Project Number:	**IST-1999-20393**
Project Title:	*Laboratories Over Next Generation Networks*
Deliverable Type:	**P – public**

CEC Deliverable Number:	IST-1999-20393/TID/WP1/DS/P/10/b11
Contractual Date of Delivery to the CEC:	M26
Actual Date of Delivery to the CEC:	February 7, 2003
Title of Deliverable:	Final Summary of Conclusions and Guidelines
Workpackage contributing to the Deliverable:	WP1 (it compiles information from all WPs)
Nature of the Deliverable:	R – Report
Author(s):	Alberto García (UC3M), Josep Mangues (UPC), Jacinto Vieira (PTIN), Cristina Peña (TID), Carlos Ralli (TID), Ruth Vázquez (TID), Tomás de Miguel (UPM), Eva Castro (UPM), Jorge Jiménez (NOR)
Editor:	Ruth Vázquez (TID)

Abstract:	The aim of this document is to draw the conclusions and results obtained in the LONG as requested in the first project review (Apr 5th 2002, Madrid). This document presents technical conclusion obtained during the whole project regarding network design, deployment and testing, user services and application migration.

Keyword List:	Conclusions, IPv6, LONG, Migration, Networks, Porting, Research, Results, Summary.

Executive Summary

This deliverable summarizes all conclusions and results that can be extracted from the work done in the LONG project. It is done at the request of the EC and reviewers after the technical review that took place in Madrid on April 5th, 2002.

TABLE OF CONTENTS

1. Introduction and General Conclusions

Along the life of the project, LONG has aimed to foresee and solve problems related to the design and deployment of IPv6 Networks and advanced user applications. In this sense, several studies and tests have been carried out and some conclusions and guidelines have been drawn.

This document presents general conclusions of all this work to make easier to understand all the work done by the LONG project. It must be understood as a quick reference guide to all documents delivered to the European Commission.

Next chapter contains conclusions extracted from WP2 and WP4 as regards to the design, deployment and testing of IPv6 networks and advanced network services that must be supported.

Following, chapter 3 explains how to deploy user services over mixed IPv4/IPv6 networks and transition mechanisms when these are needed.

After that, chapter 4 is devoted to the migration of applications. It consists of general recommendations and a "Quick Guide of Application Porting".

The summary of each deliverable is presented in Chapter 5, as an index to the LONG's technical documentation.

In chapter 6 several experiences involving real users that have been made over the LONG network infrastructure are shown.

Finally, chapter 7 sketches several action points where more effort should be put in the future, as a result of LONG project experience.

1.1 General Conclusions

The main focus of the LONG project has been the production of guidelines for the design, configuration and deployment of Next Generation Networks, so that standard network services and applications can be supported across them. LONG has addressed the design and deployment of IPv4/IPv6 transition scenarios and has validated the integration of IPv6 with advanced services such as QoS techniques, mobility mechanisms and multicast support.

In this document, the general conclusions and results obtained by all the technical WPs as well as general conclusions related to the LONG project are presented.

As one of the main aims of LONG, transition to IPv6 has been thoroughly studied. There are three points that summarize the impressions about transition to IPv6 in the LONG project:

- There are many transition mechanisms which can be used in many different scenarios of use. Most of them have been tested within the project, so there is enough information to help in the definition of each transition scenario.
- Transition to IPv6 is not only a network issue, but also an application problem. Transition to IPv6 requires the analysis of end-to-end solutions. For instance, applications which manage network addresses demand depth revision and extended code rewritten. LONG has provided general guidelines to help transition of standard and not standard applications.
- LONG has demonstrated that it is possible to start current network transition to IPv6.

More general conclusions about IPv6 and how it has evolved, drawn after more than two years working in the LONG project, are sketched in the following lines.

During the last years, most of the network equipment manufactures have implemented IPv6. This has enabled LONG to deploy stable IPv6 networks using basic network services. However, concerning to the advanced network services, the implementations are not yet stable, they are still very close to last evolutions of specification documents, being this the case of mobility, for example. Another issue is the implementation of transition mechanisms on networks in large-scale for which some problems still remain. Most of the routers from different manufactures implement transition mechanism based on tunnels (6to4, automatic and configured tunnels) but few implement transition mechanism based on translators between IPv6 and IPv4. There is also a lack for transition models that could serve seamlessly to a large number of users.

The lack of IPv4 addresses affects primarily to Europe and Asia and it is therefore a consequence that in Europe and Asia is where IPv6 is being first introduced. In this sense there is already an IPv6 backbone, NTT-Verio, which offers IPv6 communication services throughout Europe, Asia/Pacific and Australia. The ten most important ISPs in Japan use NTT-Verio, as a differentiated service offered by NTT, to provide end users services using IPv6. As an example of this commercial services there are more than 300 users who already use IPv6 ADSL access instead of the IPv4 one because the IPv6 address does not change and it gives an added value that is not possible with IPv4.

Besides this initiative in Japan there are other pre-commercial networks in Asia and Europe which aim is to introduce IPv6 to encourage Telcos and ISPs to provide full IPv6 services. One of these initiatives that has to be taken into account is Euro6IX where several LONG partners participate to pave the way for IPv6.

In fact, the introduction of the IPv6 is being done slowly. However, this process should be accelerated to prevent the lacking of addressing. This problem is more relevant for Internet solution vendors and Internet service providers, as well as for the implementations of mobile communications. In case of success of the UMTS networks the use of the IPv6 could be accelerated because of the available addressing space and, it would probably be the first public IPv6 network infrastructure in large-scale.

The introduction of IPv6 seems to be inevitable in the next years. The continuous growth of users and the emergence of services "Always-on" will lead to exhaustion of the amount of available IPv4 address today. This will compel the Telcos/ISP to migrate its networks to IPv6.

All the partners of the LONG project have deployed an IPv6 test network, which in most cases extends to outside their premises. This testbed have enabled the acquisition of operational experience for the configuration of IPv6 equipment from different manufactures and will continue to be used as a platform for applications and services testing and demonstration, being continuously updated.

As part of the work deployed, different applications, like multicast, QoS, muti-homing and mobility, were tested in order to analyse the functionalities as well as its reliability. Some of them (for example QoS and multicast), present a framework very similar to the IPv4 case. For other services, such as muti-homing or anycast, brand new mechanisms and policies are required. Also, the functional aspects of applications and network services using transition mechanisms were studied and documented. The complexity of trying to provide advanced

network services (such as QoS, multicast, etc.) while the transition is taking part is very high, and new solutions are required for this.

As result of the knowledge acquired, PTIN and TID, as the Telcos in LONG project, will conduct IPv6 consultancy and dissemination activities inside PT and Telefonica Groups, including meetings with decision staff/committees, workshops and demonstrations. Also these companies will perform various IPv6 courses and training for telecommunications engineers of the business units.

The universities involved in the project have started local networks transition to IPv6 in the near future and they have played a key role to enhance Spanish and Portuguese NRENs (RedIRIS, FCCN) migration to IPv6. They will also provide consultancy on the deployment of new networks, for which their IPv6 expertise will be considered as very valuable. Training of skilled IPv6 engineers is also of main concern for the success of the deployment of this new protocol.

As the only equipment provider partner in LONG project, Nortel Networks fully supports the move to IPv6. It is a pioneer in the implementation of IPv6 in enterprise networks since 1997 when IPv6 was first implemented in Nortel Networks BayRS product, with initial shipments in the same year. BayRS is a suite of routing protocols running on several platforms including BN (Backbone Node), ASN (Access Stack Node), ARN (Advance Remote Node), Passport 5430 and Passport 2430. The ASN was part of Nortel Networks testbed in the project. It is the aim of Nortel Networks to continue to support IPv6 in other future core networks products.

1.1.1 WP2: "Network Design and Deployment"

The WP2 aims were to define, specify and deploy an IPv6 test platform that enabled the experimentation and functional evaluation of services and applications based on this protocol. Various requirements were defined from the beginning of the project. One of these requirements was to incorporate different access and transport technologies, such as ADSL, CATV, WLAN, ISDN and ATM, in order to study and evaluate the implementation of IPv6 over these technologies and the particular characteristics of each one. Another objective was to incorporate IPv4 to IPv6 transition mechanisms and advanced network services, such as IP mobility, QoS, multicast and security, in an almost real scenario.

Studies and experimentations were carried out in order to deploy the test platform. These were divided into three phases:

1. Study and experimentation of the transition mechanisms as well as the identification of transition strategies.
2. Study of the IPv6 implementation over different access and transport technologies.
3. Study of advanced network services based on IPv6 protocol and identification of the "state-of-the-art" on standards and existing implementations.

The project focuses on collaborative work applications requirements when trying to characterize network performances because this type of applications requires high performance network. Therefore, this WP studied and analysed stable available links, of high bandwidth, to establish the connections between the partners, including the international connections between Portugal and Spain.

1.1.2 WP3: "Collaborative Work Environment"

1.1.2.1 WP3 Work Scheme

WP3 working group has been devoted to define requirements and procedures for migration of applications to the new generation networks. It includes the requirement selection on the new generation network of advanced collaborative applications and the selection and adaptation of collaborative work applications to new IPv6 and transition scenarios, including network scenarios with asymmetric links and or specialized transmission components. Therefore, the work has been divided into three phases.

The first phase has been devoted to review different kinds of applications like collaborative environments, remote experimentation (distributed labs), instrument control or distance-independent instruction (distance learning) and study how the applications can be migrated to an advanced IPv6 network. The emphasis during this phase has been to study the simple point to point applications, including the sockets programming interface. Simultaneously, a set of the most relevant services has been selected to be operative inside LONG network. Some of them have been directly installed because the migration was available. The rest has been migrated, installed and tested inside the project. The summary of this work has been described in detail in Deliverable D3.1.

The second phase has been focused in the study of more sophisticated end to end applications, mainly collaborative applications. The work has consisted on the review of different kinds of applications like collaborative environments, remote experimentation (distributed labs), instrument control or distance-independent instruction (distance learning) and how the applications can be migrated to an advanced IPv6 network. After the study, a set of general migration rules has been defined. These migration guidelines have been tested with ISABEL collaborative application. ISABEL has been migrated following the rules defined previously. After migration, some minor changes have been introduced in general recommendations. The summary of this work is included in Deliverable D3.2.

Finally, during the third phase special scenarios have been considered. The focus has been the special requirements communication networks, like asymmetric channels or specialized transmission components of some sophisticated collaborative applications. After the revision of all these scenarios migration guidelines have been produced. The summary of this work has been described in detail in Deliverable D3.3.

1.1.2.2 WP3 Main Conclusions

WP3 has been focused on the production of guidelines of applications migration to the Next Generation Networks. Applications migration process includes reviewing and rewriting code, and sometimes a partial application redesign. The recommendations for application migration are difficult to summarize in a few words. Therefore, we decided to produce a new document, identified as Deliverable WP3.4, to group the migration guidelines for all scenarios reviewed in the project.

IPv4 applications can operate with IPv6 applications using dual stack mechanism. The transmission will be carried out exchanging IPv4 packets with the help of dual stack. However, dual stack mechanisms do not, by themselves, solve the IPv4 and IPv6 interworking problems. When IPv6 packets transmission is required, the source code of existing applications must be changed in order to adapt it to the new IPv6 environment. Sometimes

addresses translators are used to the direct translation of protocols, including headers. Protocol translation often results in features loss. For instance, translation of IPv6 header into an IPv4 header will lead to the loss of the IPv6 flow label. Translation can be complemented with tunneling; which used to bridge compatible networks across incompatible ones.

However, the real migration process implies exhaustive code revision. The most important considerations that should be taken into account in the migration guidelines are the following:

- Communications management is isolated from the rest of the application components.
- Network compilation options are not dispersed all over the code.
- Application is independent of the IP addresses management.
- Application use only names but never IP addresses to select remote node.

All these requirements can be implemented as a single library. It is a middleware network interface to isolate communications from the rest of functional application modules.

The full description of previous migration guidelines and a prototype of middleware network interface can be found in D3.1, D3.2 and D3.3 documents. WP3.4 is a concentrated summary of all of them.

1.1.3 WP4: "System Exploitation, Trials and Evaluation"

To be able to fully understand the results provided by WP 4, it is convenient to present its context. This workpackage was devoted to the definition, specification, execution and analysis of tests and real user trials over the network developed as a result of WP 2. It was also expected of Workpackage 4 to generate feedback for network deployment and to provide valuable information regarding to application and advanced network services deployment in a next generation infrastructure, especially when IPv4 to IPv6 transition mechanisms are required. While WP is aimed to the theoretical analysis of transition mechanisms, technologies, and network and application services, including their basic configuration guidelines and basic functional testing, WP4 is dedicated to the execution of performance tests and advanced functional trials in close-to-real scenarios.

ISABEL was considered the reference application for evaluating and testing the LONG network, since it is a complex application ported to IPv6 with stringent network requirements such as real-time delivery. This application is used for real user trials, either used by internal users (LONG meetings) or external ones. ISABEL was kept on mind when defining the required testing scenarios: it has provided information for specifying traffic characteristics (a flow with constant bit rate, etc.), and the relevant parameters to measure for traffic performance (packet loss, delay and variation of delay). ISABEL has also been tested in scenarios in which different transition mechanisms have been deployed.

The behaviour of advanced applications like ISABEL can be enhanced by proper configuration of simple services such as DNS. It can also be achieved by tuning more advanced services such as support for quality of service, multicast, multi-homing, mobility, etc. These advanced applications are expected to be deployed over different transport technologies such as ATM, or over access technologies like ADSL, CATV, ATM, ISDN 802.11b, Ethernet, etcetera, so the implications of IPv6 deployment over these particular technologies are also worth to be deeply analyzed and exercised in Workpackage 4.

Therefore, testing these network services and technologies, and quantifying the global performance over current implementations when appropriate, is an important task for the LONG network design and deployment processes, and as a side effect, for the evaluation of IPv6 maturity. In general, the performance results can be valuable for people asking "will I loose performance when switching to IPv6?". Additionally the obtained results can be used with the convenient adaptations for each particular case as guidelines to fine tune networks in which applications demanding real-time delivery are relevant. However, it should be noted that complex scenarios lead to an extremely large number of parameters and configurations that should be exercised. In the LONG framework, we have chosen to exercise simple configurations for which preliminary relevant conclusions could be extracted; in some cases, these simple configurations have allowed us to identify that the particular technology was not mature enough at testing time.

Another set of performance tests is related with the testing of applications in order to assure that their behaviour is appropriate for being used on events in which large number of users are involved (for example, the Campus Party 2002). For this purpose, stress tests were carried out to check beforehand that the available DNS and HTTP servers would cope with the expected load.

The deployment of other application services, aside from ISABEL, was also analyzed in depth. From the LONG point of view, the deployment of the network should be performed accordingly with the real needs perceived in the deployment of applications

The strategy for the deployment of an application service in a mixed IPv4/IPv6 environment should be tailored considering its own specific communication requirements, for example, clouds of servers that communicate with each other in some specific way. From this point of view, the deployment of different application services (DNS, mail, IRC, LDAP, etc.) raises new demands for network service deployment.

On the opposite direction, the experience achieved in the operation of fundamental network services can provide valuable knowledge for application service deployment.

To fulfil the objectives presented above, two test approaches were defined: on one hand, local tests that allow exercising exhaustively some technology in a given configuration are defined; on the other hand, we have specified tests involving several partners, reflecting complex and more realistic scenarios that include heterogeneity in equipment, configuration diversity, etc.

2. Network Design, Deployment and Testing

2.1 Network Design

LONG network detailed design and deployment issues can be found in LONG D2.4 "Network Design and Deployment", while this section will try to summarize and introduce the main points and starting conditions.

Regarding basic networking work, LONG aims to interconnect the NGN laboratories of the different partners so that useful experiences on advanced services applications could be performed. For this purpose, the IPv6 protocol was selected, as it is the recommended TCP/IP next generation stack by the IETF. Then, besides the work related to applications and services, some research and tests about network services and IPv6 support were also necessary.

2.1.1 IPv6 Distributed Lab Concept and Design Alternatives

The first point to address was why are IPv6 distributed Labs needed. It has been concluded that "Stable" R&D testing platforms are needed to:

- Adapt current network and final-user services to v6.
- Deploy new network and user services in v6 Networks.
- Ensure IPv4-IPv6 interaction at Service level.
- Since some networks/scenarios are to be tested, different partner networks must join the experiment creating a single conceptual platform: **An IPv6 distributed laboratory**.

It has also been experienced before that current commercial Internet services come from IPv4 R&D stable Labs linked together.

Then, an IPv6 Distributed Lab is made of:

- v6 Nodes: which are the different partner testbeds.
- IPv6 Backbone: links, routing policy, mechanisms to access from one partner to the others.

For the backbone Links Deployment some alternatives have been found in the context of IST research projects:

- Public L2 Networks: Guaranteed BW, expensive for R&D purposes.
- Research networks: Still Based today in IPv4: Tunneling needed. Deploying QoS schemes at IPv4 level.
- NRNs/NRENs: within a single country.
- GEANT: connections over 2 or more NRNs/NRENs.
- Tunnels over Internet: Straight solution but best effort and low BW.
- Large IPv6 projects (New Alternative): Euro6IX / 6NET.

In LONG, all these alternatives have been not only considered but also tested and compared.

Some bandwidth considerations were done when deciding the right alternative for each link:

- Best effort & Low BW enough in most functionality/basic services experiments (WEB, IRC/Chat, some on-line games, LDAP, news, mail …).
- Guaranteed High BW is needed in some user services experiments (ISABEL, video streaming).

To ensure the complete description of the LONG network 3 complete maps are needed:

- LONG Backbone - Logical Links: Describes the geographical situation of each laboratory to join and which of them are directly linked.
- LONG Backbone - Physical Links: Describes the concrete implementation of each link focusing in the possible networks crossed by each link.
- LONG Distributed Lab (Global/Conceptual View).

Other network/service maps have been needed in other contexts. For instance, ISABEL meetings and events need a global map describing the connection tree at the OSI Layer 7 (Application) performed by the different ISABEL instances and components running in each remote site.

2.1.2 LONG Backbone: Logical Links

The Long backbone interconnects the access and local networks developed by the partners at their premises. During the initial phase, various possible solutions were analyzed for the interconnections of the partner's networks. The network was designed according to the available bandwidth and stable links to build a high performance network. This requirement was defined to allow the performing of conferences over the LONG project IPv6 infrastructure

The figure below presents the logical connections between the partners. These connections are provided by research and commercial network infrastructures. Within each country, Portugal and Spain, the infrastructure of National Research Networks or National Research and Education Networks (NRN/NREN) is used.

The interconnection of the NRN/NREN is supported by GEANT network. This network substituted the previous pan-European research network, TEN-155, since December 1st, 2001.

UPM, UC3M and UPC are connected through the RedIRIS (Spain's NRN) and TID is connected to UPM through Telefonica's ATM backbone.

UEV is connected to UPC. This connection is established through the RedIRIS and FCCN network, which are interconnected by GEANT network. The connection from UEV to FCCN is supported by ATM connection, where the bandwidth is shared with Internet services and traffic to other Portuguese Universities.

NOR is connected to UPC via UPCnet, the ATM backbone of the UPC.

Until November 2002, PTIN was connected permanently to TID with an IPv6 tunnel through Internet. Also, there was a connection on demand to UPM, using PIP service of GEANT network. As the GEANT PoP is located at Lisbon, on FCCN's network (Portugal 's NREN), the connection between PTIN premises and FFCN is supported by ATM commercial service provide by PT Prime.

Due to the cost of commercial connection services, PTIN–UPM connection was only established when the bandwidth was demanded, for example, to perform a conference.

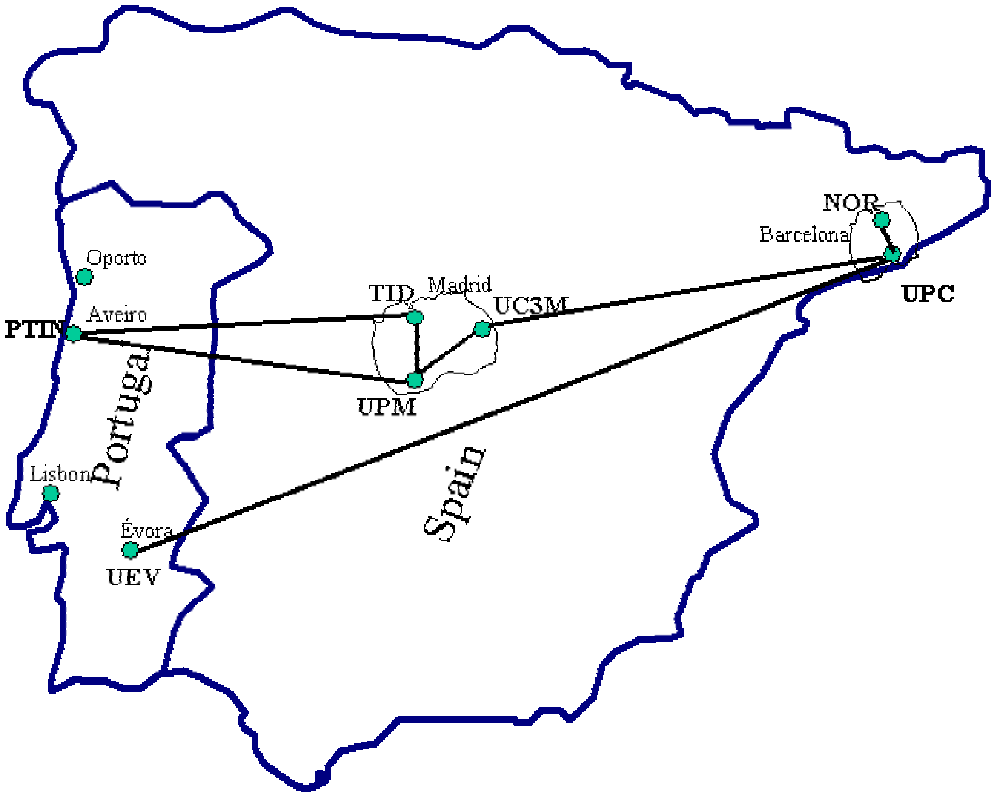


Figure 1. LONG Logical Links.

The BGP4+ routing protocol is deployed on LONG IPv6 backbone. In order to simplify the establishment of PTIN-UPM connection, the link from PTIN to TID was kept as backup. PTIN-UPM link (trough FCCN, GEANT an RedIRIS) was usually down and , in this case, the traffic from PTIN was sent through TID. When the PTIN-UPM link was established in order to assure a high bandwidth, the BGP-learned routes were update automatically and the traffic from PTIN was sent to UPM.

