College e-Science Networked Infrastructure: www.lesc.ic.ac.uk/iceni/

[84]. Internet Engineering Task Force (IETF): http://www.ietf.org

[85]. IPv6-enabled applications: http://www.ipv6forum.org.uk/navbar/links/v6apps.htm

[86]. IPv6 Forum: http://www.ipv6forum.org

[87]. IPv6 Host Operating System Implementations: http://www.ipv6forum.org.uk/navbar/links/v6oslist.htm

[88]. IPv6 porting: http://www.ipv6forum.org.uk/navbar/links/v6porting.htm

[89]. IPv6 on Linux: http://www.bieringer.de/linux/IPv6

[90]. IPv6 transition tools: http://www.ipv6forum.org.uk/navbar/links/v6trans.htm

[91]. IPv6 Wireless Internet IniTiative (6WINIT) project: http://www.6winit.org

[92]. IPv6 Working Group of Global Grid Forum: http://forge.gridforum.org/projects/ipv6-wg

[93]. Java Technology: http://java.sun.com

[94]. Java Forums - MulticastSocket setTimeToLive problem http://forum.java.sun.com/thread.jspa?threadID=587596&messageID=3320614

[95]. Job Yield Distribution Environment (JYDE): http://www.e-protein.org/e-proteinlatestd.html

[96]. Local Multipoint Distribution Service: http://www.wcai.com/lmds.htm

[97]. Microsoft IPv6 Technology Preview for Windows 2000 http://msdn.microsoft.com/downloads/sdks/platform/tpipv6.asp.

[98]. Microsoft Windows XP – IPv6 overview http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag_ip_v6_ovr_topnode.mspx

[99]. Mobile IPv6 of Linux: http://www.mipl.mediapoli.com

[100]. NAT-PT Korean implementation: http://www.ipv6.or.kr/english/natpt-overview.htm

[101]. Official Bluetooth Wireless Info Site: http://www.bluetooth.com

[102]. Online Guildline "How-to IPv6 in Globus Toolkit 3": http://www.cs.ucl.ac.uk/staff/sjiang/webpage/how-to-IPv6-Globus.htm

[103]. Online Guildline "How-to IPv6 in Globus Toolkit 4": http://www.cs.ucl.ac.uk/staff/sjiang/webpage/How-to-IPv6-in-GT4.htm

[104]. Organisation for the Advancement of Structured Information Standard: http://www.oasis-open.org

[105]. Peter Bieringer's online "IPv6 & Linux - Current Status – Distributions", http://www.bieringer.de/linux/IPv6/status/IPv6+Linux-status-distributions.html

[106]. Ping and ping6 of iputils package: http://packages.debian.org/stable/net/iputils-ping

[107]. Sun Grid Engine: http://gridengine.sunsource.net

[108]. Task Mapping Editor: http://ccse.koma.jaeri.go.jp/publicity_open_software/tme.html

[109]. The 3rd Generation Partnership Project http://www.3gpp.org

[110]. The 6bone project: http://www.6bone.net

[111]. The 6Grid project: http://www.biogrid.jp/e/research_work/gro1/6grid

[112]. The 6NET project: http://www.6net.org

[113]. The 6REN Project: http://www.6ren.net

[114]. The 6TAP project: http://www.6tap.net

[115]. The Apache Project: http://www.apache.org

[116]. The Condor project: http://www.cs.wisc.edu/condor

[117]. The Data Over Cable Service Interface Specification project: http://www.cablemodem.com

[118]. The eProtein project: http://www.e-protein.org

[119]. The GARR Conference 2005: http://www.garr.it/conf_05/conferenza-eng.htm

[120]. The High Performance Radio Local Area Network (HIPERLAN): http://portal.etsi.org/radio/HiperLAN/HiperLAN.asp

[121]. The JXTA Project: http://www.jxta.org

[122]. The KAME project: http://www.kame.net

[123]. The Legion project: http://www.cs.virginia.edu/~legion

[124]. The Long project: http://matrix.it.uc3m.es/~long

[125]. The netfilter/iptables Project http://www.netfilter.org

[126]. Test and Verification for IPv6 project: http://www.tahi.org

[127]. Uniform Interface to Computer Resources: http://www.unicore.org

[128]. Universal Mobile Telecommunications System Forum: http://www.umts-forum.org

[129]. V6 working group of Widely Integrated Distributed Environment project: http://www.v6.wide.ad.jp

[130]. Web Services: http://www.w3.org/2002/ws

[131]. Web Services Resource Framework: http://www.globus.org/wsrf

[132]. Virtual Markets and Wireless Grids Project: http://www.wirelessgrids.net

Appendix A How-to IPv6 in Globus Toolkit

This material is mainly a combination of references [102] and [103].

