Adding IPv6 support to H323: Gnomemeeting/openH323 port

Ch. Bouras[1,2], A. Gkamas[1,2], S. Josset[3], K. Stamos[1,2*]
1 Computer Engineering and Informatics Dept., Univ. of Patras, GR-26500 Patras, Greece
2 Research Academic Computer Technology Institute, Riga Feraiou 61, GR-26221 Patras, Greece
3 Alcatel Space, 26 avenue J-F Champollion, 31037 Toulouse Cedex1, France
e-mail: {bouras, gkamas, stamos}@cti.gr
email: sebastien.josset@space.alcatel.fr

Abstract: **This article deals with the port towards the new IPv6 Internet network of voice and multi-media over IP applications. We explain in detail our experience in porting Gnomemeeting, the well-known Linux based videoconference software, and more particularly the H323 library which implements the signaling protocols (call establishment, transfer, codec negotiation,…).**

1. INTRODUCTION

One of the major problems the Internet network will have to face in the near future, is the shortage of IPv4 addresses, the unique identifier of a machine on the network. If we consider the latest technological developments (like 3G Networks, UMTS, always on devices, embedded systems, etc), we can safely forecast that in the future the number of hosts will increase dramatically. We could run out of new addresses, as each new generation mobile terminal proposing Internet services requires the use of a new Internet address. The deployment of an IPv6 Internet is a natural solution for this problem of shortage of IPv4 addresses.

As a full replacement at a fixed time of every IPv4 address by an IPv6 address is not possible, the IPv6 network is set up gradually, in parallel with the IPv4 network. During this transition phase, a large effort must be dedicated on the applications, which have to be able to use either the IPv4 network, or the IPv6 network, and ideally both, to provide application bridges. Without applications, the wide addressing of the IPv6 network does not have any reason to be used one day.

Although many articles explain how to port an IPv4 program to IPv6, few are dealing with the simultaneous support of both IPv4 and IPv6. In [1] and [2] for example, some general instructions are given on porting applications to IPv6, while [1] presents a porting methodology based on the experience with the OpenH323 library.

In the case of the videoconference application Gnomeeting and openH323 library, we started with a pure IPv6 port that proved to be relatively easy and fast. Then, after having acquired some knowledge about the code design, we started to work on the simultaneous support of IPv6 and IPv4, which proved to be much more difficult.

This article starts with a presentation of the impacts of the simultaneous support of IPv4 and IPv6 on the H323 network architecture, then continues with the methodology of the port, the software design and the encountered technical difficulties.

2. H.323 VOICE AND VIDEO NETWORK

2.1 H.323 calls

H.323 is an ITU recommendation, which defines a network architecture and the associated protocols necessary to voice and multi-media calls establishment. H.323 is defined for a packet-based network, and does not impose any network protocol, which can as well be IPv4 as IPv6 or IPX. H.323 architecture makes it possible to carry out direct calls between two multimedia phones connected on the Internet or a local area network. In this case, it is necessary to know the IP address of the called party.

The main entities of an H.323 based video network are the following:

- End points: These are the H.323 clients, which are used by the end users. They can propose phone, video, fax, and application sharing functionalities.
- Gateways: Gateways can be used for the interconnection between different networks (for example an IP phone network and a traditional phone network).
- Gatekeepers: Gatekeeper makes it possible to be freed from the knowledge of called party IP address. It is then possible to call someone by his name. The gatekeeper is also able to manage the billing, and call filtering/authorization.
- Multipoint Control Units (MCUs): A Multipoint Control Unit (MCU) makes it possible to manage a conference of more than two end points. Each user connects to the MCU and then is able to discuss with all the other connected people.

Using a Gatekeeper or not, an H.323 phone call always follows the same logical order (see also):

1. Optional registration to a gatekeeper (H.225)

[*] The authors are presented in alphabetical order

2. Call setup, direct or thanks to gatekeeper (H.225)
3. Initial communication and capability exchange (H.245)
4. Establishment of audiovisual communication (H.245)
5. Call services (H.245, H.225)
6. Call termination (H.245, H.225)

The initial communication (3) is done directly between the two phones. Those will thus have to use addresses understandable by both.