In the final phase of the project, a new connection was established between PTIN and UPM through the Euro6IX infrastructure. Besides the permanent connection between PTIN and TID was changed and instead of the configured tunnel over the Internet the actual connection is also through the Euro6IX infrastructure.

As the routing on Euro6IX is not define yet, static routes are used in PTIN-UPM and PTIN-TID links and UPM advertises PTIN's prefix to the rest of the partners.

2.1.3 LONG Backbone: Physical Links

In this section we present the set of physical connections used to deploy the LONG IPv6 Backbone. The following picture present network topology deployed.

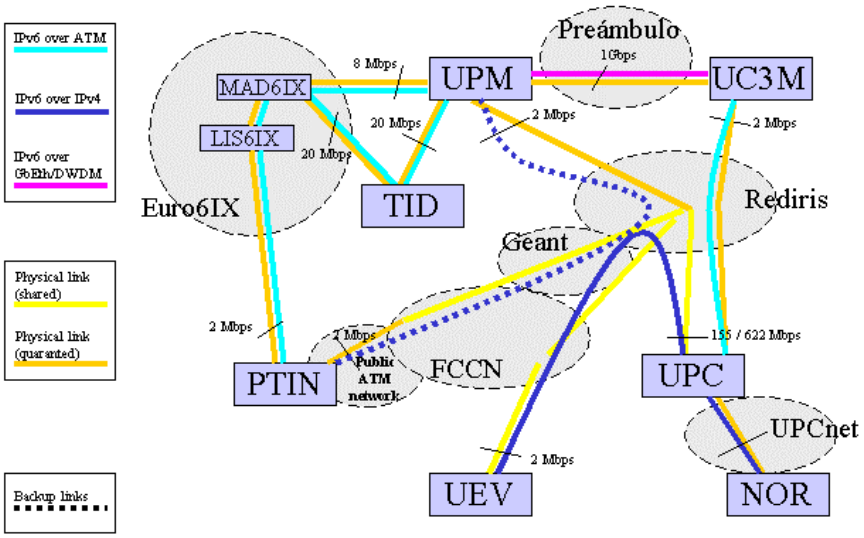


Figure 2. LONG Physical Links.

The orange an yellow colours are used to represent layer 2 links and their availability, and dark and light blue colours to represent the IPv6 links. The light blue represents the links using native IPv6 over ATM and dark blue colours represent IPv6 over IPv4 tunnels over ATM link. Another technology used, it is the DWDM links between UPM and UC3M. On this link is used IPv6 over Gigabit Ethernet.

As NRN/NRENs and GEANT networks do not provide Layer-2 connection services but IPv4 connections, IPv6 over IPv4 tunnels are established.

The most relevant changes from the picture shown in Deliverable 2.4 are the use of a DWDM link between UPM and UC3M, which belongs to the PREAMBULO project, and the Euro6IX infrastructure between PTIN and UPM and PTIN and TID.

2.1.4 LONG Distributed Lab (Global/Conceptual View)

This project focuses on the IPv6 as the main communication protocol along with the integration access and transport technologies as well as user and network services in Ipv6 environments. Another aspect that has been considered is the usage of level services in IPv4/IPv6 mixed scenarios. The conceptual design of LONG's network is shown in the following figure.

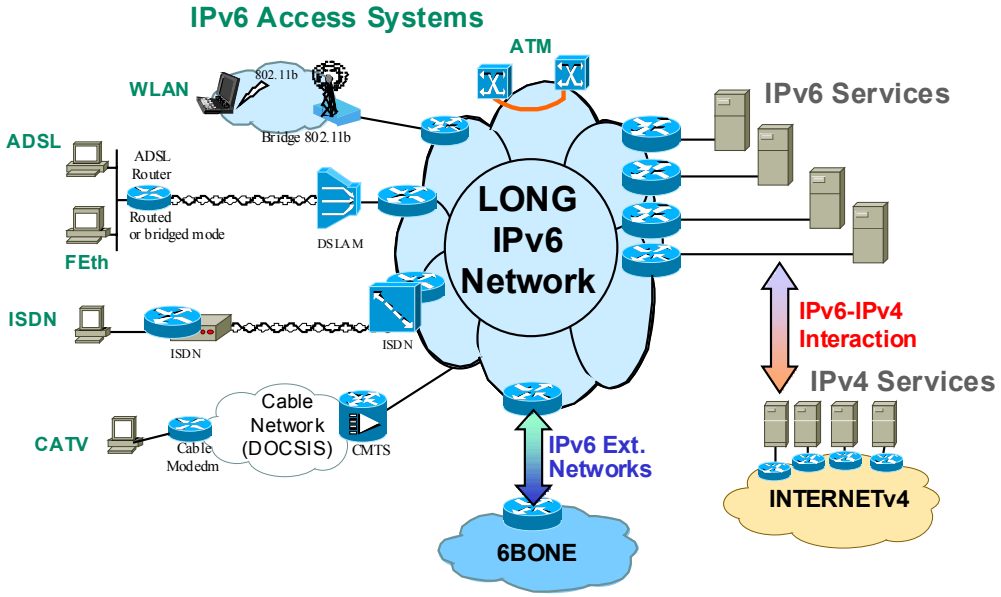


Figure 3. LONG Services and Access Systems.

This picture shows the main access technologies deployed over the LONG's network. These technologies are deployed at the premises of each partner and attached to core. The transport technologies are deployed in the point-to-point links between projects partners.

The LONG's network has allowed to test and evaluate applications and end-user services in an almost real scenario.

2.2 Transition mechanisms

The transition to IPv6 should be done as smooth as possible to prevent the disruption of the current IPv4 services, so the transition mechanisms should guarantee the gradual transition.

Regarding to IPv4/IPv6 transition mechanisms, several have been proposed, each one aimed to a different scenario. First of all, dual-stack hosts (hosts with both IPv4 and IPv6 capabilities) are expected to allow IPv4 and IPv6 applications to coexist in the same host. Additionally, if the network is deployed with dual-stack routers, exchanging routes for both the IPv4 and the IPv6 domain, transition can proceed smoothly. However, legacy IPv4 networks are expected to be common, especially at the first transition stages. For allowing the communication of IPv6 hosts or islands among IPv4 networks, tunneling can be used. Configured tunnels [RFC1933], that is, IPv6 tunnels manually established over an IPv4 network, are the most popular way of providing this type of connectivity. IPv6 route information can be distributed to deploy an IPv6 overlay network over existing IPv4 infrastructure, in a similar way as occurs in MBone. The popularity of manual tunneling relies on the administrative control they provide for network managers. However, in some cases, more automatic deployment is required and can be achieved by the 6to4 tunnel [RFC3056]. Provided that a single IPv4 address is available in a site, 6to4 tunneling provides an address range, and easy to configure connectivity. The automatic tunnel [RFC1933], another tunneling proposal for inter-domain connectivity, has the drawback of requiring one IPv4 address (a scarce resource) per host, so it is much less popular. Multiprotocol BGP has also been proposed for connecting IPv6 islands over IPv4 networks, providing another means of automatic tunnel [De Clercq 02].

Other existing solutions for providing communication among IPv6 host and routers that are connected through IPv4 infrastructure are 6over4 tunneling [RFC2529], benefiting from the IPv4 multicast service for linking host and routers, and ISATAP [Templin 02], that relies on DNS to provide information for establishing the IPv6 virtual segment between hosts and routers.

Finally, DSTM [Bound 02] has been proposed for allowing communication among IPv4 applications when one of them is running on a dual-stack host placed in an IPv6-only network. In this case, an IPv4 over IPv6 tunnel is established to carry IPv4 packets out of the IPv6-only network.

Tunneling, specially configured tunneling is a widely deployed mechanism for providing IPv6 connectivity. Its main drawback is the isolation of the IPv4 domain: the information conveyed by a routing protocol or ICMP cannot be easily propagated into the IPv6 world.

Another problem is allowing the communication of IPv6 applications with IPv4 ones. This scenario is expected to be a common one: for example, an IPv6 web browser accessing an IPv4 web server. Translation is required at some stage. SIIT [RFC2765] provides translation at IP level, on a packet to packet basis. This translation cannot preserve all the valuable information of the packet (for example, large IPv6 routing headers), and has to carefully deal with ICMP translation and fragmentation. The hardest part, address translation, can be solved by NAT-PT [RFC2766], a Network Address Translator that cares about IPv4 to IPv6 address translation and vice versa. While for NAT-PT a single translator serves a whole site, BIS [RFC2767] is a very similar mechanism intended for being embedded in the stack of the host; on the other hand BIA [Lee 02] operates in the host at the socket layer. Translation at transport level is performed by TRT [RFC3142], avoiding fragmentation problems. And the SOCKS application layer gateway can be modified for relying traffic among IPv4 and IPv6 domains, as SOCKSv64 [RFC3089] proposes.

The large experience with NAT systems, including Application Level Gateways for translating application protocols that carry network data, makes NAT-PT the most mature and popular translation tool. However, advanced implementations such as FreeBSD still do not provide support for some NAT-PT features such as state acquisition from DNS communication, required for bidirectional NAT-PT. This functional lack forces the burden of placing static configuration in the NAT box when communication is initiated from the IPv4 side. Additionally, the experience with this particular implementation is that there are still some instabilities to fix to be ready for operational environments.

Analyzing transition mechanisms, it can be stated that translation has to be avoided when possible, since it breaks the end-to-end principle, therefore affecting important services depending on it such as end-to-end security. Connections initiated by IPv4 hosts are in general handled with great difficulty. Routing among both domains is also a concern: packets can be lost although alternative paths could be available. Finally, translation mechanisms do not scale as desired: either a translator is deployed on each host, with the associated burden for the administrator, or a NAT-PT has to maintain state for a large number of hosts. There is a lack of mechanisms offering a translation service among large IPv4 and IPv6 networks (to isolate clients from this duty).

The majority of manufacturers have implemented in their IPv6 commercial equipment transition mechanisms based on tunnels. These mechanisms include manually created tunnels, such as configured tunnels, and fully automatic tunnel mechanism, such as IPv4-compatible

and 6to4. The transition mechanisms based on IP tunnels present enough stability. On the contrary, the transition mechanisms based on IP translation are still being developed and the current implementations are experimental versions.

Trials have been performed with the most relevant transition mechanism. Some of them - configured tunnels, 6to4, NAT-PT, TRT and ISATAP – are part of the stable configuration network, and are used to provide seamless access from both IPv4/IPv6 clients to a given service, as it will be detailed on the user services section. Other transition mechanisms have been tested on a more temporary basis, such as automatic tunnels, 6over4 tunnels, DSTM, or SOCKS, being less appropriate for general usage, or still on an immature state.

2.3 Access technologies

The introduction of the IPv6 in the Internet has been widely discussed. There is a generalized opinion that the introduction of IPv6 will be performed from the edge of the network to the core. On the other hand, we have to take into account that the emerging applications require a high bandwidth to the home of the client. Therefore, the implementation of the IPv6 over broadband access technologies has fundamental importance.

Various access technologies have been discussed, namely UMTS/GPRS, Wireless LAN, ADSL, Cable TV, ISDN and ATM. All these technologies support IPv6, except the CATV. However, it allows the transport of IPv6 through tunnels.

In this case, there are still open issues, mainly related to missing standardization but it is still under investigation and discussion.

As a general conclusion of the study of these new mechanisms, it is expected that many new users and even devices will be introduced and connected to the Internet in an “Always-on” fashion, so many new IP addresses will be demanded in the following years. Also, not only human users and application/information servers will be interconnected but a new variety of networks with new purposes will, such as farms of sensors available to many other devices or users to take decisions and actions, robots, etc. **All these scenarios demand IP networks with large public IP addressing to provide end-to-end connectivity and useful security, mobile and multicast schemes.**

2.4 Advanced Network services

The current IPv6 implementations from various manufactures allow the implementation of an IPv6 stable network. However, concerning the IPv6 advanced network services, the implementations are not mature enough, for example, for muti-homing, anycast and QoS.

The usage of advanced network services in transition scenarios is still under investigation and discussion. Some transition mechanisms have been proposed to solve requirements of the different transition scenarios. So far, most of these mechanisms proposed have been designed to solve basic network services and do not address the requirements of advanced services (like mobility for example).

2.4.1 IP Mobility

IPv6 mobility has implementations that follow very close to the last evolutions of that draft, which is still under discussion in the IETF. However, there are a scattering of implementations

based on different drafts, in some cases, with no compliance among them. However, it should be highlighted that in general IPv6 is expected to provide better mobility support than IPv4.

Some manufacturers have implemented the draft 13 (for example, Cisco), which is available in the experimental versions. The last version discussed in the IEFT is draft 17, in which aspects related to security are defined. This implementation is only available in experimental version on PC platforms (Linux).

The implementation that was tested (MIPL/draft 15) in the project presents some instability, mainly when several consecutive handovers are performed. Additionally, the unavailability time for handovers, sometimes as much as 3 seconds, is a bit large for some applications requiring real-time data delivery. ISABEL has been tested in an IPv6 mobile environment, as can be seen in the figure shown below. A joint effort from LONG and the MOBY DICK project has resulted in a solution to reduce handover time [RGII].

The mobility in transition scenarios is being discussed and it is still in its initial stages.

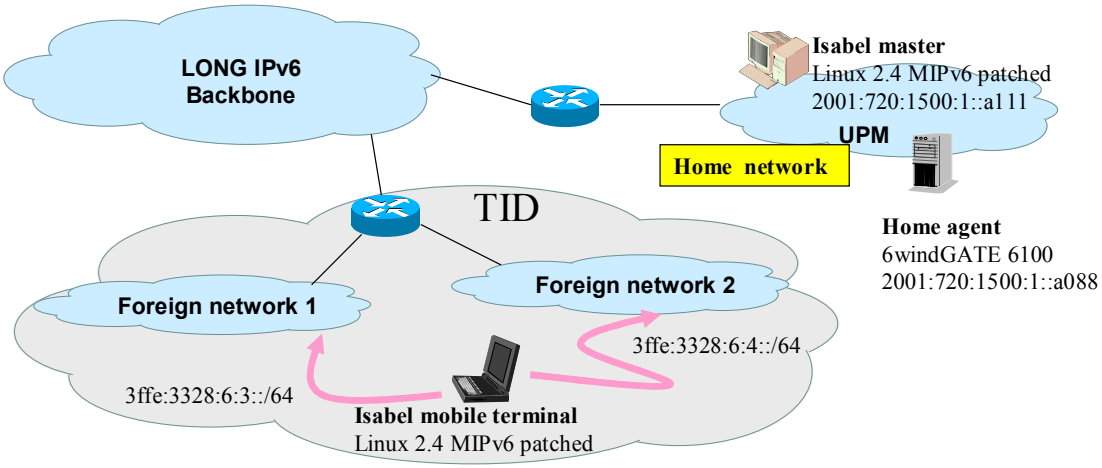


Figure 4. ISABEL and MIPv6.

2.4.2 Multicast

Like the IPv4 multicast, the multicast IPv6 can be divided into two parts: the host-router part and router-router part.

In the first part, the protocol used is the MLD (Multicast Listener Discovery), which allows the router to discover the nodes wishing to receive the multicast. This protocol was defined by the IETF and standardized in [RFC2710].

In the second part, multicast routing protocols are still being developed and they are not yet standardized. The PIMv6 is one of these protocols and has been implemented by some manufacturers.

For the deployment of a multicast network, all routers between the sender and the listener have to support the same multicast routing protocol. There are still few commercial routers with IPv6 multicast-capability. One solution to implement multicast, even across no multicast-capable routers, is the use of the multicast tunnels. The scenario deployed on the LONG network included multicast routers and tunnels over both PIM6 SM (Sparse Mode) and PIM6

DM (Dense Mode) protocols, using FreeBSD elements. The figure below shows one example of this configuration. Two members of the LONG project have joined to the M6Bone during the evolution of the project.

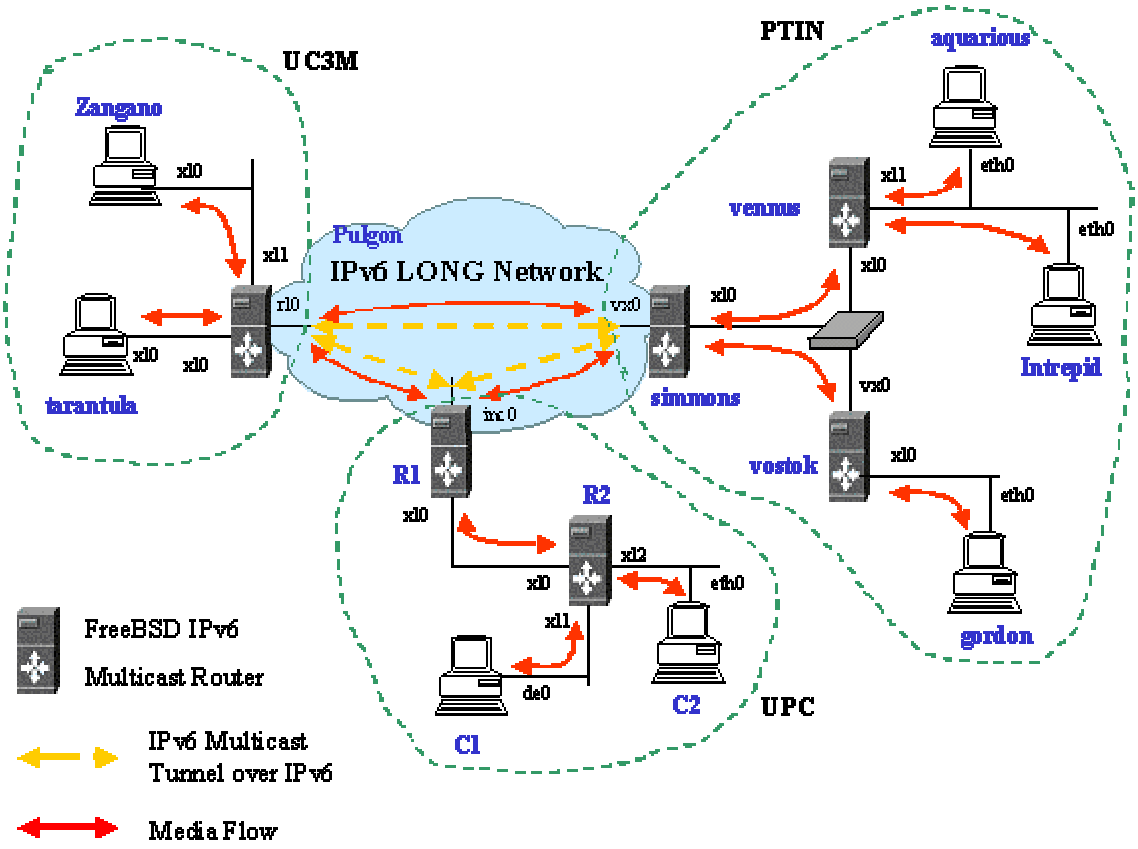


Figure 5. Multicast scenario tested at the LONG project.

Regarding to multicast in transition scenarios, a transition mechanism for multicast should be provided. This issue is under investigation and discussion. Meanwhile, two mechanisms have already been proposed namely the mBIS (multicast Bump-In-the Stack) and the MTP (Multicast Translator based on IGMP/MLD Proxying), both from Hitachi.

2.4.3 Anycast

Although there is little experience in the usage of anycast addresses, these addresses could be useful for automating tunnels and service access. In the future, the anycast addresses can be used to get fault-tolerance and sharing of servers, for example, a DNS server. There are several restrictions to its use: they cannot be used as source address so they cannot be used for TCP communication, and after the mechanism were better understood, its use should be restricted to routers (for example, to pinpoint a route to follow in a loose fashion). The main mechanism enabling anycast is the injection of prefix reachability, either just in IGP routing protocols, or in both IGP/EGP routing protocols. However, there are still open issues that need further study, such as security, scalability of the route injection procedure, and many more. In the LONG project, an experiment of a simple end-to-end use of an anycast address has been

made for locating a DNS server, using UDP access. For UDP stateless services, such as the DNS, anycast could be safe enough to be used. While this solution could ease host configuration (although other configuration alternatives could be available, for example based on multicast), it is not clear that the fault tolerance provided by the anycast mechanism exceeds the native DNS round-robin server lookup. Another issue to consider is the propagation of system/service availability, since a client would expect the anycast system to route it request to an available server, regarding the type of failure encountered (on the communication subsystem, in the host, or in the service itself), while current solutions address only communication failures. As a conclusion, it is clear that much more work has to be done on this subject.

2.4.4 Muti-homing

Muti-homing, i.e., obtaining Internet connectivity through two or more connections allows sites and hosts to improve the quality of their communications in some aspects, being fault tolerance the most obvious one. The availability of mutihoming is capital since a large number of organizations depends on critical applications built over the Internet. For IPv4, the muti-homing mechanism was clearly established: the injection of BGP reachability information over the interdomain routing system. However, the extended usage of the currently available IPv4 mutihoming solution is a main contributor to the explosive growth of the routing tables of the Network's core routers. IPv6 was initially designed to prevent massive injection of routes into the Default Free Zone, but recently the restriction for sub-TLA prefix injection has been removed, and no rules have been established to control routing table growth.

In the LONG testbed, classical BGP route injection approaches have been successfully tested, maintaining TCP connections in case of failure. However, it is expected that this solution alone could not be applied without restriction, so other options should be considered.

Other solutions have been considered. Tunneling between provider and customer site ("Muti-homing support at exit routers") provide a fault tolerance solution with limited scope, covering failures occurred in the path between both entities. Address selection through Default Address Selection (DAS) mechanisms with appropriately configured address selection tables has proven to be a convenient tool in the LONG environment in which different addresses to be used depending on the destination. DAS provided a communication path whenever one exists among two systems by sequential address selection, although it has to be considered that TCP connections cannot be preserved in case of failure.

Other proposals have been issued, although no one seems to fulfill all requirements in the common scenarios. A step further on the solution of this problem has been presented at [EXSMS].

2.4.5 DNS

New functionalities were added for the IPv6 support in DNS because of the size of the addresses. The "AAAA" records were defined to allow IPv6 hosts to be registered into the DNS tables and communicated among servers. The function of this record is similar to "A" record. Another characteristic is the possibility to have an IPv4 and IPv6 address associated with the same name record.

In the same way, to allow IPv6 hosts to be reverse looked up by IP address, the "IP6.INT" domain was introduced. Later, it was proposed to be replaced by the "IP6.ARPA" domain.