Status of this documentation

This documentation is for people who have already known Globus and want to develop IPv6 support with Globus. The normal installation and configuration of Globus are not introduced in this document. The differences between these Globus versions (GT4-alpha, 3.3, 3.2.1, 3.2, latest cvs, 3.0.2, 3.0.1 and alpha) are described individually. For the full IPv6 support, we recommend latest GT4, GT 3.2.1 or the installation from the latest cvs, which has included our IPv6 modification. In the documentation, we try to give considerations as generally as possible for IPv6 deployment. Another principle in our working is to keep our modification as little as possible while the Globus Toolkit is developing very quickly. We have identified a few further modifications, which will make IPv6 configuration and operation earlier and smoother. They will be implemented on the reasonable stabled Globus Toolkit distribution later. We will keep tracking the major official release of the Globus Toolkit and giving the corresponding guide in the future.

1. Operating System Support on Hosts

First of all, the hosts should be IPv6-enabled. The operating system needs to provide IPv6 support. Type the following command to see whether the IPv6 module is loaded on your machine or not.

 # lsmod | grep ipv6

If the IPv6 module is loaded, you can find your global IPv6 address by using "*ifconfig*" command. As long as you have IPv6 modules loaded on your hosts, your IPv6 tests can be run locally.

On Linux Red Hat 8 or later versions, IPv6 module is provided and auto-loaded as default.

On Linux Red Hat 7.3 (or earlier version), users have to re-compile the kernel to get the IPv6 support. (Reference to http://www.bieringer.de/linux/IPv6)

And IPv6 capable application API libraries need to provide support for upper-layer applications, such as java. Java SDK 1.4 (see later section 3.1) or later versions provides the IPv6 support on Solaris, Unix and Linux while Java SDK 1.5 (see later section 3.2) provide IPv6 for WinXP and Win2003 server.

2. Networking Support for IPv6

To start any IPv6 experiment, we need the IPv6 support from the platform. First we need the IPv6-enabled networking. It requires the IPv6-enabled routers, which provide forwarding and dynamic routing, and support from IPv6-enabled network services, such as IPv6 DNS, Web services, etc.

Be sure your hosts can communicate each other using IPv6 interfaces. On IPv6-enabled machines, use "ping6" command followed by destination IPv6 address or IPv6-enabled hostname to check the IPv6 communication.

 # ping6 3ffe:2101:7:4:2e0:18ff:fe34:150b *or*

 # ping6 mocha.ip6.cs.ucl.ac.uk

If there is no IPv6 DNS server, the hostname may fail. (You can configure your local hostname lookup in */etc/hosts* configuration file.)

3. Associated Applications

The Globus system also utilises external applications. These applications need to be IPv6-enabled as well.

3.1. Java SDK 1.4

In GT3, Java run-time environment needs to be IPv6-enabled as mentioned earlier. Java SDK 1.4, which has the IPv6 support, has been tested to work but requires some extra security configuration. The security problem appears to arise because Sun packaged a beta of Xalan with Java SDK 1.4, but the xml-security package requires a stable version of Xalan (v 2.2.0 or later). To resolve the problem, the stable version of xalan.jar needs to be installed into the $JAVA_HOME/jre/lib/endorsed directory.

The latest Java distribution can be downloaded from Sun website (http://java.sun.com). Follow the instructions for installation, set JAVA_HOME to the installation directory, and put $JAVA_HOME/bin on your PATH. Make sure that java can be found as a command.

You can use JRE if you don't need to build any source.

3.2. Java SDK 1.5

Java SDK 1.5 has been released January 2004 with more fully IPv6 supports. The IPv6 reverse lookup bug in Java SDK 1.4 has been fixed. The IPv6 reverse lookup functionality is necessary for GRAM of GT3. Therefore, Java SDK 1.5 is requested for all GT3 users, who need GRAM. Due to the earlier release, GT 3.0 is not compatible with Java SDK 1.5.

