Porting applications to New Generation Internet, the ISABEL case study

Tomás P. de Miguel, Juan Quemada, Eva Castro*, Santiago Pavón, Gabriel Huecas, Tomás Robles, Joaquín Salvachua, Javier Sedano, María José Perea

Department of Telematic Engineering of the Technical University of Madrid Spain,

Systems and Communications Group (GSyC) Experimental Sciences and Technology Department (ESCET) Rey Juan Carlos University (URJC) Madrid Spain*

Abstract

New Generation Internet will be based on new IP protocol version design to solve addressing problem and provide new functionalities. This paper describes guidelines to help the porting of applications to the new protocol IPv6, especially during the transition period. ISABEL collaborative application is used to exemplify the porting methodology. ISABEL has been used to support meetings and workshops in which both IPv4 and IPv6 nodes collaborate simultaneously.

I. INTRODUCTION

Ordinary users see the Internet through its applications they use daily for their work, from electronic mail to the WWW navigation. In general, users have had a great deal of success with all Internet applications, even the more simple ones such as FTP or Telnet, and many companies have decided to reorganize their networks on the Internet model by creating Intranets. The worldwide success of the Internet keeps pace with the success of the network architecture called Internet Protocol Suite, best known as TCP/IP, on which they are based. The actual IPv4 design is based on two characteristics, which have been the points of strength for Internet: universal addressing and best effort communication. These two characteristics are becoming the Internet main limit and forcing the introduction of new generation protocol IPv6.

Next Generation Internet based on IPv6 has been design to solve the actual addressing problem and obtain potential advantages mainly in fields like mobility, security or quality of service.

Protocols replacement is not easy and the transition from IPv4 to IPv6 is no exception. Protocol replacement is typically deployed by installing and configuring the new protocol on all nodes within the network and verifying that all node and router operations work successfully. Although this might be possible in a small or medium sized organization, the challenge of making a rapid protocol replacement in a large organization is very difficult. Additionally, given the scope of the Internet, rapid protocol replacement becomes an impossible task. Therefore, migration from IPv4 to IPv6 will not happen overnight. Rather, there will be a period of transition when both protocols will be in use over the same infrastructure. The designers of IPv6 [1] have been defined the following transition criteria:

- Existing IPv4 hosts can be upgraded at any time, independent of the upgrade of other hosts or routers.
- New hosts, using only IPv6, can be added at any time, without dependencies on other hosts or routing infrastructure.
- Existing IPv4 hosts, with IPv6 installed, can continue to use their IPv4 addresses and do not need additional addresses.
- Little preparation is required to either upgrade existing IPv4 nodes to IPv6 or deploy new IPv6 nodes.

To address this, the designers of IPv6 have created technologies and types of addresses so that nodes can communicate with each other in a mixed environment, even if they are separated by an IPv4 infrastructure. Network transition has been discussed in detail; however applications should be reviewed to complete the porting process.

Existing applications cannot use IPv6 without previous modifications. TCP/IP network architecture is not perfectly layered. Applications use IP addressing to identify destination nodes. Although many times symbolic names are provided, standard communications library based on sockets interface use only IP addresses and therefore IP address management should be maintained as part of application and complemented the DNS management. The new IPv6 addresses structure forces the use of new transport layer interface (the IPv6 sockets layer interface). Hence, from the applications point of view, the IPv6 deployment requires changes in the existing code and maybe the addition of new communication design concepts.

This article describes guidelines [2] to help not only the design of new IPv6 applications, but mainly the porting of existing applications to allow IPv4 and IPv6 coexistence during the transition period. The paper concludes with the analysis of ISABEL collaborative application, which is used as an exercise to identify typical porting problems and show the way to solve them.

II. APPLICATIONS PORTING PROBLEMS

When porting IPv4 applications to IPv6, there are different difficulty 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 and data structures to be ready for the new IPv6 API. However, if application makes more exhaustive use of IP addresses or advanced network facilities should be considered, such as multicasting, raw sockets, quality of service or mobility, the porting requires a complete application analysis and much more porting effort.