Currently both domains must be maintained while waiting for a definitive resolution, since most existent applications will search “IP6.INT” although best current practice is searching for “IP6.ARPA”.

Also, a new record type was defined to store an IPv6 address, and reverse lookups, in order to facilitate network renumbering. This record is called A6. The main advantage is the possibility to represent an IP address in which portion of the address can be changed without any action by the parties that controls the DNS zone.

The currently standard DNS record type for storing IPv6 is AAAA, and A6 has been sent to experimental status, due to the not fully understood risks it posses.

2.4.6 Autoconfiguration

IPv6 supports two types of autoconfiguration styles: stateless and stateful. The stateless configuration is a powerful mechanism included in IPv6 as a host to get configuration information automatically, standardized by IETF, and is widely implemented on commercial routers. It has the advantage of no requiring servers or other external entities.

The stateful configuration is a method to pass configuration parameters to a host from a server (Dynamic Host Configuration Protocol). This method is currently being defined in the IETF. The existing implementations are in general limited and unstable.

2.4.7 QoS – Differentiated Services

There are no significant differences in the conceptual framework of the Diffserv architecture between IPv4 and IPv6. The functionalities of the mechanisms implemented are the same for both protocols. Therefore, the kind of services that are being currently offered in IPv4 diffserv networks may be deployed for IPv6 networks. There are IPv6 diffserv commercial implementations available in some routers, such as routers from 6WIND, Hitachi and Thomson.

Several tests have been made on the framework of the LONG project, being the most relevant result that, in general, IPv4 and IPv6 behave similarly for the implementations that has been more extensively tested (Linux iproute2) .

The problems of implementation of Diffserv in transition scenarios can be divided into two groups.

The first group involves translation scenarios. In this case, diffserv routers are required on the IPv4 side, which use the TOS field, and diffserv routers on the IPv6 side, which use the Traffic Class field. A translator should be used to map the TOS field to the Traffic Class field and vice-versa.

The second group deals with tunneling mechanisms. In this case, the problem to solve consists of how the outer packets (IPv4) have a correct treatment with the DSCP field in the inner packets (IPv6).

2.4.8 Security

The IPv6 uses the IPSec as the generic security architecture at the IP level, and it is implemented with extension headers included in IPv6 packet. Within the IPSec, the key management protocols are the same for IPv4 and IPv6.

The scenario of the applicability of the IPsec is only possible on all-IPv4 or all-IPv6 networks.

There are various implementations of IPSec available. However, these implementations are still experimental versions and are PC based.

2.5 Network Service Experiments

A methodology has been developed for the execution of network performance tests, covering the selection of the tools, and the proposal of guidelines for test specification [D4.1]. Although the methodology is aimed to fulfil the specific LONG requirements highlighted above, it can be used for measurement in different environments.

2.5.1 Transition Mechanisms Tests

Regarding to the tests executed in [D4.2], we conclude the following:

The rate at which packet loss begins is one of the most relevant parameters measured, since it indicates when the resource reaches its limit. This value is obtained from the measured throughput vs. generated throughput graphic, when the mean throughput value becomes horizontal. Delay and jitter values for sending rates in which packets are lost have to be carefully considered, since these parameters are obtained in a saturated environment. For example, it would not be fair for the characterization of the overall delay behaviour to take into account the delay value obtained in this scenario (in the worst case, we are just measuring the effect of the particular buffer configuration of the network). But, in some other cases such as QoS testing, this is a relevant point, since it is expected that were met even in a saturated link.

In general, tests comparing IPv4 and IPv6 performance in PCs sending and receiving show similar results for both protocols, with a very small difference that stands for the transmission of the larger header size in IPv6.

Comparing the most common inter-network IPv6 to IPv6 connectivity configuration over IPv4 infrastructure (i.e. tunnels) with two Cisco routers connecting the networks, the most appropriate option, taking into account performance, is the configured tunnel. This is because it does not result in significant throughput reduction (unless a large number of tunnels are added, with traffic on them – that is not an expected scenario for a configured tunnel). Configured tunnels increase delay just faintly. 6to4 and automatic tunnels offer slightly less performance that direct IPv6 connectivity. In general, they are all acceptable solutions in terms of performance. Configured tunnels are widely used on the LONG network. 6to4 has also been used for ISABEL connectivity.

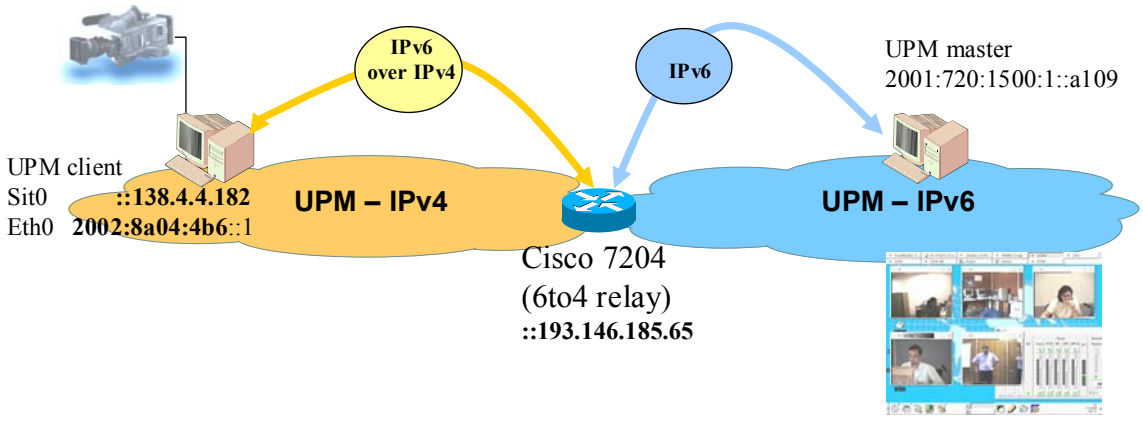


Figure 6. ISABEL access through 6to4.

For intra-network IPv6 to IPv6 connectivity in an IPv4 domain, 6over4 tunnels have been tested, showing an increase of the transmission delay. However, the requirement for IPv4 multicast makes this tunnel type unpopular. ISATAP is a protocol of growing popularity, that has been also used for this purpose.

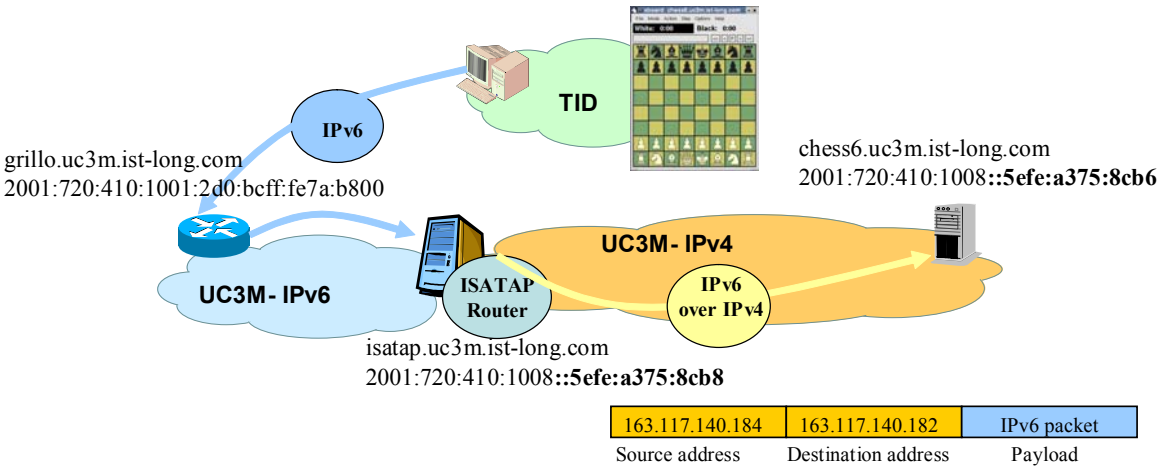


Figure 7. Chess client/server deployment through ISATAP tunneling.

Although IETF recommends avoiding translation whenever possible, in some cases it is required (for example, an IPv6-only host communicating with an IPv4-only host). The only mechanism with implementations allowing UDP packet translation in Unix environments is NAT-PT (thought it also performs TCP translation), so this option could be used if applications similar to ISABEL required translation. For the evaluation of NAT-PT, very different results are obtained regarding to the packet size used. When large packet sizes are used, around 1500 bytes, the translation overhead is negligible compared to the available link bandwidth (in these tests, Ethernet links with 100 Mbps capacity are used), and NAT-PT can

be used with a difference with the ideal case of less than 5%. Delay is also very similar to the case in which there is an intermediate router in the communication. This case is close to the ISABEL scenario, so it would be expected that ISABEL should work with NAT-PT translation, as the real experiments performed confirm (see figure below). Other real-time services has been successfully tested with NAT-PT. When packet sizes decrease, the number of processed packets per time unit increases, and the system overloads. For 62 bytes/packet, NAT-PT reduces throughput up to 60%, being the translation from IPv4 to IPv6 costlier than the opposite translation.

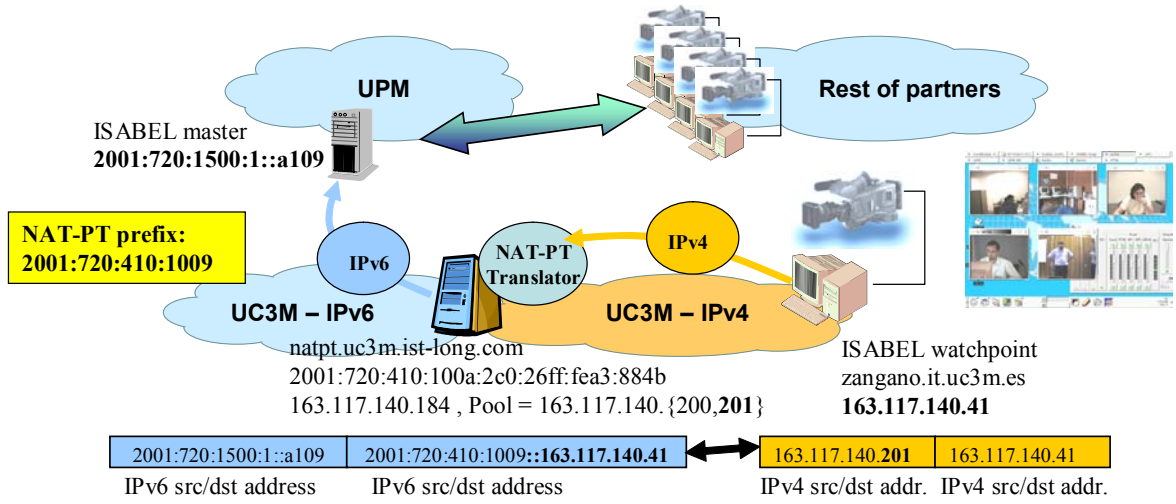


Figure 8. ISABEL with NAT-PT scenario.

TCP throughput has also been tested, using Netperf. TCP is used by a large number of applications, including ISABEL control flow. In this case, TRT has been introduced for comparison with NAT-PT, although some times is a good option for service deployment because it is simple to configure. TRT introduces a reduction in throughput compared to an IPv6 router with NAT-PT translating TCP below 20%. Delay of TRT is close to the NAT-PT case, although it is slightly worse. TRT has been used for the stable deployment of IRC service on the LONG network, as can be seen in figure 9.

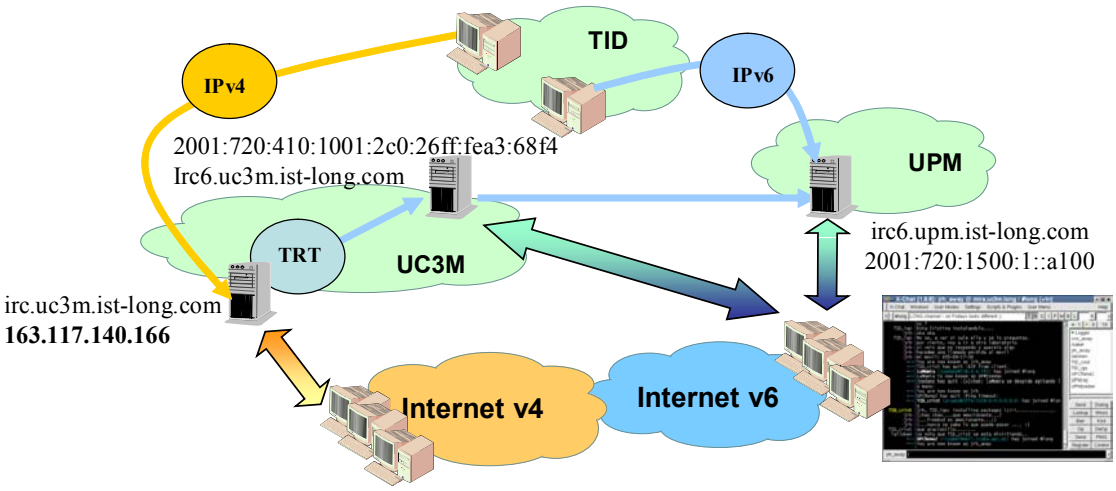


Figure 9. IRC deployment in LONG network using TRT.

We can conclude that all the tested transition mechanisms (all kind of tunnels, and NAT-PT) are appropriate for the rates required by ISABEL transmission. If other rates are to be used, it should be carefully studied the particular environment, since differences in packet size or the transport protocol used can be significant for the behaviour.

2.5.2 Access Technologies Tests

In the following paragraphs we state some conclusions obtained from the test of different access technologies.

For CATV, native IPv6 could not be used with the available equipment. The option that seems to be more convenient for providing IPv6 connectivity to a client is to establish a tunnel between the end host (or a router in the client network) to the CMTS, bypassing the cable modem device. With the tested equipment, throughput was similar for both protocols in the downstream case. The achieved rate when small packets are used is 2 Mbps, which is close to the ISABEL requirements in the common configuration. However with larger packet sizes, throughput increases up to 8 Mbps for IPv6. Downstream delay is below 10 ms for tests that do not saturate the channel. For CATV upstream traffic, restrictions are harder: IPv6 over large packet sizes transmission saturates at 1 Mbps. For small packet sizes, 900 Kbps is achieved with IPv4, and slightly over 700 Kbps for IPv6. Mean delay is below 50 ms, with a difference of around 40% between IPv4 and IPv6 transmission. With this available bandwidth, the ISABEL application would have to tackle explicitly with the asymmetry of the channel.

Regarding to ADSL, three different sets of tests have been performed, taking into account three different standard configurations (*Basic*, *Class*, *Premium*). For usual ISABEL transmission, no configuration – even Premium - is suited, since although Premium can deliver 2 Mbps downstream, it can only provide 320 Kbps upstream. The tested equipment does not offer native IPv6 support, so the most appropriate configuration to provide IPv6 connectivity to the customer relies in establishing a tunnel (IPv6 over IPv4) to the router that receives the resulting ATM traffic. For this case, throughput is affected mostly due to the increased overhead of carrying 20 additional bytes.

We must highlight the lack of native IPv6 support in the tested CATV and ADSL equipment; mature IPv6 implementations will still take some time for these technologies. This is a major obstacle for the deployment of IPv6 in the home and SOHO environments.

Experimenting with the 802.11b technology, we see that performance is reduced compared to 10 Mbps Ethernet (although the theoretical maximum throughput of 802.11b is 11 Mbps). For the 1452 bytes/packet test the available rate it is about 6,1 Mbps, while for the 62 bytes/packet test this value is near 600 kbps. These low values, especially those measured for small packet sizes, are due to the high overhead added by the link layer encoding. Delay depends on the packet size, but it reaches 100 ms. In this case, the overhead introduced by IPv6, compared to the use of IPv4, is insignificant, and variations are below 2%.

Regarding IPv6 over ATM, IPv6 measured throughput and delay are very close to the IPv4 case. Achievable throughput exceeds by far the rate required by ISABEL, and it is limited by CPU performance at the ends (saturating at 12 Mbps for small packet sizes, for both IPv4 and IPv6). It should be noted that the tested equipment currently do not provide some advanced features such as IPv6 support for SVC (only PVCs can be used), or traffic shaping for IPv6 flows.

Finally, a DWDM link traffic has been included in the LONG infrastructure. The link protocol used is Gigabit Ethernet and no special problem has been identified on its use.

3. User Services

Different application services have been deployed in the LONG network. The deployment of these services had led us to some conclusions about the deployment of user services over mixed IPv4/IPv6 networks.

First of all, different problems can be identified: IPv6 applications communicating across an IPv4 network (note that in a further transition stage, the opposite problem will arise), and IPv6-enabled applications interacting with IPv4 applications.

- The first case is solved by the network, through the use of tunnels, and is transparent to the application. We will not discuss this issue further.
- The second case, IPv6-enabled applications interacting with IPv4 applications, can be solved by the deployment of applications allowing communication using both protocols, provided that the applications run over dual-stack hosts, or by the deployment of translators.

The deployment of applications that can communicate with both IPv4 and IPv6 is a neat solution. This is especially useful for the deployment of servers. If a server is to be deployed in this way, either IPv4 or IPv6 clients can access the service. The restrictions due to translation (limited support for extension header translation, broken end-to-end relationship, etc.) are not an issue in this case. However, this model demands dual protocol connectivity for the host acting as server. And we have to consider that adding support for both protocols increases slightly the cost of application development. In the LONG network the deployment of DNS, Web, FTP, IRC, Mail, News and ISABEL follows this approach. To consider the particular case of ISABEL, IPv4 clients and IPv6 clients can connect to a FlowServer running in a dual stack host, since this component handles communication with both network protocols.

If the network does not allow communication using both protocols, but a dual-stack machine is hosting a dual protocol application, tunnels can be used to establish communication with another host using the protocol not natively supported by the network. For example, if an ISABEL FlowServer running on a dual-stack machine were placed inside an IPv6 only network, DSTM could be used for tunnelling IPv4 packets over IPv6 from a border router to reach the FlowServer (or a static tunnel could be configured for this purpose). This latter scenario is not being tested in the LONG network.

If the application does not support dual protocol communication, it is hosted in a single stack machine, or it is hosted in a dual stack host but the network does not allow by any means the communication for both protocols, protocol translation is required. Several services have been tested in such way in the LONG network. One of the design criteria is to place the burden of translation close to the server, not in the client side. The deployment of IRC, News and LDAP in the LONG network achieves this goal by offering different servers for IPv4 and IPv6 access, and translating the communication among the servers with a transition mechanism. IPv4 clients access IPv4 servers, and the same occurs for IPv6. When client to server translation cannot be avoided, as is the case for Web or FTP, proper DNS configuration is required for allowing easy access. Regarding to the particular translation technology to use on each case, either in server-to-server or client-to-server communication, the applicability of the translator and performance have to be checked for each application. For example, if translation should be required for ISABEL, NAT-PT would be the only choice, since it is the only available mechanism for translating both UDP and TCP traffic (NAT-PT is transport

independent). NAT-PT is a general mechanism that can be used in many cases, although it should be noted that, with the NAT-PT FreeBSD version available for testing, special configuration is required if an IPv4 node is to start communication with an IPv6 one. NAT-PT is deployed in the LONG network for translating ISABEL, LDAP, or games. Additionally, it supports FTP application level translation, so it is also deployed for FTP translation. TRT, that can be easily configured, is another option that is used for IRC deployment.

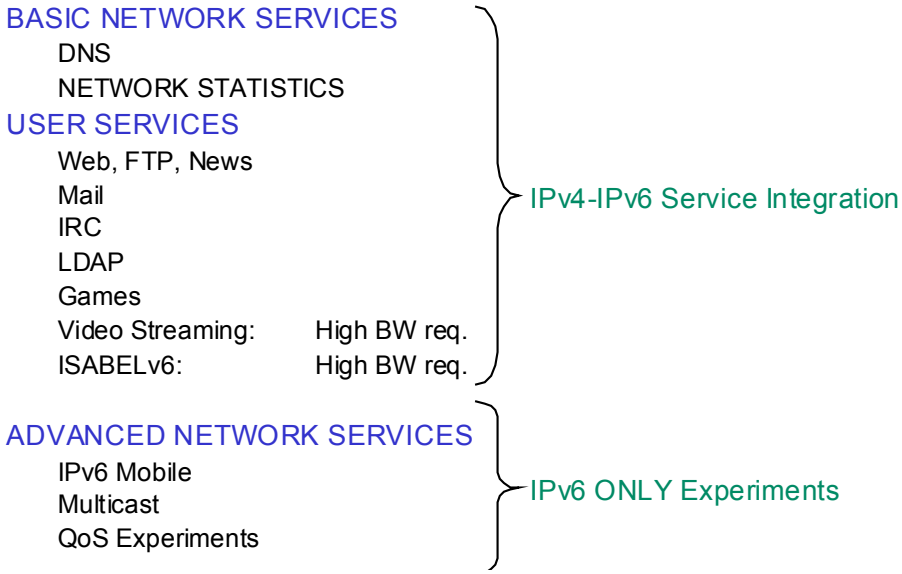


Figure 10. LONG Services List.

3.1 Basic Network Services

The services that are needed for the normal function of the LONG network are shown in the following figures.