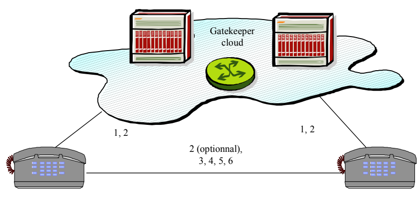


Figure 2: Call phases

2.2 Impact of simultaneous IPv4 & IPv6 address on H.323

The current specification of the H323 protocol makes it possible to manage calls independently of the network type. An H323 transmission channel is defined by two communication terminations named endpoints: A source and a destination, which are defined by an address (IPv4, IPv6, IPX, …) and a Service Access Port (TCP/UDP Port).

In general, it is possible to achieve communication between IPv4 and IPv6 hosts, as shown on Figure 3.

	IPv4 server	IPv6 server
IPv 4 client	Communicate using IPv4	Communicate using IPv4, server sees IPv4-mapped IPv6 address
IPv 6 client	Can communicate using IPv4 if the IPv6 client uses an IPv4-mapped IPv6 address	Communicate using IPv6

Figure 3: Interoperability between IPv4 and IPv6 versions running on dual-stack hosts

It can however be a challenge to have an H.323 Call seamlessly using both IPv4 and IPv6 networks. When an IP phone software starts, it opens a communication port (Service Access Port) on which it could receive H.323 signaling. A dual-stack terminal can choose to open an IPv4 port, an IPv6 port or a dual protocol port. A dual port makes it able to receive both IPv4 and IPv6 call at the same time. On the other hand if it registers at a Gatekeeper, it has to make a choice and to specify an IPv4 or IPv6 address network.

As a gatekeeper maintains a list of the connected users (names or aliases), and corresponding endpoints (address+port), a dual terminal stack could be registered twice, once in IPv4 and another in IPv6.

It is possible to carry out an IPv4 and IPv6 mixed call between two dual stack terminals. Terminal A opens an IPv4 port and calls the terminal B, which opens an IPv6 port. The messages from A to B will be carried by the IPv6 network, and the messages from B to A will be in IPv4.

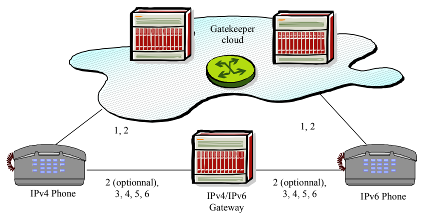


Figure 4: Call thanks to an IPv4/IPv6 Gateway

Problems arise if one wants to carry out a call between a pure IPv4 terminal and a pure IPv6 terminal. The use of mapped addresses is not possible, as H.225 messages at application layer clearly exchange IPv6 addresses.

A solution could be to use a gatekeeper connected to each network, able to detect this conflict and being used as Gateway for the transmission between the two networks.

Another solution could be to use at least one dual stack H323 client, able to detect an IPv4 only party, and sending the corresponding IPv4 address in H.225 messages instead of the mapped IPv6 address it is listening on.

3. GNOMEMEETING & OPENH323 PROJECT

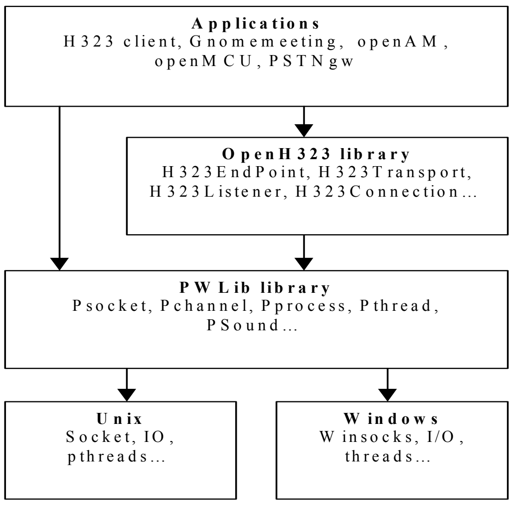
The main objective of the openH323 project [4] is to develop and provide the Internet community with a library implementing the H323 recommendation. This library, distributed under MPL license (Mozilla public license), can be used for free in commercial software. This project is led by Equivalence Pty Ltd, an Australian company.