As a conclusion, for who wants IPv6-enabled GRAM, please use the latest cvs installation with Java SDK 1.5.

We have a small test program for the IPv6 reverse lookup functionality (http://www.cs.ucl.ac.uk/staff/sjiang/webpage/reverse-lookup-test.htm):

 # java lookupAndReturn -6 hostname

If you have IPv6 address displayed and the reverse lookup hostname is not the address, your system is ready for the IPv6-enabled GRAM.

3.3. PostgreSQL (JDBC)

A JDBC compliant database is required for use of the Reliable Transfer Service (RFT) and Master Managed Job Factory Service (MMJFS). Currently, PostgreSQL is recommended. For IPv6 support, an IPv6 patch needs to be applied. It can be downloaded from http://www.lbsd.net. Download the patch, apply to PostgreSQL source code, then build and install PostgreSQL from the source code.

3.4. Web Containers

A Web container is the execution infrastructure for operating OGSA services. The container environments need to provide IPv6 Web services for OGSA.

GT3 provides a stand-alone web container. But it is only for test purpose. Jakarta Tomcat has been recommended by the Globus implementation group. A few other web service containers may be used as well. The configuration of stand-alone web container is introduced in section 4.1.

Tomcat5 is recommended with fully IPv6 support. For Tomcat4, since we use Java SDK 1.4 for IPv6 support, the "lightweight edition" of Tomcat4, which excludes several libraries that have already been included in the Java SDK1.4 distribution, has to be used instead of the "full edition" of Tomcat4. However, there are still problems for Tomcat4 to respond to a literal IPv6 address call. Tomcat can download from apache website http://jakarta.apache.org/tomcat.

For deploying IPv6-enabled GT3 on Websphere, please refer to:

http://www.ecs.soton.ac.uk/~dgm/work/work.php

4. IPv6 Configuration of GT3

For the general installation and configuration of GT3, please follow the Globus admin guide (http://www-unix.globus.org/toolkit/3.0/ogsa/docs/admin). We only describe the particular steps for IPv6.

4.1. Stand-alone Web container configuration

By default, the Globus stand-alone Web container binds to the IPv4 address of local host. However, the webcache port is connectable through both IPv4 and IPv6 interface.

But in IPv6 calls, Globus server side will translate the IPv6 communication into IPv4 since the port is binding with IPv4 address of local host.

To use hostname instead of IP address, set up the configuration option "publishHostName" to be "true" in the globalConfiguration section of server-config.wsdd as follows:

```
<parameter name="publishHostName" value="true"/>
```

Then, the Globus stand-alone Web container looks up the hostname of local host and uses it in the initial processing. On the dual-stack machine, the hostname can bind to both IPv4 and IPv6 address. Afterwards, the Globus server is IP independent and can communicate with client according to the IP family that client uses.

If there are other hostnames that can link to the host, such as a particular IPv6 hostname, you can set up the configuration option "logicalHost" in the globalConfiguration section of server-config.wsdd and client-server-config.wsdd (the client config is for GRAM only) as follows:

```
<parameter name="logicalHost" value="mocha.ip6.cs.ucl.ac.uk"/>
```

In the above example, *mocha.ip6* is a hostname that binds only to IPv6 address. The Globus stand-alone Web container will bind to IPv6. The IPv4 calls are still acceptable on dual-stack machines. But Globus server side will translate the IPv4 communication into IPv6.

In the original Globus coding, literal IPv6 address is not acceptable. To use a literal IPv6 address in a URL [See RFC 2732], the literal address should be enclosed in "[" and "]" characters. To use literal IPv6 address, see section 4.3.

4.2. PostgreSQL IPv6 configuration

As mentioned in section 3.2, an IPv6 patch needs to be applied with PostgreSQL source code. After that, you need to make sure that postmaster will be started with the "-i" flag to allow TCP/IP based connections. This will be in /etc/init.d/postgresql. If your postgresql service script uses pg_ctl to start postmaster, use "-o 'i'" to pass argument to postmaster. Secondly, if you want to allow remote hosts to connect to

your DB, you will need to edit pg_hba.conf. By default, this will only allow connections from 127.0.0.1. Put IPv6 loopback address in as a new entry with IP-Mask ffff:ffff:ffff:ffff:ffff:ffff as following:

```
host    all         all         ::1         ffff:ffff:ffff:ffff:ffff:ffff         trust
```

The invoking of JDBC database from GT3 is configured in the *server-config.wsdd*. The JDBC databases are different in GT3 alpha and in GT3.0.