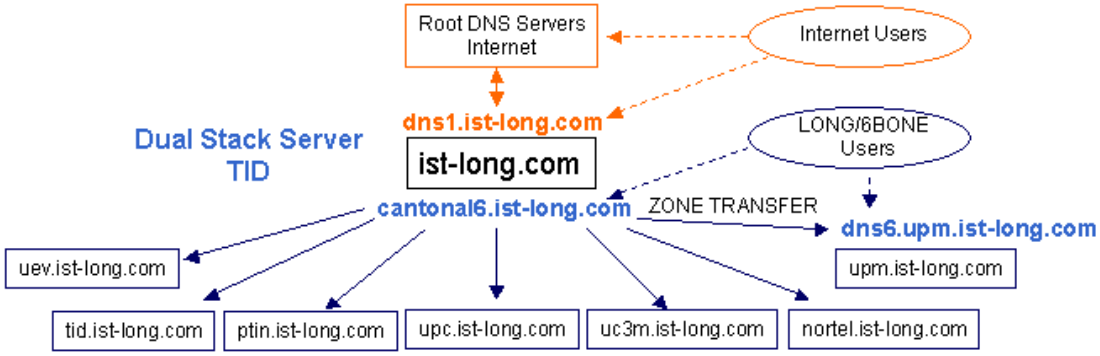
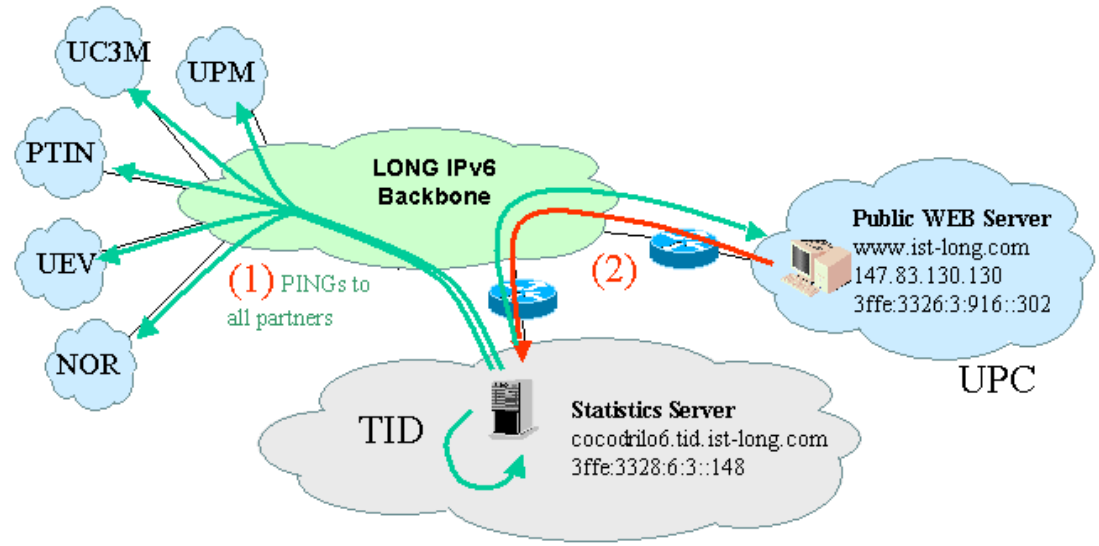


Figure 11. DNS in LONG network



Network Statistics

(1) TID collects statistics with the results from ping to all the partners, every 15 minutes.

(2) UPC gets periodically those statistics, builds graphics and makes them available both in the IPv4 WEB Server and the IPv6 WEB Server.

Figure 12. Operating of Network Statistics in LONG network

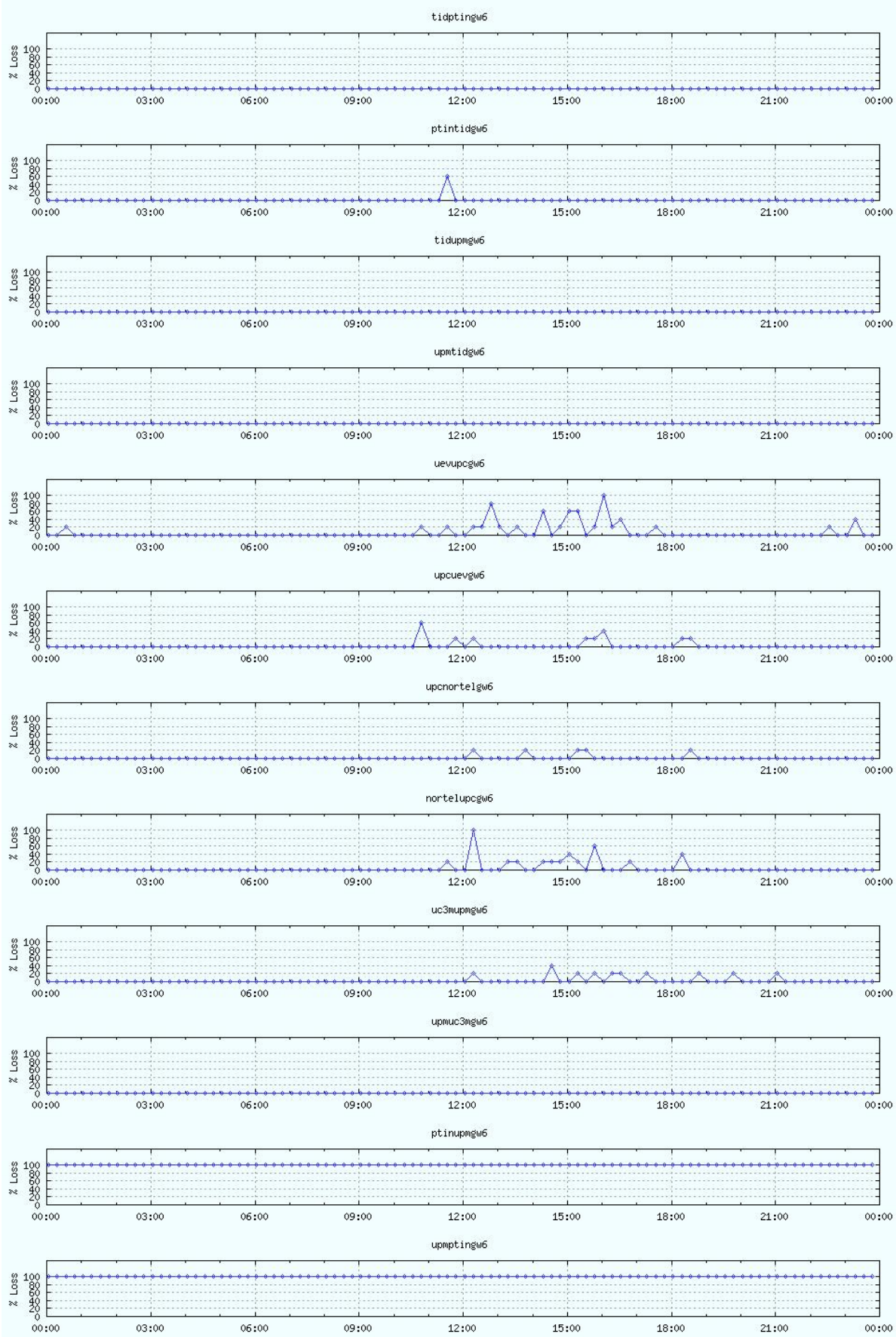


Figure 13. Example of Losses in LONG network

3.2 User Services

The user services are those that give a meaning to communication networks. The following pictures show the operation of these services.

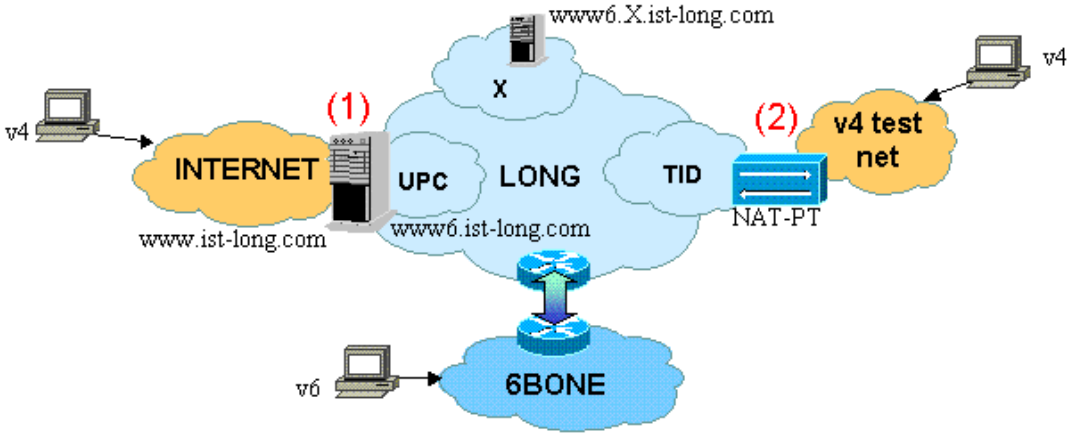


Figure 14. LONG WEB Service

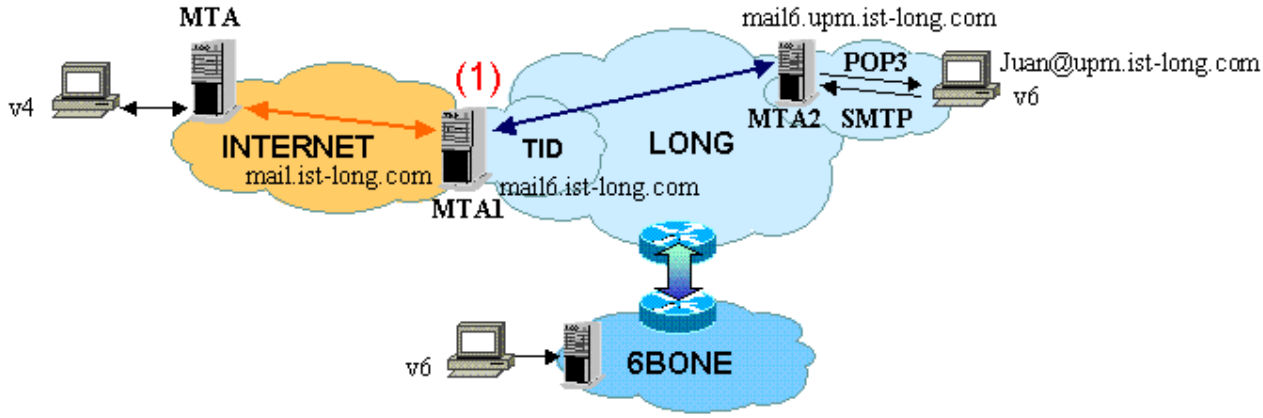


Figure 15. LONG Mail Service

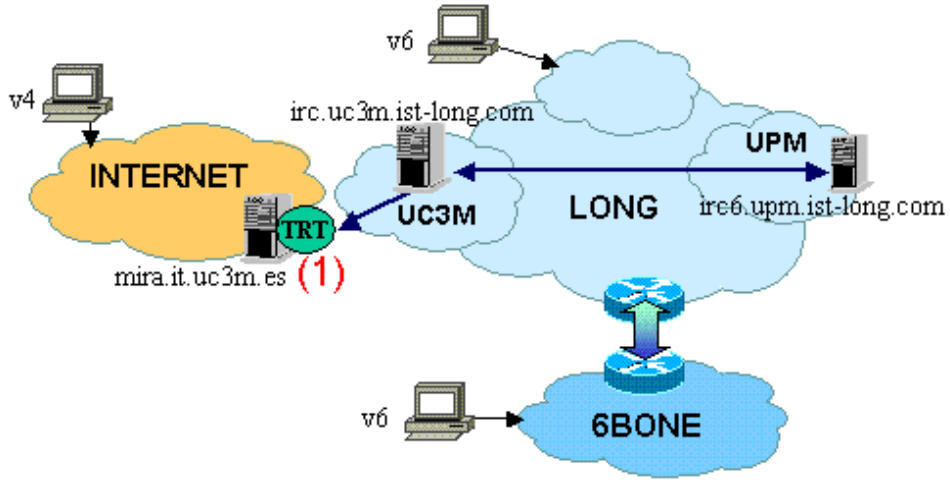


Figure 16. LONG IRC Service

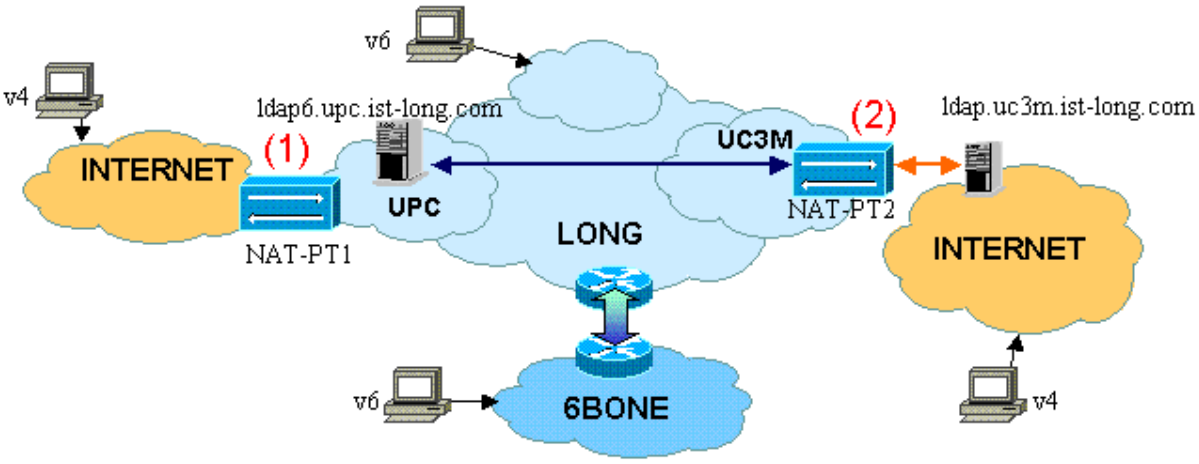


Figure 17. LONG LDAP Service

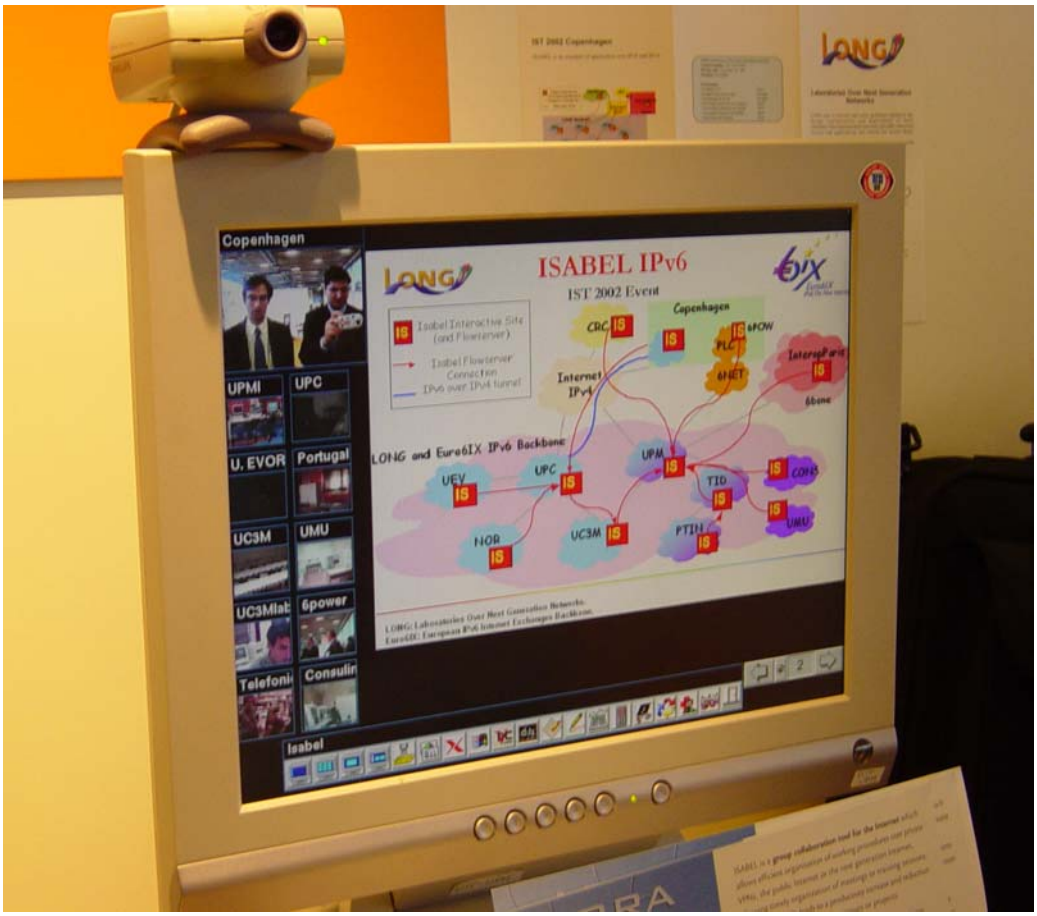


Figure 18. ISABEL demonstration at IST-2002

3.3 Advanced Network Services

Some advanced network services have been tried through the LONG network that have given the opportunity of experiment with the IPv6 features that differ of those of IPv4, such as mobility, which is not an add-on in IPv6, and multicast.

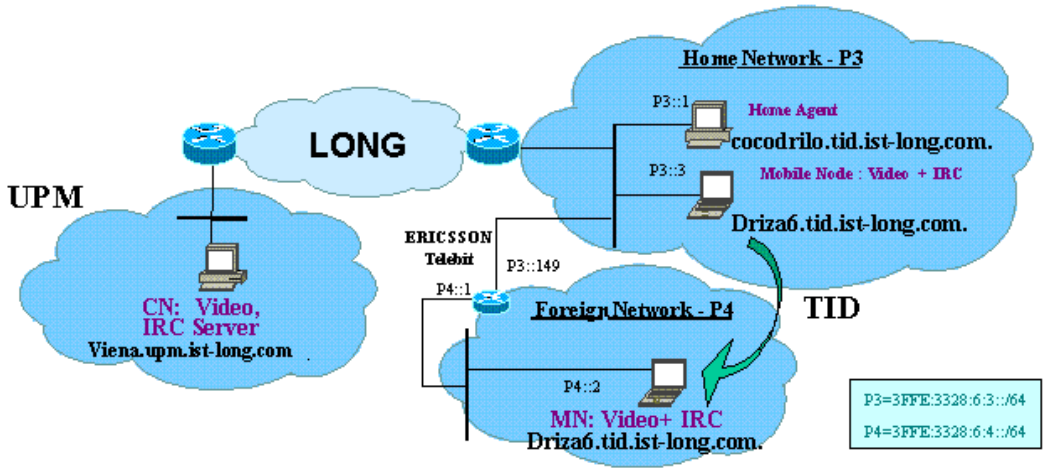
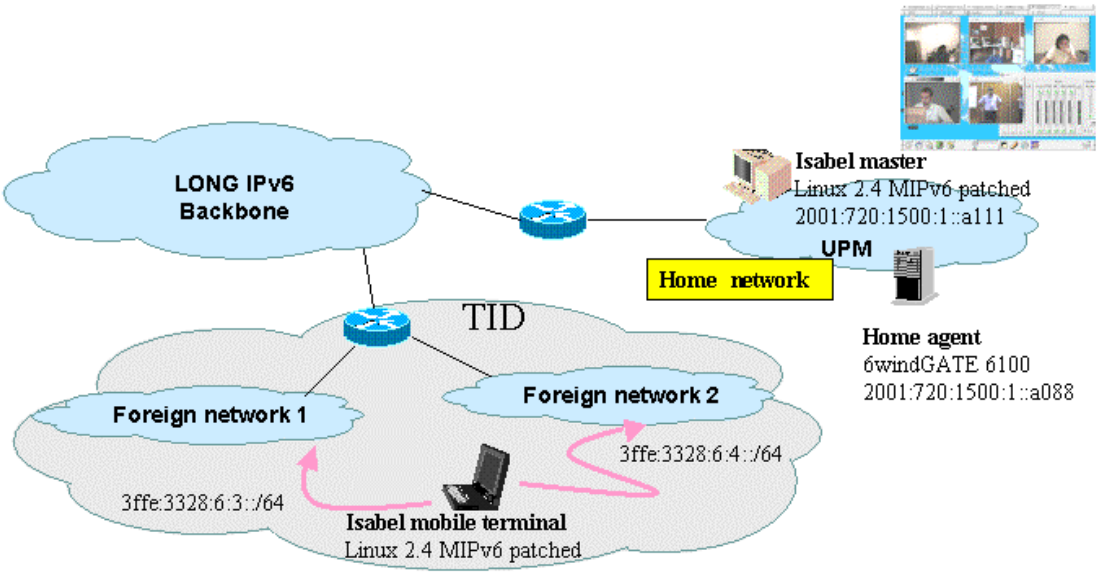


Figure 19. Example of IPv6 Mobility in LONG



ISABEL + MIPv6:

(1) The Isabel mobile terminal, in foreign network 1, connects to the Isabel master at UPM
(2) The Isabel mobile terminal moves to foreign network 2, and gets a new CoA.
(3) The Isabel session keeps running!!