Standard transport API based on sockets interface makes the version protocol visible to the application and therefore if

protocol changes address data structures should be adapted to the new environment. Moreover, other facilities such as conversion functions between hostnames and IP addresses are different in the new IPv6 environment too.

Differences between IPv4 and IPv6 sockets APIs are related to a new socket address structure to carry IPv6 addresses, new address conversion functions and several new socket options [3]. 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. There are additional recommendations to access more advanced features like raw sockets or header configuration [4].

However, in other applications not only communications block but other modules or application parts, include dependencies on IP addresses and must be reviewed deeply. All these dependencies can be grouped in one of the following subjects:

- Parsing IP addresses.
- Use of special IP addresses.
- Local IP address selection.
- Application Data Unit (ADU) fragmentation.
- Use IP addresses to identify application elements.

A. Parsing IP addresses

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, the resolver. Applications call functions in a system library to access it. Developers should change the use of the IPv4 resolution functions by the new IPv6, or by the protocol-independent ones if it is possible.

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 names resolver until one of them works. However, applications could have a different behavior when using IPv4, IPv6, IPv4-compatible-IPv6 or IPv4-mapped, etc. addresses.

Sometimes applications accept IP addresses and include parsers to translate from textual to binary form and validate inputs. 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 138.4.2.10. 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. However, it is used as a separator between the address and the service port number in IPv4 networks. Applications can use the same format as the literal IPv6 addresses in URLs, enclosing the IPv6 address within square brackets, to solve the ambiguity [5], for instance: http://[2001:720:1500:1::a100]:80/index.html

B. Use of special addresses

There are some special addresses, which some applications use in certain circumstances. The most frequent is localhost interface. Although it is possible to use the symbolic name, most IPv4 applications use the IP address.

When moving application from IPv4 to IPv6, hard coded IP addresses fail trying to establish connections. So the use of names instead of IP addresses is recommended, because names could be reconfigured without changing application source code. Developers should review application source code to include names instead of IP addresses as can be found in the table below:

Symbolic name	IPv4 address	IPv6 address
INADDR_ANY	0.0.0.0	::
IN6ADDR_ANY_INIT	0.0.0.0	::
INADDR_LOOKBACK	127.0.0.1	::1
IN6ADDR_LOOKBACK_INIT	127.0.0.1	::1
INADDR_BROADCAST	255.255.255.255	Not exist

C. 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 behavior 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. 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 [6].

D. ADU fragmentation

Application Data Unit (ADU) is the block of data sent or received in a single communication operation at application level. Sometimes ADU is different from the amount of data that can be handled on the network interface; the Transfer Unit (TU) and therefore ADU is fragmented.

The problem is how to select the most 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, IPv6 recommends [7] that all IPv6 nodes should implement PMTU Discovery (PMTU-D) to optimize the throughput of fragmented ADUs. PMTU-D is a mechanism to use the longest IPv6 packet size than 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 message to the source address and the source host adjusts the packet size based on the ICMP message.

Unfortunately, PMTU-D is only a recommendation not a mandatory network module. There are applications, which realize their own packetization process. 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.

E. Use IP addresses to identify elements

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.

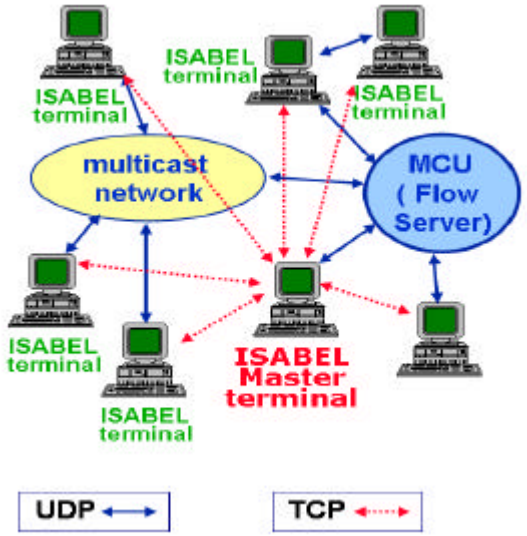
The best solution to this problem is the use of identifiers independent of the network layer or maybe the FQDN. The FQDN remains invariable over the time although it's associated IP address can change.