The applications that have been developed on top of the OpenH323 library include a command line H.323 client, an H.323 videoconferencing server (MCU), H.323 answering machine, H.323 gatekeeper, H.323 to PSTN and fax modem to T.38 gateways, and GnomeMeeting, a graphical H.323 client for Unix.

The openH323 library is written in C++, and implements more than 100 classes in more than 350.000 lines of code. It proposes classes to handle H.323, H.225, H.245, RTP protocols, some audio codecs and video such as G.711 or G.722. The library is divided in two parts, one being generated from the IDL definition of the standards, the other implementing the behavior of these generated classes in C++.

The openh323 library relies on the PWLib library for all its systems calls. PWLib offers platform independent C++ classes for socket management, thread, tables, sort algorithms, null strings... It is thus possible to compile at the same time on a Unix platform and a Windows platform without modifying a single line of code. This library contains more than 300 classes implemented in more than 150.000 lines of source code.

Gnomemeeting [5] is the most famous Linux based videoconference software used over the Internet. Developed by the University of Leuven, it is an open source H.323 software which can interoperate with popular H.323 products including Microsoft's Netmeeting. It relies on the openh323 library for the H.323 protocol management.



Applications
H323 client, Gnomemeeting, openAM, openMCU, PSTNgw
OpenH323 library
H323EndPoint, H323Transport, H323Listener, H323Connection...
PWLib library
Psocket, Pchannel, Pprocess, Pthread, PSound...
Unix
Socket, IO, pthreads...
Windows
Winsocks, I/O, threads...

Figure 6: OpenH323 function stack

4. METHODOLOGY OF AN IPV6 PORT

4.1 Need for a Methodology

After having taken the decision to carry out the IPv6 port of the Gnomemeeting software, and thus the port of openH323 and pwlib libraries, we found ourselves facing a mountain of more than 500.000 lines of source code. Without a clear methodology of work this task was dedicated to failure.

Below we present a step-by-step methodology that can be used in order to port a large project to IPv6:

- Study and understand the source code, highlighting the points where a change in the program's logic is probably necessary.
- Parse the source code with an automatic tool like Checkv4.exe [6].
- Modify the source code lines reported by the automatic tool, which are probably going to be rather straightforward.
- Make any other necessary modifications in more subtle places not reported by the automatic tool.
- Test and debug the code, correcting any issues that arise.
- Verify completeness of porting effort.

4.2 Simple Windows 2000 Both IPv4 &IPv6 support

The first thing to be made was to cut out work in small tasks. We thus focused initially on the port of a basic application using the openH323 library: simpleH323, a voice only internet phone without graphical interface, which uses only a subset of H.323 functionalities and which works perfectly over IPv4.

We carried out this port in the Microsoft Visual Studio 6 development environment, under Windows 2000 with IPv6 SDK. We have incorporated the IPv6 modifications using conditional compilation directives, directly modifying the code of the H.323 classes, even those that had been generated from IDL files of the H.323 standard. The library thus compiled was IPv4 only, or IPv6 only. This port was relatively short, about a few days. Really useful in this phase can be an automatic tool like Microsoft's Checkv4.exe [6] that parses the source code and reports IPv4 dependencies, and a text editor with 'find keyword in file' functionality able to recursively search in the whole source tree.. We then could make a summary of the modifications to be done for a simultaneous support of IPv4 and IPv6.

We then set out again of a clean and synchronized CVS snapshot of the sources. We incorporated the modifications needed to limit the redundancies and not to modify the generated files. This phase was relatively long (many weeks). The results were incorporated in the OpenH323 project's CVS repository.

4.3 Simple Linux and GnomeMeeting Both IPv4 &IPv6 support

Once the current CVS sources were validated under Windows 2000 with a simple application, we started to work under Linux. Almost no modification was necessary for the simple set of functions necessary to the simpleH323 application. We then tested another simple applications: openAM, which is a vocal answering machine. The compilation of the current CVS version of Gnomemeeting required very recent versions of the graphic libraries, and we had to upgrade our entire Linux platform. The first tests showed us that we needed more functions than those already validated by the simpleH323 application.

4.4 Results & Mailing list

Communicating regularly our results on the openH323 mailing list allowed us to exchange with other teams interested by the subject and ready to make tests. To allow the realization of these tests by a maximum of readers of the mailing list, we wrote an 'openH323 IPv6 how to', and the binaries were placed at the disposal on Internet. Part of this 'how to' is now included in the openH323 'readme.txt' file.