Figure 20. Example of ISABEL Mobility in LONG

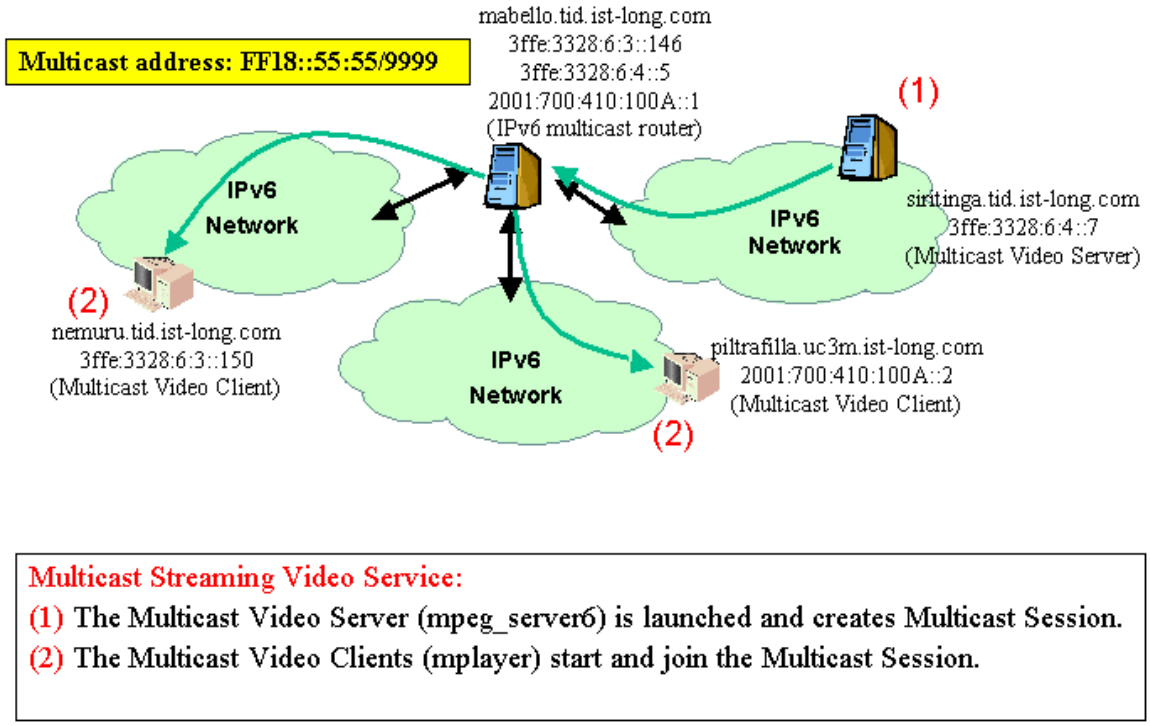


Figure 21. Example of Multicast in LONG

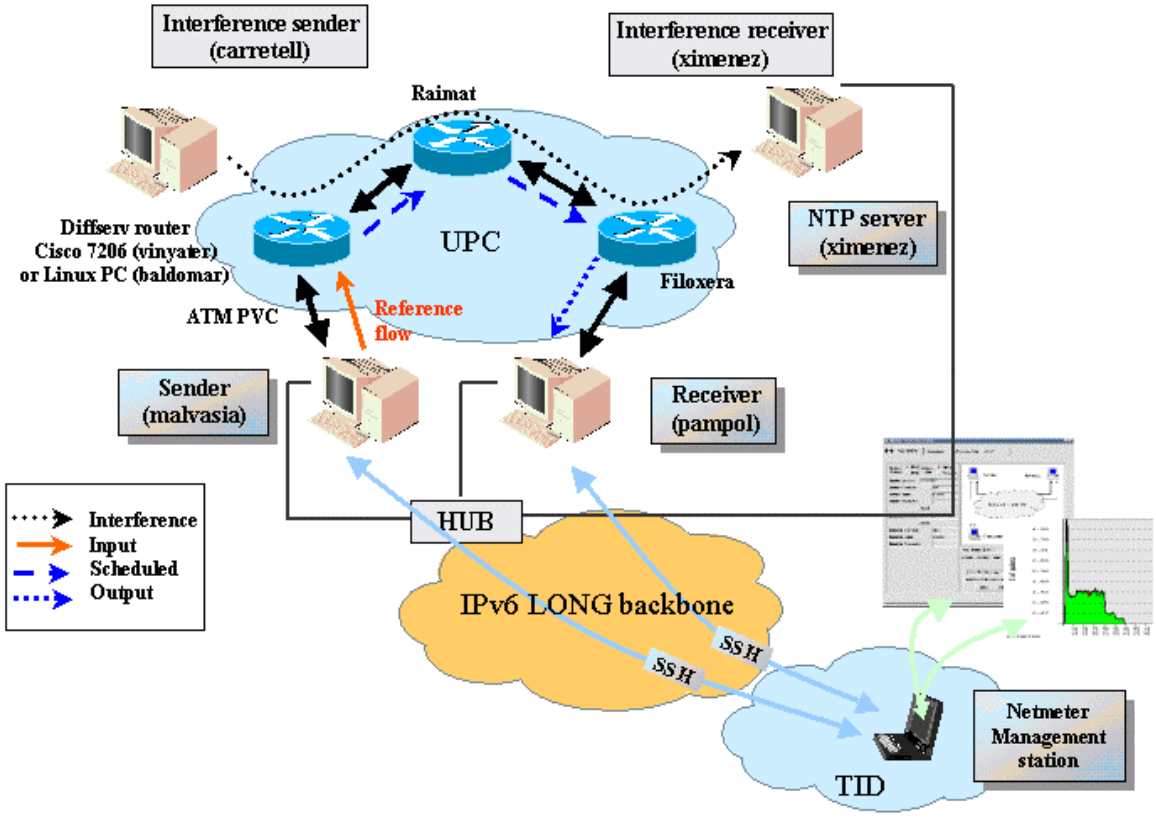
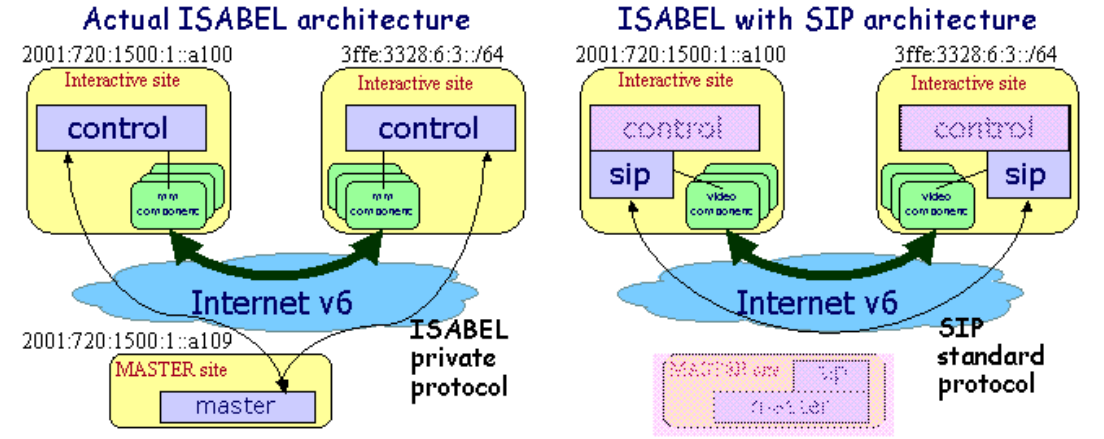


Figure 22. Example of Diffserv Demostration in LONG

In the technical review that took place in Madrid on April 5th, 2002 LONG reviewers recommended the use of SIP new protocol in the CSCW ISABEL. As a result, the study of migrating ISABEL private control protocol to SIP has been sketched.



ISABEL control & SIP:

ISABEL control functionality:
- Register new sites
- Starts multimedia components
- Active interaction modes.

ISABEL SIP control:
- Improves sites identification
- Simplifies master behaviour
- Allows integration with others

Figure 23. ISABEL and SIP

To conclude, we can state that the LONG project has deployed relevant standard network applications (DNS, Web, FTP, News, LDAP, Mail, IRC, and ISABEL) proving that in most cases they can work seamlessly for both IPv6 and IPv4 clients. In general, good support has been provided in UNIX environments for this applications, and Windows have also been used for accessing to some of them. However, DHCPv6 and NFS are not properly supported at this stage; for the first one, there is still an on-going debate in the technical community about its requirement. Regarding to applications, the next step would be to wait for application developers to adapt their proprietary or less standard applications to IPv6.

4. Application Migration

Nowadays, most existing network applications are written assuming IPv4. In general, most of them can be converted without too much effort. Many changes could be done automatically and there are some scripts to do it. However, there are applications that make special use of IPv4 or include advanced features, such as multicasting, IP options or raw sockets. In such cases, exhaustive code revision should be done.

LONG Deliverable D3.1 “Requirements and guidelines for distributed laboratories application migration” includes detailed rules and sockets interface description for most popular programming languages.

Also, most part of the analysis about general application migration is covered by LONG deliverable D3.2A “Guidelines for migration of collaborative work applications”. This document is divided in three parts. The first analyzes existing applications to look for characteristics that usually should be reviewed during migration process. The second is devoted to show the basic socket interface extensions for IPv6, fully described in RFC2553. Finally the last part provides general recommendations for new IPv6 applications.

The mentioned deliverable D3.2A includes some examples of code porting, used to illustrate the required changes in the client and server components. Migration guidelines are valid for any programming language, however for simplicity, application migration examples are provided only in C language.

Finally, more conclusions are drawn in D3.3 “Guidelines for migration of collaborative work application working over asymmetric channels”.

4.1 General Porting Recommendations

Unless all transition mechanisms are valid in a early phase, finally applications should be ported to the new scenario. Sometimes IPv6 version can be developed in parallel with other programmed modifications. However, in most cases we only want to maintain the initial functionality and adapt communications module to enable IPv6.

There are two probable scenarios during transition. The first is based on maintaining two application versions and the second with only one dual version.

We can forget old IPv4 applications and maintain them only to communicate with IPv4 nodes during transition period. The transition methodology consists in developing a complete set of applications designed to work only over IPv6 transport layer. The main advantage is to be ready with new applications after transition period. To finish IPv6 transition it is necessary only to remove IPv4 applications and dual stack network. However, the selection of suitable applications during transition period is a user responsibility. When tests, configuration or management of applications are considered, the application selection is not a problem, because their users normally have technical knowledge. However, with most of end-user applications the user has not enough knowledge or does not want to know which is the proper application version that should be used in each connection. Moreover, during transition period will be necessary to provide data servers working both IPv4 and IPv6 to accept connections not only from new but also from old nodes.

The second transition scenario looks for changing existing applications, not developing new ones. The porting process takes the original IPv4 application, reviews source code and produces a new version adapted to work over both IPv4 and IPv6 environments. The porting

process is a little bit more complex than IPv6 only support, but the result is more flexible to be used during transition period. Transition period will be probably a long period during which there will be islands, not only IPv6 but also IPv4, interested for users in the new environments. This is because during a long period most of data servers will be maintained in the IPv4 only environment.

Reviewing existing applications to work both IPv4 and IPv6 simultaneously requires two phases: code analysis and code rewriting. Code examination is required to locate the following issues:

- IP addresses management.
- Network information storage: data structures.
- Communications API functions and pre-defined constants.

After code identification, code-rewriting can take place. Many changes can be done automatically and there are some applications to do it [SUN, HP]. However, many applications make special use of IPv4 addresses and socket structures, consequently automatic rewriting is not possible. The most important issues that should be taken into account are the following:

- Names resolution.
- Sockets address structures.
- Sockets programming interface.
- Address conversion functions.
- Multicast interface.

Finally, if a design of a new IPv6 application is considered, some recommendations should be taken into account. First of all, it should be decided if the new application will be ready for IPv6 only or it will be a dual one. IPv6 only application does not know IPv4 protocol, therefore v6 to v4 adaptation should be added to the system when IPv4 interaction is demanded. However, a dual application is a little more complex than previous one.

Applications should include code to select the proper interface depending on the correspondent node. To isolate all these operations from the rest of the application a transport module is recommended. The transport module provides a clear and uniform interface to the rest of the application hiding all protocol selection details.

This complex scenario can be summarized in the following items:

- Service continuity: IPv6 transition is not only an address or routing issue but also mainly a service enhancement. All commercial services deployed recently such as QoS, Intranets, group collaboration or IP telephony, have to be continuously provided whatever the IP infrastructure might be.
- Mixed scenario during transition: The introduction of IPv6 will be slow compared with the size of Internet. When IPv4 and IPv6 have to coexist, keeping transition under control is essential to avoid a final scenario with two parallel Internet infrastructures. Therefore, the application porting process should be included with enough resources in the transition plan. In standard environments, the application porting process has a

reduced cost because most of typical applications are already IPv6 enabled. The LONG Project has made an updated catalogue of such applications.

- Transition is not always a necessary solution: Deploying new applications not demanding a complete development to support IPv4 in combination with IPv6 users. Next generation applications will be available only in the new environment. Deploying transition mechanisms at a large scale can lead to scalability issues that could heavily limit the IPv6 performance compared to a native solution. Moreover, the availability of new applications only in the new environment is the best mechanism to accelerate transition from actual IPv4 environment to the new IPv6 one.

4.2 Quick Guide of Application Porting

Most existing applications are written assuming IPv4. In general, most of them can be converted without too much effort. Many changes could be done automatically and there are some scripts to do it. However, there are applications, which make special use of IPv4 or include advanced features, such as multicasting, raw sockets or other IP options. In such cases, an exhaustive code revision should be done.

Besides, there are applications that are not only affected by modification of function calls but also by the logic behind their usage. Such code is the hardest to deal with and cannot be automatically ported. This code can be misleading if developers do not consider the logic of the program during the porting process.

Dual stack allows applications to use both types of addresses, IPv4 and IPv6. Developers will only need to port their IPv4 applications to the new IPv6 API, considering that applications could communicate using both protocol versions, depending on the destination node.

In the general case, porting an existing application to IPv6 requires to examine the following issues in the application source code:

- Network information storage: data structures.
- Resolution and conversion address functions.
- Communication API functions and pre-defined constants.

Many applications use IP addresses to store references to remote nodes. This is not recommended because IPv6 lets addresses change over time (this technique is named renumbering). Applications should use stable identifiers for other nodes, for instance, host names. If applications store IP addresses they should redo the mapping from host names to IP addresses in order to solve inconsistencies.

Client applications should be prepared to connect to multi-homed severs, nodes that have more than one IP address. Hence, when a communication channel to a multi-homed server fails, client applications should try another IP address in the multi-homed server list of IP addresses until they find one that is working.

Applications using URLs should change definitions in RFC2732 which specifies square brackets to delimit IPv6 addresses: http://[IPv6Address]/index.html

4.2.1 Analyzing existing programs

Once the IPv6 network is deployed, applications must be designed to work over it. Existing applications cannot use IPv6 without previous modifications. Hence, from the applications point of view, the IPv6 deployment requires changes in the existing applications and new communication design concepts to be included for the new ones.

When porting IPv4 applications to IPv6, there are different degrees to carry out this task. If applications only use basic communications facilities, developers only have to identify the application communication module and change some functions in order to use the new IPv6 API. However, if IPv6 advanced facilities should be considered, such as quality of service or mobility, a redesign of some parts of the application must be done. In this last case, the porting process requires a complete knowledge of the application architecture and this process is similar to the design of a new application over IPv6.

Providing basic IPv6 communication facilities to an existing application is an easy task if the application architecture is well designed, with isolated functional modules, so that the transport module could be easily identified. In the other hand, if the application has not isolated modules, adaptation to the new API will require many changes in different parts of source code, making difficult the reusability and the maintenance.

The application transport module is in charge of implementing the communication paradigm selected by the application and must be adapted to the new IPv6 communication API. Usually, the communication API makes visible to the application the version of protocol it is using, since address data structures are completely different. Besides, other facilities such as conversion functions between hostnames and IP addresses are different in the new IPv6 environment too.

Besides, there could be other modules or application parts different from transport module, which can include dependencies on the IP addresses, and they must be reviewed:

- IP address parsers.
- Use of special addresses.
- Local IP address selection.
- Application Data Unit (ADU) fragmentation.
- Register systems based on IP addresses.

In the following subsections, some modifications to be made in the transport module and in other modules, which include dependencies on the IP addresses, are studied.

4.2.1.1 Application transport module

When porting to IPv6, the first module to be reviewed is the one related to the management of the remote communication channels, named transport module. This module will surely contain dependencies on the IP version used by the application.

The application transport module establishes communication channels required by a specific application when transmitting data to remote sites. This module makes applications network independent, providing a generic communication module to allow applications exchange information between remote participants.

Changes in the application transport module are required since the kernel communication API has been changed to support IPv6. The module must be changed to use the new address structures and functions to create IPv6 communications.

During the gradual transition phases from IPv4 to IPv6, the same application will work over IPv4 and IPv6 networks. Hence, portability is one of the main features of applications, which should work in both environments. One of the best ways to make applications independent of the protocol used (IPv4 or IPv6) is to design a generic communication library, which hides protocol dependencies and lets the application transport module be simpler. The use of the library allows reusability and easy maintenance of the communication channel abstraction. Applications can forget about communication problems, since the library takes charge on behalf of them, see the following figure.

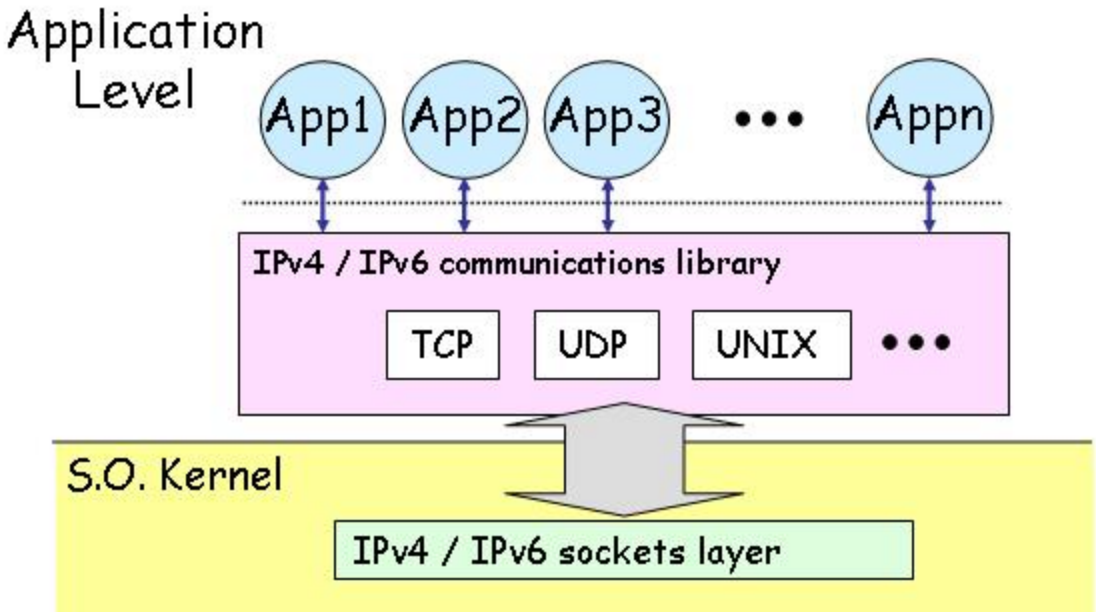


Figure 24. Applications protocol independent of IPv4 or IPv6.

A generic communication library allows applications to be independent of the lower level protocol, and provides a common communication interface to every application. Applications should be able to access to different communication protocols from the generic library API functions: TCP, UDP, IP, multicast, etc., independently of the IP version.

4.2.1.2 Other application modules with IP address dependencies

The application transport module is directly related to the communication establishment and the information transmission. However, there could be other application modules, which have IP address dependencies, so they are IP version dependant. In the next subsections these other dependencies are analysed.

4.2.1.2.1 IP address parsers

Many applications require an IP address as an argument to establish a new connection to this address, for example using the client/server model the client needs to know the IP address or the hostname where the server is running.

The use of Fully Qualified Domain Name (FQDN) instead of IP address is preferable since nodes can change their addresses and this process should be transparent to applications. Applications can store and use FQDN, delegating the resolution of the IP addresses to the name resolution system, which will return the associated IP address at the moment of the query.

From applications point of view, the name resolution is a system-independent process. Applications call functions in a system library, known as the resolver, which is linked into the application when the application is built. Developers should change the use of the IPv4 resolution functions by the new IPv6 ones, or by the protocol-independent ones if they exist.

Typically applications do not require knowing the version of the IP version they are using, hence applications only should try to establish communications using each address returned by resolver until it works. However, applications could have a different behaviour when using IPv4, IPv6, IPv4-compatible-IPv6 or IPv4-mapped, etc. addresses.

There could be scenarios where the use of IP addresses is required and applications utilise a parser to analyse addresses. In these cases, IP address parsers must be modified in order to include the new IPv6 string address format.

IPv4 addresses are represented using dotted quad format, each decimal integer represents a one octet of the 4 octet address, value between 0 and 255; for instance 138.4.4.161. The written length of the IPv4 address string varies between 7 and 15 bytes. IPv6 addresses are represented using hexadecimal notation which can be abbreviated, requiring between 3 and 39 bytes; for instance 2001:720:1500:1::a100. So, IPv4 uses dot (“.”) to separate octets and IPv6 uses colon (“:”) to separate each pair of octets. Application address parser code should be reviewed to be in conformance with the IPv6 address representation.

There could be an ambiguity with the colon character. Colon character is used in the IPv6 addresses as a separator between each pair of address octets and it is used in the IPv4 networks as a separator between the address and the service port number. Applications can use the same format as the literal IPv6 addresses in URLs, enclosing the IPv6 address within square brackets, to solve the ambiguity. The RFC 2373 describes literal IPv6 addresses in URLs, for instance: http://[2001:720:1500:1::a100]:80/index.html

4.2.1.2.2 Use of special addresses

There are some special addresses, which are found hard coded within the application source, instead of using symbolic names. For instance, the addresses 127.0.0.0/8 is referred to the localhost interface. When moving an application from an only IPv4 host to an only IPv6 host, hard coded addresses will fail when establishing connections. So the use of names instead of IP addresses is recommended, names could be reconfigured without changing application source code.

Developers should review application source code to change the IPv4 addresses in Table 1 to consider portability to IPv6 only hosts.

Table 1: Special addresses.

Symbolic Name (IPv4 / IPv6)	IPv4 Address	IPv6 Address
INADDR_ANY / IN6ADDR_ANY_INIT	0.0.0.0	::
INADDR_LOOPBACK / IN6ADDR_LOOPBACK_INIT	127.0.0.1	::1
INADDR_BROADCAST	255.255.255.255	Not exist

4.2.1.2.3 Local IP address selection

IPv6 allows many IP addresses by each network interface with different reachability scope (link-local, site-local or global). Hence, there should be mechanisms to select which source and destination addresses applications should use in order to know the behaviour of the systems.

Normally, the name resolution functions return a list of valid addresses for a specific FQDN. Applications should iterate this list to select the address to be used in the communication channel configuration (for instance, to select the source address, for bind() function, and to select the destination address, for `sendto()` or `connect()` functions). Source address selection is a critical operation that gives information to the receiver about the address to send the reply. If the selection is not appropriated the backward path could be different from forward path, even if the addresses are administratively scoped, the reply may be lost and communication between applications will fail.

When choosing source address, some applications use unspecific address to let the OS kernel make this selection, named default address selection. When choosing destination address, some criteria could be used to prefer one address based on the pair source/destination values. The default address selection algorithm returns a preferred address from a set of candidates, based on a policy to make the best choice. Administrators can configure this mechanism to override the default behaviour. There are still some works in progress around this topic (draft-ietf-ipv6-default-addr-select.txt).

4.2.1.2.4 ADU fragmentation

Application Data Unit (ADUs) is the block of data sent or received in a single communication operation at application level. Applications alternate between computation and communication phases, and during the communication phase they send or receive one or more data structures, each forming an ADU. However, since the amount of data that can be handled at any given time in a single unit of operation is limited by the available resources on the network interface node, the ADU must be fragmented in transfer units (TUs).

The problem is to select the more adequate TU size. Long packets are transmitted more efficiently, as the application overhead of the end systems is reduced. On the other hand, longer packets tend to increase transit delays because of the intermediate relaying process, which is not good in real time applications. The size of the TUs is directly related to the maximum size of the IP packet used over a network (PMTU, Path Maximum Transmission

Unit) and the IP fragmentation process. Then, longer packets are likely to be fragmented to adapt the packet size to the link layer.

Since IPv6 fragmentation is an end-to-end algorithm, [RFC 1981] recommends that all IPv6 nodes should implement PMTU Discovery (PMTU-D) to optimise the throughput of fragmented ADUs. PMTU-D is a mechanism to use the longest IPv6 packet size that fits the IPv6 minimum MTU through all the networks traversed, increasing the efficiency of transmission and guaranteeing that the IPv6 packets can travel through all networks to reach the destination. If a packet is too large for a router to forward on to a particular link, the router must send an ICMP "Destination Unreachable -- Fragmentation Needed" message to the source address an the source host adjusts the packet size based on the ICMP message.