III. ISABEL PORTING TO IPV6

The ISABEL CSCW application [8] is a group collaboration tool for the Internet, which uses standard TCP-UDP/IP protocols. ISABEL has been developed at the Department of Telematic Engineering of the Universidad Politécnica de Madrid (DIT-UPM).

ISABEL uses an innovative service concept, which adapts collaboration sessions to user needs, by including tailored service definitions as fairly free way of interaction (tele-meeting), strict interaction control by the lecturer or educator (tele-class) or strict distributed congress driven based on centralized control (tele-conference). A service definition language can be used to defined any ISABEL service as an easy modification if the existing services or complete definition of new ones.

ISABEL platforms are constructed over IP networks using either IP unicast, IP multicast or mixtures of both, as shown in the next figure, where a mixture of unicast and multicast is used. The Flow Server is the ISABEL equivalent to a videoconferencing MCUs and must be used for creating multi-points over unicast connectivity. The terminals can be used as flow servers as they always include one. Dedicated flow servers can be also used.

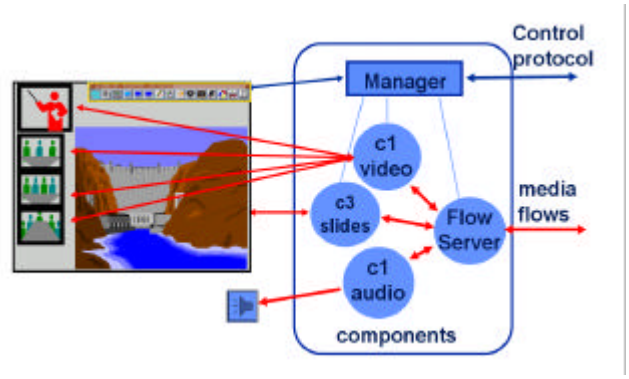


An ISABEL session is coordinated by a control unit (the Master) of the session, which acts also as participant's session registry for the rest of the ISABEL terminals. The session participants connect via an ISABEL terminal, which has to request to the Master the acceptance into the session. Once the ISABEL terminal has been allowed by the Master to enter the conference, it will enter the session and participate in the activities taking place in the session. An ISABEL terminal is a multimedia PC with audio video and network interfaces, which runs Isabel.

The architecture of an Isabel terminal includes a set of media components, which are controlled by a management agent [9]. The management agent is written in TCL-Tk and implements the management policy of a given service. The media components are written in C/C++ and manage the media flows

and the media presentation at the Isabel terminals (sites) connected to the session. The media components are: audio, video, shared workspace, VNC windows OS, etc.

The next figure shows the application architecture, which is composed of two module types. The Manager devoted to connect to the Master (using TCP connection) and control the local ISABEL behavior. The second module type is media component used to exchange UDP datagram flows between ISABEL terminals. The UDP flows of the media components are aggregated by the local flow-server. The UDP flows can be exchanged over unicast or multicast.