In GT3 alpha, JDBC database is used for both ManagedJobFactoryService and ReliableTransferFactoryService. The URL connection in both services should be changed as following (the entries in local-server-config.wsdd and altered-server-config are used under some particular conditions):

```
<service name="base/gram/ManagedJobFactoryService" provider="Handler"
style="wrapped" use="literal">
<parameter name="dbConnectionURL"
value="jdbc:postgresql://[::1]/jobManagerDb"/>

<service name="base/reliabletransfer/ReliableTransferFactoryService"
provider="Handler" style="wrapped" use="literal">
<parameter name="connectionURL"
value="jdbc:postgresql://[::1]/testDatabase"/>
```

In GT3.0, GRAM starts to use embedded database Xindice, while RFT renamed after MultiFileRFTFactoryService. The URL connection should be changed as following:

```
<service name="base/multirft/MultiFileRFTFactoryService"
provider="Handler" style="wrapped" use="literal">
<parameter name="connectionURL" value="jdbc:postgresql://[::1]/rftDb"/>
```

4.3. Special for literal IPv6 dealing

As mentioned earlier, in the original Globus coding, literal IPv6 address is not acceptable. This is mainly caused by using String to create URLs. In order to be compatible with literal IPv6 address, Java standard method - "new URL" - should be used whenever create a URL. A few java files have been identified and modified in

our porting. Globus 3.0 invokes a shipped axis library (apache project) to create SOAP scheme, but that axis does not support IPv6. Therefore, we modified axis (only the piece of code that Globus invokes) as well. Axis library file is provided besides our modified ogsa.jar. The latest axis has fixed the problem. Globus has already included the fixed axis in their cvs, GT 3.3 and GT 3.2.

IPv6 modification has already included in GT3.3, GT 3.2 and the latest cvs. To get our modification to enable literal IPv6 address usage in earlier version of GT3, download our modified ogsa.jar (and axis.jar) from the following URL. Replace the existing ogsa.jar (and axis.jar) in $OGSA-DIR/lib.

http://www.cs.ucl.ac.uk/staff/sjiang/webpage/GT3-IPv6-Download.htm

Afterwards, you can use the global IPv6 address with the configuration option "logicalHost" in the globalConfiguration section of server-config.wsdd as follows:

```
<parameter name="logicalHost"
value="2001:630:13:101:2e0:18ff:fe34:150b"/>
```

Then Globus stand-alone Web container will bind to IPv6. The IPv4 calls are still acceptable on dual-stack machines, but Globus server side will translate the IPv4 communication into IPv6.

4.4. Tomcat Configuration for IPv6

The server side of GT3 can be deployed on other Web containers. The deploying processing generates the configuration file from the temp, not the using configuration file. Therefore, you have to repeat what we have mentioned in Sections 4.1 & 4.2.

Tomcat version 5 has been tested with fully IPv6 support. Tomcat4 has problems to correctly respond to the call with a literal IPv6 address.

5. Running IPv6 Tests on GT3

If the server side of Globus has bound to literal IPv6 address or IPv6-only hostname, no matter which IP family the client uses, the communication will be through IPv6 interface. In the following tests, that the client chooses IP family is based on the

fundamental that server side binds to the hostname that has both IPv4 and IPv6 address.

5.1. Start Web Container

To start Globus stand-alone Web container, run "ant startContainer" at the Globus installed directory. To start the Tomcat, run "service tomcat5 start" ("service tomcat4 start" for Tomcat4) as root for the rpm installation or run "source bin/startup.sh" at the Tomcat installed directory.

5.2. Running GUI (OGSA Service Browser)

To start OGSA Service Browser, type the following ant command. It will start with the local host and the default port 8080.

 # ant gui

In the graphic browser, you can change the host and port to browse the OGSA services on remote machines or on other web container. Change *service.root* and *service.port* in ogsa.properties to configure the default host and port.