However, the PMTU-D is implemented in the packetization protocol, but it is only a recommendation not a mandatory network module, and routers do not always rely on ICMP packets, see RFC2923. So, applications can send their TUs using a TU size longer than the MTU of an intermediate link on the current path, between the source and the destination nodes and it is likely none of the packets will reach the destination node. This problem is known as the black hole. For example, in the next figure, packets are not fragmented at the source because they can be sent over this first link layer. But, when the packets reach the intermediate link R_{inter}, which is using an outgoing MTU smaller than the packet size, packets will be discarded and will not reach destination node. In the next figure TU size is longer than the MTU_3 of the intermediate link, at this point packet will be discarded.

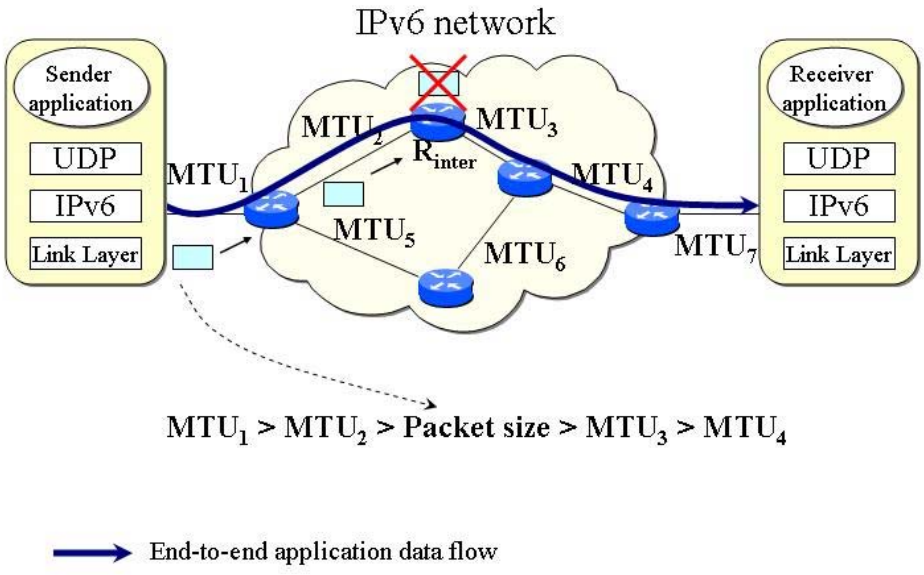


Figure 25. Transfer Unit (TU) size problem if not PMTU-D implemented.

TCP is a packetization protocol and some implementations solve this problem implementing black hole detection and sending smaller packets until the transmission is done.

However, there are applications, which realize their own packetization process, maybe sending UDP packets. If PMTU discovery is not properly working, data will not reach destination. In this situation, applications must implement their own mechanisms to detect black hole problem and send smaller packets, or use the minimum supported MTU for IPv6 1280 octet packets.

4.2.1.2.5 Register systems based on IP addresses

When analysing network applications, dependencies in the source code with the IP address are frequently found. For instance, one of the most popular ways to register remote nodes in a collaborative system is based on using the IP addresses as keys for searching in a registry system. Group communication is often related to a group membership concept based on a participant registry system.

The registry system provides an identification method to allow connections from different remote participants to a session. The problem is that an IP address cannot be used to identify a peer node since IP addresses can change over time, for instance after a renumbering process. Renumbering should be an infrequent event, but sometimes it will happen and it should be a transparent process for applications.

The best solution to this problem is the use of identifiers independent of the network layer, for instance random numbers. However, some applications require that the identifiers were dependent on network address. One of the flexible solutions is the use of FQDN as stable identifiers for participant nodes in the registry system. The FQDN remains invariable over the time although its associated IP address can change. The resolver code will provide the IP address if it was required at any time.

Sometimes applications require IP addresses instead of FQDN. When porting to IPv6, this dependency will provoke some changes in the source code to support the new data structures. In these cases, applications can get addresses from FQDN and store them in configuration files for an unlimited time. This information should be refreshed periodically to actualise IP address modifications and to avoid no longer valid information. The refreshing process will guarantee the use of the last assigned IP address to a node.

4.2.2 IPv4/IPv6 Interoperability

The mechanism to select the appropriate IP address is decided by the name service. When a network node wants to reach another, it asks the name service for its IP address. If the answer is an IPv4 address, it is assumed that there is a path through Internet to link with the remote node and that this remote node is capable of receiving IPv4 connections from the source node. The same applies for a node that has only IPv6 address. It is assumed that it understands IPv6 packets. If both source and destination nodes have dual stack, the communication will use the type of address returned by the name service.

During first step of transition phase there is no IPv6 network. This document is not devoted to study how to solve other communication aspects, which are not visible to the application layer. If IPv6 addresses should be used during connection but IPv6 routers were not part of the network infrastructure, a basic IPv4 framework should be used. This is achieved by building IPv6 tunnels. They encapsulate IPv6 packets inside IPv4 header and send them through the IPv4 network.

However, dual-stack should be used during most of the transition periods, because not all IPv6-only implementations allow the interaction with any kind of network node. Following we will study the interoperability between IPv4-only nodes, IPv6-only nodes and dual stack nodes using the client/server model.

4.2.2.1 IPv6/IPv4 clients connecting to an IPv4 server at IPv4-only node

There is an IPv4 server application running on IPv4-only node and we will analyze all connection possibilities from clients.

If an IPv4 client running on IPv4-only node connects to the IPv4 server, it will work as the usual IPv4 client/server connection, since both of them are IPv4-only nodes.

When using an IPv4 client running at dual stack node, the client will request the server IP address to the resolver and it will return the server IPv4 address. The client running at dual stack will work as if it was running at IPv4-only node and will exchange IPv4 packets.

When using an IPv6 client running at IPv6-only node, since this node can only exchange IPv6 packets, it cannot connect to the server, which can only exchange IPv4 packets. They are using incompatible protocols.

If an IPv6 client is running at dual stack node, the client will request to the resolver the server IP address. Since the server is running at an IPv4-only node, the resolver will return the IPv4-mapped IPv6 address. The IPv6 client will use this address to connect to the IPv4 server and the dual stack node will send IPv4 packets when the IPv4-mapped IPv6 address is used. Then, IPv6 client will work as it was connected to an IPv6 server and the IPv4 server will work as it was connected to an IPv4 client.

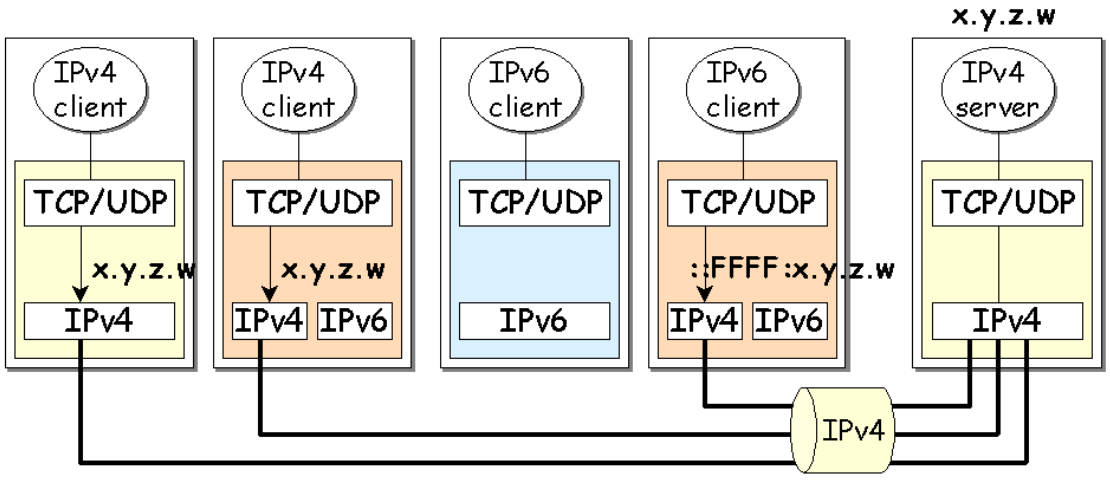


Figure 26. IPv4/IPv6 clients connecting to an IPv4 server at IPv4-only node.

4.2.2.2 IPv6/IPv4 clients connecting to an IPv6 server at IPv6-only node

There is an IPv4 server application running at IPv6-only node and we will analyze all connection possibilities from clients, see next figure.

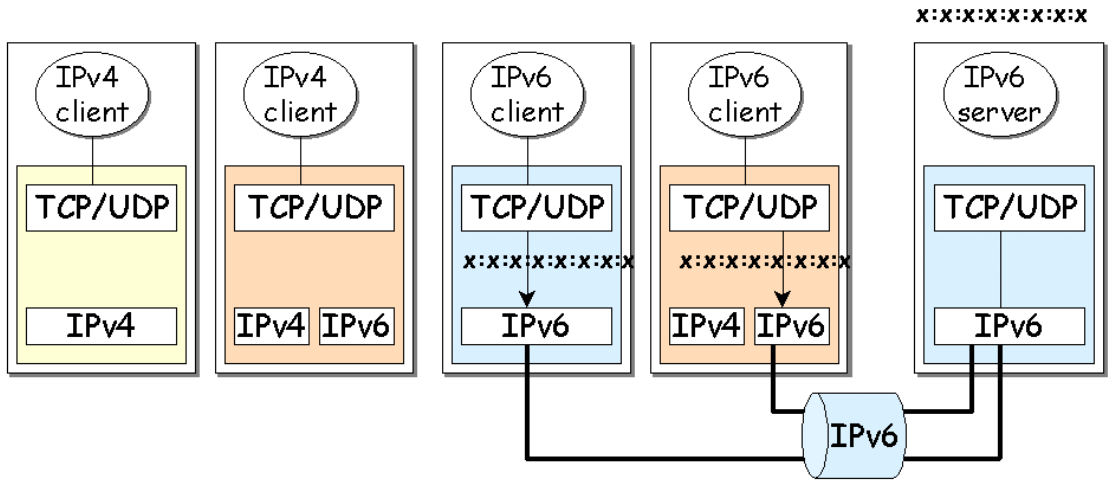


Figure 27. IPv4/IPv6 clients connecting to an IPv6 server at IPv6-only node.

IPv4 clients cannot connect to an IPv6 server running at an IPv6-only node, since IPv4 clients cannot use IPv6 addresses. Then, only IPv6 clients can connect to the server using the IPv6 address of the server node.

4.2.2.3 IPv6/IPv4 clients connecting to an IPv4 server at dual stack node

There is an IPv4 server application running at dual stack node and we will analyze all connection possibilities from clients, see next figure.

When using IPv4 clients, the mechanisms are similar to the 4.1.1 section, the IPv4 server at only-IPv4 node case. Clients use the server IPv4 address. IPv4 packets are exchanged between clients and server.

An IPv6 client at IPv6-only node cannot connect to the IPv4 server, since the server cannot use IPv6 addresses. Although both nodes could communicate using IPv6 protocol, since the server only uses IPv4, the client cannot connect to the server.

An IPv6 client at dual stack node can connect to the IPv4 server using IPv4 network. The IPv6 client request to the resolver the server IP address, the resolver will return the IPv4-mapped IPv6 address to the IPv6 client. As we analyzed at the 4.1.1 section, the IPv6 client will work as it was connected to an IPv6 server and the IPv4 server will work as it was connected to an IPv4 client.

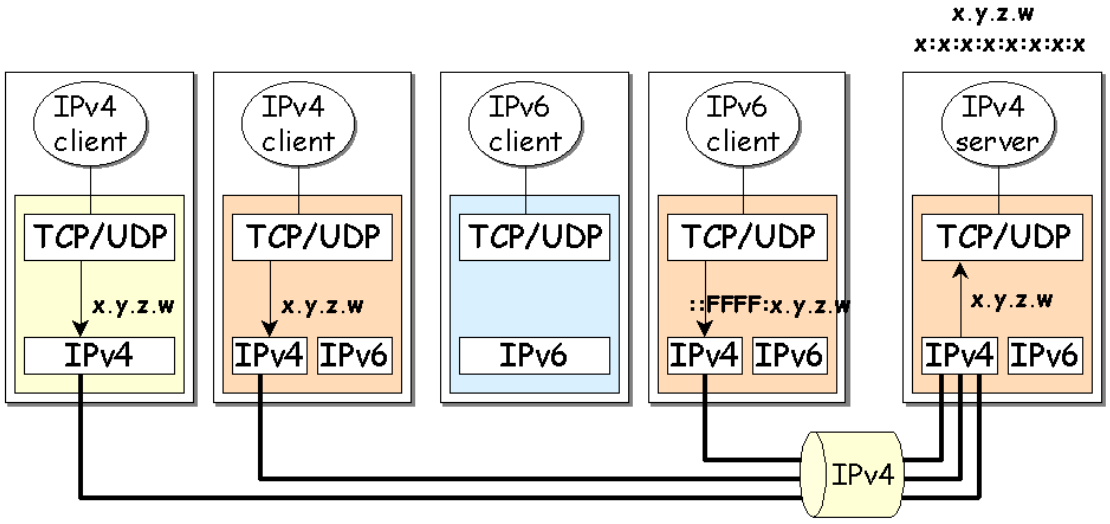


Figure 28. IPv4/IPv6 clients connecting to an IPv4 server at dual stack node.

4.2.2.4 IPv6/IPv4 clients connecting to an IPv6 server at dual stack node

There is an IPv6 server application running at dual stack node and we will analyze all connection possibilities from clients, see next figure.

IPv4 clients will connect using IPv4 protocol. They will use the IPv4 server address and exchange IPv4 packets. These packets are delivered to the IPv6 server using IPv4-mapped IPv6 addresses. Also, the server will use the IPv4-mapped IPv6 addresses when answering the IPv4 client requests. The dual stack at the server node will send IPv4 packets when the IPv4-mapped IPv6 address is used.

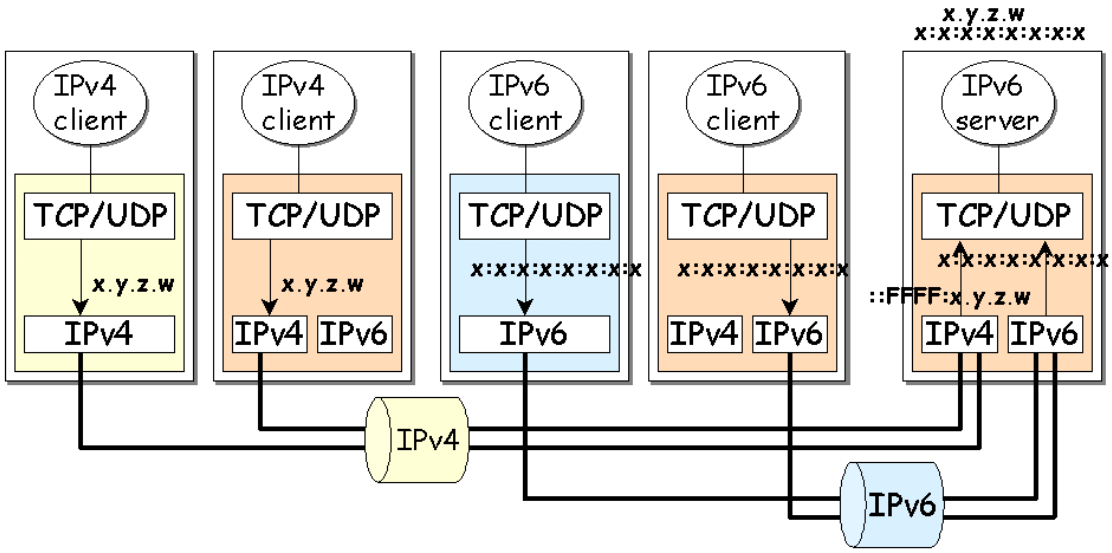


Figure 29. IPv4/IPv6 clients connecting to an IPv6 server at dual stack node.

IPv6 clients will use the server IPv6 address to connect to it and will exchange IPv6 packets.

4.2.2.5 IPv4/IPv6 clients connecting to an IPv4-only server and IPv6-only server at dual stack node.

During transition there could be running different versions of the same application on a dual stack node, the IPv4-only and the IPv6-only versions. The IPv4-only server will only accept connections from IPv4 clients and the IPv6-only server will only attend to IPv6 clients, see next figure.

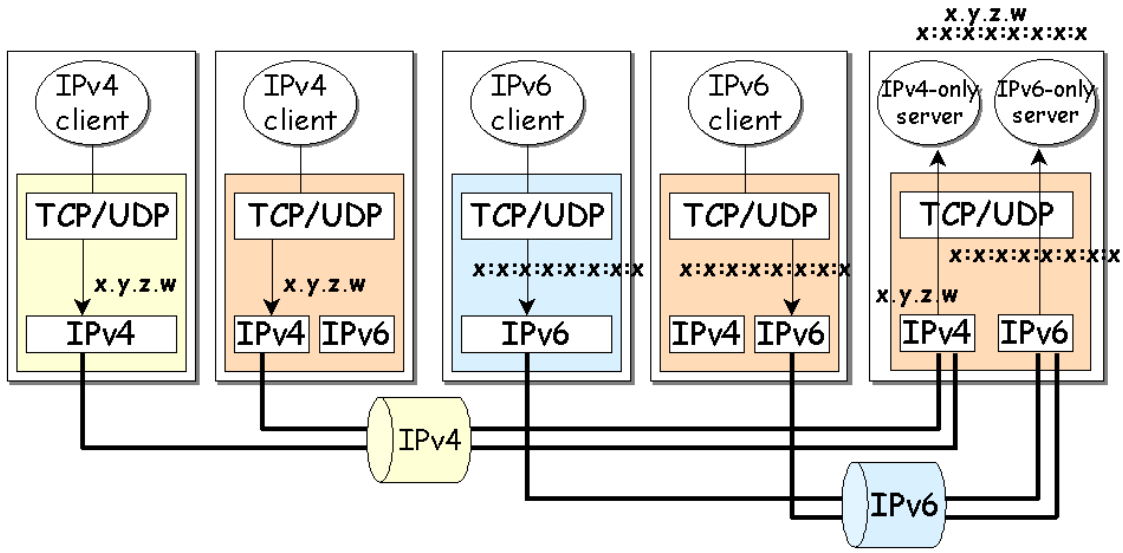


Figure 30. IPv4/IPv6 clients connecting to an IPv4-only server and IPv6-only server at dual stack node.

This case allows separating the IPv4 connections from IPv6 ones. If only IPv6 connections must be accepted, the IPv6-only server will be started. Notice that 4.2.4 is similar to this case, but in that case the IPv6 server will accept all connection requests, from IPv4 and IPv6 clients and cannot be separated connection from different protocol versions.

4.2.2.6 Client/server and network type interoperability

The Table 2 summarizes the combinations to connect clients and servers running at different kind of nodes: IPv4-only node, IPv6-only node and dual stack node. The combinations signed with “X” denote that communication between such kinds of nodes is not possible. However, dual-stack combinations allow network communication in almost all circumstances. There is only an exception: When the server is IPv4 and an IPv6 client tries to communicate with it, the connection is only possible if client address is an IPv4-mapped into IPv6 address. In this case, if the client chooses a pure IPv6 address, the server will not be able to manage the client address.

Table 2: Client server and network type combinations.

		IPv4 server application		IPv6 server application	
		IPv4 node	Dual-stack	IPv6 node	Dual-stack
IPv4 client	IPv4 node	IPv4	IPv4	X	IPv4
	Dual-stack	IPv4	IPv4	X	IPv4
IPv6 client	IPv6 node	X	X	IPv6	IPv6
	Dual-stack	IPv4	IPv4 / X	IPv6	IPv6

Therefore, four application types can be distinguished:

IPv4-only: An application that is not able to handle IPv6 addresses i.e. it cannot communicate with nodes that do not have an IPv4 address.

IPv6-aware: An application that can communicate with nodes that do not have IPv4 addresses i.e. the application can handle the larger IPv6 addresses. In some cases this might be transparent to the application, for instance when the API hides the content and format of the actual addresses.

IPv6-enabled: An application that, in addition to being IPv6-aware, takes advantage of some IPv6 specific features such as flow labels. The enabled applications can still operate over IPv4, perhaps in a degraded mode.

IPv6-required: An application that requires some IPv6 specific feature and therefore cannot operate over IPv4.

During the gradual transition phase from IPv4 to IPv6, the same application should be run in an IPv4 or IPv6 nodes. Hence, portability is one of the main features of applications, which should work in both environments.

4.2.3 Porting Source Code

Some changes are needed to adapt the socket API for IPv6 support: a new socket address structure to carry IPv6 addresses, new address conversion functions and several new socket options that are explained in RFC-2553. These extensions are designed to provide access to the basic IPv6 features required by TCP and UDP applications, including multicasting, while introducing a minimum of change into the system and providing complete compatibility for existing IPv4 applications. Access to more advanced features (raw sockets, header configuration, etc.) is addressed in RFC-2292.

The following subsections are an overview of changes in the basic BSD socket API to support IPv6 (see RFC-2553 for details) and how the applications source code should be changed in order to make it portable and protocol independent code.