A public IPv6 answering machine was deployed in the context of the IST project Satip6 [5]. This answering machine allows everybody to test remote IPv6 phone connections.

5. SOFTWARE DESIGN FOR IPV6 & IPV4 SUPPORT

5.1 Software Design

Facing the great number of possibilities ahead of us at the time of the software redesign, the main selection criteria was systematically the backward compatibility with the already existing IPv4 applications.
Because the applications use the OpenH323 classes that give an abstract view of the networking facilities, SimpleH323 and openAM did not require modifications of their source code, except for the makefiles. Some light modifications are necessary for Gnomemeeting.

The code of the openH323 library that is automatically generated from the already IPv6 aware H323 IDL files did not have to be modified, on the other hand we had to add handlers for the effective handling of the IPv6 addresses.

The core of the openH323 library relying on the PWLib classes to handle addresses and UDP/TCP connections only needed some light modifications in its internal management of the addresses.

The most significant modifications were made in the PWLib library, which provides a PIPSocket class allowing the handling of the sockets. This PIPSocket class was strongly redesigned., as can be seen in the Appendix. PIPSockets are IPv4 by default They turn into an IPv6 socket only when they are explicitly mapped onto an IPv6 address.

5.2 Impact of Multi-platform support

Most of the functions related to IPv6 addresses are defined in RFC 2553 and are common to all the platforms, although there are minor incompatibilities and differences between the platforms [6].

On the other hand network related functions do not have a common API. It is then necessary not only to implement calls specific to the platforms, but generally specific to the IPv6 SDK and the version of the platform.

Thus the management of the IPv6 routing tables, or the lists of IPv6 networks interfaces, is not implemented the same way under Linux, Sun, Windows 2000 and XP.

5.3 Use and set of #Define

As PWLib and openH323 libraries have to continue to work on systems being not IPv6, the use of conditional compilation directives was the natural solution.

The major disadvantage is the very strong need for distributing a version of the libraries compiled with the same options as the applications.

Thus, we encountered problems while trying to use the Gnomemeeting application compiled without the IPv6 flag, with openH323 and PWLib libraries compiled with this flag.

The Sockets and H323 classes defined in Gnomemeeting and openH323 had neither the same sizes, nor the same offsets for the method calls. Memory leaks and crashes at runtime when methods are called make such bugs difficult to identify.

Such a problem is due to a difference in the compilation flags setting strategy, between openH323 and the applications. Under Linux, openH323 determines itself its default options by testing the system, and thus compiles the library with IPv6 if the machine is IPv6 enabled (detection of /proc/af_inet6). Gnomemeeting requires a manual configuration of the IPv6 flags.

The solutions would be that the library could specify its compilation flags to the applications.

5.4 Textual address translation

Most of the modifications relate to the management of the addresses for End Points. In particular conversion between addresses in text formats towards binary addresses.

An H.323 IPv4 endpoint is internally stored as a textual string '256.56.34.92:5010', to specify 256.56.34.92 as the IP address and 5010 as the TCP or UDP port. The internal parsers just seek to the first double dot to split this string in two sub-strings.

An H.323 IPv6 endpoint being internally written '[x:x:x:x:x::x]:8080', it is necessary to rewrite these parsers as the first double dot is in the middle of the IPv6 address. The task becomes more complicated with IPv6 scoped addresses that can be written '[x:x:x:x:x:x%le0]:8080', and require in addition to manage the percent sign.

6. DEVELOPMENT CONCERNS

6.1 IPv6 SDK and RFC compliance

The IPv6 standard having evolved quickly, the IPv6 SDK are not all up to date and not always compatible with the IPv6 drivers of the operating system kernel.

Thus Linux 2.2 kernel and glibc, and Visual C++ 6 define the old RFC 2133 sockaddr_in6 structure that does make only 24 bytes, when the RFC 2553 requires a 28 bytes structure.

It is necessary to upgrade to Linux 2.4, and patch manually Visual C++ 6 to be RFC 2553 compliant, which defines the field sin6_scope_id necessary to the support of the scoped addresses. In the same time, the IPv6 kernel drivers of

Windows 2000 are RFC 2553 compliant, and wait for 28 bytes structure parameters.