5.3. Running GRAM

There is still a bug for GRAM (see Globus Bugzilla #1924). As a temporary solution, user can manually edit $GLOBUS_LOCATION/var/uhePortMapping: either add brackets [see RFC 2732] or use hostname instead of literal IPv6 address. After this, user may need to remove the old uhe under the .globus directory in user home directory. User may also need to reboot.

To run the GRAM job on remote machines or on other web container, use the following command according to your GT version. Replace the "localhost" by your remote hostname and replace port "8080" by your actually port number. Notice: the GT3 alpha and GT3.0 cannot communicate with each other.

GT 3.3, 3.2.1 and GT 3.2:

 # java org.globus.ogsa.impl.base.gram.client.GlobusRun -factory

http://localhost:8080/ogsa/services/base/gram/MasterForkManagedJobFactoryService -file schema/base/gram/examples/test.xml

GT 3.0, GT 3.0.1 and cvs installation:

 # java org.globus.ogsa.impl.base.gram.client.GlobusRun -factory http://localhost:8080/ogsa/services/base/gram/MasterForkManagedJobFactoryService -file etc/test.xml

GT 3 Alpha:

 # java org.globus.ogsa.impl.base.gram.client.GramClient http://localhost:8080/ogsa/services/base/gram/MasterManagedJobFactoryService etc/test.xml

6. Running IPv6 Tests on GT4

If the server side of Globus has bound to literal IPv6 address or IPv6-only hostname, no matter which IP family the client uses, the communication will be through IPv6 interface. In the following tests, that the client chooses IP family is based on the fundamental that server side binds to the hostname that has both IPv4 and IPv6 address.

6.1. Start Web Container

To start Globus stand-alone Web container, run "globus-start-container". At the terminal, we can see the web container start with service URLs, which include hostname or IP address:

Starting SOAP server at:
https://[2001:630:13:101:2e0:18ff:fe34:150b]:8443/wsrf/services/

With the following services:

[1]:
https://[2001:630:13:101:2e0:18ff:fe34:150b]:8443/wsrf/services/TriggerFactoryService

To start the Tomcat, run "service tomcat5 start" ("service tomcat4 start" for Tomcat4) as root for the rpm installation or run "source bin/startup.sh" at the Tomcat installed directory.

6.2. Start GridFTP Server

The re-implemented GridFTP is based on the Globus_XIO, which is an extensible input/output library for the Globus Toolkit. It provides a single POSIX-like API (open/close/read/write) that supports multiple wire protocols, with protocol implementations encapsulated as drivers. The XIO drivers distributed with 4.0 include TCP, UDP, file, HTTP, GSI, GSSAPI_FTP, TELNET and queuing. Although IPv6 driver is still under testing, it is distributed with Globus_XIO. UCL has tested it working in a proper dual-stack environment.

To start GridFTP server, run "globus-gridftp-server". It binds the GridFTP daemon to both IPv6 and IPv4 on the dual-stack machine. Afterwards, the server is able to respond both IPv6 and IPv4 client call.

 # globus-gridftp-server

Notice: if the user uses xinetd or inetd, make sure that the version of xinetd or inetd is IPv6-enabled.

6.3. Running GRAM

To run the GRAM job on remote machines or on other web containers, use the following command. Replace the "localhost" by your remote hostname/IP address and replace port "8443" by your actually port number. Notice: if a literal IPv6 address is used in the command, it should be included in a pair of square brackets [See RFC 2732].

 # globusrun-ws -F
https://localhost:8443/wsrf/services/ManagedJobFactoryService -submit -f
test-job.xml

In the above command, "test-job.xml" is a job descript file. Users can find the samples in Globus online WS GRAM manual (http://www-

unix.globus.org/toolkit/docs/development/3.9.4/execution/wsgram/user/index.html).

6.4. Running GridFTP client

IPv6 has been tested and is exposed only in globus-url-copy. Since it is still considered experimentally, user needs to explicitly enable it with the parameter"-ipv6" (which allows both, ipv4 and IPv6, whatever the user's system indication is the default).