4.2.3.1 Socket Address Structures

Functions provided by socket API use socket address structures to determine the communication service access point. Since different protocols can handle socket functions, a generic socket address structure is used as argument of these functions for any of the supported communication protocol families, `sockaddr`.

```
struct sockaddr {
    sa_family_t sa_family;      /* Address family */
    char sa_data[14];           /* protocol-specific address */
};
```

Although socket functions handle generic socket address structure, developers must fill the adequate socket address structure according to the communication protocol they are using to establish the socket. Concretely, the IPv4 sockets use the following structure, `sockaddr_in`:

```
typedef uint32_t in_addr_t;
struct in_addr  {
   in_addr_t s_addr;                /* IPv4 address */
};

struct sockaddr_in {
    sa_family_t sin_family;          /* AF_INET */
    in_port_t sin_port;              /* Port number.  */
    struct in_addr sin_addr;         /* Internet address.  */

    /* Pad to size of `struct sockaddr'.  */
    unsigned char sin_zero[sizeof (struct sockaddr) -
                           sizeof (sa_family_t) -
                           sizeof (in_port_t) -
                           sizeof (struct in_addr)];
};
```

And the IPv6 sockets use the following structure, `sockaddr_in6`, with a new address family `AF_INET6`:

```
struct in6_addr {
   union {
      uint8_t u6_addr8[16];
      uint16_t u6_addr16[8];
      uint32_t u6_addr32[4];
   } in6_u;

   #define s6_addr              in6_u.u6_addr8
   #define s6_addr16             in6_u.u6_addr16
   #define s6_addr32             in6_u.u6_addr32
};

struct sockaddr_in6 {
   sa_family_t sin6_family;    /* AF_INET6 */
   in_port_t sin6_port;        /* Transport layer port # */
   uint32_t sin6_flowinfo;     /* IPv6 flow information */
```

```
  struct in6_addr sin6_addr;  /* IPv6 address */
  uint32_t sin6_scope_id;     /* IPv6 scope-id */
 };
```

The `sockaddr_in` or `sockaddr_in6` structures are utilized when using respectively IPv4 or IPv6. Existing applications are written assuming IPv4, using `sockaddr_in` structure. They can be easily ported changing this structure by `sockaddr_in6`. However, when writing portable code, it is preferable to eliminate protocol version dependencies from source code. There is a new data structure, `sockaddr_storage`, large enough to store all supported protocol-specific address structures and adequately aligned to be cast to the a specific address structure.

```
/* Structure large enough to hold any socket address (with the historical exception of AF_UNIX). 128 bytes reserved.  */

#if ULONG_MAX > 0xffffffff
# define __ss_aligntype __uint64_t
#else
# define __ss_aligntype __uint32_t
#endif
#define _SS_SIZE        128
#define _SS_PADSIZE     (_SS_SIZE - (2 * sizeof (__ss_aligntype)))

struct sockaddr_storage
  {
    sa_family_t ss_family;      /* Address family */
    __ss_aligntype __ss_align;  /* Force desired alignment.  */
    char __ss_padding[_SS_PADSIZE];
  };
```

Hence, portable applications should use `sockaddr_storage` structure to store their addresses, IPv4 or IPv6 ones. This new structure hides the specific socket address structure that the application is using.

4.2.3.2 Socket functions

The socket API has not been changed since it handles generic address structures, independent from the protocol it is using. However, applications should change the value of arguments used to call the socket functions. First, applications should allocate enough memory to store the appropriate socket address structure. And second, before calling the socket functions, the specific socket address structure should be cast to the generic one, which is accepted by the socket functions as an argument.

```
int     socket  (int domain, int type, int protocol);

int     listen  (int s, int backlog);

ssize_t write   (int fd, const void *buf, size_t count);

int     send    (int s, const void *msg, size_t len, int flags);

int     sendmsg (int s, const struct msghdr *msg, int flags);
```

```
ssize_t read    (int fd, void *buf, size_t count);

int     recv    (int s, void *buf, size_t len, int flags);

int     recvmsg (int s, struct msghdr *msg, int flags);

int     close   (int fd);

int     shutdown(int s, int how);
```

Socket calls where a socket address structure is provided from application to kernel.

```
int bind   (int sockfd, struct sockaddr *my_addr, socklen_t addrlen);

int connect(int sockfd, const struct sockaddr *serv_addr,
        socklen_t addrlen);

int sendto (int s, const void *msg, size_t len, int flags,
        const struct sockaddr *to, socklen_t tolen);
```

Socket calls where a socket address structure is provided from kernel to application.

```
int  accept     (int s, struct sockaddr *addr, socklen_t *addrlen);

int  recvfrom   (int s,  void  *buf, size_t len, int flags,
            struct sockaddr *from, socklen_t *fromlen);

int  getpeername(int s, struct sockaddr *name, socklen_t *namelen);

int  getsockname(int s, struct sockaddr *name, socklen_t *namelen);
```

4.2.3.3 Required modifications when porting to IPv6

When porting to IPv6, some modifications related to the socket API are required in the network applications.

Three modification types to be made when porting source code to IPv6 have been identified:

a) Creating a socket.

b) Socket calls where a socket address structure is provided from application to kernel.

c) Socket calls where a socket address structure is provided from kernel to application.

There are some examples below of these three types of operation.

a) Creating a socket

The difference between creating an IPv4 and an IPv6 socket is the value of the family argument in the socket call.

- IPv4 source code:

```
socket(PF_INET, SOCK_STREAM, 0);  /* TCP socket */
```

```
socket(PF_INET, SOCK_DGRAM,  0);  /* UDP socket */
```

- IPv6 source code:

```
socket(PF_INET6, SOCK_STREAM, 0); /* TCP socket */
socket(PF_INET6, SOCK_DGRAM,  0); /* UDP socket */
```

b) Socket calls where a socket address structure is provided from application to kernel

Socket address structure is filled before calling the socket function.

- IPv4 source code (a complete example is included in appendix in the ListenServer4.cpp file):

```
struct sockaddr_in addr;
socklen_t         addrlen = sizeof(addr);

/*
  fill addr structure using an IPv4 address before calling socket
  funtion
*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

IPv6 source code:

```
struct sockaddr_in6 addr;
socklen_t         addrlen = sizeof(addr);

/*
  fill addr structure using an IPv6 address before calling socket
  function
*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

- Portable source code (a complete example is included in appendix, in the ListenServer.cpp file):

```
struct sockaddr_storage addr;
socklen_t               addrlen;

/*
  fill addr structure using an IPv4/IPv6 address and
  fill addrlen before calling socket function
*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

c) Socket calls where a socket address structure is provided from kernel to application

When calling this kind of socket functions, the socket address structure is filled in with the address of the source entity.

- IPv4 source code (a complete example is included in appendix, in the TCPDayTimeServer4.cpp file):

```
struct sockaddr_in addr;
socklen_t         addrlen = sizeof(addr);

accept(sockfd,(struct sockaddr *)&addr, &addrlen);

/*
  addr structure contains an IPv4 address
*/
```

- IPv6 source code:

```
struct sockaddr_in6 addr;
socklen_t          addrlen = sizeof(addr);

accept(sockfd,(struct sockaddr *)&addr, &addrlen);

/*
  addr structure contains an IPv4 address
*/
```

- Portable source code (a complete example is included in appendix, in the TCPDayTimeServer.cpp file):

```
struct sockaddr_storage addr;
socklen_t               addrlen = sizeof(addr);

accept(sockfd,(struct sockaddr *)&addr, &addrlen);

/*
  addr structure contains an IPv4/IPv6 address
  addrlen contains the size of the addr structure returned
 */
```

4.2.3.4 Address conversion functions

The address conversion functions convert between binary and text address representation. Binary representation is the network byte ordered binary value, which is stored in the socket address structure and the text representation, named presentation, is an ASCII string.

The IPv4 address conversion functions are the following ones:

```
/*
  From text to IPv4 binary representation
*/
int       inet_aton (const char *cp, struct in_addr *inp);
in_addr_t inet_addr( const char *cp);

/*
  From IPv4 binary to text representation
*/
char        *inet_ntoa(struct in_addr in);
```

The new address conversion functions which work with both IPv4 and IPv6 addresses are the following ones:

```
/*
   From presentation to IPv4/IPv6 binary representation
*/
int inet_pton(int family, const char *src, void *dst);

/*
   From IPv4/IPv6 binary to presentation
*/
const char *inet_ntop(int family, const void *src,
                      char *dst, size_t cnt);
```

- IPv4 source code (a complete example is included in appendix, in the TCPDayTimeServer.cpp file):

```
struct sockaddr_in addr;
char *straddr;

memset(&addr1, 0, sizeof(addr));
addr.sin_family = AF_INET;        /* family */
addr.sin_port = htons(MYPORT);    /* port, networt byte order */

/*
   from text to binary representation
*/
inet_aton("138.4.2.10", &(addr.sin_addr));


/*
   from binary to text representation
*/
straddr = inet_ntoa(addr.sin_addr);
```

- IPv6 source code:

```
struct sockaddr_in6 addr;
char straddr[INET6_ADDRSTRLEN];

memset(&addr, 0, sizeof(addr));
addr.sin6_family = AF_INET6;      /* family */
addr.sin6_port = htons(MYPORT);   /* port, networt byte order */

/*
   from presentation to binary representation
*/
inet_pton(AF_INET6, “2001:720:1500:1::a100",
        &(addr.sin6_addr));
```

```
/*
   from binary representation to presentation
*/
inet_ntop(AF_INET6, &addr.sin6_addr, straddr,
        sizeof(straddr));
```

4.2.3.5 Resolving names

Applications should use names instead of addresses for hosts. Names are easier to remember and remain the same; however numeric addresses could change more frequently.

From applications point of view the name resolution is a system-independent process. Applications call functions in a system library known as the resolver, typically `gethostbyname` and `gethostbyaddr`, which is linked into the application when the application is built. The resolver code is the burden of making the resolution dependent of the system configuration.

There are two new functions to make name and address conversions protocol independent, `getaddrinfo` and `getnameinfo`. Besides, the use of these new ones instead of `gethostbyname` and `gethostbyaddr` is recommended because the latter are not normally reentrant and could provoke problems in threaded applications.

The `getaddrinfo` function returns a linked list of `addrinfo` structures, which contains information requested for a specific set of hostname, service and additional information stored in an `addrinfo` structure.

```
struct addrinfo {
   int     ai_flags;         /* AI_PASSIVE, AI_CANONNAME */
   int     ai_family;        /* AF_UNSPEC, AF_INET, AF_INET6 */
   int     ai_socktype;      /* SOCK_STREAM, SOCK_DGRAM ... */
   int     ai_protocol;      /* IPPROTO_IP, IPPROTO_IPV6 */
   size_t  ai_addrlen;       /* length of ai_addr */
   struct sockaddr ai_addr;  /* socket address structure */
   char    ai_canonname;     /* cannonical name */
   struct addrinfo ai_next;  /* next addrinfo structure */
};
```

```
/* function to get socket address structures */

int getaddrinfo(const char *node, const char *service,
            const struct addrinfo *hints,
            struct addrinfo **res);
```

When writing a typical client application, node and service are normally specified. When writing a server application, they both can be specified too, but in many cases only service is specified, allowing clients to connect to any node interfaces.

Applications should examine the linked list returned by `getaddrinfo` to use the adequate structure. In some cases, not all the addresses returned by this function can be used to create a socket.

The `getaddrinfo` function allocates a set of resources for the returned linked list. The `freeaddrinfo` function frees these resources.

```
/* function to free the resources allocated by getaddrinfo */

void freeaddrinfo(struct addrinfo *res);
```

Next, an example of `getaddrinfo` and `freeaddrinfo` usage (complete examples are included in appendix, in the listenServer.cpp and connectClient.cpp files):

```
n = getaddrinfo(hostname, service, &hints, &res);

/*
   Try open socket with each address getaddrinfo returned,
   until getting a valid socket.
*/

resave = res;

while (res) {
   sockfd = socket(res->ai_family,
                   res->ai_socktype,
                   res->ai_protocol);

   if (!(sockfd < 0))
      break;

   res = res->ai_next;
}

freeaddrinfo(ressave);
```

The `getnameinfo` function provides from a socket address structure, the address and service as character strings. Next, an example of use (a complete example is included in appendix, in the UDPDayTimeServer.cpp file):

```
char clienthost   [NI_MAXHOST];
char clientservice[NI_MAXSERV];

/* ... */

/* listenfd is a server socket descriptor waiting connections
   from clients
*/

connfd = accept(listenfd,
                (struct sockaddr *)&clientaddr,
                &addrlen);

getnameinfo((struct sockaddr *)&clientaddr, addrlen,
```

```
        clienthost, sizeof(clienthost),
        clientservice, sizeof(clientservice),
        NI_NUMERICHOST);

printf("Received request from host=[%s] port=[%s]\n",
    clienthost, clientservice);
```

Typically, applications do not require knowing the version of the IP they are using. Hence, applications only should try to establish the communication using each address returned by resolver until it works. However, applications could have a different behaviour when using IPv4, IPv6, IPv4-compatible-IPv6 or IPv4-mapped, etc. addresses.

There are defined some macros to help applications to test the type of address they are using, see Table 3.

Table 3: Macros for testing type of addresses.

```
int  IN6_IS_ADDR_UNSPECIFIED      (const struct in6_addr *);
int  IN6_IS_ADDR_LOOPBACK         (const struct in6_addr *);
int  IN6_IS_ADDR_MULTICAST        (const struct in6_addr *);
int  IN6_IS_ADDR_LINKLOCAL        (const struct in6_addr *);
int  IN6_IS_ADDR_SITELOCAL        (const struct in6_addr *);
int  IN6_IS_ADDR_V4MAPPED         (const struct in6_addr *);
int  IN6_IS_ADDR_V4COMPAT         (const struct in6_addr *);
int  IN6_IS_ADDR_MC_NODELOCAL     (const struct in6_addr *);
int  IN6_IS_ADDR_MC_LINKLOCAL     (const struct in6_addr *);
int  IN6_IS_ADDR_MC_SITELOCAL     (const struct in6_addr *);
int  IN6_IS_ADDR_MC_ORGLOCAL      (const struct in6_addr *);
int  IN6_IS_ADDR_MC_GLOBAL        (const struct in6_addr *);
```

4.2.3.6 Multicasting

When using UDP multicast facilities some changes must be carried out to support IPv6. First application must change the multicast IPv4 addresses to the IPv6 ones, and second, the socket configuration options.

IPv6 multicast addresses begin with the following two octets: FF0X.

The multicast socket options are used to configure some of parameters for sending multicast packets, see Table 4.

Applications using multicast communication open a socket and need to configure it to receive multicast packets. This is the main difference between the multicast and unicast socket use. Once the socket is opened and a multicast address is bind, it is required to configure at least the membership option to join to the multicast group. It will allow receiving all information sent to this multicast group.

Table 4: Multicast socket options.

IPv6 OPTION	
IPV6_MULTICAST_IF	Interface to use for outgoing multicast packets.
IPV6_MULTICAST_HOPS	Hop limit for multicast packets.
IPV6_MULTICAST_LOOP	Multicast packets are looped back to the local application.
IPV6_ADD_MEMBERSHIP	Join a multicast group.
IPV6_DROP_MEMBERSHIP	Leave a multicast group.

In the following example it is shown how can be configured, in both cases with IPv4 and IPv6, three multicast socket options: LOOP, MEMBERSHIP and TTL. The same function, `setsockopt`, is used in both cases. The main difference is the option constant values and the multicast group addresses. In IPv4 the multicast group is represented using a `sockaddr_in` structure and in IPv6 it is represented using a `sockaddr_in6`.

```
int
joinGroup(int sockfd, int loopBack, int mcastTTL,
        struct sockaddr_storage *addr)
{
   int r1, r2, r3, retval;

   retval=-1;

   switch (addr->ss_family) {
      case AF_INET: {
         struct ip_mreq      mreq;

         mreq.imr_multiaddr.s_addr=
            ((struct sockaddr_in *)addr)->sin_addr.s_addr;
         mreq.imr_interface.s_addr= INADDR_ANY;

         r1= setsockopt(sockfd, IPPROTO_IP, IP_MULTICAST_LOOP,
                  &loopBack, sizeof(loopBack));
         if (r1<0)
            perror("joinGroup:: IP_MULTICAST_LOOP:: ");

         r2= setsockopt(sockfd, IPPROTO_IP, IP_MULTICAST_TTL,
                  &mcastTTL, sizeof(mcastTTL));
         if (r2<0)
           perror("joinGroup:: IP_MULTICAST_TTL:: ");
```

```
        r3= setsockopt(sockfd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
                  (const void *)&mreq, sizeof(mreq));
        if (r3<0)
           perror("joinGroup:: IP_ADD_MEMBERSHIP:: ");

    } break;

    case AF_INET6: {
      struct ipv6_mreq    mreq6;

      memcpy(&mreq6.ipv6mr_multiaddr,
           &(((struct sockaddr_in6 *)addr)->sin6_addr),
           sizeof(struct in6_addr));

      mreq6.ipv6mr_interface= 0; // cualquier interfaz

      r1= setsockopt(sockfd, IPPROTO_IPV6, IPV6_MULTICAST_LOOP,
                  &loopBack, sizeof(loopBack));
      if (r1<0)
         perror("joinGroup:: IPV6_MULTICAST_LOOP:: ");

      r2= setsockopt(sockfd, IPPROTO_IPV6, IPV6_MULTICAST_HOPS,
                  &mcastTTL, sizeof(mcastTTL));
      if (r2<0)
         perror("joinGroup:: IPV6_MULTICAST_HOPS::  ");

      r3= setsockopt(sockfd, IPPROTO_IPV6,
                  IPV6_ADD_MEMBERSHIP, &mreq6, sizeof(mreq6));
      if (r3<0)
        perror("joinGroup:: IPV6_ADD_MEMBERSHIP:: ");

    } break;

    default:
       r1=r2=r3=-1;
  }

  if ((r1>=0) && (r2>=0) && (r3>=0))
    retval=0;