6.2 Address resolution with and without square brackets

The choice to implement a function as platform dependent or independent is sometimes clever. Even if the interface is identical, the behavior of some name resolution functions under the various platforms is not homogeneous. Linux accepts the IPv6 addresses with or without the square bracket (e.g. [x:x:x:x::x]), whereas Windows manages only clean addresses without the square bracket (e.g. x:x:x:x::x).

The solution here is to systematically clean the addresses on all the platforms, even if it is not really needed.

7. APPLICATIONS

During the IPv6 port, we focussed on three applications. Pre-compiled binary for Windows and Linux are available on Internet for tests:

- SimpleH323: SimpleH323 proved to be very useful as very simple. It enabled us to focus on the H323 library. It is now fully functional under IPv6 and IPv4.
- OpenAM: OpenAM wasn't more complicated than simpleH323, it proved to be useful for both local and remote tests. It enabled us to start the first IPv6 debugging of Gnomemeeting with a stable remote IPv6 phone with a predictable behavior.
- Gnomemeeting: This application proved to be the most complex in particular because it uses almost all functionalities of the h323 library and many advanced functions of pwlib.

8. CONCLUSION & FUTURE WORK

The porting of the Gnomemeeting software and the openH323 and PWLib libraries, lead us to focus on some technical points that must be taken into account while in the design phase of an application, when this application is supposed to benefit from the new IPv6 networks.

It appears clearly that the use of an object software architecture based on a class able to manage simultaneously IPv4 and IPv6 networks makes it possible to develop applications for IPv4 which require very few modifications to work also on IPv6 simultaneously.

Currently only the IPv6 and IPv4 addresses are stored in the PIPSocket::Address class. An additional field scope should be added so that scoped addresses can be taken into account. Also, many parts of the PWLib library are not used by GnomeMeeting and probably contain IPv4 dependencies that have to be identified and modified.

9. REFERENCES

[1] "Adding IPv6 capability to Windows Socket Applications", Microsoft Corporation

[2] "Porting Networking Applications to the IPv6 APIs", Sun Microsystems

[3] C. Bouras, A. Gkamas, K. Stamos, "From IPv4 to IPv6: The case of OpenH323 Library", in Proceedings of SAINT 2003 Workshop, 2003

[4] OpenH323 project, http://www.openh323.org

[5] GnomeMeeting, http://www.gnomemeeting.org/

[6] Microsoft IPv6 Technology Preview for W2000, http://msdn.microsoft.com/downloads/sdks/platform/tpipv6.asp

[7] IST SatIP6, http://satip6.tilab.com

10. APPENDIX: DEFINITIONS IN SOURCE

```
TransportAddress ::= CHOICE
{
      ipAddress         SEQUENCE
      {
              ip        OCTET STRING (SIZE(4)),
              port      INTEGER(0..65535)
      },
      ip6Address        SEQUENCE
      {
              ip        OCTET STRING (SIZE(16)),
              port      INTEGER(0..65535),
      },
      …
}
```

ASN definition of an H225 transport address (h225.asn)

```
class PIPSocket : public PSocket
{
      class Address : public PObject {
              unsigned version;
                      union {
                              in_addr four;
#if P_HAS_IPV6
                              in6_addr six;
#endif
              } v;
      } ;
}
```

C++ partial definition of the PIPSocket class (ipsock.h)

It was necessary to allow the choice between the use of the structures 'sockaddr_in' or ' sockaddr_in6 ' at the call time for many functions, such as getpeername, bind, getnameinfo, accept, recvfrom... Rather than to use conditional compilation, we defined a Psockaddr class, shown below, based on 'sockaddr_storage', which was dynamically casted in 'sockaddr_in' or 'sockaddr_in6 '.

```
class Psockaddr
{
  public:
    Psockaddr(const PIPSocket::Address & ip, WORD port);
    …
    socklen_t           GetSize()        const;
    PIPSocket::Address  GetIP()          const;
    WORD                GetPort()        const;
  private:
    sockaddr_storage storage;
};
```

C++ partial definition of the local Psockaddr class (sockets.cpp)