 # globus-url-copy -ipv6 gsiftp-file-source gsiftp-file-destination

7. To choose the IP family on client

To choose the client IP family, run the above Java command with the following arguments:

-Djava.net.preferIPv6Addresses=true

Use IPv6 addresses if it is possible

-Djava.net.preferIPv4Stack=true

Use IPv4-only Stack on client side

Appendix B Globus Toolkit Porting Bugs

We list both the IPv6 and non-IPv6 relevant bugs as follows, referring the Bugzilla numbers. We also give the status and a brief description for each bug. All were reported to the official Globus, Java and Axis developers, and have been resolved successfully.

1. IPv6 Relevant Bugs

- Globus Bugzilla #1032, Porting GT3 IPv6-enabled - Solved Issues 1 (Java SDK 1.4, JDBC IPv6, Tomcat edition and GT3 Container Configuration)

  [Resolved] It is about to get the IPv6-enabled environment for GT3.

- Globus Bugzilla #1033, Porting GT3 IPv6-enabled - Identified Issues 1 (Invoking JDBC and Gatekeeper Configuration)

  [Resolved] It is the configuration for GT3 to use IPv6 APIs.

- Globus Bugzilla #1034, Porting GT3 IPv6-enabled – Identified Issues 2 (CoG)

  [Resolved] Java Commodity Grid Kit provides access to Grid services. GT3 needs the IPv6 support from it.

- Globus Bugzilla #1035, Porting GT3 IPv6-enabled – Identified Issues 3 (Hard-coded IP Addresses and literal IPv6 address handling

  [Resolved] Approximately ten Java files have hard-coded IPv4 addresses. It has been marked for GT 3.2. For literal IPv6 address handling, see Globus Bugzilla #1361.

- Globus Bugzilla #1256, cannot submit job to localhost

[Resolved] Since GT 3.0.1, the localhost is not a valid target to submit jobs. For submitting job through IPv6 interface, we meet the Java reverse lookup problem, see below.

- Globus Bugzilla #1342, IPv6 non-compatible Axis library in GT3 - and fixed axis

  [Resolved] The Globus-shipped axis library (axis.jar) has a few IPv4 dependencies. We have helped Axis developers to fix it, see axis-Apache Bugzilla #23576. It has been marked for GT 3.2.

- Globus Bugzilla #1361, IPv6 non-compatible URIs and URLs generating

  [Resolved] The source code needs to be changed to meet the particular literal IPv6 address format in URIs and URLs.

- Globus Bugzilla #1924, IPv6 incompatible - proxyStarter uses string to handle IP address

  [Resolved] This issue has been eliminated in 4.0 and an upgrade is recommended.

- Axis-Apache Bugzilla #23576, IPv4 dependencies in AXIS - org.apache.axis.types.URI

  [Completed] URI functions were IPv4-only. This has been fixed to be IP independent.

- Report IPv6 reverse look up bug to Sun Java

  The reverse lookup function does not work properly in Java 1.4. In Java 1.4.0 and 1.4.1, both IPv4 and IPv6 reverse lookup do not work. Java 1.4.2 fixed the IPv4 reverse lookup, but the IPv6 reverse lookup does not work yet. The bug has been fixed in Java SDK 1.5.

- Globus Bugzilla #2577, ServiceHost.hostcache is not initialized when container start {GT4}

  [Resolved] It causes that "logicalHost" or "publishHostName" configuration

option is skipped in some components running. Then IPv4 address is used as default.

- Globus Bugzilla #2573, IPv6 incompatible bug in ServiceThread {GT4}

  [Resolved] There is an IPv6 incompatible bug in wsrf/container/ServiceThread.java. It uses a string to create a URL. It has been fixed by using "new URL" to create the URL.

2. Non-IPv6 Relevant Bugs

- Globus Bugzilla #1140, UserContainer loops continuously during initialisation (JVM)

  [Resolved] GT3 was hard-corded to use shipped IBM JVM. It causes problems on some machines to use MMJFS.

- Globus Bugzilla #1246, cannot resolve symbol - when compiling mmjfs from cvs

  [Resolved] Some of these cvs build.xml do not consider different configuration well.

- Globus Bugzilla #1410, local part cannot be "null" when creating a QName

  [Resolved] Globus uses JAX-RPC library, which has a bug in checking the validation of QName. GT3 has problems caused by the invalid QName with the after-fixed JAX-RPC library.

- Globus Bugzilla #1531, no such algorithm: MD5/RSA for provider Cryptix

  [Resolved] One of the un-compatible problems between GT3 and Java 1.5.