  return retval;
}
```

5. Summary of Deliverables

This chapter is intended to provide and index and short descriptions of the LONG technical documents generated during the project life.

5.1 WP1: Project Management

5.1.1 Deliverable 1.1: "Guide of project management and comm. facilities"

In the context of the LONG project this document provides the guidelines for managing technical documentation, costs statements, deliverables, etc.

5.1.2 Deliverable 1.2: "Progress report"

Progress reports from [which ones] contain the summary of technical achievements, meetings, expenditures and other interesting information related to the project management in the period that they comprise.

5.1.3 Deliverable 1.3: "Summary of extensive conclusions and guidelines"

This deliverable summarizes all conclusions and results that can be extracted from the work done in the LONG project until the time of writing (M22, September 2002). It was done at the request of the EC and reviewers after the technical review that took place in Madrid on the April 5th, 2002. It is a draft document for this deliverable 1.4 “Final summary of conclusions and guidelines”.

5.1.4 Deliverable 1.4: "Final summary of conclusions and guidelines"

The present document. This deliverable summarizes all conclusions and results that can be extracted from the work done in the LONG project. It is done at the request of the EC and reviewers after the technical review that took place in Madrid on April 5th, 2002.

5.1.5 Deliverable 1.5: "Final report"

This deliverable is intended to list the main activities and technical achievements carried out by LONG consortium between December 2000 and January 2003. Also, the deliverable lists the project documents delivered during the same period of time. It is important to remark that this document is not a technical one with technical conclusions of the project but a compilation or list done as a final management report.

5.2 WP2: Network Design and Deployment

5.2.1 Deliverable 2.1: "Description of IPv4/IPv6 available transition strategies"

This document focuses on transition mechanisms and their applicability. The transition mechanisms defined in the context of the IETF (Internet Engineering Task Force), at the time of writing, and the IPv4 to IPv6 scenarios where these mechanisms can be used were described.

A theoretical evaluation of each mechanism, concerning the performance and the scalability, were discussed and presented in the document. During the study, functional tests were carried out and the management costs for each mechanism were evaluated. Also configuration guidelines were defined for the implementations selected. The performance tests of these mechanisms were carried out in the WP4.

In order to understand where and how the mechanisms could be applied, a set of scenarios trying to cover all possible real situations during transition from IPv4 to IPv6 were defined. The transition of an IPv4 network to IPv6, for example ISP network, corporate network, or others, crosses various stages. A scenario may be interpreted as a snapshot of the network at some stage of its evolution, from an only-IPv4 to only-IPv6 network.

Some mechanisms do not make sense or are more appropriate in determinate scenarios. In this context, the applicability of the main transition mechanisms is evaluated.

5.2.2 Deliverable 2.2: "Access Technologies in LONG Project"

Deliverable 2.2 describes the IPv6 implementations over different access and transport technologies and analyses the particular characteristics of each one. Also, the state-of-the-art of the available IPv6 equipment for access technologies is presented.

The main access technologies discussed and presented in this document were UMTS, Wireless LAN ADSL, Cable TV, ISDN and ATM. These technologies were tried and tested, except in the case of the UMTS where only theoretical studies were performed.

The introduction of IPv6 is expected to be from the edge of the network to the core. In this way, the access technologies assume a relevant role in the introduction of IPv6. In fact, almost all technologies referred to above can support IPv6, except the CATV where there is no native IPv6 equipment available. The transmission of the data over CATV is based on DOCSIS specification and this standard has not yet been adapted to IPv6. However, some manufacturers, like Cisco, provide the IPv6 connectivity through a tunnel from CMTS (Cable Modem Terminal System) to the client equipment.

5.2.3 Deliverable 2.3: "Advanced Network Services: description and support on LONG network"

This document focuses on the study of functional aspects related to the deployment of advanced network services in IPv4 and IPv6 mixed scenarios. The advanced network services, that have been discussed, are mobility, multicast, anycast, muti-homing, QoS, security, DNS and DHCPv6.

The survey includes the deployment of these network services on IPv6, the comparison to IPv4 and their working in transition scenarios; in other words, scenarios where IPv6 networks are interconnected to an IPv4 network. Also, the standards, namely IETF drafts, are already referred.

Besides, some user services, like FTP, Mail, teleconference, news, IRC, RADIUS and LDAP, implemented in IPv4/IPv4 mixed scenario are described. The transition mechanisms used by theses services are described in the deliverable 2.1.

The studies carried out led to the deployment of the LONG network. In this way, local tests were performed and configuration guidelines are presented in this document.

5.2.4 Deliverable 2.4: "Network Design and Deployment"

This document describes the LONG network that will be in operation until the end of the project. The network design is presented, including the physical connections, the core and access network, the addressing plan and the basic network services as the IP routing and DNS.

Besides, the advance network services and the transition mechanisms that are supported in the LONG network are described.

The services that are not stable enough at this moment are not considered in LONG network. However, these services will be tested and evaluate until the production of the D4.4

Additionally, in this document, the test-bed network of each partner is presented in detail.

5.3 WP3: Collaborative Work Environment

5.3.1 Deliverable 3.1: "Requirements and guidelines for distributed laboratories application migration"

This deliverable provides requirements and guidelines for migrating applications and services inside distributed laboratories. Applications should be classified in different categories based on the complexity to be ported to IPv6. Scenarios and strategies for point-to-point application migration are analysed in this deliverable.

How existing IPv4 applications can be migrated to use the new IPv6 facilities is studied. If application source code is available, the migration main effort is to change application to use the new IPv6 communication API. However, this solution does not include new IPv6 features (QoS, mobility, etc.) in the existing applications. This solution only adapts applications to use the new protocol version, but using the same IPv4 features. In most cases, such kind of adaptation can be made using special scripts. A adaptation script replaces the standard IPv4 API by the new IPv4/IPv6 version in the source code automatically.

If IPv6 advanced facilities should be considered, an application reengineering is required and developers must know the structure and details of application before starting.

The document includes detailed description of communication APIs of the most popular programming languages: porting C programs (Berkeley socket interface and Winsock interface), porting C++ programs (utilising the same APIs as C programs, but using class abstraction to hide implementation details of protocol version), migration of SOCKs based applications, Java porting and Perl scripting programming language. Many applications only use the basic socket interface extensions for IPv6 and the migration of this kind of applications is simpler than those which use the advanced socket interface. Implementation details of these APIs are studied in the appendixes A and B. An example of the use of the sockets API is presented with some examples taken from the porting of MGENv6.

The last part of the document provides the migration guidelines, mainly related to the replacement of the standard IPv4 API by the new IPv4/IPv6 one. When redesigning existing applications or building new ones is considered, some additional recommendations should be taken into account. These considerations are focused on the availability of services working over heterogeneous scenarios, a mixture between IPv4 and IPv6 islands. Besides, application performance analysis has been included to evaluate the new network scenarios. The analysis is devoted to study delays and the overheads introduced when IPv6 protocol is used.

5.3.2 Deliverable 3.2: "Guidelines for migration of collaborative work applications"

This deliverable is focused on the description of general characteristics during the migration process of distributed collaborative applications. Avoiding application dependencies with IP protocol is one of the main points of porting to IPv6. The document is divided in two parts: description and evaluation of migration guidelines.

After the revision of the group communication used by collaborative applications, the first part of the deliverable is focused on the description of migration guidelines for these applications. The revision of the group communication is based on the analysis of the layered collaborative applications architecture to find the subsystems that will be changed during the migration process: application control, flow management, end-to-end QoS and network subsystem. The group communication is studied to understand the interaction between collaborative applications. Some dependencies with IP protocol are identified in collaborative applications which can be resolved using FQDN (Fully Qualified Domain Name) instead of IP addresses.

The document includes a second part devoted to demonstrate the viability of the previous proposed guidelines. ISABEL collaborative is used as the best case study because we have the source code and it has been used to maintain some technical co-ordination meetings inside the project, working over the IPv6 network provided by the LONG project.

ISABEL porting to IPv6 requires changes in two of its subsystems: the session co-ordination layer and the reliable/unreliable transport layer.

The ISABEL session co-ordination layer contains a user registry system based on the participant IP addresses. When IPv6 is used, the FQDN should be considered as the symbolic names to identify participants in the new network environment. This solution is valid during transition period, with ISABEL IPv4 and IPv6 participants collaborating simultaneously.

The ISABEL unreliable transport layer is migrated using an general communication library which hides communication implementation details to applications, in particular the IP protocol version. The ISABEL reliable transport layer is migrated using an IPv4/IPv6 translator at application level. For the reliable transport layer another experimental solution is explained. This solution is based on the group communication model using a reliable multicast protocol.

The ISABEL migration process includes a test plan to check the functionality over the new network environment. The evaluation is completed with the use of the ISABEL application during the LONG distributed meetings.

5.3.3 Deliverable 3.3: "Guidelines for migration of collaborative work applications working over asymmetric channels"

The third work package document studies special application components and network scenarios. The aim is to provide guidelines to simplify migration process in such cases.

There are network scenarios, in which application functionality could be affected when IPv6 is used, especially relevant in ADSL asymmetric channels. As a result of the developed study, the application packet size has been found as a critical parameter. The IPv6 networks do not allow fragmentation in intermediate nodes. Therefore, the IPv6 fragmentation is only in the source node. PMTUD mechanism is recommended in order to guarantee application datagrams will be sent from source to destination without problems. If datagrams sent to a

destination must be fragmented because an intermediate link has a small MTU value, all packets will be resized to go through this link. In the real time applications the performance can be affected when the packet size changes.

When working with heterogeneous networks, many collaborative applications often use logical intermediate components. They are designed not only to solve previous described interconnection problems but also to adapt different application constraints. These intermediate components are relevant when interconnecting IPv4 with IPv6 nodes in the same application. Therefore, dual stack interface is required in the node that supports such intermediate component. The document establishes that standard guidelines defined in the previous deliverables are also valid for porting intermediate components to IPv6.

Finally, the deliverable presents some collaborative applications that have been migrated in the fields of streaming video and network games during this period.

5.4 WP4: System Exploitation, Trials and Evaluation

5.4.1 Deliverable 4.1: "First phase trials scenario specifications"

Deliverable 4.1 specifies the trials that are to be carried out in the first phase (local tests). These tests are mainly focussed on performance, and deal with IPv4/IPv6 transition mechanisms, and access and transport technologies. The results of the specified tests are described in Deliverable 4.2.

To obtain homogeneous performance results among all partners, a methodology has to be established. This includes identifying the most relevant performance descriptors, that for multimedia applications like ISABEL are the transmission rate at which packet loss begins (that in general represents the maximum rate achievable in normal conditions), delay and jitter. The tests should be defined in such way that all relevant parameters (configuration, topology, traffic conditions, etc.) were clearly specified. Since absolute performance values are highly dependent on the available hardware, software version, etc., comparison among interesting scenarios is the most valuable result. To approximate the behaviour of ISABEL, the MGEN UDP traffic generator and measurement tool is used. MGEN has been ported to IPv6 by the LONG consortium. The parameters to vary on each test are packet size (to be able to provide feedback to application deployment), and the amount of traffic generated, since traffic load is paramount for network behaviour. The traffic pattern generated is a constant bit rate one, since the tests aim to characterize the performance of multimedia applications similar to ISABEL (large adjustable constant bit rates). These tests are complemented with TCP traffic measurement using the Netperf tool, since applications using TCP are very popular, and even ISABEL uses TCP for part of its communication. When several network technologies or configurations are tested, the number of cases considered in the proposed tests is so large that the addition of a new parameter would be unfeasible with the precision required.

Considering that measurement tools are applications (running on top of a given OS, etc.), as opposed to specialized measurement hardware, they are expected to measure the behaviour that other applications will experiment. However, as a disadvantage, the results may not characterize accurately the behaviour of the network itself, but with results that include the end system behaviour.

Tests for IPv4/IPv6 transition mechanisms, and access and transport technologies are presented in Deliverable 4.1. Tests are gathered into appropriate categories that allow comparison among them: in transition mechanisms, we can distinguish among solutions that solve similar problems: inter-network IPv6 to IPv6 connectivity over IPv4 legacy infrastructure, intra-network IPv6 to IPv6 connectivity, and IPv6 to IPv4 connectivity. Regarding to access and transport technologies, different configurations are exercised for IPv6 over each one considered (ISDN, ADSL, CATV, 802.11b, ATM).

5.4.2 Deliverable 4.2: "Report on first phase trials and evaluation report"

Deliverable 4.2 presents the results obtained after performing the tests specified in [D4.1]. Conclusions related with this deliverable are presented in the WP4 conclusions included in the chapter of general conclusions of this document.

5.4.3 Deliverable 4.3: "Second phase trials specification"

In this deliverable the specifications for the second phase trials, dealing with distributed tests, are presented, focussing on advanced application and network services in a transitioning environment. We first describe the application testbed methodology, and later the network testing framework.

The advanced network services (DNS, multicast, diffserv, security, mobility, anycast or muti-homing) are intended to be used by the application services considered in this document (mainly ISABEL, but also HTTP, FTP, IRC, News, Mail, LDAP). IPv6, and the IPv4/IPv6 transition network, are expected to appropriately support services widely deployed in IPv4, such as DNS or muti-homing, and to foster the deployment of currently less common services such as multicast, diffserv, security, mobility or anycast. This claim has to be confirmed.

On the other hand, it is assumed that IPv6 success will highly depend on applications, first on its availability, and later, on the possible enhancements that IPv6 can deliver to them. Deploying IPv6 applications such as ISABEL in complex networks will provide valuable knowledge for both application programming and deployment.

Several partners will test the applications deployed in the LONG network. A special case of applications that will be locally tested (provided that appropriate implementations are available) are AAA with RADIUS, that enhances support for ISDN access, and NFS/RPC, that enables convenient file access in IPv6-only hosts. These two applications are naturally deployed as a local user service, but they are considered as important for IPv6 deployment by LONG consortium.

Network service testing, is in general performed in two steps: first, exhaustive tests are carried out locally by one partner, to obtain information about how the service should be better deployed in the network. These local tests usually include performance evaluation if performance is a relevant parameter for the service (for example, when diffserv or multicast are considered). The behaviour of some relevant application services (for example, ISABEL) is also used to test the network services: for example, ISABEL can be used for evaluating different solutions for muti-homing, since some of them will not preserve a TCP session, thus incurring in ISABEL service disruption. Then several partners will be involved in the stable deployment of the network services, that usually involves several transition scenarios, and in field trials. Some services that are not considered mature enough for distributed deployment, such as DHCPv6 and security, are just locally tested.

5.4.4 Deliverable 4.4: "Conclusions and Guidelines from Experiments"

This document reports the results of the performance and functional experiments proposed in D4.3. It also summarizes the trials performed during the project.

5.5 WP5: Dissemination and Implementation

5.5.1 Deliverable 5.1: "Dissemination and Use Plan"

This report states the planned activities that were to be carried out by the project LONG within the workpackage 5 about dissemination and implementation. This document was delivered at month 6 since the beginning of the project. It includes the first dissemination activities performed during this period of time and a plan of the ongoing and planned activities for the rest of the duration of the project.

5.5.2 Deliverable 5.2: "Web site of the project for internal use including the relevant documents of the project" & Deliverable 5.3: "Public Web site with general information about the project and related topics and projects"

This document tries to describe all the characteristics of the two LONG official Web sites: the public site and the internal one.

The public server is, of course, accessible to all Internet users, while user's login and password are needed to access the Internal server. To get this password the internal site maintainer has to be contacted.

Some of the partners have developed also some partner-internal Web servers to do some internal dissemination in their own organization. Commonly, these Web servers are only accessible from their own internal networks so that they are not described in this document.

5.5.3 Deliverable 5.4: "Publication of the contributions to the Workshop and/or other dissemination events"

This document is a compilation of the dissemination activities of the project. It includes papers published in Journals, papers presented at Conferences, RFC submitted to IETF, presentations done in several international events, presentations of the activities of the project given in meetings of other projects, Guidelines published, list of attendance to co-ordination activities with other projects including IPv6 Cluster meetings, demonstrations organized directly by the project and demonstrations where LONG participated. All these activities are well documented in the website of the project: long.ccaba.upc.es.

5.5.4 Deliverable 5.5: "Final version of the Web site including all the planning, set-up and results of the experiments along with the guidelines and recommendations"

This report presents a brief description of the structure and contents of the web sites of the project, the internal one and the public one. A more detailed description of the public one is presented.

6. Real User Experiences

Several experiences involving real users have been made over the LONG network infrastructure. Among them, we highlight the following:

- IPv6 Forum 2001 - Distribution with ISABEL of IPv6 Forum 2001. January 2001. Participants: UPM (Spain), UPC (Spain), UC3M (Spain), UEV (Portugal), PTIN (Portugal), UNAM (Mexico), ETRI (Korea), ICSI (USA), CRC (Canada), ULB (Belgium), MCLAB (Switzerland).

  This event was distributed using ISABEL.

- IPv6 Forum 2002 - Distribution with ISABEL of IPv6 Forum 2002. March 2002. Participants: PTIN (Portugal), UPC (Spain), MCLab GmbH (Switzerland), Berkeley University (USA), Université Libre de Bruxelles (Belgium), ETRI / PEC (Korea), University of Evora (Portugal), UPM (Spain), UC3M (Spain), TID (Spain), RedUNAM (Mexico), Communications Research Centre Canada (Otawa).

  The event was distributed using ISABEL over IPv6. A demonstration of the LONG services was also available.

- 2002 Valencia Campus Party - The Campus Party is an event that joins together people from Europe and Latin America to share experiences related with computers and communications. This event, that was first organized in 1997, was held in August 2002 with 3000 users. This year, the Euro6IX project provided IPv6 connectivity, and the services provided by the LONG were open to all the participants: DNS, IRC IPv6, mail SMTP IPv6, the aFoto application, video streaming, ISABEL, web, ftp and some games.

- IST2002 – In the IST conference held at Copenhagen, several demos were performed, including ISABEL connectivity with most of the members of the consortium, and access to the services provided by the LONG services: DNS, IRC IPv6, mail SMTP IPv6, the aFoto application, video streaming, web, ftp, etc. Connectivity to the LONG network was provided by Euro6IX (as long as part of its booth space).

It should also be noted that most of LONG project meetings have been hosted using ISABEL over IPv6.

In general, both the network and the applications have proven to be stable in the experiences that have been performed.

7. Future Action Points

During LONG project many efforts have been put in improving IPv6 and bringing this new protocol nearer to reality. However there are several action points where more effort should be put that are listed in this section.

7.1 Access Networks

Emerging applications require "Always-on" connections. Also new mobile terminals connected to the Internet are expected to appear in the coming years. Therefore, many new IP addresses will be demanded in the next years. This is one reason why introduction of IPv6 will be performed from the edge to core of the networks. So far, almost all access technologies support IPv6. The evolution of the IPv6 implementation over this technology and new mobile devices, such as PDA, should be followed.

7.2 Transition to IPv6

The main focus of LONG project has been the production of guidelines to simplify networks transition to IPv6. The project has produced guide documents not only for networks but also applications transition. These results will be used to start transition of current IPv4 networks as part of LONG exploitation plan.

The simplest application field will be universities. University partners have enough experience to start the IPv6 network deployment. In concrete Spanish universities have received notification from core network administrators in order to allow IPv6 network connection to European IPv6 network. Therefore, it is possible to start networks transition in the near future.

A future action point that has been identified is to write a document of guidelines for coexistence or migration to IPv6, as the document for porting applications that LONG has already deployed. To work and write about this topic, expertise, real scenarios and pilot networks are needed as part of real networks transition. This document should include the guidelines for BGP configuration, muti-homing issues, mobility, multicast and all services demanded by real users.

Before any organization would think of migrating to IPv6, guarantees are needed to assure that the equipment is ready to support at least the same load than the current one and that migration could be done in a smooth way. In this sense the performance evaluation of commercial routers under high loads and transition mechanisms is another issue to be considered in the future.

Although new networks can be defined as IPv6 only networks, transition will start with current IPv4 networks. Therefore, the IPv4 and IPv6 will coexist for several years where the identification of possible transition scenarios is very important. The requirements for the deployment of theses scenarios on ISP/Telcos networks are different from corporative networks.

In this context, activities like IPv6 Task Force studies must be addressed to provide a dissemination mechanism of LONG project results, which will be used to help coordination between ISP/Telcos and final users organizations.

For all transition mechanisms, better implementations for the proposed mechanisms are required. For example, DNS support should be integrated with NAT-PT free software

implementations, to allow bidirectional NAT. Application Level Gateway functionality, i.e., translation at application level for typical application protocols conveying IP address information should be also provided at NAT-PT.

7.3 Advanced Services

7.3.1 QoS

Another interesting point is the evaluation of end-to-end QoS on IPv6 (including comparison with IPv4) and hybrid scenarios with IPv4 QoS and IPv6 QoS. The use of an integrated tool such as NetMeter (with synchronization via GPS) could be very practical.

Performance evaluation of commercial routers under high loads with QoS should be also considered in the future.

7.3.2 Mutihoming solutions.

Mutihoming is a service that should be properly addressed for making IPv6 ready to operational usage. Current IPv4 solutions based on BGP route injection presents scalability limitations, since it contributes to the exponential growth of the BGP table size. Alternative solutions should be addressed, either based on separation of the identifier/location roles, improvement of the route management at the routing system, addressing of the capabilities for fault detection and re-routing control at the end-hosts, or a combination of them.

7.3.3 Anycast.

Further study on the routing mechanisms for provisioning the anycast service in the network must be addressed. End-to-end communication through anycast addresses should be studied, especially in the case of the UDP transport protocol, and stateless communication.

However, network level support for anycast service detection, even over stateful communication (for example, using the TCP transport protocol, or using UDP in services for which the application has to preserve some state) should be very valuable.

7.3.4 DHCPv6.

Although there is still debate on whether DHCP is required for IPv6, it is expected that such easy to control administrative tool, with the capability of providing different configuration to different users would be a very attractive option for network managers. KAME, the reference IPv6 free software developer, does not plan nowadays to deploy and maintain such software. Therefore, coding and maintaining this service would be an added value to the IPv6 community.

7.3.5 Renumbering models and tools.

The current addressing model relies on the assignment to customer of part of the address space of the current provider, requiring a complete change of the addresses when the provider is changed. In a competitive commercial scenario, the change in the provider should not be penalized by the current technology. New addressing paradigms or tools for the current addressing model should be explored in order to overcome this problem.

7.3.6 Mobility

Current specifications and implementations for mobile IPv6 suffer from packet loss when handovers are to be performed. To this effect contribute several issues: mobile packets routing, the requirement of performing Duplicate Address Detection when switching to another network, etc. More research is required in this field.

The security issues related to mobility were recently defined. The last mobility IPv6 draft will become RFC, consequently, new implementations will emerge from different manufactures. The deployment of realistic mobility scenarios, as well as the interoperability of different implementations should be tested and evaluated.

The commercial deployment of Mobile IP services will require that mobility IP solutions support different administrative domains. The solution that has been proposed is based on AAA protocols as RADIUS or Diameter. The evaluation of this solution or others implementation models should be researched.

So far, the mobility and the multicast have been discussed separately. However, a mobile node should not have less functionality in relation to another IPv6 node. One solution presented in last mobility-drafts is the establishment of a tunnel between the Home Agent and Mobile Node to send the MLD messages and multicast data. The evaluation of this solution and alternative solutions should be addressed.

8. Glossary and Abbreviations

ADSL Asymmetric Digital Subscriber Line

ATM Asynchronous Transfer Mode CATV

BGP Border Gateway Protocol

BIS Bump In the Stack

CATV Cable Television

CMTS Cable Modem Terminating System

CSCW Computer Supported Collaborative Working

DAS Default Address Selection

DHCP Dynamic Host Configuration Protocol

DSCP DiffServ Codepoint

DNS Domain Name System

DOCSIS Data Over Cable Service Interface Specification

DSTM Dual Stack Transition Mechanism

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IGMP Internet Group Management Protocol

IGP/EGP Internal/External Gateway Protocol

IETF Internet Engineering Task Force

IPv6 Internet Protocol version 6

ISATAP Intra-Site Automatic Tunneling Protocol

ISDN Integrated Services Digital Networks

IRC Internet Relay Chat

ISP Internet Service Provider

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

mBIS Multicast extensions to BIS

MIPL Mobile IPv6 for Linux

MLD Multicast Listener Discovery

MTP Multicast Translator based on IGMP/MLD Proxying

MTU Maximum Transfer Unit

NAT-PT Network Address Translation - Protocol Translation

NFS/RPC	Network File System / Remote Procedure Call
NGN	Next Generation Networks
NRN	National Research Network
NREN	National Research and Education Network
NTP	Network Time Protocol
PDA	Personal Digital Assistant
PIM	Protocol Independent Multicast
PMTUD	Path MTU Discovery
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RFC	Request for Comment
SIIT	Stateless IPv4-IPv6 Translator
SOHO	Small Office Home Office
SVC/PVC	Switched Virtual Circuit / Permanent Virtual Circuit
TCP	Transmission Control Protocol
TLA	Top Level Aggregation
TOS	Type Of Service
TRT	Transport Relay Translator
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
WP	Work Package

9. References

[Bound 02] *"Dual Stack Transition Mechanism (DSTM)"*. J.Bound. February 2002. (28814 bytes) (Status: Internet Draft)

[D4.1] *First phase trials scenario specifications*. LONG project deliverable 4.1.

[D4.2] *Report on first phase trials and evaluation report.* LONG project deliverable 4.2.

[De Clercq 02] *Connecting IPv6 Domains across IPv4 Clouds with BGP*. Tri Nguyen, Gerard Gastaud, Francois Le Faucheur, Dirk Ooms, Jeremy De Clercq, Stuart Prevost. January 2002. (23825 bytes) (Status: Internet Draft)

[EXSMS] *"Extension Header for Site-Multi-homing Support"*. http://www.rfc-editor.org/internet-drafts/draft-bagnulo-multi6-mhexthdr-00.txt. October 2002

[Lee 02] *Dual Stack Hosts using 'Bump-in-the-API' (BIA)*. Seungyun Lee. April 2002. (32534 bytes) (Status: Internet Draft)

[RGII] *"Random generation of interface identifiers"*, http://www.ietf.org/internet-drafts/draft-soto-mobileip-random-iids-00.txt

[RFC1933] *Transition Mechanisms for IPv6 Hosts and Routers*. R.Gilligan, E.Nordmark. April 1996. (Format: TXT=47005 bytes) (Obsoleted by RFC2893) (Status: PROPOSED STANDARD)

[RFC2529] *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels.* B.Carpenter, C.Jung. March 1999. (Format: TXT=21049 bytes) (Status: PROPOSED STANDARD)

[RFC2710] *Multicast Listener Discovery (MLD) for IPv6.* S.Deering, W.Fenner, B.Haberman. October 1999. (Format: TXT=46838 bytes) (Status: PROPOSED STANDARD)

[RFC2765] *Stateless IP/ICMP Translation Algorithm (SIIT).* E.Nordmark. February 2000. (Format: TXT=59465 bytes) (Status: PROPOSED STANDARD)

[RFC2766] *Network Address Translation - Protocol Translation (NAT-PT).* G.Tsirtsis, P.Srisuresh. February 2000. (Format: TXT=49836 bytes) (Updated by RFC3152) (Status: PROPOSED STANDARD)

[RFC2767] *Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS).* K.Tsuchiya, H.Higuchi, Y.Atarashi. February 2000. (Format: TXT=26402 bytes) (Status: INFORMATIONAL)

[RFC3056] *Autonomous System Confederations for BGP.* P.Traina, D.McPherson, J.Scudder. February 2001. (Format: TXT=20529 bytes) (Obsoletes RFC1965) (Status: PROPOSED STANDARD)

[RFC3089] *A SOCKS-based IPv6/IPv4 Gateway Mechanism.* H.Kitamura. April 2001. (Format: TXT=25193 bytes) (Status: INFORMATIONAL)

[RFC3142] *An IPv6-to-IPv4 Transport Relay Translator.* J.Hagino, K.Yamamoto. June 2001. (Format: TXT=20864 bytes) (Status: INFORMATIONAL)

[Templin 02] *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP).* Dave Thaler, Tim Gleeson, Mohit Talwar, Fred Templin. April 2002. (32908 bytes) (Status: Internet Draft)