The idea which guides ISABEL porting to IPv6 (ISABEL$_{v6}$) is the interoperability between IPv6 and IPv4 sites. The simplest way is to run dual stack to access both IPv4 and IPv6 resources. Besides usual changes in the network dependent parts of the program, three main developments had to be addressed:

1. The adaptation of the ISABEL Control Library (TCP
2. The adaptation of the ISABEL Media Library (UDP)
3. The elimination of all IPv4 addresses dependencies

The adaptation of the ISABEL TCP library needed a significant amount of work because TCL-Tk does not supported IPv6 when the port was made. Several options were considered, but the realization of an IPv4 to IPv6 translator proved to be the most effective and was used in ISABEL$_{v6}$. The name of this module is "toTcp6". "toTcp6" is written in C++ and allows an IPv4 application to establish an application IPv6 tunnel to the remote terminal which is again translated to IPv4 by the "toTcp6" module. Therefore ISABEL still uses IPv4 internally, but is able to work over and IPv6 only network, due to internal double stack support.

The adaptation of the Isabel UDP library was very standard because this library is in C/C++ and therefore only the adaptation to the IPv4/IPv6 capable socket interface was made following standard procedures.

Finally the elimination of the IPv4 address dependencies in ISABEL was by far the most difficult task. As in many other applications, the IPv4 address was used in ISABEL as the site identifier. Therefore a new label had to be selected as unique identifier in the ISABEL Session. This change obliged to rewrite significant parts of the management component of ISABEL. A report of the porting can be found in [10]. Therefore, no only IPv6 but also IPv4 sites which pretend to interact with IPv6 sites must run the new ISABEL$_{v6}$.

The ISABEL$_{v6}$ is therefore an application working over dual stack, although it can work with IPv6 connectivity only. It is prepared to work in most kind of IPv6 network transition scenarios.

ISABEL has been used to distribute many events. Development started in 1993 and it was first used in the RACE Distributed Summer School on Advanced Broadband Communications ABC93, in July of 1993. Since then many events have been performed mainly within European research projects of RACE, ACTS and IST. The best known ones were the RACE Summer Schools on Advanced Broadband Communication, the tele-work experiments within Ericsson, the various Global 360 events, the IDC conferences or Madrid Global IPv6 Forum Summits. During all this trials ISABEL has been evolving and tuned to achieve it present shape.

ISABEL$_{v6}$ was developed in the context of the European IST LONG project [11]. The distribution of the Madrid Global IPv6 Summit in May 2002 was designed as a validation of maturity of IPv6 technologies, connecting 15 sites and including therefore native IPv4, native IPv6, as well as IPv6 over IPv4 tunnels. More recently, The New Generation Internet Workshop distributed in February 2003 to 25 universities and research laboratories in Europe and Canada has been used to demonstrate the ISABEL$_{v6}$ flexibility in real scenarios during transition to IPv6.

V. REFERENCES

[1] S. Bradner, A. Mankin. *The Recommendation for the IP Next Generation Protocol*. RFC 1752. January, 1995.

[2] T. P. de Miguel, E. Castro. *Programming Guidelines on Transition to IPv6*. North American IPv6 Task Force (NAV6TF), January, 2003.

[3] R. Gilligan et al. *Basic socket interface extensions for IPv6*. RFC 2553. March 1999.

[4] W. Stevens, M. Thomas. *Advanced sockets API for IPv6*. RFC 2292. February 1998.

[5] R. Hinden, S. Deering. *IPv6 addressing architecture*. RFC 2373. 1998.

[6] R. Draves. *Default address selection for IPv6*. http://www.ietf.org/internet-drafts/draft-ietf-ipv6-default-addr-select-09.txt. August 2002.

[7] J. McCann et al. *Path MTU discovery for IPv6*. RFC 1981, August 1996.

[8] *Isabel CSCW Application*. http://isabel.dit.upm.es

[9] T. P. de Miguel, et al. *Isabel - Experimental Distributed Cooperative Work Application over broadband Networks*. Springer-Verlag, vol. LNCS 868, pp. 353-362, Sep. 1994.

[10] E. Castro, T. P. de Miguel. *Guidelines for migration of collaborative work applications*. LONG D3.2_v2. July 15 2002.

[11] *LONG Laboratories Over Next Generation Networks* IST-1999-20393. http://long.ccaba.upc.es